Automated Identity based Approach to verify Data Possession in Public Cloud

Pooja G. Natu
Department of Computer Engineering
AISSMS COE, Pune
Pune, India

Shikha Pachouly
Department of Computer Engineering
AISSMS COE, Pune
Pune, India

ABSTRACT
Cloud-based outsourced storage relieves the client's load of storage management and preservation by providing an equivalently flexible, inexpensive, location-independent platform. As clients no longer have physical control of data, outsourced data integrity checking is of crucial importance in cloud storage. It allows the clients to verify data intactness and correctness without downloading the entire data. As the verification is to be done at client end, the integrity checking protocol must be efficient to save client's time. Another aspect of the protocol is flexibility, which improves the quality of integrity checking by allowing user specific block partition size. Moreover in case of company oriented scenario, maintaining log records of each verification request can help in security analysis. Taking these three points into consideration, we have proposed the flexible, automated and log based RDPC model as: Auto ID-RDPC model for single-cloud storage. The proposed model is based on bilinear pairings and RDPC technique. The approach eliminates certification management with the help of Identity management and additionally provides log management towards data integrity. The model makes client free from initiating verification request and keeps track of previous records which reduces client's time. The principle concept here is to make data integrity checking a painless job for any client. Our results demonstrate the effectiveness of our approach.

General Terms
Cloud Computing; Cloud Storage Security; Data Integrity checking

Keywords
Remote data Possession Checking; Identity Based Management; MD5 Technique;

1. INTRODUCTION
The cloud computing facilitates many straight benefits to clients as on demand service, location independence, elasticity, network based model, resource pooling and so on. The cloud storage provisioning is one of the important services of cloud computing. The cloud storage facilitates massive amount of data storage which magnetize small and medium scale organizations to utilize remote storage for efficient and economic storage management. It is a model of data storage where the data is stored in logical pool, the physical storage spans multiple servers and the physical environment is actually owned and managed by a hosting entity. The tasks like keeping the data available and accessible, and the physical environment protected and running is done by cloud storage providers. Though the cloud storage is grabbing the market, many security issues hinder the client to move their data on remote server. The critical issue of data integrity comes whenever client uploads data on un-trusted servers. In such scenarios, clients need to implement strategies to prove originality of data. Also the client must not need to retrieve entire data to check integrity in order to have reduced network and computational overhead. The central problem of remote data security is ensuring integrity of remotely located data which is out of client reach. The companies are moving their sensitive data over cloud in order to gain economic and operational benefits. Ensuring cloud users that their data is intact is especially important when users are companies. The remote data possession checking (RDPC) is a primarily designed to address the data integrity checking issue in company environment.

1.1 Motivation
Storing the data in cloud environment becomes natural and essential too. But, security is one of the major concerns for all entities in cloud services. The data owners need to worry about misuse of data, unauthorized access to the data and data loss. Moreover, the cloud service providers (CSP) may be untruthful and they may discard the data which has not been accessed or rarely accessed to save the storage space or hide considerable data loss caused during migration or for any other reason to maintain good reputation in market. As a result, data owners need to be convinced that their data are correctly stored in the Cloud. The previous studies considered many parameters but lacking in reducing client interaction and logging.

1.2 Related Work
In cloud computing, remote data integrity checking is an important security issue in today's taxonomy. The client's massive data is not in local environment and outside control. The malicious cloud server may damage the client's data in order to gain more benefits and to maintain their reputation. Many researchers proposed the equivalent system model and security model to work on security issue. Latest work was proposed by Wang H. which was an ID based RDPC approach for multi cloud storage [1]. In 2007, provable data possession (PDP) paradigm was proposed by Ateniese et al. [2], where the verifier can check remote data integrity with a high probability. Based on the RSA technique, they provided two provably secure PDP schemes. As a continuation, Ateniese et al. proposed dynamic PDP model and concrete scheme [3] which failed in providing a support for insert operations. Taking this limitation in consideration, in 2009, Erway et al. proposed a full-dynamic PDP scheme based on the authenticated flip table [4]. Then F. Seb´e et al. did the

