A Distributed Key Based Security Framework for Private Clouds

Ali Shahbazi
Dept. of Computer Science
East Carolina University
Greenville, NC 27858 USA

Ali Karahroudy
Dept. of Computer Science
East Carolina University
Greenville, NC 27858 USA

Julian Brinkley
Dept. of Computer Science
East Carolina University
Greenville, NC 27858 USA

Nasseh Tabrizi
Dept. of Computer Science
East Carolina University
Greenville, NC 27858 USA

***Abstract*—Cloud computing in its various forms continues to grow in popularity as organizations of all sizes seek to capitalize on the cloud's scalability, externalization of infrastructure and administration and generally reduced application deployment costs. But while the attractiveness of these public cloud services is obvious, the ability to capitalize on these benefits is significantly limited for those organization requiring high levels of data security. It is often difficult if not impossible from a legal or regulatory perspective for government agencies or health services organizations for instance to use these cloud services given their many documented data security issues. As a middle ground between the benefits and security concerns of public clouds, hybrid clouds have emerged as an attractive alternative; limiting access, conceptually, to users within an organization or within a specific subset of users within an organization. Private clouds being significant options in hybrid clouds, however, are still susceptible to security vulnerabilities, a fact which points to the necessity of security frameworks capable of addressing these issues. In this paper we introduce the Treasure Island Security Framework (TISF), a conceptual security framework designed to specifically address the security needs of private clouds. We have based our framework on a Distributed Key and Sequentially Addressing Distributed file system (DKASA); itself borrowing heavily from the Google File System and Hadoop. Our approach utilizes a distributed key methodology combined with sequential chunk addressing and dynamic reconstruction of metadata to produce a more secure private cloud. The goal of this work is not to evaluate framework from an operational perspective but to instead provide the conceptual underpinning for the TISF. Experimental findings from our evaluation of the framework within a pilot project will be provided in a subsequent work.**

Keywords—private cloud security framework; distributed key; dynamic metadata reconstruction; cloud security

I. INTRODUCTION

Cloud computing, in its varying incarnations, continues to emerge as an attractive deployment option for enterprises and organizations seeking ways to reduce and better manage the costs associated with application deployment. Commercial cloud services allow organizations to consume computing resources in a manner similar to traditional utilities like electricity or water; paying for computing resources in a matter commensurate with their use. This Platform as a Service (PaaS) model additionally externalizes the costs associated with infrastructure and systems administration while providing a potentially more scalable and reliable deployment environment [1]. These significant benefits have created an impression in the minds of many consumers and organizational decision makers that "the cloud" is the answer to any number of software dilemmas.

But while the aforementioned benefits are undoubtedly attractive, these public cloud services are not without significant drawbacks within certain usage scenarios. In circumstances involving highly confidential, sensitive or secret data, security issues inherent to public clouds render their use inadvisable, impractical or even impossible depending upon legal and regulatory requirements. Government entities and health care organizations for instance often face legally mandated data security requirements that nearly all cloud services are incapable of satisfying due to a host of real and perceived security related issues [2-5]. The perception of the security and confidentiality vulnerabilities of public clouds has been reinforced by a number of data breaches reported in the media [6]. While governmental entities, regulatory bodies and medical organizations may benefit from the cloud given the large volumes of data generally involved with their respective activities, the risk of a single data breach often outweighs the potential benefits. Although some cloud providers continue to address these security and regulatory issues, as Microsoft has with the addition of Health Insurance Portability and Accountability Act (HIPAA) compliance features added to its Window's Azure cloud service [7], public clouds still possess too many security unknowns for many organizations.

As a result of these issues hybrid clouds have emerged as a middle ground between the aforementioned benefits of cloud computing and the identified security issues. Private clouds, significant aspects of private clouds, are developed and administered by an organization's internal IT department for the exclusive use by specific users or user groups within the organization [1, 8]. It is presumed that this greater degree of control *guarantees* an elimination of the security and regulatory issues posed by public clouds. But these private clouds may

also suffer from security issues, leading to a number of proposals designed to address these issues. In this paper we introduce the Treasure Island Security Framework (TISF) which builds upon existing thinking to provide a scalable security framework for private clouds.

