Data Confidentiality in Cloud Storage Protocol Based on Secret Sharing Scheme: A Brute Force Attack Evaluation

Alexandru Butoi, Mircea Moca, Nicolae Tomai

Data Confidentiality in Cloud Storage Protocol based on Secret Sharing Scheme: A brute force attack evaluation

Alexandru Butoi[1], Mircea Moca[1] and Nicolae Tomai[1]

Babeş-Bolyai University of Cluj-Napoca, Business Information Systems Department
{alex.butoi,mircea.moca,nicolae.tomai}@econ.ubbcluj.ro **

Abstract. Outsourcing a company's data storage and management system increases data security risk and generates a mistrust among adopters, that their data will remain secure and undisclosed. Due to limitations of the SLAs in this direction, the challenge is to build mechanisms that provides *data security by design* and assures data nondisclosure protection implicitly. In this paper we present an evaluation of the Data Confidentiality in Cloud Storage using a brute-force setup with the aim of studying its effectiveness. Through experimentation on the CloudSim environment we found that the probability of unauthorized data reconstruction can be exponentially decreased by using a sufficiently large number of chunks. Also, increasing the number of used storage volumes leads to a linear decrease of unauthorized data reconstruction.

Key words: cloud data, security, confidentiality, secret sharing scheme, brute force evaluation

1 Introduction

Certain studies like [1] show that data security still remains the main barrier for cloud service adoption in business. While it is true that nowadays public clouds represent affordable data storage alternatives, outsourcing a company's storage increases data security risk and emphasizes the problem of unauthorized data disclosures performed by malicious insiders. This indirectly leads to mistrust among possible adopters, as complex or unclear SLAs fail to reduce the risks of information disclosure after moving it into the cloud. The need of built-in interaction mechanisms for achieving *security by design* is mandatory while it can build trust between the two involved parties [2]. The challenge is to build mechanisms for public cloud infrastructures such that the confidentiality attribute is guaranteed through their own definition.

This paper is a continuation of the work conducted on Data Confidentiality in Cloud Storage Protocol (DCCSP) [3]. In this paper we evaluate the security

** This work was cofinanced from the European Social Fund through Sectoral Operational Programme Human Resources Development 2007-2013, project number POSDRU/159/1.5/S/134197 "Performance and excellence in doctoral and postdoctoral research in Romanian economics science domain".

strength of DCCSP using a brute force attack setup as a proxy. The implementation of DCCSP in a real system would help cloud adopters to assure a protection against unauthorized disclosure of sensitive data by malicious insiders or other attackers.

The remainder of this paper is structured as follows: Section 2 describes the addressed problem, section 3 overviews the DCCSP protocol, section 4 presents the conducted experiments and findings, while section 5 gives an overview of related or similar approaches. Section 6 concludes our work.

2 Background

In this section we give a short description of the main concepts used in our discussion. We consider two interacting parties: the **cloud consumer** and the **cloud provider**. The cloud consumer can be any user which contracts and use cloud storage resources to store her data. The cloud provider is an organization that owns, fully administers and rents cloud resources to the cloud consumer. Resources delivery to the cloud consumer is made according to a previously agreed SLA (cost, availability, security policies, usage etc.).

In our context the cloud consumer is willing to move data to a public storage cloud service. Due to cloud-specific resource elasticity achievable mainly through virtualization, the storage resources are managed and delivered in form of storage volumes. These may also be virtualized or not. We assume that the cloud consumer has access to *multiple storage volumes* from the cloud provider in order to save her data, according to the agreed SLA. The public cloud provider is a storage service provider which grants access for a cloud consumer to n distinct storage volumes$(V_1, V_2, ..., V_n)$. Each such volume has a certain amount of storage space that the cloud consumer uses for storing data.

In our setup the cloud consumer holds and fully controls a private cloud infrastructure (like own managed servers and storage). On the other side, the cloud consumer has only limited control over the public cloud resources regarding data administration. This aspect may be considered a source of security risk and mistrust[4].

