Deliverance from Trust through a Redundant Array of Independent Net-storages in Cloud Computing

Gansen Zhao[‡], Martin Gilje Jaatun[*],Athanasios Vasilakos[§]
Åsmund Ahlmann Nyre[*], Stian Alapnes[†], Qiang Yue[¶], Yong Tang[‡]

[‡]South China Normal University, China
Email: {gzhao, ytang}@scnu.edu.cn
[*]SINTEF ICT, Norway
Email: {martin.g.jaatun,asmund.a.nyre}@sintef.no
[§]National Technical University of Athens, Greece
Email: vasilako@ath.forthnet.gr
[†]Telenor Corporate Development, Norway
Email: stian.alapnes@telenor.com
[¶]GDEII, China
Email: yueqiang@gdeii.com.cn

***Abstract*—Cloud storage services are gaining more and more attention. Surveys suggest that the confidentiality issue is one of the major obstacles for users to use cloud storage services to keep sensitive data. This paper proposes to deploy a Redundant Array of Independent Net-storages (RAIN) for confidentiality control in Cloud Computing. The RAIN approach splits data into segments and distributes segments onto multiple storage providers, without having to trust each provider. By keeping the distribution of segments and the relationships between the distributed segments private, the original data cannot be re-assembled. When the segments are small enough, each segment discloses no meaningful information to others. Hence RAIN is able to ensure the confidentiality of data stored on clouds. A formal model of the proposed approach has been developed to articulate the process. Security analysis has been performed, indicating that the proposed approach can implement confidentiality protection without the need of encrypting the data.**

I. Introduction

Security concerns are frequently cited [1], [2] as one of the major obstacles to cloud computing adoption. In a traditional outsourcing scenario, technical and organizational security mechanisms contribute to protect a customer's data, but the most important factor is that the customer establishes a trust relationship with the provider. This implies that the customer acknowledges that if the provider is evil, the customer's data may be used improperly [3].

Cloud Computing can be thought of as outsourcing taken to the extreme, where both storage and processing is handled by one or more external providers, and where the provider(s) may be in a different jurisdiction than the customer. Not knowing where your data is physically located may be uncomfortable to the customer, and personal data may even be illegal to export from some jurisdictions [4]. Settling disputes is more challenging when the provider may be on a different continent, which is all the more reason to limit the degree to which the customer has to trust the provider. This is the "need to know" principle in a nutshell - if the provider does not need to read the information, why should it be allowed to? [5]

In this paper, we continue on a path toward a Cloud Computing scenario where the dependency on *trust* will be reduced by ensuring that each actor gets access to sufficiently small units of data so as to minimize confidentiality concerns. Thus, our approach is the opposite of the aggregation problem in database security since we *de-aggregate* the sensitive data.

The remainder of the paper is structured as follows: In Section II we outline the background for our contribution, and in Section III we sketch our solution. We present a formal model in Section IV, and provide a security analysis in Section V.We discuss our contribution in Section VI, outline further work in Section VII, and offer our conclusions in Section VIII.

II. Background

Cloud computing provides on-demand services delivered via the Internet, and has many positive characteristics such as convenience, rapid deployment, cost-efficiency, and so on. However, we have shown [5] that such off-premise services cause clients to be troubled by a number of common security concerns:

- Data Availability
- Data Confidentiality
- Data Integrity

In previous work [1], we identified five deployment models of cloud services designed to ease users' security concerns:

- **The Separation Model** separates storage of data from processing of data, at different providers.
- **The Availability Model** ensures that there are at least two providers for each of the data storage and processing

978-1-4244-9920-5/11/$26.00 ©2011 IEEE

tasks, and defines a replication service to ensure that the data stored at the various storage providers remains consistent at all times.
- **The Migration Model** defines a cloud data migration service to migrate data from on storage provider to another.
- **The Tunnel Model** defines a data tunneling service between a data processing service and a data storage service, introducing a layer of separation where a data processing service is oblivious of the location (or even identity) of a data storage service.
- **The Cryptography Model** extends the tunnel model by encrypting the content to be sent to the storage provider, thus ensuring that the stored data is not intelligible to the storage provider.

We have shown that through duplication and separation of duty, we can alleviate availability and integrity concerns, and to some extent also confidentiality, by implementing encrypted storage. However, even with encrypted storage, we still have to trust the encryption provider with *all* our data.

The main motivation for confidentiality control in the cloud is currently various privacy-related legislation forbidding the export of sensitive data out of a given jurisdiction, e.g. the Privacy legislation in the EU [4]. The current solution to this problem has been to offer geolocalized cloud services, where a customer may request the cloud provider to ensure that the sensitive data is only stored and processed on systems that are physically located in a geographically defined area, e.g., within the borders of the European Union. However, since cloud service providers typically run global operations, even data that physically reside in one jurisdiction will in principle be accessible from anywhere in the world.