II. Related Work

There is a significant body of work which documents the challenges and proposed solutions to the issue of cloud security; both public and private [9-12]. Many of these works take substantially different approaches to the issue of cloud security given the broad topic that is cloud computing. The approach taken within this works revolves primarily around an overlapping use of encryption, distributed key methodology, sequential chunk addressing and dynamic metadata reconstruction to improve system security.

Distributed key methodology is not a new concept having significant support with the literature albeit in significantly different conceptualizations and implementations [13-14]. A form of distributed key methodology serves as the backbone of the security effort proposed within this work whereby the key necessary to decrypt individual file chunks and reconstruct a stored file is distributed within our proposed system. This methodology stands in contrast to the some of the more common methods of cloud authentication such as those based primarily on password protection and Private Key Infrastructure (PKI). While these techniques are relatively easy to implement, they have a number of deficiencies which have been documented in the scientific literature and the media. Password protection for instance is dependent upon the user's ability to maintain confidential information against social engineering attacks wherein information enabling the reconstruction of one or more passwords may be divulged inadvertently by a user [15]. Password methodology is additionally troublesome given the average user's penchant for password reuse [16]. While the reuse of an existing password minimizes the cognitive load that memorizing a number of different passwords for different system creates, reuse effectively means that a password compromised for one system allows malicious users to access a host of other user accounts.

This is especially troublesome for email accounts which often serve as key link in the user verification process for many systems. Ticket based authentication using the Kerberos protocol is an improvement over pure password based protection however this methodology still possesses security risks [17]. A breach of a system's authentication server will result in the exposure of all user accounts due to the centralization of authentication management [18]. Public Key Infrastructure (PKI) addresses some of these issues through the use of digital certificates for entity identity verification [19]. This approach however also has a number of flaws [20]. The distributed key approach utilized within our proposed system addresses many of these issues by using a decentralized form of authentication that eliminates the single point of failure found in password protection scheme given the use of a segmented, dispersed key. This methodology is further bolstered by our use of a form of dynamic metadata reconstruction, which protects information about the stored data, and chunk encryption [21].

III. The Treasure Island Security Framework (TISF)

While the Google File System (GFS) [22-23] and the Hadoop Distributed File System (HDFS) [24-26], a GFS derivative, are commonly utilized with private clouds, we have proposed an alternative cloud architecture upon which the Treasure Island Security Framework is based. The Distributed Key and Sequentially Addressing Distributed file system (DKASA), which builds upon aspects of both GFS and HDFS, improves the security of data storage and file distribution in a private cloud primarily through the introduction of dynamic metadata reconstruction, sequential addressing and distributed key methodology.

A. Distributed Key and Sequentially Addressing File System

The Distributed Key and Sequentially Addressing Distributed file system (DKASA), as illustrated abstractly in Fig. 1, has a number of characteristics borrowed from the GFS including a single master configuration, use of fixed chunk sizes and chunk replication. We provide assumptions with respect to the configuration of DKASA within our proposed framework to contextual the security risk model discussed in Section 3 C.

1) Single Master Server

Both Single Master (SM) and Dual Main Server (DMS) configurations are possible within the DKASA file system although our proposal is based on the use of the former; mirroring the GFS. The DMS configuration, which involves the use of a management server and a file retrieval server, is potentially less robust than the SM configuration given its relatively poor performance under stress. It is likely, with moderate to high levels of network traffic and significant numbers of large files, that the retrieval server in the DMS configuration becomes a bottleneck for the entire system.

2) Fixed Chunk Size and Multiple Replicas

We anticipate the use of a fixed chunk size, which enables the use of the GFS mutation and lease method to reduce network traffic. Unlike the GFS which uses a fixed size of 64MB we have consciously chosen to leave the size of the chunk ambiguous as both large and small chunk sizes have advantages and disadvantages. A large chunk size for instance will reduce the number of chunk servers needed for each client while also reducing the client's interaction with the master for reading metadata and namespaces. This large chunk size however is also incompatible with smaller files. A smaller chunk size is compatible with smaller files however it may result in greater data fragmentation. In an actual implementation of our framework a system architect would determine the appropriate chunk size for the specific usage scenario.

