



Data Security in Cloud by Using Blowfish Algorithm

ABDULLAH TH ABDALSATIR¹, MOHAMMED FAROOQ HAMDI², ALI NOORI KAREEM³

¹Research Scholar, Dept of Computer System and Engineering, SHIATS, Allahabad, India,
E-mail: Abdullah.thaier@yahoo.com.

²Research Scholar, Dept of Computer System and Network, SHIATS, Allahabad, India,
E-mail: mohamedfhazal@ymail.com.

³Asst Prof, Dept of computer Engineering Techniques, AL-Yarmouk University College, Baghdad-Iraq,
E-mail: Eng.ali.n.k@gmail.com.

Abstract: Cloud computing has great potential of providing robust computational power to the society at reduced cost. It enables customers with limited computational resources to outsource their large computation workloads to the cloud, and economically enjoy the massive computational power, bandwidth, storage, and even appropriate software that can be shared in a pay-per-use manner. Storing data in a third party's cloud system causes serious concern over data confidentiality. General encryption schemes protect data confidentiality, but also limit the functionality of the storage system because a few operations are supported over encrypted data. Constructing a secure storage system that supports multiple functions is challenging when the storage system is distributed and has no central authority. We propose a threshold proxy re-encryption scheme and integrate it with a decentralized erasure code such that a secure distributed storage system is formulated. The distributed storage system not only supports secure and robust data storage and retrieval, but also lets a user forward his data in the storage servers to another user without retrieving the data back. The main technical contribution is that the proxy re-encryption scheme supports encoding operations over encrypted messages as well as forwarding operations over encoded and encrypted messages. Our method fully integrates encrypting, encoding, and forwarding. We analyze and suggest suitable parameters for the number of copies of a message dispatched to storage servers and the number of storage servers queried by a key server.

Keywords: Threshold Proxy Re-encryption, Cryptography, Secure Storage System, Cloud Storage system, Straight Forward Integration.

I. INTRODUCTION

Cloud Computing provides convenient on-demand network access to a shared pool of configurable computing resources that can be rapidly deployed with great efficiency and minimal management overhead. One fundamental advantage of the cloud paradigm is computation outsourcing, where the computational power of cloud customers is no longer limited by their resource-constrained devices. By outsourcing the workloads into the cloud, customers could enjoy the literally unlimited computing resources in a pay-per-use manner without committing any large capital outlays in the purchase of hardware and software and/or the operational overhead there. Despite the tremendous benefits, outsourcing computation to the commercial public cloud is also depriving customer's direct control over the systems that consume and produce their data during the computation, which inevitably brings in new security concerns and challenges towards this promising computing model. High-speed networks and ubiquitous Internet access become available in recent years, many services are provided on the Internet such that users can use them from anywhere at any time. For example, the email service is

Probably the most popular one. Cloud computing is a concept that treats the resources on the Internet as a unified entity, a cloud. Users just use services without being concerned about how computation is done and storage is managed.

In Existing System we use a straightforward integration method. In straightforward integration method Storing data in a third party's cloud system causes serious concern on data confidentiality. In order to provide strong confidentiality for messages in storage servers, a user can encrypt messages by a cryptographic method before applying an erasure code method to encode and store messages. When he wants to use a message, he needs to retrieve the Codeword symbols from storage servers, decode them, and then decrypt them by using cryptographic keys. General encryption schemes protect data confidentiality, but also limit the functionality of the storage system because a few operations are supported over encrypted data. A decentralized architecture for storage systems offers good scalability, because a storage

server can join or leave without control of a central authority.

II. PROPOSED SYSTEM

In our proposed system we address the problem of forwarding data to another user by storage servers directly under the command of the data owner. We consider the system model that consists of distributed storage servers and key servers. Since storing cryptographic keys in a single device is risky, a user distributes his cryptographic key to key servers that shall perform cryptographic functions on behalf of the user. We propose a new threshold proxy re-encryption scheme. The encryption scheme supports encoding operations over encrypted messages and forwarding operations over encrypted and encoded messages.