3 DCCSP definition overview

In this section we give a brief description of the algorithms underlying the DCCSP protocol, detailed in [3]. This protocol insures that prior to the data storing into the public cloud, it will be secured by computing the secret sharing scheme on the user's private cloud. The secret is the file that holds sensitive information and the sharers are public cloud storage volumes. By this, only the cloud consumer knows the distribution pattern of the data into the public cloud. The protocol is transparent to any cloud infrastructure that fulfills the above assumption of instantiating multiple storage volumes, being implemented on the private cloud side as an interaction layer with the public cloud storage services.

Figure 1 depicts an overview of the protocol, which defines two main phases: the file send and file retrieve.

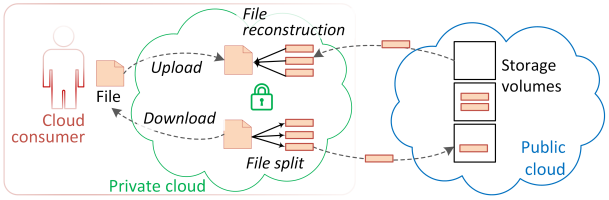

Fig. 1. Protocol overview.

The file send phase of the protocol takes place when the user wants to store or update a sensitive file in the public cloud storage. Before the upload of the file, the protocol splits the file into several smaller data chunks, denoted by $C_1, C_2, ..., C_m$. These chunks are stored into available storage volumes in public cloud in a way that the probability of reconstructing the initial file by a malicious insider or by an attacker is minimized. The protocol uses two algorithms to fulfill its purpose:

I.**A secret splitting algorithm** [3, p. 993] that implements the strategy of computing each chunk with the optimum size from the information security point of view. It uses the Shanon entropy and Kullbach-Leibler information for discrete distributions as metrics for minimizing the useful information content carried by each chunk relative to the whole information in the original file. The total amount of information in the file is quantified by the Shanon entropy which computes the average information contained in a sequence of data [5, p. 367-368], while the Kullbach-Leibler measures "the distance" between informational content of a new chunk ($I(c_i)$) and the entire file informational content($I(f)$) [5, 6]:

$$I(f) = -\sum_{i=1}^{s} P(x_i) ln P(x_i);\ I(f, c_i) = \sum_{i=1}^{s} P(x_i) ln(\frac{P(x_i)}{Q(x_i)}); \tag{1}$$

$P(x_i)$ = probability of byte x_i occurrence in file f, $P(x_i) = \frac{k_{x_i}}{s}$ where k_{x_i} equals with the number of x_i occurrences in file f, and s is the file size in number of bytes; ln is the natural logarithm function; $Q(x_i)$ = probability of byte x_i occurrence in chunk c_i when x_i is selected from the original file; $Q(x_i) = \frac{kc_{x_i}}{s}$, kc_{x_i} = number of occurrences for byte x_i in chunk c_i;

As plot (a) in figure 2 shows, each chunk size is selected with regards of having it's calculated K-L metric as higher as possible compared with the original file entropy $I(f)$: $I(f, c_i) >> I(f)$ [3]

II.**A chunk distribution algorithm**[3, p. 995] which implements the distribution of computed file chunks in the public cloud storage. These are distributed among available storage in a way that the probability of unauthorized reconstruction of the file is constantly minimized (eq. 2):

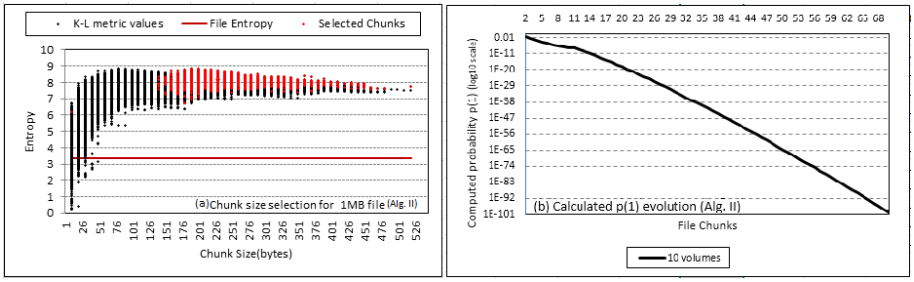

Fig. 2. Chunk optimal size computation[3]