Each chunk within the system will have a number of replicas (k) to ensure data availability; each replica chunk exists in isolation from the original. In the event that the original chunk is unavailable the replica system will retrieve a replica and use that data to reconstruct the original file.

3) Encryption

Each chunk's data is encrypted in the client machine and sent via secure communication using RSA [27] and Advanced

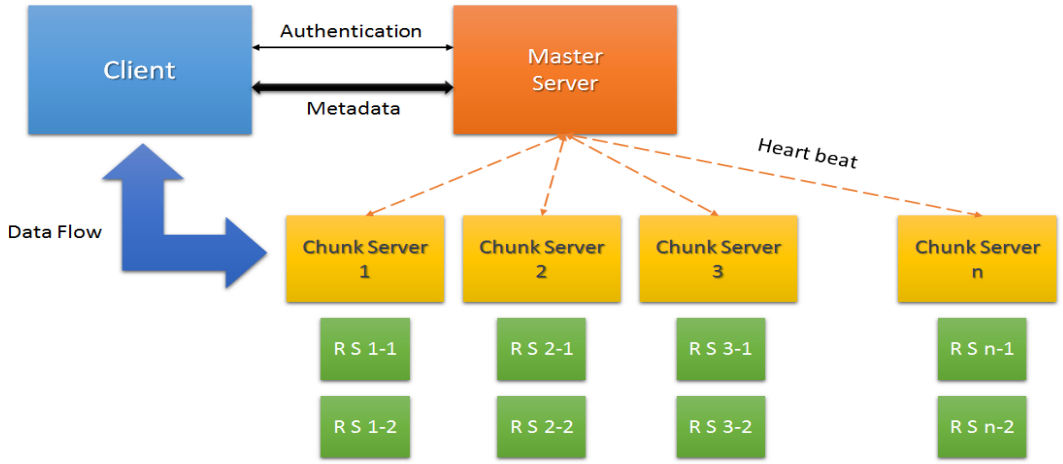

Fig. 1. Treasure Island Security Framework Abstract Model

Encryption Standard (AES) encryption [28]. The key differentiation between the method proposed in this work and traditional approaches is the use of a distributed key approach as opposed to the use of Public Key Infrastructure (PKI).

4) Distributed Key and Sequential Addressing

The use of a distributed key methodology has been driven primarily by two factors, a desire to increase security while introducing a level of flexibility whereby different security levels exist within the system. In terms of the latter, this type of security granularity facilitates varying degrees of file and user level security as opposed to a methodology within which the level of security within the system as whole is the only manipulable value.

Distributed key methodology as proposed within the DKASA system involves the distribution of a cryptographic key into four isolated parts. The first two parts of the key are stored in the master server and the client, the third part is stored in each chunk (n) and the final part of the key is stored in the previous chunk (n -1).

A file to be stored in the private cloud will be divided into a series of sequentially addressed chunks with distinct, appended headers and footers. The header of each encrypted chunk contains the following information:

- 128 bit local deciphering key
- 128 bit remote deciphering key
- The address of the next chunk
- 128 bit status code identifying the originality property of the chunk
- 1024 bits of audit data

This header data is used by the Cloud Management Server (CMS) and the user's client during the file retrieval process to locate file chunks, decipher them and rebuild the original file using the distributed key and sequential addressing approach. The full key necessary to decrypt each encrypted chunk is produced as a result of the concatenation of the parts of the key stored on the master server, the client, the current chunk server and the previous chunk server as illustrated in Fig. 2.