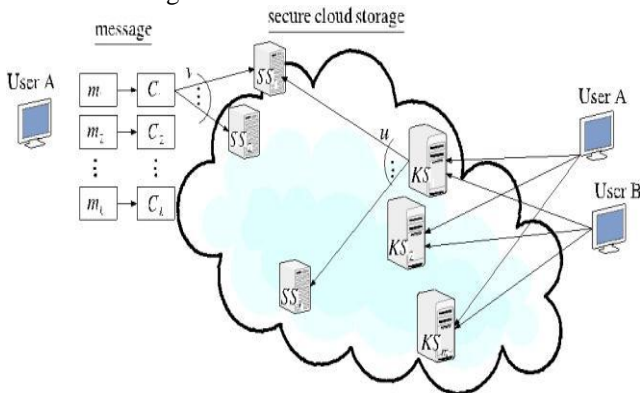


Fig.1. A General system model of our work.

III. A SECURE CLOUD STORAGE SYSTEM WITH SECURE FORWARDING

As described in Section III, there are four phases of our storage system.

A. Data Encryption Module

In cloud login module the user can login his own details. If the user cannot have the account for that cloud system first the user can register his details for using and entering into the cloud system. The Registration process details are Username, E-mail, password, confirm password, date of birth, gender and also the location. Then the user will go to open his account and view the code that can be generated from the cloud system. In folder creation process the cloud system may ask one question for that user. The user should answer the question and must remember that answer for further usage the server from the cloud can give the encrypted form of the uploading file.

B. Data storage

In Admin Module the admin can login to give his username and password. Then the server setup method can be opened. In server setup process the admin first set the remote servers Ip-address for send that Ip-address to the receiver. Then the server can skip the process to activate or Dis-activate the process. For activating the process the storage server can display the Ip-address. For Dis-activating the process the storage server cannot display the Ip-address. The activated Ip-addresses are stored in

available storage server. By clicking the available storage server button we can view the currently available Ip-addresses.

C. Data forwarding

In forward module first we can see the storage details for the uploaded files. When click the storage details option we can see the file name, question, answer, folder name, forward value (true or false), forward E-mail. If the forward column display the forwarded value is true the user cannot forward to another person. If the forward column display the forwarded value is false the user can forward the file into another person. In file forward processes contains the selected file name, E-mail address of the forwarder and enter the code to the forwarder. Now Then the current user has login to the cloud system and to check the receive details.

D. Data retrieval

In Download module contains the following details. There are username and file name. Now, the client has to download the file to download the file key. In file key downloading process the fields are username, filename, question, answer and the code. Now clicking the download option the client can view the encrypted key

E. Analysis

We analyze storage and computation complexities, correctness, and security of our cloud storage system in this section. Let the bit-length of an element in the group G_1 be l_1 and G_2 be l_2 . Let coefficients $g_{i,j}$ be randomly chosen from $\{0, 1\}^{l_3}$

1. Storage cost: To store a message of k blocks, a storage server SS_j stores a codeword symbol $(b, \alpha_j, \tau, \gamma_j)$ and the coefficient vector $(g_1, j, g_2, j, \dots, g_k, j)$. They are total of $(1+2l_1 + l_2 + kl_3)$ bits, where $\alpha_j, \tau \in G_1$ and $\gamma_j \in G_2$. The average cost for a message bit stored in a storage server is $(1+2l_1 + l_2 + kl_3) / kl_2$ bits, which is dominated by l_3/l_2 for a sufficiently large k

2. Computation cost: modular exponentiations in G_1 and G_2 , modular multiplications in G_1 and G_2 , and arithmetic operations over $GF(p)$. These operations are denoted as Pairing, Exp1, Exp2, Mult1, Mult2, and Fp, respectively. Computing an Fp takes much less time than computing a Mult1 or a Mult2. The time of computing an Exp1 is $1.5[\log p]$ times as much as the time of computing a Mult1, on average,

3. Correctness: There are two cases for correctness. The owner A correctly retrieves his message and user B correctly retrieves a message forwarded to him. The correctness of encryption and decryption for A can be seen in (1). The correctness of re-encryption and decryption for B can be seen in (2). As long as at least k storage servers are available, a user can retrieve data with an overwhelming probability.

