ISSN: 2350-0743

www.ijramr.com

International Journal of Recent Advances in Multidisciplinary Research

Vol. 04, Issue 11, pp.3050-3055, November, 2017

RESEARCH ARTICLE

EFFICIENT AND DYNAMIC PROOF OF STORAGE FOR MULTI-USER CLOUD ENVIRONMENT

***Gopichand, G., Santhi, H., Gayathri, P. and Geraldine Bessie Amali, D.**

School of Computer Science and Engineering, VIT University, Vellore, Tamil Nadu, India

ARTICLE INFO

Article History:

Received 19th August, 2017
Received in revised form
07th September, 2017
Accepted 04th October, 2017
Published online 30th November, 2017

Keywords:

Proof of Storage,
Cryptography, Deduplication,
Homomorphic Authenticated Tree,
Multi-user.

ABSTRACT

The Dynamic Proof of Storage is a very useful cryptographic scheme which enables a user to update the file or check for the integrity of the data in a cloud server. There exists dynamic schemes in the single user environment but for the multiuser environment, the algorithm is still not been analysed yet. For a multi user cloud storage system, a secure multi user deduplication technique is needed which would enable the user to skip the file upload and gain the instant automated ownership when the other owner of the file have uploaded the file on the server. This paper introduces the concept of deduplicatable dynamic proof of storage and also propose an efficient construction in order to achieve a dynamic and secure cross user deduplication simultaneously.

INTRODUCTION

Outsourcing is the strategy which is been practiced by different companies to perform some tasks and provide the services to the other companies. The most important outsourcing been done these days is the Storage Outsourcing. It is becoming popular and attractive to many industries and academic institutions because of the fact that it is of low cost and the files stored can be easily shared among different devices. Cloud storage gains attention as it is one of the most important form of storage outsourcing. Cloud storage is the form of distributed storage system which contains data stored digitally in form of logical blocks and actual data is been stored in different servers all over the geographic locations. The whole environment is been managed by some hosting company. Many different IT companies use cloud storage services of their own. The users of the company uses the servers to save their data on it, read/write the data through different devices. Cloud services are been adopted widely but the security threats and hacks remain with it even now. When the user uploads his file on the server, he should be convinced that the file which he has uploaded is not tampered or lost. Thus making data integrity the most important property while using the storage space. Traditionally Digital Signatures and Message Authenticated Codes (MACs) were employed for maintaining the data integrity. These techniques had some disadvantages. The files need to be downloaded for the integrity check. The downloading of the files incurred heavy communication costs. These methods are suitable to some extent but cannot be used when the verification requirement of the user is very frequent.

Corresponding author:* **Gopichand, G.,
School of Computer Science and Engineering, VIT University, Vellore, Tamil Nadu, India.