$$p(1) = \frac{t!}{(t-1)!} P(1-P)^{t-1} = t \cdot P(1-P)^{t-1} \tag{2}$$

$p(1)$ is the probability of 1 successful reverse engineering modeled as Bernoulli trials; t is the number of trials; P is the event probability of choosing the right chunk in a trial calculated as a product of the two event probabilities: the probability of reconstructing the volume set used to store the file and the probability of reconstructing the right sequence of chunks in the original file, $P = \frac{1}{C_{n_v}^{k_v}} \cdot \frac{1}{A_{n_c}^{k_c}}$ [3, p.994-995].

The above strategy stores the file chunks in a manner that probability $p(1)$ of obtaining 1 success of file reconstruction to be as small as possible regardless the number of trials t. The plot (b) in figure 2 depicts the evolution of calculated $p(1)$ in the process as more chunks are being stored among 10 storage volumes. The $p(1)$ values have an exponential decrease and are represented on a log10 scale chart for easier reading purposes.

A **Chunk Distribution Dictionary** (CDD) is used for each data chunk upload operation to be recorded into it. Each record in this dictionary has three key fields: (1) The **index of the chunk in the original file**. This provides the exact position of the chunk within its belonging file; (2) The **virtual volume(s)** where the chunk is stored. (3) The **hash sum of the chunk** for integrity check purposes. The file splitting process takes place on the private cloud side, the resulting chunks being sent to the public cloud in a random order and associated CDD is stored on the private cloud infrastructure too. By this, only the cloud consumer which is the owner of data is aware of the distribution scheme.

The file retrieve phase takes place when a file is accessed or downloaded from the public cloud infrastructure. The operation can be triggered only through the private cloud which stores the corresponding dictionary. The CDD is used here to identify every data chunk of the file together with its associated storage volume. Similar to the send strategy, the chunks are randomly selected for retrieval. Finally, based on the chunk index, the original file is reconstructed.

4 Experiments and results

In this section we present a set of experiments performed to evaluate the effectiveness of our DCCSP protocol. We used the CloudSim[7] environment running

the experiments. As experimental data we used files exported from an e-learning platform, containing records with sensitive user account details.

4.1 Experimental setup

The strategy of storing chunks into the public cloud storage is based on constantly minimizing the probability of successful reconstruction of the original file, mainly by a malicious insider. In order to evaluate the security strength of our protocol we have implemented a brute force attack simulation model which tries to reverse engineer the process of reconstructing the original file without prior knowledge of the chunk distribution dictionary. The simulation is based on a brute force search strategy which searches the storage volumes for data chunks files while trying to reconstruct the original file. As an output for bench-marking, it counts the number of trials needed for a complete retrieval of the original file.

The main issue in mistrusting the data protection in public cloud system is mainly related to the problem of malicious insiders, super-users or administrators which can easily gain unauthorized access to already stored data.

Assumptions:

a the attacker is a malicious insider which knows the exact total set of volumes used in storing the target file;
b the attacker does not have knowledge about how chunks are distributed;
c the attacker does not have knowledge of the right chunk order in the file;
d the storage volumes are storing only the chunks of the targeted file.

The approach is an optimistic one which can be differentiated from a real world scenario by certain aspects like: the attacker does not have knowledge of the distribution records for chunk trial validation purpose after each chunk selection, the volumes may contain other data than targeted chunk files or it is likely for the attacker to not be able to know the exact volume set used in storing the targeted file. All the above are hardening the process of reverse engineering in a real world scenario.

