ABHINAV
NATIONAL MONTHLY REFEREED JOURNAL OF REASEARCH IN SCIENCE & TECHNOLOGY
www.abhinavjournal.com

A RELIABLE APPROACH TOWARDS SECURE DISTRIBUTED STORAGE, DYNAMIC FILE OPERATIONS AND SECURITY IN CLOUD COMPUTING

Christina Beemer[1], Saurabh Jain[2]. Jitendra Gosavi[3] and Kaustubh Bhole[4]

Student, AISSMS's IOIT, Pune, India
Email: [1]beemerchristina@gmail.com, [2] saurabhjain507@gmail.com, [3]jitugosavi@gmail.com, [4]kaustubhbhole0808@gmail.com

ABSTRACT

Cloud storage allows users to store data in a remote location thus decreasing the cost of hardware. The data is stored at remote location, thus, there is a security risk about the correctness of data. In order to address the data correctness problem we propose a system, in which the data to be stored is broken down into blocks and encrypted using the Blowfish algorithm. The proposed system does not have a third party to check the integrity; absence of TPA results in better security provided by effective data security algorithms by the user itself. The proposed system also supports dynamic file operations like update and delete.

Keywords: Cloud Computing, Cloud Service Provider, Blowfish Algorithm.

INTRODUCTION

Cloud computing is a new trend which provides users to store their data at remote location. End users access the cloud based applications through the internet. Nowadays business organizations are using cloud in order to store data, the main reason being the reduction in the hardware cost. Moving data to cloud provides a lot of flexibility to users since they do not have to care about the hardware management. Amazon, Google are some of the cloud service providers. However it eliminates the responsibility of local machines to maintain data, there is a chance of loss of data due external or internal attacks. External attacks are done by professional hackers for their own purposes. Internal attacks are the ones which occur due to the short comings of cloud service providers. Internal attacks may involve the deletion of data by the cloud service provider which is very rarely accessed by the user. Hence it is very important to ensure that there is no data loss. In this paper we provide a scheme to store data on distributed servers. In the proposed scheme the data to be stored is divided into blocks known as tokens. Then these blocks are encrypted on the client side using Blowfish algorithm and stored on the servers. This will ensure the security of the data being stored. The integrity of data is checked using mac authentication codes (MAC). The proposed system also provides dynamic operations such as update and delete.

PROBLEM STATEMENT

System Model

Fig.1 shows the cloud storage architecture

It mainly consists of two entities

1. User: It is the intended client who will store data to the cloud
2. Servers: These are the servers which provide storage as a service.

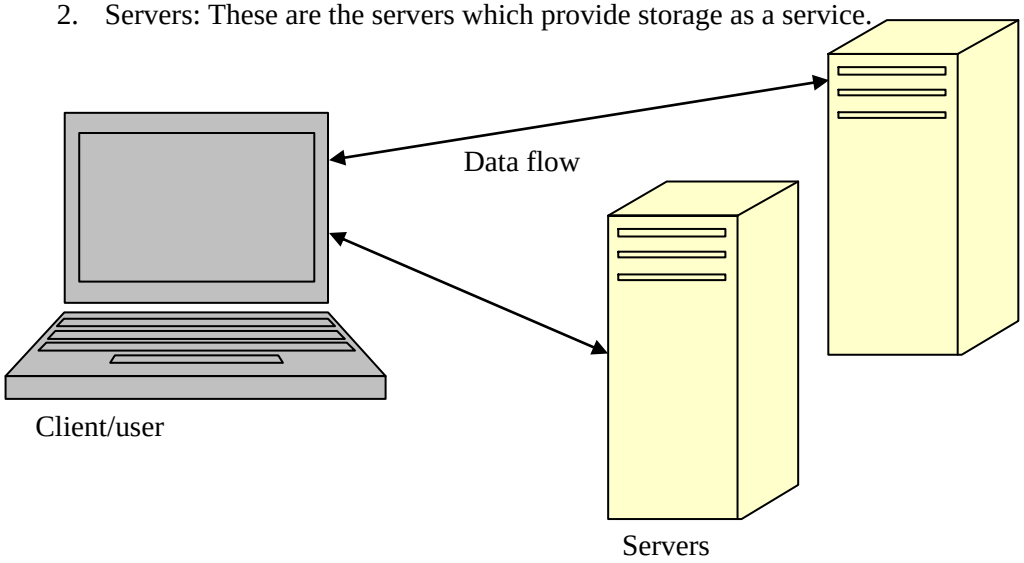

Fig 1. Cloud storage architecture

Design Goals

The proposed system aims to achieve

1. Data correctness: To ensure the user that the data is stored appropriately.
2. Dynamic data support: namely deletion and updation of data. 3] Store data on multiple servers.