similar work in [5]. The central idea of PDP is it allows a verifier to verify the remote data integrity even with no retrieving or downloading the complete data. It is based on probabilistic proof of possession by sampling random set of blocks from the server, which reduces I/O costs considerably. The verifier which may be the client needs to preserve small metadata to carry out the integrity checking tasks. The novel approach towards PDP was proposed by Wang in 2012 which was the concrete scheme of proxy PDP in public clouds [6]. As the era was moving towards multi cloud environment, Zhu et al. proposed the cooperative PDP in the multi-cloud storage which is for hybrid cloud [7]. The multiple replica PDP approach was then proposed by Ateniese et al [8].

In reference with the Ateniese et al.'s revolutionary effort, many remote data integrity checking models and protocols have been proposed [9], [10], [11], [12], [13]. Shacham presented the first proof of retrievability (POR) scheme with provable security in 2008 [14]. The idea of POR is, the verifier can check the remote data integrity and even can retrieve the remote data at any time. The state of the art can be found in [15], [16], [17], [18]. In some scenarios, the client can delegate the remote data integrity checking job to the third party which is commonly referred as the third party auditing in cloud computing [19], [20], [21], [22].

One of benefits of cloud storage is to enable global access to data with location independence. This implies that the end devices may be mobile and limited in computation and storage. We have presented the comparison on currently available PDP systems in [33] along with advantages and disadvantages of the schemes.

1.3 Our Contribution

In data integrity checking tasks, the preprocessing of file, generation of challenges and verification of proof are key activities. This paper focuses on providing highly flexible security provisioning approach for company oriented surroundings where client may need to check integrity of data on timely basis. We recommend the novel RDPC model as AutoID-RDP. With the help of Identity management the protocol is ended fairly efficient. Additionally, the protocol is automated and produces and maintains log file in order to preserve verification records for an organization. The protocol provides flexibility in data preprocessing phase where user can select the size of file blocks to have projected security over the file.

1.4 Paper Organization

The rest of the paper is organized as follows. Section 2 formalizes the Auto ID-RDPC model. Section 3 presents our Auto ID-RDPC protocol with a detailed description of techniques used in the model. Section 4 presents the results of the proposed model. Finally, Section 5 concludes the paper.

2. SYSTEM ARCHITECTURE AND MODELING

The Auto ID-RDPC system model and the detailed description of the protocol are presented in this section. The model comprises of various entities and these entities perform required operations. The entities of system are shown in Figure 1 and can be given as:

1. Client: an entity, which has massive data to be stored on the single-cloud for maintenance and computation, in our scenario, the client is any employee of an organization.
2. CSS (Cloud Storage Server): an entity, which is remotely located and managed by cloud service provider. It has significant storage space and computation resource to manage the client's data. More specifically these are un-trusted and remotely located entities.
3. PKG (Private Key Generator): an entity, when receiving the identity, it outputs the corresponding private key to the client. It reduces certification management overhead drastically.

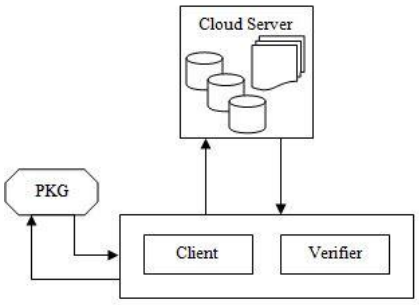

Figure 1: System Model of ID RDPC

In the cloud paradigm, clients can be relieved from the burden of storage and computation by putting the large data files on the remote cloud servers. As the clients no longer possess their data locally, it is of critical importance for them to ensure that their data are being correctly stored and will not get altered or damaged. That is, clients should be equipped with efficient security means so that they can periodically verify the correctness of the remote data even with no existence of local copies. We next formally define an Auto IDRDPC scheme. We have then specified a security definition to capture security requirements of the model.