This requirement introduced the concept called Proof of Storage which doesn't require downloading the files from the server for checking their integrity. While maintaining the proof of storage capability, the users also require to perform insertion and updation in any of the file uploaded by him on server. This required the existing proof of storage to be modified into Proof of Storage mechanism which is dynamic in nature. The Dynamic PoS scheme uses the structures such as Merkle tree. Whenever the dynamic operations are been executed by the user, the tags are been regenerated. These tags are been used for checking the integrity of the files been uploaded. In this scheme, every block of the file has a tag attached to it. This tag is required for the integrity check. For the integrity check the user randomly selects the block index which is present in the server and sends those indexes to the server for the verification. The server then receives the challenged indexes and returns the corresponding block and their tags to the user. The verifier then gets the tags and checks for the block integrity. This verification scheme varies in PoS and Dynamic PoS. Most of the schemes encodes the block indexes into a tag and thus the user can check the integrity of the block and its index simultaneously. The Dynamic scheme cannot encode the block indexes into tags. The reason behind is that dynamic PoS involves multiple dynamic operations and hence there will be a change in the indexes of the blocks not updated. This change in the index of the non-updated blocks incurs many computation cost as well as communication cost. This concept can be best understood by the following example. Consider there are about 100 blocks of the file which is been uploaded on the server. After the upload is complete, a new block is been uploaded by the user right behind the 5th block of the original file. As a result the block indexes of 95 block are been changed and the user has to generate and send to the server 96 updated indexes even though there is only one block added newly. The above problem is been handled well in the dynamic PoS as the authenticated structures are been introduced. In this scheme, the tags are been associated with the authenticated structures. With this advantage, dynamic PoS is yet to be improved for a cross-user deduplication (Pietro and Sorniotti, 2012). The application on the client side will allow the user to skip uploading of the file which is already present in the server and get the ownership of the file instantly. Thus resulting in the less wastage of the storage space and saving the transmission bandwidth of users. In the computing world, deduplication is been employed mainly on the data. Thus the data deduplication is the techniques which is similar to the compression technique. Deduplication technique is applied to eliminate the duplicate copies of the stored data. This technique improves the storage utilization and is widely been employed on the network which has a lot of data transfer from one node to another. In this process the chunk of data which is unique are identified and then they are stored on the server. As seen from the Figure 1, when a server is been shared by multiple users, then the same data or the pattern of data may repeat some dozens of time. This huge amount of same data is then analysed and only the unique data is been stored on the server by this deduplication process. From the figure 1, we can see that multiple copies of C++/doc/txt files exists which belongs to different users. When the deduplication technique is been employed on these copies of data, the unique pattern from various files are identified and then they are stored on the server. Thus the amount of data that is been stored or need to be transferred is been reduced.

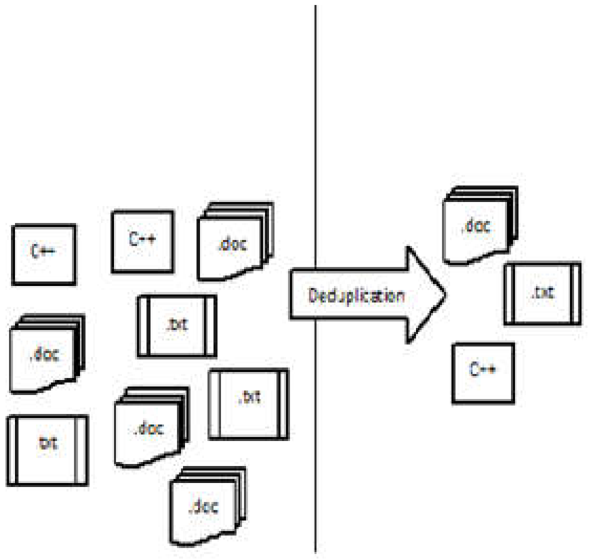

Figure 1. Deduplication

To achieve secure deduplication, there are 2 challenges. One is structure diversity and another is private tag generation. The first challenge, structure diversity is given its name as the authenticated structure can have some conflicts. Suppose if there exists a file on server F. If the user changes some of the contents of the file then the already present structure F will change to F’ (this is slightly different than original structure F). Now if a new user wants to upload the same changed file as that of the file with structure F’, then the new structure which is already present on the server has some different structure than the one which the user intends to upload. This is known as the structure diversity problem. The existing dynamic PoS, the secret key is used for the generation of the tag for checking the integrity of the file. Hence, the owner who got the ownership of the file without uploading the file himself will not be able to generate the new tag when he updates the existing file. These problems may lead to fail the dynamic PoS schemes. If the deduplication technique is combined with the dynamic PoS scheme then the problem of structure diversity will be solved. In order to solve the private tag generation problem, a separate private tag is been generated for every owner for the file and then upload the authenticated structure to the server along with the file associated with it. When the file is been updated by the user, then the corresponding authenticated structure need to be changed as well resulting into a new authenticated structure. With this we can solve both of the cross-user deduplication problems but it will lead to some unnecessary computations as well.