Upon successful completion of this process an interim copy of the file is available to the user on the client machine. After the user completes file manipulation (read, update, delete, etc) based on the mutation and lease method as employed within the GFS the chunks are stored on new servers. The complete file access algorithm is as follows:

procedure FileRetrieval()

1) Client machine sends authentication request to master server
2) Master server checks and approves client
3) Master server sends SID and service lists to client
4) Client asks master server for the address of the first chunk
5) Master server sends first chunk server address and first chunk (n-1) code part to client
6) Client send chunk server request, 128 bits of deciphering key, code part (n-1) and first chunk address
7) Based on internal algorithm client partition is reinterpreted to new deciphering code
8) Client reads and deciphers first chunk from the file server based on the reconstructed key
9) LOOP: Client refers to the next server based on the read data from the current server
10) Client reads and deciphers chunk from the file
11) server based on the reconstructed key
12) IF: File is complete
13) END LOOP
14) END IF
15) END LOOP:

end FileRetrieval

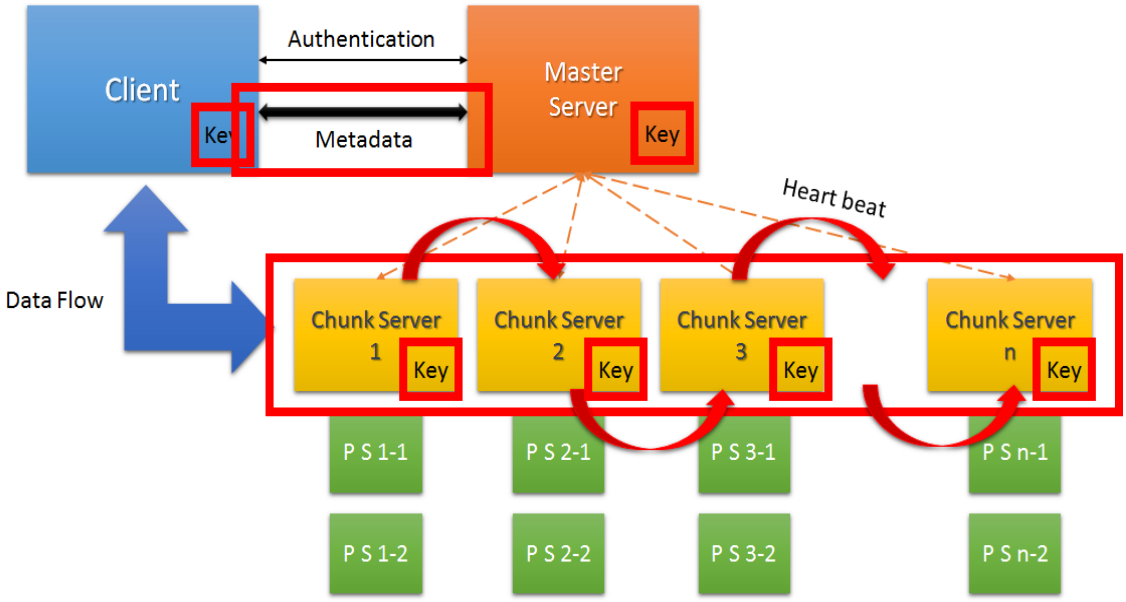

Fig. 2. TISF Sequential Addressing and Distributed Key Methodology

B. *Security Risk Model*

Our approach adds security beyond that found in security schemes using PKI and single password methodologies in that the likelihood of system compromise from a single attack is largely eliminated. We evaluate this claim using the previously outlined assumptions and the presumption of a single file broken down into many chunks (*N*), each possessing many replicas (*K*) within a distributed key architecture. We define the distributed key as

$$D_K = D_i + D_i\text{-}1 + D_{CM} + D_{MS} \quad (1)$$

All four isolated parts of the key are necessary to construct the full cryptographic key required to decipher each chunk. To evaluate the likelihood of comprise it is thus necessary to calculate the availability of each of the four key components with respect to their individual locations; the master server, the client, the original chunk and the previous chunk. Within the working system the master server is constantly operational therefore the availability of this component to an attacker is equal to 100% or

$$P_A(D_{MS}) = 1 \quad (2)$$

For the client machine the window for an attack is based on the total time of connection. For the purpose of this analysis we assume a client connection duration that is represented as *TCM*. For the final two key components, it is necessary for an attacker to successfully attack two different chunks, the current chunk and the previous chunk, to assemble all of the components necessary to reconstruct the full cryptographic key. The probability of a successful attack in this scenario is