Although misappropriation of data by cloud providers have not been documented, Jensen et al. [6] show that current cloud implementations may be vulnerable to attack. Ristenpart et al. [7] demonstrate that even supposedly secret information such as where a given virtual machine is running may be inferred by an attacker, highlighting another attack path.

III. Approach

We have extended the deployment models [1] with a new concept where data is split up and kept by several independent (non-colluding) storage providers in a Redundant Array of Independent Net-storages (RAIN) [5], in such a manner that a single chunk does not compromise confidentiality. The data can then be stored using one or several cloud storage providers (duplicated according to the deployment models).

A. Using Botnets for Non-nefarious Purposes

We propose to organize the various elements in our distributed cloud architecture as a traditional multi-tier Command & Control (C&C) botnet, e.g. as described by Wang et al. [8].

We introduce a new type of cloud service provider which assumes the role of the botnet C&C node, and which is in charge of assembling the information and presenting it. This keeps all processing in the cloud, but leaves us with the problem that we have to trust this provider with our information. The resulting configuration is illustrated in Figure 1.

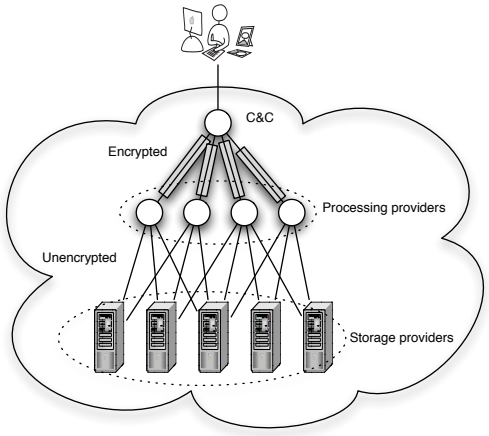


Fig. 1: Divide-and-conquer Cloud Security

The key property of the solution is that all the subnodes (cloud processing providers) and leaf nodes (cloud storage providers) only get to observe a small subset of a user's data, and that these nodes are prevented from associating a given piece of data with a specific user, or with other pieces of the same dataset. Ultimately, it will be like breaking open a large number of jigsaw-puzzles and distributing the pieces among storage providers – a single provider will not be able to determine where the pieces come from, or even if they are part of the same puzzle.

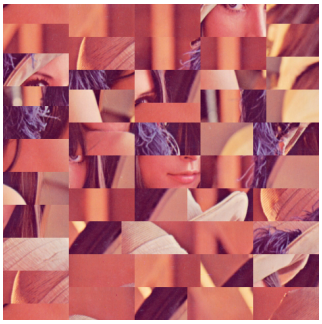
Referring to Figure 1 again, we would need to employ three different cloud cryptography providers for the tunnels from the C&C node to the subnodes. Of course, if the number of subnodes becomes large enough, it would be necessary to introduce additional tiers in the hierarchy, in order to minimize re-use of cryptography provider in any one node. We also assume that communication from the user to the C&C node is protected by, e.g., conventional TLS.

Note that we do not propose to make the cloud processing provider work with encrypted data; the confidentiality control is achieved through the de-aggregation of information, and hiding the relationships between the processing providers. Also note that assuming the volume of such "botnet computations" is large enough (i.e., many enough users employ this technique), it is also possible to re-use providers, since it will not be possible for a provider to relate two different processing tasks with each other.

B. Example

To illustrate the concept, we will in the following consider the storing of bitmap images in the cloud. Figure 2a shows a 480X480 image. The image is sliced into a 10X5 grid, of 48X98 pixels each. For our purposes, a single slice of the picture does not reveal much useful information to the observer, and this information can be stored unencrypted as

(a) Original Lena Image



(b) Randomly Segmented Lena Image

Fig. 2: Segmentation and Randomization

Fig. 3: An Individual Slice

long as it is not possible to combine it with the other slices. sample slice from the original image. Figure 3 does not give much information about the original image, not to mention any information on the girl Lena in the image.

The C&C node performs the slicing of the image, and randomly distributes the slices among (say) 10 subnodes. Each subnode then stores the slices independently using as many cloud storage providers as available (ideally one for each slice, but even for this small example we would probably be hard pressed to find 50 independent providers). To prevent observability, the subnodes may use an encrypted tunnel to transfer the data to the storage providers.

It is the responsibility of the C&C node to keep track of which subnode has received which slices, but also to record the location. When the image is to be retrieved, the user will instruct the C&C node to fetch all the slices.

Admittedly, this is a toy example, with all the real processing being performed by the C&C node – the real challenge comes when it is required to perform complicated processing on each subnode. This will also introduce the need for more sophisticated "slicing" of data.