4. Security: Security has to be compared and contrasted with other related concepts: Safety, continuity, reliability. The key difference between security and reliability is that security must take into account the actions of people attempting to cause destruction.

IV. SYSTEM MODEL

A. Decentralized code

A decentralized architecture for storage systems offers good scalability, because a storage server can join or leave without control of a central authority. We construct a secure cloud storage system that supports the function of secure data forwarding by using a threshold proxy re-encryption scheme. Our system is highly distributed where storage servers independently encode and forward messages and key servers independently perform partial decryption. Our method fully integrates encrypting, encoding, and forwarding. We analyze and Suggest suitable parameters for the tight integration of encoding, encryption, and forwarding makes the storage system efficiently meet the requirements of data robustness, data confidentiality, and data forwarding. Accomplishing the integration with consideration of a distributed structure is challenging .Our system meets the requirements that storage servers independently perform Encoding and re-encryption and key servers independently perform partial decryption

B. Integration

In an integration processes, the splinted message is joined into an m number of blocks, and stored into lager storage server. User A encrypts his message M is decomposed into k number of blocks m_1, m_2, \dots, m_k and which has an identifier ID. User A encrypts each block m_i into a cipher text C_i and sends it to v randomly chosen storage servers. Note that a storage server may receive fewer than k message blocks and we assume that all storage servers know the value k in advance. Then forward to user B. Data which is encrypted by using single key. This is produced by using hash key algorithm. In the data storage phase, user A encrypts his message M and dispatches it to storage servers. A message M is decomposed into k number of blocks m_1, m_2, \dots, m_k and which has an identifier ID. User A encrypts each block m_i into a cipher text C_i and sends it to v randomly chosen storage servers.

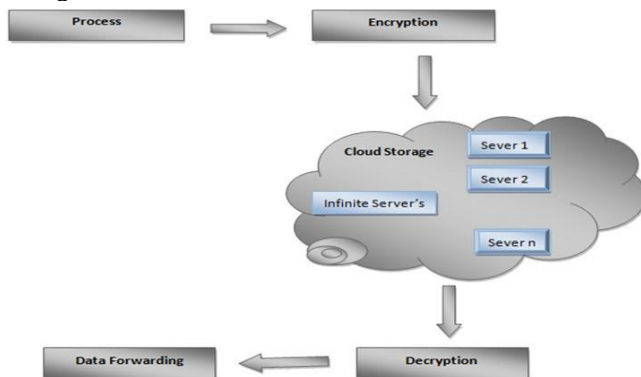


Fig.2. Overview architecture

C. Encryption

This is used to encrypt a plain text into a cipher text. Cipher text is produced along with a single key. The integrated data is encrypted with a single key using

random key generation method. Whenever users encrypt the text, each session time a new key is generated. Storing data in a third party cloud does not provide Confidentiality in cloud storage. Data confidentiality is provided by threshold proxy re-encryption scheme. This is used to generate the more than 10,000 key at the session time.

D. Data forwarding

In forward module first we can see the storage details for the uploaded files. When click the storage details option we can see the file name, question, answer, folder name, forward value (true or false), forward E-mail. If the forward column display the forwarded value is true the user cannot forward to another person. Another user can check his account properly and view the code forwarded from the previous user. Then the current user has login to the cloud system and to check the receive details. In receive details the forwarded file is present then the user will go to the download process

E. Login

Log in page make user to access an account in a cloud server. When user has an account in the cloud server for accessing data and provides other services. User can sign up the page directly else users needed to create new account using create account option.

F. Uploading File

In cloud login module the user can login his own details. If the user cannot have the account for that cloud system first the user can register his details for using and entering into the cloud system. The Registration process details are Username, E-mail, password, confirm password, date of birth, gender and also the location. After entering the registration process the details can be stored in database of the cloud system. Then the user has to login to give his corrected username and password the code has to be send his/her E-mail. Then the user will go to open his account and view the code that can be generated from the cloud system

G. Data Retrieval

Date retrieval is the final module of this project In Download module contains the following details. There are username and file name. First, the server process can be run which means the server can be connected with its particular client. Now, the client has to download the file to download the file key. In file key downloading process the fields are username, filename, question, answer and the code. Now clicking the download option the client can view the encrypted key.