LITERATURE SURVEY

The main concept called the Proof of Storage was been first used by Ateniese and Kaliski. Proof of Storage is the concept and is mainly useful as it lets the user to randomly choose some of the data blocks which he want to verify and expects the server to send back the file blocks along with the tags associated with them. In the paper “Security and Privacy in Cloud Computing” (Xiao *et al.,* 2013), the author have highlighted the benefits of cloud computing and the various data centers associated with it. According to the author, the key issue on using the cloud server for storage of files is that of the security. The author have analyzed different security algorithms which can be employed on the cloud server for handling the security concern. Keeping in mind the data integrity for the data stored on cloud, (Jiang *et al.,* 2016; Juels and Kaliski, 2007) authors have tried to analyze the collusion attack in the existing employed data integrity schemes and tried to provide the user with the more secure public integrity scheme (Stefanov *et al.,* 2012; Ren *et al.,* 2015) with the added advantage of the group user revocation in a more secure way. They have used Vector commitment and local revocation group signature as the verifier in their scheme. Finally they have proved their invention better than the existing schemes by giving the experimental analysis for the same. Their analysis have proved their scheme a much secure and efficient one. To support the Proof of Storage concept, (Xu *et al.,* 2012; Cui *et al.,* 2017; Zheng, Yifeng *et al.,* 2016) have proposed a linearly homomorphic signature scheme which would have the lattice assumptions. They have constructed a lattice based PoS protocol which would let the verifier to only need to store the tags required for verification than storing the whole data blocks. Another concept called the Proof of Retrievability (PoR) (Liu *et al.,* 2015; Dodis *et al.,* 2009; Cash *et al.,* 2013; Azraoui *et al.,* 2014; Ren *et al.,* 2015; Xu and Chang, 2012) was been evaluated by Dongxi and John Zic. The main idea was to propose a scheme which will be using homomorphic encryption scheme. The scheme will check for the reliability by generating the probabilistic and homomorphic message authenticators. Then the authors have evaluated the performance of the proposed scheme with that of the existing one. To support the dynamic operations, the existing Proof of Reliability need to be modified. The authors Ren, Lina Wang and Qian Wang (Ren *et al.,* 2015) have proposed the dynamic Proof of Reliability with efficient recovery for the data which is been corrupted. The proposed system is been proved for the practical use by the experimental results. The authors Yan Zhu and Gail-Joon Ahn (Zhu *et al.,* 2012) has proposed construction of Provable Data Possession (PDP) known as Cooperative Provable Data Possession (CPDP) which will support the scalability of the services. They have proved the security of their proposed system by using multiprover zero knowledge proof system. They have also presented the efficient method for selecting the required optimal parameters in order to minimize the communication costs. Narm-Yin Lee and Yun-Kuan Chang have focused on the core issues of the cloud server (Lee *et al.,* 2011; Kaaniche *et al.,* 2015; Motegaonkar *et al.,* 2016). When the user stores his files on the server then the user need to check the integrity of the files been stored. If the server which stores the data turns up to be faulty and untrusted, then the proposed system that uses symmetric encryption of data keep the data secured to some extent.

According to Junxiang, Wang, and Liu Shengli (Junxiang *et al.,* 2012), for checking the integrity of the file on the untrusted servers Provable Data Possession (PDP) model is used. This model is static model. In order to support dynamic data, dynamic provable data model is required. In this model the file updation and the integrity proof of the file is possible as the same time. Homomorphic MAC is applied to achieve the same. The authors have studied the existing model and have proposed a batch update verifiable Dynamic Provable Data Possession model. In this model, a batch of block updates are done rather than update per block. This proposed model was more efficient than the existing one. In addition to this, author have applied BLS-like signatures for enabling verification. When the proposed model was simulated, the result was that this scheme achieved high performance for the dynamic updates in comparison to the existing scheme. Liu, Feifei, Dawu Gu, and Haining Lu (2011) proposed an improved version of the dynamic provable data possession model proposed in (Junxiang *et al.,* 2012; Ateniese *et al.,* 2011; Zhu *et al.,* 2012). The authors divided the file into various blocks. Every block had the tag associated with it and every tag had hash values linked to them. The tags were used for checking the integrity of the blocks and the hash values were used for checking the integrity of the tags. The authors compared their model with the previous model (Junxiang *et al.,* 2012) and they came to a conclusion that the computational complexity of the proposed system is reduced to a constant value. The previous model has complexity of logn. If the user need to store some secret data, so some additional space is required for storing the secret key after encryption of that secret data. This extra storage space was just about 0.02% of the original file. Thus the model was acceptable by the users in most of the cases. Chen, Bo, and Reza Curtmola proposed another (Chen *et al.,* 2012) dynamic data possession model for data auditing. They have proposed a Remote Data Checking scheme which can provide robustness and dynamic updates at the same time. They have derived many construction versions. The first construction was efficient when encoding is concerned but the updates required high communication costs. These drawbacks led them to develop the second construction which could overcome the first construction drawbacks. They employed combination of techniques like RS codes on Cauchy matrices and decoupling the encoding scheme which they used for achieving robustness. They further reduced the use of inert and delete operation and replaced them with append and modify operation for the updation of the parity data.