Note that, without the proper knowledge of the distribution of the slices, it is very unlikely that the original image can be reconstructed. Given all the slices, it is still not easy to reconstruct the original image, if the number of slices is large enough. Figure 2b is an example of the reconstructed image without knowledge of the slices' order. The reconstructed image does not look much like the original image. If the image is sliced into even smaller slices, the reconstructed image would be even more different from the original image.

C. Criteria

Since we perform the slicing and distribution of data in order to achieve data confidentiality, it is important that the slicing and distribution process adheres to the following criteria:

- Data must be sliced into segments small enough such that each segment bears no meaningful information to malicious entities. With data sliced in this way, malicious entities may be able to access an individual data segment, but the access to the data segment should not compromise the confidentiality of the data as a whole.
- Data segments must be distributed in a random manner, such that it is not possible to establish the relationships between data segments without knowledge of the original data. The relationships between data segments are kept secret by the data owner.

IV. Formal Model

Let D be a piece of data to be split and stored on a cloud, $split$ be a function that splits D into a sequence H of smaller segments such that $H = (d_1, d_2, ..., d_n)$ where n is the number of segments D shall be split into.

H can be represented as $H =< D_s, R_s >$ where

- $D_s = \{d_i | i = 1, ..., n\}$
- $R_s = \{< d_i, d_{i+1} > | i = 1, ..., n-1\}$.

Note that D_s is the set of all segments D is split into. R_s is the set of relations between the segments in D_s, specifying the order of the segments.

Let M be the set of cloud providers that are providing cloud storage services. $\forall m_i \in M$, there is $D_i \subseteq D$ such that

$$\bigcup_{i=1}^{n} D_i = D \quad (1)$$

$$D_j \bigcap D_i = \oslash \quad (2)$$

Hence, $\forall d_i \in D$ and $d_i \in D_j$, d_i is stored on $m_j \in M$.

The criteria that must be imposed can then be formalized as follows.

- $\forall d_i \in D$, d_i does not disclose any information about D.
- $D_D = \{D_i | i = 1, \ldots, n\}$. D_D is a set representing a random partitioning of D such that

$$p(d_i \in D_j) \leq \frac{|D_D|}{D} \quad (3)$$

where $p(d_i \in D_j$ denote the possibility that $d_i \in D_j$ holds, $|D_D|$ and $|D|$ represent the number of elements in the set D_D and D respectively.

The above model specifies that, for a given piece of data D, it would be splitted into multiple segments, which comprise of the set D_s. The segments are then organized into groups, let say D_1, D_2, ..., D_k. The groups comprise the set D_D.

The sequential relations between the elements in D_s is represented by the set R_s, which specifies the predecessors and the successors of each elements in D_s.

To protect the confidentiality, data segments are managed by different cloud storage services, with D_i managed by cloud storage service m_i.

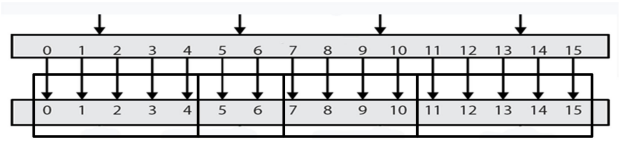


Fig. 4: Sequential Segmentation

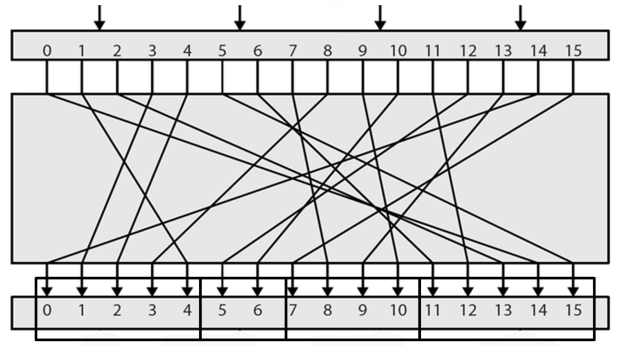


Fig. 5: Random Segmentation

A. Data Segmentation

The criteria stated in section III mandate that the data segmentation must make sure that each data segment bears no sensitive information of the original data.

The segmentation does not mandate the size of each each segment as the size has not direct relationship with the information of the segment bears. Commonly, the bigger a piece of segment, the more information it may bear. Thus a bigger segment is more likely to bear sensitive information.

The criteria do not impose any restriction on the way the original data is segmented. A most simple way to segment data is a sequential segmentation, by which the original data is treated as a binary stream and is divided into multiple substreams in their original order, as illustrated by Figure 4. An alternative way to segment the original data is to pick digits from random positions of the original data and put them into different segments. Figure 5 shows a case of data segmentation based on the random segmentation approach.

The segmentation process is in fact a permutation of the original data, followed by dividing of the permutated data.