$$C_n = \{(Ns - b)\ 2\text{-}2\} / \{Ns! / (Ns - 2)!\ 2!\} \quad (3)$$

Where the denominator is the number of all choices and the numerator is the likelihood that an attacker successfully selects the server containing the first chunk. Thus the chance of gaining access to all the necessary items for deciphering a chunk will be:

$$P_{FA} = \{(N_s - b)\ 2\text{-}2\} / \{N_s!/(N_s - 2)!2!\} \times \{T_{CM}/86400\} \quad (4)$$

The availability of a file for a user, in the event of a server failure or malware attack, depends upon the number of replica chunks that exist for each original. When there are $K < n$ replica chunks a high risk situation exists for the integrity of the user data in that file chunks exist without a backup. Where $K = n$ all chunks have a backup therefore a full copy of the entire file exists. The total number of full file backups may be determined by *(K/ (n+K))*. The chance of successful file retrieval after any malware attack server failure is thus:

$$AI = K\ P_p / (n+K) \quad (5)$$

where *Pp* equals availability of a server in the system, *K* the quantity of replica control chunks, and *n* the number of chunks per file.

IV. Conclusion And Future Work

In this paper we have introduced a security framework for private clouds called the Treasure Island Security Framework (TISF) which is based upon a Distributed Key and Sequentially Addressing Distributed file system. We have introduced DKASA and the methodology behind its proposed implementation while evaluating the security risks inherent in our approach. We believe that our proposed approach enhances both data availability and integrity while providing a higher degree of security and backup control at both the user and file level. Perhaps the most significant advantage of the DKASA cloud as proposed is the avoidance of the most common public cloud security and data availability issues; issues which have been chronicled exhaustively in both the press and the related scientific literature. Our subsequent work will seek to evaluate our claims within a pilot project which will be documented in a future paper.

REFERENCES

[1] R. Prodan and S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers", in 10th IEEE/ACM International Conference on Grid Computing, Alberta, Canada, 2009, pp. 17-25.

[2] K. Ren, C. Wang and Q. Wang, "Security Challenges for the Public Cloud," IEEE Internet Computing, vol. 16, pp. 69–73, Jan. 2012.

[3] R. Iglesias, R. Nicholls, A. Travis and W. Henderson, "Private Clouds with No Silver Lining: Legal Risk in Private Cloud Services," Digiworld Economic Journal, no. 85, pp. 125–140, 1st Q. 2012.

[4] L. Kaufman, "Data Security in the World of Cloud Computing," IEEE Security and Privacy Journal IEEE, vol. 7, pp. 61-64, Jul. 2009.

[5] S. Subashini and V. Kavitha, "A Survey on Security Issues in Service Delivery Models of Cloud Computing," Journal of Network and Computer Aplications, vol. 34, no. 1, pp. 1-11, Jan. 2011.

[6] J. Galante, O. Kharif and P. Alpeyev, "PlayStation security breach shows Amazon's cloud appeal for hackers," The Seattle Times, May, 16, 2011. [Online], Available: http://seattletimes.com/html/businesstechnology/2015071863_amazoncloudhackers17.html, [Accessed Jun. 29, 2013].

[7] B. T. Horowitz, "Microsoft Adds HIPAA Compliance Features to Azure for Cloud Health Data," eweek.com, para. 1, Jul. 27, 2012. [Online]. Available: http://www.eweek.com/c/a/Health-Care-IT/Microsoft-Adds-HIPAA-Compliance-in-Windows-Azure-for-Cloud-Health-Data-446671/. [Accessed Jun. 29, 2013]

[8] C. Babcock, "Time to Believe in 'Private Clouds'," Information Week, vol. 28, no. 12, pp. 27-30, Apr. 2009.

[9] K. W. Nafi, T. S. Kar, S. A. Hoque and M. M. A. Hashem, "A Newer User Authentication, File encryption and Distributed Server Based Cloud Computing security architecture," International Journal of Advanced Computer Science and Applications, vol. 3, no. 10, pp. 181-186, Oct. 2012.