According to Kaaniche, Nesrine, and Maryline Laurent in paper (Kaaniche *et al.,* 2015) and other authors in (Bowers *et al.,* 2009; Di Pietro and Sorniotti, 2012; Wang *et al.,* 2010), with increase in the amount of data been stored on Cloud promotes the need of the security challenge. The authors have proposed a model for the verification of the data blocks which are been stored on different servers. The model is called set homomorphic proof of data possession, called SHoPS. This model supports the verification of the aggregated proofs. The authors have further explained the advantages of using their proposed model. Of that the first one is that the owner of the file gives authority to the third entity for the verification of his files. This releases the user from the work of verification at regular interval of time. The second advantage of the proposed model is that it can aggregate different proofs and also the verification of the blocks with a very less communication overhead. According to Douceur, John R., et al. in paper (Douceur *et al.,* 2002), the Farsite distributed file system replicates every file on multiple desktops. Replication of the file requires a lot of storage space. There should be a mechanism which can reclaim the used space from different computers (Stefanov *et al.,* 2012). When the authors conducted a survey, they found that a high number of desktop file systems have near to half of the consumed space is been consumed by the duplicate files. This proves that there is a need of a controlled file replication. They have employed two mechanisms to reclaim the space. First one is that they have used encryption which enables the duplicate files on server to condense into the space of one single file. This mechanism works even when the files belong to different users and are encrypted using different user keys. The second mechanism employed is SALAD. It stands for Self Arranging, Lossy, Associative Database. This mechanism aggregates the file contents and the location information in fault-tolerant manner. The authors have conducted simulations of this proposed system. The results show that the system is highly scalable and is very effective.

Problem Statement

Many researchers have proposed Dynamic PoS schemes. But all these schemes have been proposed for the single user cloud environment. In this type of environment, dedulication will take place only in the user's own data. The data set scheme of one user does not interfere or affect the data set of the other user. This can be well understood with the following example- Suppose a user A uploads the file F1 to the server for storage. After the upload is done, another user B tries to upload his file (File F2) to the file server. The file of the user B has the same content as that of the file F1 uploaded by the user A. Since the current environment is single user CLOUD environment, so the upload of the file by one user doesn't affect the uploading of the same file by another user. Here the deduplication takes place only in the current user data set rather than all the users as a whole. Multi-user cloud storage scheme is different than the single-user storage mechanism. Here the deduplication takes place for the files with all the user together. In order to achieve this, a secure cross side deduplication technique is required while can skip the file upload process and get the ownership of the already uploaded file immediately if the file is already uploaded by any another user. There are many shortcomings of the file upload process in the single-user cloud environment. The first and the most important of them is that the CLOUD storage space is not used efficiently. The same content with different file names uploaded by different users are uploaded to the server thus duplicating the storage space on the server. Second important shortcoming is that the deduplication in the single user environment is been handled at the file level. A secure means is needed which could inspect already uploaded contents to the server before uploading any new file content. This scheme can save a considerable amount of space on the server which can be used for other purpose.