V. ALORITHMS

A. Blowfish Algorithm

The data transformation process for Pocket Brief uses the Blowfish Algorithm for Encryption and Decryption, respectively. Blowfish is a symmetric block cipher that can be effectively used for encryption and safeguarding of data. It takes a variable-length key, from 32 bits to 448

bits, making it ideal for securing data.[1][4]Blowfish was designed in 1993 by Bruce Schneider as a fast, free alternative to existing encryption algorithms. Blowfish is unpatented and license-free, and is available free for all uses. Blowfish Algorithm is a Feistel Network, iterating a simple encryption function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits. The actual encryption of data is very efficient on large micro processors. Blowfish is a variable-length key block cipher. It is significantly faster than most encryption algorithms when implemented on 32-bitmicroprocessors with large data caches Feistel Networks A Feistel network is a general method[1][3] of transforming any function Feistel Network is given below:

- Split each block into halves
- Right half becomes new left half
- New right half is the final result when the left half is XOR'd with the result of
- Applying f to the right half and the key.
- Note that previous rounds can be derived even if the function f is not invertible.[4]

The Blowfish Algorithm:

Manipulates data in large blocks

- Has a 64-bit block size.
- Has a scalable key, from 32 bits to at least 256 bits.
- Uses simple operations that are efficient on microprocessor The Blowfish Algorithm:

B. Data Encryption Standard (DES)

The algorithm is designed to encipher and decipher blocks of data consisting of 64 bits under control of a 64-bit key. IP , then to a complex key-dependent computation and finally to a permutation which is the inverse of the initial permutation IP^{-1} . [9] The key-dependent computation can be simply defined in terms of a function f , called the cipher function, and a function KS , called the key schedule. A description of the computation is given first, along with details as to how the algorithm is used for encipherment.

VI. EXPERIMENTAL RESULTS

In the experiments, we first examine the time taken to create a log file and then measure the overhead in the system. With respect to time, the overhead can occur at three points: during the authentication, during encryption of a log record, and during the merging of the logs. Also, with respect to storage overhead, we notice that our architecture is very lightweight, in that the only data to be stored are given by the actual files and the associated logs.

A. Creation Time

In the first round of experiments, we are interested in finding out the time taken to create a log file when there are entities continuously accessing the data, causing continuous logging. Results are shown in Fig.3. It is not surprising to see that the time to create a log file increases linearly with the size of the log file. Specifically, the time to create a 100 Kb file is about 114.5 ms while the time to

create a 1 MB file averages at 731 ms. With this experiment as the baseline, one can decide the amount of time to be specified between dumps, keeping other variables like space constraints or network traffic in mind.

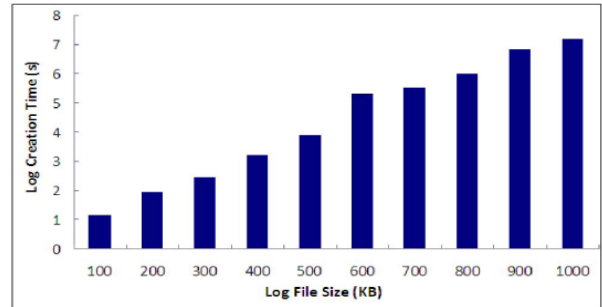


Fig.3. Time to create log files of different sizes.

B. Merging Time

To check if the log harmonizer can be a bottleneck, we measure the amount of time required to merge log files. In this experiment, we ensured that each of the log files had 10 to 25 percent of the records in common with one other. The exact number of records in common was random for each repetition of the experiment. The time was averaged over 10 repetitions. We tested the time to merge up to 70 log files of 100 KB, 300 KB, 500 KB, 700 KB, 900 KB, and 1 MB each. The results are shown in Fig.4. We can observe that the time increases almost linearly to the number of files and size of files, with the least time being taken for merging two 100 KB log files at 59 ms, while the time to merge 70 1 MB files was 2.35 minutes.