Definition 1(Auto IDRDPC): An Auto IDRDPC protocol is a set of six polynomial time algorithms (Setup, Extract, FileBreak, TagGen, GenProof, CheckProof) which run as follows:

1. (*params*, *mpk*, *msk*) ← Setup(1^k) is the parameter-generation algorithm. It accepts the security parameter *k* as input and outputs three things: the system public parameters *params*, the master public key *mpk* and the master secret key *msk*. This action is performed by PKG in order to generate security parameters.
2. (pk_{ID}, sk_{ID}) ← Extract(1^k, *params*, *mpk*, *msk*, *ID*) is a probabilistic key-extraction algorithm that is run by PKG to extract the client's private key. It takes three inputs as: the public parameters *params*, the master public key *mpk*, the master secret key *msk*, and the identity *ID* of a client. As a consequence, Extract(.) output the private key sk_{ID} equivalent to the client with identity *ID*.
3. F ← *FileBreak(F,n)* is an algorithm that is run by client to break the file into required number of blocks which depends on expected data security. The function takes a file and factor n as parameter and gives a set of file blocks as an output. This is an improvement which makes the model more flexible.
4. T_m ← TagGen(*m*,*t*) is an algorithm that is run by the client to produce the verification metadata. It takes

two input parameters: a file block *m*, and a time factor t and returns the verification tag T_m which is retained by client for data integrity verification.

5. $V \leftarrow$ GenProof(*F, chal*) is run by the CSS to produce a proof of data possession. It receive as inputs the prearranged collection *F* of file blocks, a challenge *chal.* The function returns a proof of data possession *V* for the blocks in *F* that are determined by the challenge *chal*.
6. *{* "*success*", "*failure*"*}* ← CheckProof(*chal, V*) is run by the client to validate a proof of data possession. It takes inputs: a challenge *chal* and a proof of data possession *V* . Following verification process it returns "success" or "failure", representing that *V* is a correct proof or not. This entry is upholded in the log which is addition over standard ID RDPC approach.

There are two variations of data integrity checking policy: Public verifiability and Private verifiability. In the CheckProof, if the private key sk_{ID} is necessary, the ID-RDPC protocol is considered as a Private AutoID-RDPC. As we are considering company oriented environment, the ID-RDPC we propose in this paper belongs to this type.

In accumulation to communication and computation overheads as low as possible, an Auto ID-RDPC protocol should satisfy the following requirements:

1. The verifier should not be required to keep an entire copy of the file(s) to be verified on local machines. It would be impractical and infeasible for a verifier to replicate the whole content. Also, the cost will be improved. Storing a reduced-size digest of the data at the verifier should be adequate for verification of the CSS-generated proof.
2. The protocol has to stay protected even if the prover is malicious. A malicious prover is interested in proving awareness of some data that he does not entirely know. Here, security means that such a prover will fail in convincing the verifier on his authenticity.
3. It must to be possible to run the verification an limitless number of times as employees may check the data any number of times.

In order to record above specified security requirements, we define here the security of an Auto ID-RDPC protocol as follows:

Definition 2 (Unforgeability): We can say an ID-RDPC protocol is safe and sound if for any (probabilistic polynomial) adversary *A* there is negligible probability that *A* wins the ID-RDPC game on a set of file blocks i. The ID-RDPC game among the adversary *A* and the challenger *C* can be illustrated as follows:

- Setup: The challenger executes (*params, mpk, msk*) ← *KeyGen*(1^k), throws (*params, mpk*) to the adversary *A* and keeps the master secret key *msk* confidential.
- First-Phase Queries: The adversary *A* adaptively makes a number of diverse queries to the challenger *C*. Each query can be one of the following:
- Extract queries: The adversary can ask for the private key of any false identity *ID*. The challenger obtains the private key sk_{ID} by executing Extract(*params, mpk, ID*) and sends sk_{ID} to the adversary. And denote the extracted identity set by S_1.
- Hash queries: The adversary adaptively formulates *hash* function queries. The challenger responds with the *hash* values to the adversary.
- Tag queries: The adversary makes block-tag pair queries adaptively. For a query *m* acknowledged from the adversary, the challenger calculate the tag $T_m \leftarrow$ TagGen(*m,t*) and sends it back to the adversary. Without defeat of generality, let $\{(m_i, T_i): i \in I_1\}$ be the set of queried block-tag pair.
- Challenge: The challenger generates a challenge *chal* which describe a ordered collection $\{ID^*, i_1, i_2, \ldots, i_c\}$, where *ID* not in S*1 is the identity of a non-corrupted client, $\{i_1, i_2, \ldots, i_c\}$ not a subset of I_1, and *c* is an positive integer. The adversary is essential to offer a proof of data possession checking for the blocks as $m_{i1}, .., m_{ic}$.
- Second-Phase Queries: Similar to the First-Phase Queries. Suppose that the Extract query identity set is S_2 and $\{(m_i, T_i): I \text{ not in } I_2\}$ is the set of queried block tag pairs in this next phase. The restriction is that $\{i_1, i_2, \ldots, i_c\}$ is not subset of $I_1 \cup I_2$ and *ID* not in S*1∪*S*2.
- Forge: The adversary *A* computes a proof of data possession examination *V* for the blocks indicated by *chal* and returns *V*.