Every file can be considered as a data chunk arrangement. The brute force strategy searches for a valid arrangement of data chunks. When the candidate arrangement is identical with the chunk arrangement of the original file, the search stops:

1. *while unconsidered storage volumes exists, randomly select one;*
2. *on selected volume, search for the next valid chunk: a selected chunk is considered valid if it's position in the candidate arrangement built so far, is the same as in the original file arrangement;*

We used two setups for a 1MB target file:

(I) we varied the number of used volumes, keeping the number of file chunks constant at 4361;

(II) we varied the number of chunks from 100 to 100000 keeping the used volumes number constant at 50.

The output of these simulations were the number of *Volume trials* and number of *Chunk trials*. The number of *Volume trials* represents the total number

of volume selections made in the brute force reverse engineering strategy. The number of *Chunk trials* is the total number of chunk selections made in the brute force search process.

4.2 Brute force trials

I. For the first setup, results are depicted in figure 3. We can see that volume trials have a linear evolution rapidly ascending from 44000 trials to 650000 trials as used volumes increase from 10 to 150. As for the chunk trials we can see that the figures are maintained at a high level, even when 10 volumes are used. Both charts indicate a low probability of unauthorized reconstruction of the file while the number of unsuccessful trials increase.

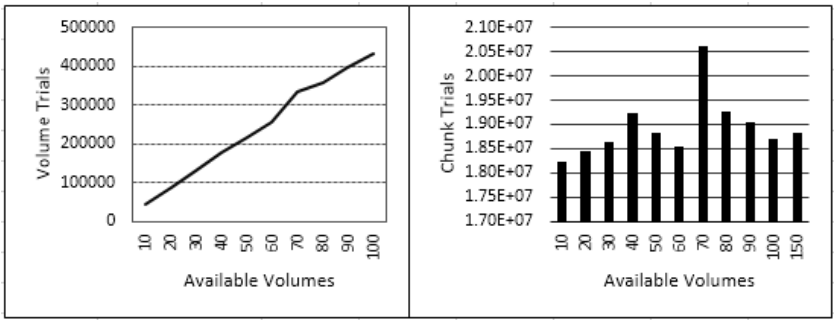

Fig. 3. Volume and chunk trials for 1 MB[4361 chunks] file when varying the number of used storage volumes.

II. In the second simulation setup, when the number of chunks is increased, figure 4 shows that both output indicators follow the same pattern of an exponential growth. Furthermore, we observe that when the number of chunks is

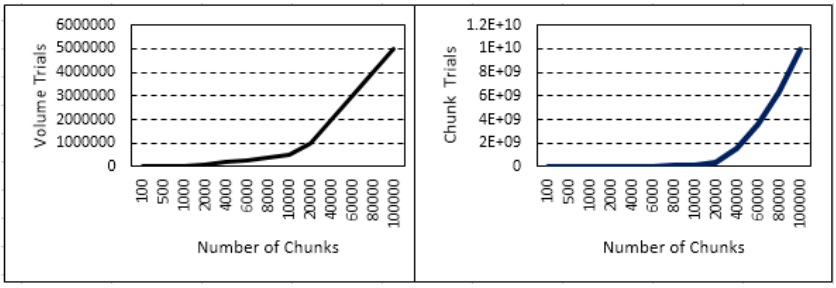

Fig. 4. Volume trials when varying the number of chunks.

above 10000, the number of trials rapidly increases towards infinity and rapidly lowering the probability of reverse engineering of the original file.

Splitting a file in a higher number of chunks seems to be a better strategy because if we increase the chunks number, we can exponentially decrease the chances of breaking the protocol, while increasing the used volumes number, we decrease the chances of a security breach only linearly.

5 Related work

The existing data protection oriented cloud systems like Depsky [8] and Belisarius [9] may be vulnerable to malicious insiders. These approaches use classic security concepts to protect data, like encryption which can be efficient but when comes to data processing, the access to data can be difficult. In contrast, DCCSP aims to minimize the issue of malicious insiders and protect data from unauthorized disclosure even against cloud administrators, without using a mandatory encryption step.

Other secret shared scheme approaches to data protection in cloud [10, 11] make use of multiple cloud providers, while DCCSP uses multiple virtual storage from one single provider.