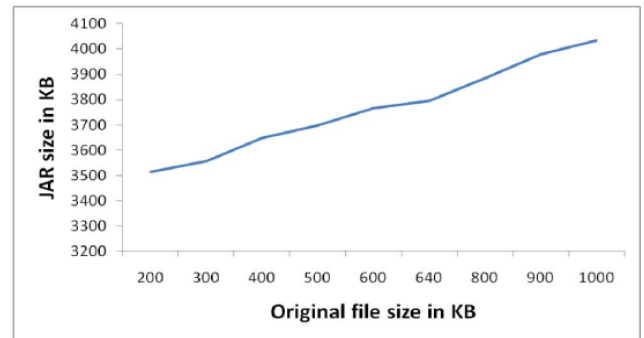


Fig.4. Size of the logger component.

VII. CONCLUSIONS

In this paper, we consider a cloud storage system consists of storage servers and key servers. We integrate a newly proposed threshold proxy re-encryption scheme and erasure codes over exponents. The threshold proxy re-encryption scheme supports encoding, forwarding, and partial decryption operations in a distributed way. To decrypt a message of k blocks that are encrypted and encoded to n codeword symbols, each key server only has to partially decrypt two codeword symbols in our system. By using the threshold proxy re-encryption scheme, we present a secure cloud storage system that provides secure data storage and secure data forwarding functionality in a decentralized structure. Moreover, each storage server independently performs encoding and re-encryption and each key server independently performs partial decryption.

Our storage servers act as storage nodes in a content addressable storage system for storing content addressable blocks. Further study on detailed cooperation is required.

VI. REFERENCES

- [1] J. Kubiawicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An Architecture for Global-Scale Persistent Storage," Proc. Ninth Int'l Conf. Archivers Support for Programming Languages and Operating Systems (ASPLOS), pp. 190-201, 2000.
- [2] P. Druschel and A. Rowstron, "PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility," Proc. Eighth Workshop Hot Topics in Operating System (HotOS VIII), pp. 75-80, 2001.
- [3] A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer, and R. Wattenhofer, "Farsite: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment," Proc. Fifth Symp. Operating System Design and Implementation (OSDI), pp. 1-14, 2002.
- [4] A. Haeberlen, A. Mislove, and P. Druschel, "Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures," Proc. Second Symp. Networked Systems Design and Implementation (NSDI), pp. 143-158, 2005.
- [5] Z. Wilcox-O'Hearn and B. Warner, "Tahoe: The Least-Authority Filesystem," Proc. Fourth ACM Int'l Workshop Storage Security and Survivability (StorageSS), pp. 21-26, 2008.
- [6] H.-Y. Lin and W.-G. Tzeng, "A Secure Decentralized Erasure Code for Distributed Network Storage," IEEE Trans. Parallel and Distributed Systems, vol. 21, no. 11, pp. 1586-1594, Nov. 2010.
- [7] D.R. Brownbridge, L.F. Marshall, and B. Randell, "The Newcastle Connection or Unixes of the World Unite!," Software Practice and Experience, vol. 12, no. 12, pp. 1147-1162, 1982.
- [8] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network File system," Proc. USENIX Assoc. Conf., 1985.
- [9] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage," Proc. Second USENIX Conf. File and Storage Technologies (FAST), pp. 29-42, 2003.

Author's Profile:



Abdullah Th Abdalsatir Has received his Bachelor degree in computer Engineering Techniques from Al Yarmouk University College -Iraq in 2009. He's obtained M-Tech in Computer System and Engineering. From SHIATS (Formerly Allahabad Agriculture Institute, Allahabad-India) in 2013. He worked one year in computers and networks and six month in IT security and Ethical hacking as well as he worked one year in embedded system .presently he is working as Assistant Lecturer at computer Engineering Techniques Department, AL-Yarmouk University College, Baghdad-Iraq.



Mohammed farooq hamdi, From Sam Higginbottom institute of agriculture, technology & sciences - deemed university.



Ali Noori Kareem Has received his Bachelor degree in computer Engineering Techniques from Al Yarmouk University College -Iraq in 2010. He's obtained M-Tech in Computer System and Network. From Kharkov National University of Radio and Electronics in 2013. He worked seven years in computers and networks. Presently he is working as Assistant Lecturer at computer Engineering Techniques Department, AL-Yarmouk University College, Baghdad-Iraq.