In this security identification, we say that the adversary A wins in the Auto IDRDPC game if CheckProof(*chal, V*) ="*success*". And in this protocol the chances of getting result as success are negligible.

Definition 2 states that, for the challenged blocks, an untrusted PCS cannot construct a proof of data possession if the blocks have been modified or deleted. As the definition does not affirm clearly the status of the blocks that are not challenged, this is not sufficient for the Auto IDRDPC protocol. Practically, a secure RDPC protocol even needs to guarantee that after validating the PCS-generated proof, a client can be convinced that all of his outsourced data have been kept unbroken with a high probability. This observation gives the following security definition.

Definition 3 ((ρ, δ) security): We can say the protocol is (*ρ, δ*) secure if, given a fraction *ρ* of PCS corrupted blocks, the probability that the corrupted blocks are detected is at least *δ.*

The notations used throughout this paper along with the descriptions are listed in Table 1.

Table 1: Notations and meanings

Notation	Description
G_1	Cyclic Multiplicative group with order q
G_2	Cyclic Multiplicative group with order q
Z_q^*	{1,2,3,4,5,...,q-1}
G	Generator of group G_1
D	Generator of group G_2
h(x)	Cryptographic hash function

(x,Y)	Master secret/ Public key pair
(ID,sk_{ID})	User's Identity/Private key Pair
(R,σ)	User's Private Key sk_{ID}=(R, σ)
N	Total Number of blocks
F=(f_1,f_2,..f_n)	File F divided into n blocks
CSP	Cloud Storage Provider
∑	Tag metadata created per block
T_{cl}	User side table to store metadata
C	Client (here an employee)

3. PROPOSED AUTO IDRDPC PROTOCOL

The Auto IDRDPC approach is presented in this section. The model is based on bilinear pairings, which are reviewed below.

3.1 Bilinear Pairing

Let $G1$ and $G2$ be two cyclic multiplicative groups with the same prime order q, *i.e.*, $|G1| = |G2| = q$.

Let $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear map [24], which satisfies the following properties:

1. Bilinearity:
   $\forall g_1, g_2, g_3 \in G_1$ and $a, b \in Z_q$,

   $$e(g_1, g_2g_3) = e(g_2g_3, g_1) = e(g_2, g_1)e(g_3, g_1)$$

   $$e(g_1^{\,a}, g_2^{\,b}) = e(g_1, g_2)^{ab}$$

2. Non-degeneracy:
   $\exists\, g_4, g_5 \in G_1$ such that $e(g_4, g_5) \neq 1_{G2}$.

3. Computability:
   $\forall g_6, g_7 \in G_1$, there is an efficient algorithm to calculate $e(g_6, g_7)$.

A bilinear map e can be constructed with the modified Weil or Tate pairings on elliptic curves. Our Auto IDRDPC scheme is constructed on the gap Diffie-Hellman group, where the computational Diffie-Hellman (CDH) difficulty is hard while the decisional Diffie-Hellman (DDH) problem is easy.