Proposed System

The major two shortcomings in the single user cloud environment discussed above is addressed and solved in our system model. This model consists of 2 major entities. These entities are the cloud server and the corresponding users. Under the entity user, two different types of users are been defined. One is the Original User who is the one whose files are uploaded at the first onto the server. Another is the Subsequent User, who does not actually upload the file on the server but proves the ownership of that file. Both the users, the original as well as the subsequent users can update the files uploaded by them. Our deduplicatable dynamic PoS system has a total of five phases. These are pre-process phase, deduplication phase, upload phase, update and the proof of storage phase. Refer figure 2 for the flow of control as per requirement. As we can see form the figure, that the pre-process phase, deduplication and the upload phase will take place only once for a particular file. If the user makes some changes to the file, the user will edit the file and then upload it back on the server. Before making any changes in the file the user has to prove the ownership of the file. This means that any other user cannot edit the file which he has not uploaded. Only after proving the ownership of the file, the user will be allowed to edit the file and then upload it back. Pre-process phase- In this phase, the user intends to upload his file on the cloud server. It is the cloud server which decides whether the file which the user want to upload should actually be uploaded onto the cloud server or not. If the file upload decision is granted by the server, then it goes to the upload phase otherwise it will go directly to the deduplication phase. The upload phase is executed only when the file been selected by the user is not already present in the cloud server ie it is not been uploaded already. The original user encodes the local file and uploads it on the server. In the deduplication phase, file which is been selected by the user is already uploaded by another user. The subsequent user has the file locally and the server stores the structure of the local file on the server. The subsequent users need to provide the proof of ownership (Chen *et al.,* 2016) of the file without uploading it to the server. The above three phases (pre-process, deduplication and upload phase) occur only once in the whole life cycle as seen from the user's perspective. These phases occur only when the user want to upload the file to the server. If the above three phases terminate normally ie when the user has finished uploading the file to the server (upload phase) or else when the user have passed the verification of the file for uploading (deduplication phase), then it is said that the user has the ownership of the file. The upload phase is executed again if the user want to edit the file which is already been uploaded by him or if he want to check for the integrity of the file existing. The user can demand the application for the blocks of file which he want to check the integrity for.

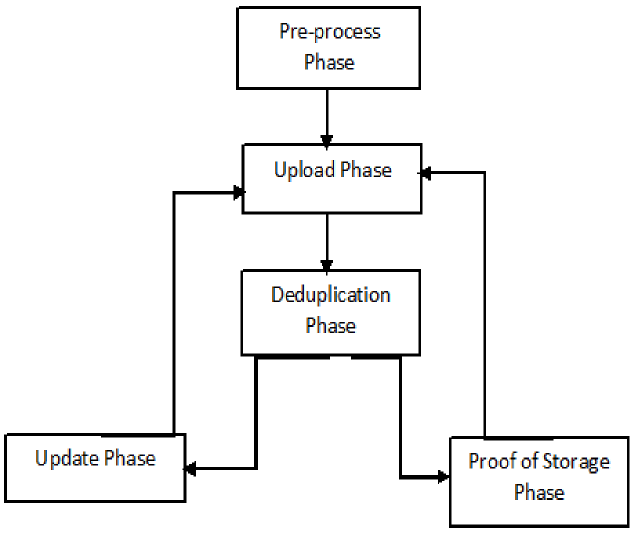

Figure 2. 5 phases of life cycle of a file

The application should then return the blocks for which user has demanded. The user can modify, insert or delete any content from the file or the whole file. In order to update the content of the file, the user has to prove the ownership of the file or should pass the verification phase first. The user updates the content of the file. For every update, the server has to save the original file and its structure if there exists the subsequent users for that file. After the updation is done, the server saves the updated part and correspondingly alter the authenticated structure of the updated part and save it for the user. Thus in this model the updation of the file is very easy as the updation is only attached with the original file and its structure. Once when the Proof of Storage phase is reached, the user can check for the integrity of the uploaded file to the server even without downloading them to the local storage. Before coming to this phase, the user has already passed the deduplication phase and proved his ownership for the file. The update and the proof of storage phase is the phase which can be passed any number of times for any file on the server. After proving the ownership of the file, the user can anytime update and check integrity without even keeping the original file locally with them.