[10] T. Wood, A. Gerber, K. K. Ramakrishnan, P. Shenoy and J. Van der Merwe, "The case for enterprise-ready virtual private clouds," in Proc. of the 2009 Conference on Hot Topics in Cloud Computing, San Diego, CA, 2009, pp. 4-9.

[11] D. W. Chadwick, M. Casenove and K. Siu, "My private cloud—granting federated access to cloud resources," Journal of Cloud Computing: Advances, Systems and Applications, vol. 2, no. 1, pp. 1-16, Feb. 2013.

[12] A. A. Karahroudy. "Security Analysis and Framework of Cloud Computing with Parity-Based Partially Distributed File System." M.S. thesis, East Carolina University, Greenville, USA, 2011.

[13] R. Geambasu, A. A. Levy, T. Kohno, A. Krishnamurthy and H. M. Levy, "Comet: An active distributed key –value store," in 9th USENIX Symposium on Operating Systems Design and Implementation - OSDI, British Columbia, Canada, 2010 pp. 323-336.

[14] J. Resch. "Authenticating Cloud Storage with Distributed Keys," presented at the Storage Developer Conference (SNIA), Santa Clara, CA, 2011.

[15] S. P. Maan and M. Sharma, "Social Engineering: A Partial Technical Attack," International Journal of Computer Science Issues (IJCSI), vol. 9, no. 2, pp. 557-559, Mar. 2012 ----MAYBE FIND A SOURCE THAT IS BETTER WRITTEN

[16] B. Ives, K. R. Walsh and H. Schneider, "The domino effect of password reuse," Communications of the ACM, vol. 47, no. 4, pp. 75-78, Apr.. 2004.

[17] S. P. Miller, B. C. Neuman, J. I. Schiller and J. H. Saltzer, "Kerberos authentication and authorizatio system" in In Project Ahtna Technical Plan. 1987 -----FIX FORMAT OF CITATION

[18] S. M. Bellovin, M. Merritt, "Limitation of the Kerberos authentication system," ACM SIGCOMM Computer Communication Review, vol. 20, no. 5, p. 119-132, Oct. 1990.

[19] D. Solo, R. Housley and W. Ford, "X. 509 public key infrastructure certificate and CRL profile," Jan. 1999 [Online]

[20] C. Ellison and B. Schneier, "Ten risks of PKI: What you're not being told about public key infrastructure," Computer Security Journal , vol. 16, no. 1, pp. 1-7, Nov. 2000.

[21] A. Waqar, A. Raza, H. Abbas and M. Khurram Khan, "A Framework for Preservation of Cloud Users' Data Privacy using Dynamic Reconstruction of Metadata," Journal of Network and Computer Applications, vol. 36, no. 1, pp. 235-248, Jan. 2013.

[22] S.E. Arnold, "MapReduce, Chubby and Hadoop", KM World, vol. 19, no. 10, pp. 1-18, Nov. 2010.

[23] S. Ghemawat, H. Gobioff and S. Leung, "The Google File System," in Proc. ACM SIGOPS Operating Systems Review, Bolton Landing, NY, USA, 2003, pp. 29-43.

[24] M. Bhandarkar, "MapReduce programming with Apache Hadoop", IEEE International Symposium on Parallel & Distributed Processing, Taipei, Taiwan, 2010, p. 1.

[25] K. Shvachko, Hairong Kuang, S. Radia and R. Chansler, "The Hadoop Distributed File System",in IEEE 26th Symposium on Mass Storage Systems and Technologies, Reno, USA, 2010, pp. 1-10.

[26] T. White, Hadoop: The Definitive Guide, Sebastopol, CA: O'Reilly Media, 2009.

[27] R. L. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Ditial Signature and Public-Key Cryptosystems," Laboratory for Computer Science, Massuchesetts Institute of Technology, Cambridge, MA, Nov. 1977.

[28] J. Daemen, V. Rijmen, "Announcing the Advanced Encryption Standard (AES)," Federal Information Processing Standards OPublication 197, Nov. 2001.