3.2 Auto IDRDPC Protocol Construction

This protocol comprises the procedures Setup, Extract, FileBreak, TagGen, SetTimer, GenProof, CheckProof and LogRecord. The protocol architecture is described as follows:

1. Initially *PKG* creates the public and private key for the client along with public parameters. In Extract phase, client sends the ID (unique identity) to PKG.
2. The PKG then generates the secret key sk_{ID} and sends back to client.
3. The client generates the file blocks along with corresponding block tag pairs and uploads them to PCS. Client then deletes all data from local machine and keeps only related metadata.
4. The client starts the timer for specific milliseconds.
5. The verifier who can be the client generates challenge and forwards the challenge to PCS on timer off event.
6. PCS then creates the appropriate possession proof for requested file block.
7. PCS sends the possession proof to the verifier or client.
8. The verifier checks the possession proof and logs the particular entry to log file.

The architecture of proposed model is illustrated in figure 2 below:

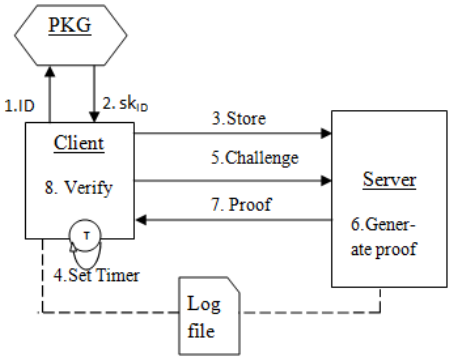

Figure 2: Architecture of our Auto IDRDPC Approach

Suppose that the number of file blocks is n. It is described below the procedures of the Auto IDRDPC scheme.

- Setup: PKG chooses a random number $x \in Z_q^*$ and sets $Y = g^x$, where g is a generator of the group G_1. PKG chooses a random item $u \in G_1^*$. Define two cryptographic hash functions as: $H : \{0, 1\}^* \rightarrow Z_q^*$, $h : Z_q^* \rightarrow G_1^*$. Let f be a pseudo-random function and let π be a pseudo-random permutation as given in equation (1):

  $$f : Z_q^* \times \{1, 2, \ldots, n\} \rightarrow Z_q^*$$
  $$\pi : Z_q^* \times \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\} \quad (1)$$

  Finally, PKG publishes
  $\{G_1, G_2, e, q, g, Y, u, H, h, f, \pi\}$ and keeps x as the *master key*.

- Extract. A client submits his identity *ID* to PKG. PKG picks $r \in Z_q^*$ and computes equation (2) as:

  $$R = g^r,\ \sigma = r + xH(ID,R) \bmod q \quad (2)$$

  PKG sends the private key $skI_D = (R, \sigma)$ to the client by a secure channel.

- FileBreak(F,n): This function uses a simple iterative algorithm to break the file in given number of blocks. The boundary of a block is fixed and must not be overloaded. The factor, that is how many blocks are to be generated is taken as n and F is actual file to be loaded on CSP. The set F can be

defined as $\{f_1,f_2,..,f_n\}$ where n is number of blocks generated from single file F.

- TagGen(sk_{ID}, *F*, *i*): For simplicity one must consider that the client generates the tags successively according to the counter *i*. That is, the client generates a tag for a message block m_2 after m_1, which imply that the client maintains the latest value of the counter *i*. For m_i, the client performs MD5 hashing along with time stamp and the TagGen procedure is as follows in equation (3):

  1) Compute
  $T_i = (h(i)).t$ *(3)*
  2) Output T_i and send (m_i, T_i) to the PCS.

- GenProof($F = (m_1,m_2,. . . ,m_n)$, *chal*,): In this procedure, the verifier who can be the client herself queries the PCS through local server for a proof of data possession of *c* file blocks whose indices are randomly selected using a pseudo-random permutation keyed with a fresh randomly-chosen key for each challenge. Let $chal = (c, k_1)$. Then, the PCS does:
  1) For $1 \leq j \leq c$, compute the indices of the blocks for which the proof is generated: $i_j = \pi_{k1}(j)$.

  In this step, the challenge *chal* defines an ordered set as $\{c, i_1,\ldots,i_c,\}$.

  2) Compute the tag for the selected file block with same equation used in Tag Generation phase.

  3) Output $V = (T, m)$ and send *V* to the client as the response to the *chal* query.