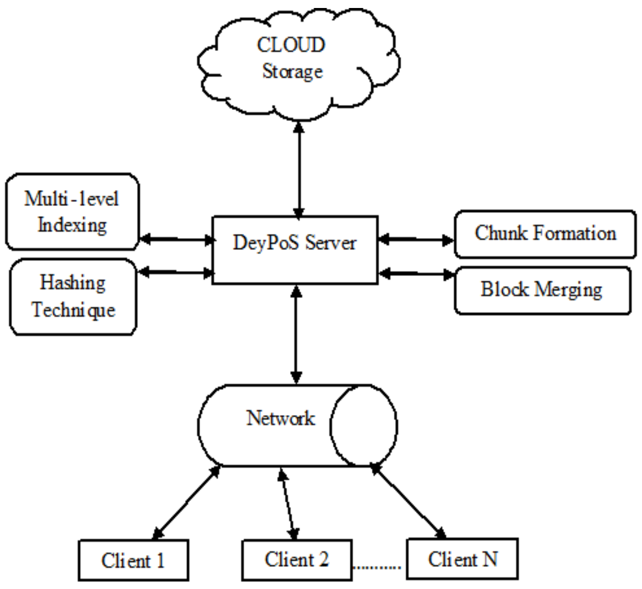

Figure 3. System Architecture

To achieve the output of the proposed system, many individual processes are employed. Refer Figure 3 for different process which is been employed in the system. It is clear from the figure that many client interact with the application for the data storage via a network. The network undergoes some process needed to execute the 5 phases of the file. The processes are Chunks formation, Block Merging, Multi-level indexing and Hashing. The subset of the above processes execute for every request by the client. The proposed system model has 5 phases of the life cycle of the file. One of the phase, Upload phase does the file uploading onto the server. In the file uploading process, the user selects the file from the client system and then the file will be sent to the file chunks formation module. The output of the file chunks formation will be the file block which then will be uploaded to the Cloud server by File Transfer Protocol. The chunks formation process will divide the whole file into many different blocks which will be then analysed for the uploading. Once the file is been uploaded, the logical block addressing will then be maintained in the database.

The file chunk formation process takes the files as the input, breaks it in the block based on the packet size and then finally designates the chunks by the hash code which are maintained as the file chunks for the user in the database. Every chunk is assigned a hash code which uniquely identifies that block. The hash code is been assigned by the hashing process. A 16 bit hash code is assigned to every block. The deduplication phase follows multilevel indexing process for the deduplication check. The block has code is divided into three parts. When the user requests for the file upload, the deduplication phase matches the first part of the hash code with the first level indexing. If it matches then the comparison will take place for the second level of indexing. It the hash code doesn't match then the block is considered as the new one. After the first level of indexing and hash code match only then the second level of indexing is matched. If the second level also matches then the third level of the indexing is matched. If it doesn't matched then the block is considered as the new one. If the third level of the indexing matches with that of the third part of the hash code then the file is considered to be already uploaded and then the instance of the block is been increased in the database. The multi-level indexing is useful as it saves a lot of execution time while matching the hash code. If the multi-level indexing is not employed then the whole 16 bits of the hash code need to be matched in order to come to the conclusion whether the block exists or not. But with the multi-level indexing technique if suppose the first part of the hash code doesn't match then the conclusion is drawn that the block doesn't exist. This reduces the number of matches and the execution time to derive a conclusion.

If the user want to read/write the file uploaded by him, he can download the file in the file download process. The user will select the file which he want to download, get the logical block addressing from the database and based on it, the respective blocks will downloaded from Cloud. The downloaded blocks will then undergo block merging process. In this process all the downloaded chunks will be merged and the contents of the blocks will be written into one single file and then the file will be presented to the user. The logical address of every block will be searched using the Logical Block Addressing scheme and the blocks will be merged. The proof of storage phase of the life cycle lets the user to check for the integrity of the file uploaded on the server. The user can challenge the application with the particular chunk to be verified. Here the integrity of the blocks are been done by using the hash code. Every block is been assigned a 16 bit hash code which stands unique for every block. The hash code demanded by the user and that which exists on server is the deciding factor for the block integrity check.