Similar to our brute force attack evaluation strategy, Galbally et al. [12] employ brute force attack schemes to evaluate the strength of a HMM-based dynamic signature verification system. Also, in [13] the authors propose a hybrid RSA+Difie-Hellman symmetric encryption strategy which is mathematically evaluated against brute force attacks. [14] evaluates the encryption efficiency to digital images against brute force attacks, statistical and differential attacks while [15] evaluates simple block ciphers using different strategies for brute force search like genetic algorithms or particle swarm optimisation.

6 Conclusions

This paper presents a continuation of work presented in [3] introducing DCCSP protocol for assuring a lower as possible probability of unauthorized data disclosure in public cloud storage services. The current work addressed the problem of evaluating the strength of DCCSP using a brute force search strategy. We found that the probability of unauthorized data reconstruction can be exponentially decreased by using a sufficiently large number of chunks. Also, we observed that increasing the number of used storage volumes leads to a linear decrease of unauthorized data reconstruction. This leads to the idea that DCCSP can be cost efficient by using a constant number of storage volumes (fixed cost) and leveraging the number of chunks for obtaining high security levels. As future work we aim to development of a new flavor of the splitting algorithm operating at the word level for text files. Also, we plan to experiment with this protocol into a real cloud infrastructure.

References

1. Subashini, S., Kavitha, V.: A survey on security issues in service delivery models of cloud computing. Journal of Network and Computer Applications **34**(1) (2011) 1–11
2. Abbadi, I., Alawneh, M.: A framework for establishing trust in the cloud. Computers & Electrical Engineering (2012) 1073–1087
3. Butoi, A., Tomai, N.: Secret sharing scheme for data confidentiality preserving in a public-private hybrid cloud storage approach. In: Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on. (2014) 992–997
4. Rocha, F., Abreu, S., Correia, M.: The final frontier: Confidentiality and privacy in the cloud. Computer **44**(9) (2011) 44–50
5. Hsu, H.P.: Probability, random variables, and random processes. McGraw-Hill (1997)
6. Burnham, K.P., Anderson, D.R.: Model selection and multimodel inference: a practical information-theoretic approach. Springer (2002)
7. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience **41**(1) (2011) 23–50
8. Bessani, A., Correia, M., Quaresma, B., André, F., Sousa, P.: Depsky: dependable and secure storage in a cloud-of-clouds. In: Proceedings of the sixth conference on Computer systems, ACM (2011) 31–46
9. Padilha, R., Pedone, F.: Belisarius: Bft storage with confidentiality. In: 2011 10th IEEE International Symposium on Network Computing and Applications (NCA), IEEE (2011) 9–16
10. Fahl, S., Harbach, M., Muders, T., Smith, M.: Confidentiality as a service–usable security for the cloud. In: Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on, IEEE (2012) 153–162
11. Kajiura, Y., Kanai, A., Tanimoto, S., Sato, H.: A file-distribution approach to achieve high availability and confidentiality for data storage on multi-cloud. In: Computer Software and Applications Conference Workshops (COMPSACW) 2013 IEEE 37th Annual, IEEE (2013) 212–217
12. Galbally, J., Fierrez, J., Martinez-Diaz, M., Ortega-Garcia, J.: Evaluation of brute-force attack to dynamic signature verification using synthetic samples. In: Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on. (2009) 131–135
13. Mandal, B., Bhattacharyya, D., Bandyopadhyay, S.: Designing and performance analysis of a proposed symmetric cryptography algorithm. In: Communication Systems and Network Technologies (CSNT), 2013 International Conference on. (2013) 453–461
14. Ahmed, H., Kalash, H., Allah, O.: Encryption efficiency analysis and security evaluation of rc6 block cipher for digital images. In: Electrical Engineering, 2007. ICEE '07. International Conference on. (2007) 1–7
15. Nalini, N., Rao, G.R.: Attacks of simple block ciphers via efficient heuristics. Information Sciences **177**(12) (2007) 2553 – 2569