- CheckProof(*mpk, skID, chal, V*): Upon receiving the response *V*, the verifier (who can be the client herself) does confirmation of challenge and received verification.

- Log entry is saved in log file for as file block id, time and result as success or failure.

4. SYSTEM PERFORMANCE ANALYSIS

The model needs to get analyzed based on two major parameters: Computational cost and communication overhead. Initially, we analyze the performance of our proposed Auto ID-DPDP protocol from the computation and communication overhead. Afterwards, we examine our proposed model with ID-DPDP protocol's properties of flexibility and verification logging.

The proposed model does not incur major additional *computation* costs than standard ID-RDPC approach. The model provides flexible approach in pre processing which cause considerable computational overhead. If we have file f and client wants to break it into n blocks, the computational complexity is O(n) to divide the file. Practically, client provide value of n<c for standard file storage where c is a constant. The client computations are carried out on timer basis which added a minute complexity to deal with the timer. The approach supports logging activity which needs less amount of time to perform I/O operations on client end.

The *communication* overhead caused varies which is based on number of blocks generated by client. We have used Identity management to reduced additional communication. The U.S. National Bureau of Standards and ANSI X9 have determined the shortest key length requirements: 1024 bits for RSA and DSA, 160 bits for ECC.

Simulation: To study the prototypal implementation of Auto IDRDPC approach, we have simulated the protocol by using Java programming language with JPair Library (JPair v1.03). In the simulation, Cloud Storage is simulated on FUJITSU Lifebook A Series Laptop with the following settings:

- CPU: Intel Core i3-2020M @ 2.40GHz
- Physical Memory: 4GB DDR3 1600MHz
- OS: Windows 8 OS 8.1 Pro

The client works on Lenovo Laptop with the following settings:

- CPU: CPU I PDC E6700 3.2GHz
- Physical Memory: DDR3 2G
- OS: Windows 7 OS

In the simulation, we choose an elliptic curve with 160-bit group order. The web service is hosted on client side server. The service provides all facilities to ensure data integrity to client. Here the client is considered as an employee of organization which has public cloud storage. The client and server software specifications are as follows:

- Web Server: Apache Tomcat 7
- Spring Web MVC 3.0
- Java 7 or above
- XML specifications
- HTML 5 and CSS 3
- Eclipse Juno IDE

We have compared our approach with previous studies as listed below in table 2.

Table 2. Comparison of our scheme with other techniques

Schemes	Query	Response	Storage	Automated	Log Based
[1]	log2 n+2log2 q	$1G_1$+slog$_2$ q	O(n)	No	No
[2]	$3Z^*_q$ (480)+c	$1G_1$+$1Z^*_q$ (480)+c	O(1)	No	No
Our	b_i+*16n*	b_i+255+c	O(1)	Yes	Yes

As per the standards, we analyze the communication overhead caused by our model, which mainly comes from the queries and responses. In an Auto ID-RDPC query, the client needs to send the block id of available blocks to CS. In case of response, the CS needs to respond with 1 element of set T* to the client. The total communication is about (16)*n+255+c bits where n is number of blocks requested, in this approach we are assuming single query submission. The scheme [2] gives considerable performance but causes additional cost c for client request initiation and major communication overhead due to client reference. If client needs to generate challenges with small durations, the constant value grows rapidly.

The following graph indicates the query generation time comparison. As our approach provides automated query generation, the client interaction is less and hence as query generation goes increasing our approach provides steady behavior as shown below.

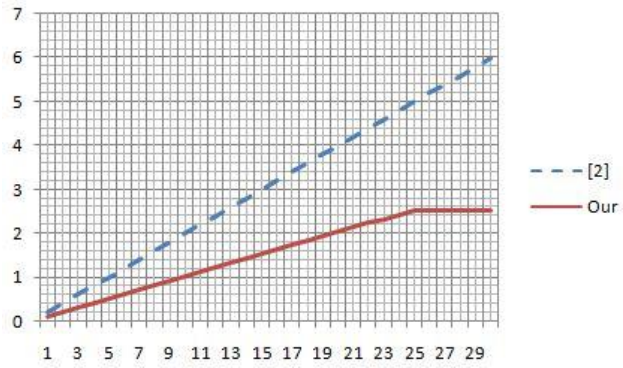

Figure 3: Graph to represent query generation of our scheme and [2].