Conclusion

In our proposed system the major requirement of the multi-user cloud storage system is been handled and the deduplication of the dynamic PoS model is been used. Dynamic scheme lets the user to make changes to the file and upload it again. With the deduplication scheme, only that block will be uploaded on server which is unique. The whole file will be searched for the unique block chunk and only that chunk will be uploaded leaving the redundant portion of the file. While downloading the file, the various chunks of the original file will be merged together in a single file and then the user will be able to download the file which he had uploaded. The proof of ownership concept of the file is useful as the security of the file is a major concern in the untrusted storage spaces. The storage space has become the major concern for the users these days. The proof storage concept is helpful in letting the user check for the integrity of the file. The user need not worry about the file blocks and download the file for repeated checking. The hash code for every block lets the user get the satisfaction that the file which he has uploaded is still there on server and is not lost. The deduplication process completes the 5 phases of the life cycle of the file resulting in the efficient use of the storage space on cloud.

The values of the simulation are as follows

Table 1. Simulation Results

File Size (bytes)	First Upload time (ms)	Second Upload time (ms)
8	35	49
45	41	57
300	27	38
357	26	34
496	27	38

Above results can be visualized as

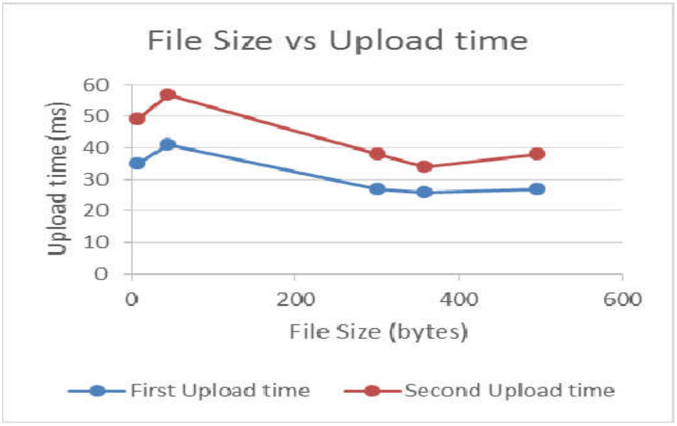

Figure 4. Simulation graph

We can see from the above graph, the file when uploaded by the user for the first time takes less time compared to the upload done for the same file second time. This is because if the same content file exists on the server, the check is been done for every block on each file upload. The deduplication check for every block of the file on server results for the time difference on the two uploads. In future a more efficient model can be implemented which will employee multi-level indexing scheme implemented with the more authenticated data structure. The combination of different data structure can also be considered for implementing the system. The new system can then be evaluated with the proposed system based on the different file sizes chosen for uploading.

REFERENCES

Ateniese, G., R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, 2011. “Remote data checking using provable data possession,” *ACM Transactions on Information System Security,* vol. 14, no. 1, pp. 1–34.

Azraoui, M., K. Elkhiyaoui, R. Molva, and M. ¨Onen, 2014. “StealthGuard: Proofs of Retrievability with Hidden Watchdogs,” in Proc. of ESORICS, pp. 239–256.

Bowers, K. D., A. Juels, and A. Oprea, 2009. “HAIL: A high-availability and integrity layer for cloud storage,” in Proc. of CCS, pp. 187–198.

Cash, D., A. Ku¨pc¸u¨, and D. Wichs, 2013. “Dynamic proofs of retrievability via oblivious RAM,” in Proc. of EUROCRYPT, pp. 279–295.

Chen, Bo, and Reza Curtmola. 2012. "Robust dynamic provable data possession." Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on. IEEE.

Chen, J., L. Zhang, K. He, R. Du, and L. Wang, 2016. “Message-locked proof of ownership and retrievability with remote reparing in cloud,” Security and Communication Networks.

Cui, Hui, et al. 2017. "Attribute-Based Storage Supporting Secure Deduplication of Encrypted Data in Cloud." IEEE Transactions on Big Data.

Di Pietro, R. and A. Sorniotti, 2012. “Boosting Ef ciency and Security in Proof of Ownership for Deduplication,” in Proc. of ASIACCS, pp. 81–90.