Cloud Data Storage

File division

The file(F) selected to be stored on the cloud is divided into blocks(B1,B2…Bn) known as tokens. The number of blocks in which the file must be divided totally depends upon the user. The main advantage of dividing the file is that even if an attacker accesses a single block it will be of no use.

Encryption of file

The file (F) which is divided into blocks is encrypted using Blowfish algorithm before deploying it onto the cloud. Encryption is done at the client end.

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. 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. Although there is a complex initialization phase required before any encryption can take place, the actual encryption of data is very efficient on large microprocessors. Blowfish is a variable-length key block cipher. It is suitable for applications where the key does not change often, like a communications link or an automatic file encryptor. It is significantly faster than most encryption algorithms when implemented on 32-bit microprocessors with large data caches. A Feistel network is a general method of transforming any function (usually called an F-function) into a permutation. It was invented by Horst Feistel and has been used in many block cipher designs.

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.

For applications with a small key size, the trade-off between the complexity of a brute-force attack and a differential attack make a large number of iterations superfluous. Hence, it should be possible to reduce the number of iterations with no loss of security (beyond that of the reduced key size). Uses subkeys that are a one-way hash of the key. This allows the use of long passphrases for the key without compromising security. Has no linear structures that reduce the complexity of exhaustive search. Uses a design that is simple to understand. This facilitates analysis and increase the confidence in the algorithm. In practice, this means that the algorithm will be a Feistel iterated block cipher.

Encryption: Blowfish has 16 rounds. The input is a 64-bit data element, x. Divide x into two 32-bit halves: xL, xR.

Then, for i = 1 to 16:

xL = xL XOR Pi

xR = F(xL) XOR xR

Swap xL and xR

After the sixteenth round, swap xL and xR again to undo the last swap.

Then, xR = xR XOR P17 and xL = xL XOR P18.

Finally, recombine xL and xR to get the ciphertext.

Decryption is exactly the same as encryption, except that P1, P2,..., P18 are used in the reverse order. Implementations of Blowfish that require the fastest speeds should unroll the loop and ensure that all subkeys are stored in cache.

Generating the Subkeys:

The subkeys are calculated using the Blowfish algorithm:

1. Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of pi (less the initial 3): P1 = 0x243f6a88, P2 = 0x85a308d3, P3 = 0x13198a2e, P4 = 0x03707344, etc.
2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.)
3. Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).
4. Replace P1 and P2 with the output of step (3).
5. Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.
6. Replace P3 and P4 with the output of step (5).
7. Continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm. In total, 521 iterations are required to generate all required subkeys. Applications can store the subkeys rather than execute this derivation process multiple times.

Decryption

Once data is accessed by the user it is decrypted using the Blowfish algorithm. Decryption will take place at the client end. Since data will be in blocks they need to be merged into one which is the original file (F).

Dynamic Operations

So far, we assumed that F represents static or archived data. This model may fit some application scenarios, such as libraries and scientific datasets. However, in cloud data storage, there are many potential scenarios where data stored in the cloud is dynamic, like electronic documents, photos, or log files etc. Therefore, it is crucial to consider the dynamic case, where a user may wish to perform various block-level operations of update, delete and append to modify the data file while maintaining the storage correctness assurance. Since data do not reside at users' local site but at cloud service provider's address domain, supporting dynamic data operation can be quite challenging.

CONCLUSION

In this paper we see the various security issues in cloud storage and provide a flexible way in which the data can remain secure at the servers. We also provide a way through which dynamic operations can be done on a cloud based system.

REFERENCES

1. C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in Proc. of IWQoS'09, July 2009, pp.1–9

2. Juels and J. Burton S. Kaliski, "Pors: Proofs of retrievability for large files," in Proc. of CCS'07, Alexandria, VA, October 2007, pp.584–597.
3. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peter-son, and D. Song, "Provable data possession at untrusted stores," in Proc. of CCS'07, Alexandria, VA, October 2007, pp. 598–609.
4. M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Audit-ing to keep online storage services honest," in Proc. of HotOS'07. Berkeley, CA, USA: USENIX Association, 2007
5. M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008, http://eprint.iacr.org/.
6. G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proc. of SecureComm'08,2008, pp. 1–10.
7. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud com-puting," in Proc. of ESORICS'09, volume 5789 of LNCS. Springer-Verlag, Sep. 2009, pp. 355–370.
8. C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proc. of CCS'09, 2009, pp. 213–222
9. B. Schneier, Applied Cryptography, John Wiley & Sons, New York, 1994.
10. B. Schneier, Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish) Fast Software Encryption, Cambridge Security Workshop Proceedings (December 1993), Springer-Verlag, 1994, pp. 191-204.