Our model reduces this constant time by providing timer based approach for challenge generation trigger. This amount of communication is reasonable with current communication technologies and will be fixed for any number of files uploaded on server. In Ateniese *et al.*'s scheme, it uses as many as 6*1024=6144 bits [3]. In Table 2, we compare the communication overheads of our Auto ID-RDPC protocol with respect to query and response time needed for the integrity checking. In our approach Query generation is faster and Response formation is little bit burdened.

4.1 User Interface Design

The UI of proposed system is generated with JSP technology and shown as below.

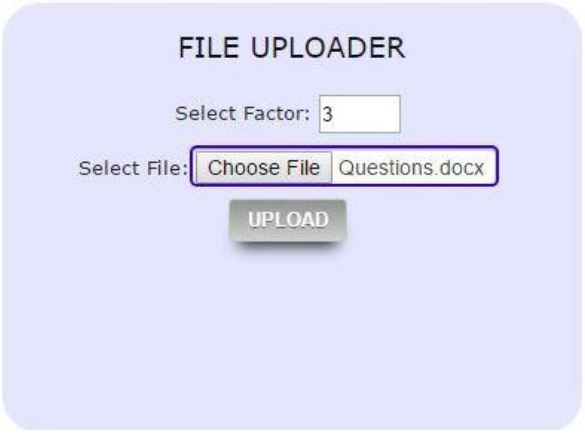

Figure 5: Upload File option

This upload file screen provides a way to enter factor, which is used for file breaking and a file which user wants to upload on remote cloud server.

Figure 6: Timer Setting Screen

This screen ensures that the possession will be requested after given time duration and will be recorded in verification log file for future use

Figure 7: Verification Log

The verifications are generated automatically for random blocks to maintain data possession and the records can be viewed as shown above.

5. CONCLUSION

This paper formalizes an Auto IDRDPC model appropriate for company-oriented cloud storage. We present the novel Auto ID-RDPC protocol proven secure under the assumption that the CDH problem is hard.

The protocol maintains the log records which facilitates verification analysis. The approach allows clients to set time interval after which the challenge generation will be performed. Hence the client time is saved and data verification is performed automatically. The modified tag generation is algorithm is used to ensure uniqueness of tag. The flexibility is provided by allowing user to select factor for file breaking activity.

In addition to the structural advantage of removal of certificate administration and verification, our protocol gives enhanced performance and provides logging, automation and flexibility.

6. REFERENCES

[1] Wang H., "Identity-Based Distributed Provable Data Possession in Multi-Cloud Storage", Services Computing, IEEE Transactions 2014, (Volume:PP, Issue 99.)

[2] Huaqun Wang, Qianhong Wu, Bo Qin , Domingo-Ferrer, J., "Identity-based remote data possession checking in public clouds", Information Security, IET (Volume:8 , Issue: 2), pp. 114 – 121, 2014.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson,

D. Song, "Provable Data Possession at Untrusted Stores", *CCS'07*, 2007.

[4] G. Ateniese, R. DiPietro, L. V. Mancini, G. Tsudik, "Scalable and Efficient Provable Data Possession", *SecureComm 2008*.

[5] C. C. Erway, A. Kupcu, C. Papamanthou, R. Tamassia, "Dynamic Provable Data Possession", *CCS'09*, pp. 213-222, 2009.

[6] F. Seb´e, J. Domingo-Ferrer, A. Mart´ınez-Ballest´e, Y. Deswarte, J. Quisquater, "Efficient Remote Data Integrity checking in Critical Information Infrastructures", *IEEE Transactions on Knowledge and Data Engineering*, 20(8), pp. 1-6, 2008.

[7] H.Q. Wang, "Proxy Provable Data Possession in Public Clouds," *IEEE Transactions on Services Computing*, 2012.