Dodis, Y., S. Vadhan, and D. Wichs, 2009. “Proofs of retrievability via hardness ampli cation,” in Proc. of TCC, pp. 109–127, 2009.

Douceur, J., A. Adya, W. Bolosky, P. Simon, and M. Theimer, 2002. “Reclaiming space from duplicate les in a serverless distributed le system,” in Proc. of ICDCS, pp. 617–624.

Douceur, John R., et al., 2002. "Reclaiming space from duplicate files in a serverless distributed file system." Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on. IEEE.

Jiang, Tao, Xiaofeng Chen, and Jianfeng Ma, 2016. "Public integrity auditing for shared dynamic cloud data with group user revocation." IEEE Transactions on Computers, 65.8, 2363-2373.

Juels, A. and B. S. Kaliski, 2007. “PORs: Proofs of retrievability for large les,” in Proc. of CCS, pp. 584–597.

Junxiang, Wang, and Liu Shengli, 2012. "Dynamic provable data possession with batch-update verifiability." Intelligent Control, Automatic Detection and High-end Equipment (ICADE), 2012 IEEE International Conference on. IEEE.

Kaaniche, Nesrine, and Maryline Laurent, 2015. "SHoPS: Set Homomorphic Proof of Data Possession Scheme in Cloud Storage Applications." Services (SERVICES), 2015 IEEE World Congress on. IEEE.

Lee, Narn-Yih, and Yun-Kuan Chang, 2011."Hybrid provable data possession at untrusted stores in cloud computing." Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on. IEEE.

Liu, Dongxi, and John Zic, 2015. "Proofs of Encrypted Data Retrievability with Probabilistic and Homomorphic Message Authenticators." Trustcom/BigDataSE/ISPA, 2015 IEEE. Vol. 1. IEEE.

Liu, Feifei, Dawu Gu, and Haining Lu, 2011. "An improved dynamic provable data possession model." Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on. IEEE.

Motegaonkar, Sonali B., and Chaitanya S. Kulkarni, 2016. "To develop secure deduplication of data using hybrid cloud methodology." Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on. IEEE.

Ren, Z., L. Wang, Q. Wang, and M. Xu, 2015. “Dynamic Proofs of Retrievability for Coded Cloud Storage Systems,” IEEE Transactions on Services Computing, vol. PP, no. 99, pp. 1–1.

Ren, Zhengwei, et al. 2015. "Dynamic proofs of retrievability for coded cloud storage systems." IEEE Transactions on Services Computing.

Shacham, H. and B. Waters, 2008. “Compact proofs of retrievability,” in Proc. of ASIACRYPT, pp. 90–107.

Stefanov, E., M. van Dijk, A. Juels, and A. Oprea, 2012. “Iris: A scalable cloud le system with ef cient integrity checks,” in Proc. of ACSAC, pp. 229–238.

Wang, C.,Q. Wang, K. Ren,and W. Lou, 2010. “Privacy-preserving public auditing for data storage security in cloud computing,” in Proc. of INFOCOM, pp. 1–9.

Xiao, Zhifeng, and Yang Xiao, 2013. "Security and privacy in cloud computing." IEEE Communications Surveys & Tutorials, 15.2, 843-859.

Xu, J. and E.C. Chang, 2012. “Towards ef cient proofs of retrievability,” in Proc. of ASIACCS, pp. 79–80.

Xu, Wei, Dan Feng, and Jingning Liu, 2012. "Public verifiable proof of storage protocol from lattice assumption." Intelligent Control, Automatic Detection and High-End Equipment (ICADE), 2012 IEEE International Conference on. IEEE.

Zheng, Yifeng, et al., 2016. "Toward Encrypted Cloud Media Center With Secure Deduplication." IEEE Transactions on Multimedia.

Zhu, Y., H. Hu, G.J. Ahn, and M. Yu, 2012. “Cooperative provable data possession for integrity veri cation in multicloud storage,” IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 12, pp. 2231–2244.

Zhu, Yan, et al. 2012. "Cooperative provable data possession for integrity verification in multicloud storage." IEEE transactions on parallel and distributed systems 23.12, 2231-2244.

****