[8] Y. Zhu, H. Hu, G.J. Ahn, M. Yu, "Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage", *IEEE Transactions on Parallel and Distributed Systems*, 23(12), pp. 2231-2244, 2012.

[9] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu, S. S. Yau, "Efficient Provable Data Possession for Hybrid Clouds", *CCS'10*, pp. 756-758, 2010.

[10] R. Curtmola, O. Khan, R. Burns, G. Ateniese, "MR-PDP: Multiple Replica Provable Data Possession", *ICDCS'08*, pp. 411-420, 2008.

[11] A. F. Barsoum, M. A. Hasan, "Provable Possession and Replication of Data over Cloud Servers", CACR, University of Waterloo, Report2010/32, 2010. Available at http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-32.pdf.

[12] Z. Hao, N. Yu, "A Multiple-Replica Remote Data Possession Checking Protocol with Public Verifiability", *2010 Second International Symposium on Data, Privacy, and E-Commerce*, pp. 84-89, 2010.

[13] A. F. Barsoum, M. A. Hasan, "On Verifying Dynamic Multiple Data Copies over Cloud Servers", *IACR eprint report 447, 2011*. Available at http://eprint.iacr.org/ 2011/447.pdf.

[14] A. Juels, B. S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files", *CCS'07*, pp. 584-597, 2007.

[15] H. Shacham, B. Waters, "Compact Proofs of Retrievability",b*ASIACRYPT 2008*, LNCS 5350, pp. 90-107, 2008.

[16] K. D. Bowers, A. Juels, A. Oprea, "Proofs of Retrievability: Theory and Implementation", *CCSW'09*, pp. 43-54, 2009.

[17] Q. Zheng, S. Xu. Fair and Dynamic Proofs of Retrievability. *CODASPY' 11*, pp. 237-248, 2011.

[18] Y. Dodis, S. Vadhan, D. Wichs, "Proofs of Retrievability via Hardness Amplification", *TCC 2009*, LNCS 5444, pp. 109-127, 2009.

[19] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu, "Zero-Knowledge Proofs of Retrievability", *Sci China Inf Sci*, 54(8), pp. 1608-1617, 2011.

[20] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing", *INFOCOM 2010*, IEEE, March 2010.

[21] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", *IEEE Transactions on Parallel And Distributed Systems* , 22(5), pp. 847-859, 2011.

[22] C. Wang, Q. Wang, K. Ren, N. Cao, W. Lou, "Toward Secure and Dependable Storage Services in Cloud Computing," *IEEE Transactions on Services Computing*, 5(2), pp. 220-232, 2012.

[23] Y. Zhu, G.J. Ahn, H. Hu, S.S. Yau, H.G. An, S. Chen, "Dynamic Audit Services for Outsourced Storages in Clouds," *IEEE Transactions on Services Computing*, 2011. http://doi.ieeecomputersociety.org/10.1109/TSC.2011.51

[24] O. Goldreich, "Foundations of Cryptography: Basic Tools", Publishing House of Electronics Industry, Beijing, 2003, pp. 194-195.

[25] D. Boneh, M. Franklin, "Identity-based Encryption from the Weil Pairing", *CRYPTO 2001*, LNCS 2139, 2001, 213-229.

[26] A. Miyaji, M. Nakabayashi, S. Takano "New Explicit Conditions of Elliptic Curve Traces for FR-reduction", *IEICE Transactions Fundamentals*, pp. 1234-1243, 2001.

[27] D. Boneh, B. Lynn, H. Shacham, "Short Signatures from the Weil Pairing", *ASIACRYPT 2001*, LNCS 2248, pp. 514-532, 2001.

[28] H. W. Lim, "On the Application of Identity-based Cryptography in Grid Security", *Ph.D. dissertation*, University of London, London, U.K., 2006.

[29] S. Yu, K. Ren, W. Lou, "Attribute-based On-demand Multicast Group Setup with Membership Anonymity", *Calculater Networks*, 54(3), pp. 377-386, 2010.

[30] P. S. L. M. Barreto, B. Lynn, M. Scott, "Efficient Implementation of Pairing-based Cryptosystems", *Journal of Cryptology*, 17(4), pp. 321- 334, 2004