The dividing of the data is to control the length of each segment to avoid having too much information in a single segment. The permutation of the original data is to diffuse the data in a way that the new presentation bears very limited information on the original data. Combining the division and permutation could greatly reduce the amount of information beared by a single segment.

B. Randomize Distribution

Once the original data is transformed into segments, the segments need to be stored by different cloud storage services. Segment distribution process is the process to decide each segment be assigned to a specific cloud storage service for storing. The distribution process needs to make sure the distribution is random to avoid tracking of the segments. Otherwise, segments can be identified by malicious users with limited cost.

An ideal distribution would be a complete random distribution of the segments over the available cloud storage services. A possible way to achieve random distribution is to randomly pick a cloud storage service for each segment. This can be implemented by using a sequence of random numbers. Each of the random number in the sequence can be used to specify which cloud storage service to store a data segment. For a sequence of random number, denoted by $R = r_1, r_2, \ldots, r_n$, the ith data segment is stored in a cloud storage service specified by r_i.

Another possible way to achieve random distribution is to use a secret number s, and the data segment d to generate a number n, where

$$n = hash(s, d)$$

By keeping s in private, the distribution of the data segments will be close to random distribution.

C. Data Re-assembling

The re-assembling of the original data requires two pieces of information.

1) The segment distribution information. The segment distribution information allows the picking of related segments from all the cloud storage services.
2) The order relations of the segments, R_S. With R_S, the picked data segments can be permutated back to the original order to construct the original data.

V. Security Analysis

Malicious users can either collect individual data segment or re-assemble the complete data to compromise the data confidentiality.

A. Compromization by a single data segment

Malicious users can randomly pick up individual data segments if they have excessive access privilege to the cloud storage services. Any individual data segment that is picked by a malicious user should disclose no information on the original data, according to the criteria of data segmentation. Therefore, it is not possible for malicious users to compromise the confidentiality by any single data segments.

B. Compromization by Re-assembling

Malicious users can also re-assemble the original data. The re-assembling consists of a few steps as follows.

1) Picking all related data segments.
2) Permuting the data segments.

Picking all the related data segments requires a malicious users to have excessive access privileges to access all the involved cloud storage services, and also requires that the malicious users to pick up all the data segments from all the involved cloud storage services.

Suppose that for each $m_i \in M$, where M is the set of all the cloud storage services, the set of all data segments stored by the cloud storage service m_i is N_i.

The number of data segments stored in M is $TotalSegs$ where

$$TotalSegs = \sum_{i-1}^{|M|} |N_i| \tag{4}$$

To be able to re-assemble the original data, a malicious user must be able to pick all the segments and permutate the segments into the right order.

If the malicious user does not know the number of segments the original data has been splitted into, the total number of possible re-assembled data is $NumOfAllReassembled$, where

$$NumOfAllReassembled = \sum_{i=1}^{U} P^i_{TotalSegs} \tag{5}$$

$$= \sum_{i=1}^{U} P^i_{\sum_{i-1}^{|M|} |N_i|} \tag{6}$$

Where R is the upper limit of the number of segments that a data is likely to be splitted into.

If the malicious user knows, s, the number of segments the original data has been splitted into, the total number of possible re-assembled data is $NumOfReassembled$, where

$$NumOfReassembled = \sum_{i=1}^{s} P^i_{TotalSegs} \tag{7}$$

$$= \sum_{i=1}^{s} P^i_{\sum_{i-1}^{|M|} |N_i|} \tag{8}$$

Both cases require a large amount of computation to brute-force search the complete space. Hence it is not trivial for a malicious user to compromise the data confidentiality by re-assembling the original data.

Though the computation complexity of the above brute-force searching is far less than those of cryptographic algorithms, the actual performance will be similar. For cryptographic algorithms depend on computation complexity for security, while the scheme proposed in the paper depends on both computation complexity, bandwidth cost and storage space for security. To brute-force search the complete space, the attacker will have to retrieve all data segments from all the cloud storage services. The number and the whole volume of the data segments is expected to be paramount. To permutate the data segments, the attacker will have to have extremely large storage capability. Hence the proposed scheme is strong in against re-assembling.

VI. Discussion

Cloud service providers have been identified as potential targets of attack simply because of the vast amounts of data they store on behalf of their multitude of customers. In this sense, it may be in the providers' best interest to "know less" - if even the provider cannot access the customers' data directly, there is little point in attacking them.

Strictly speaking, most users would probably be happy if it were possible to impose universal usage control [9] on data submitted to providers (a sort of "reverse DRM", where end-users get to control how multi-national corporations use their data), but despite Krautheim's efforts [10], we do not believe this will be a reality in the foreseeable future. Thus, it would seem that the easiest way to control what a provider does with your information is to hide it - either through encryption (as previously proposed for the storage providers) or through separation.

We have specified the use of multiple cryptography providers as well as storage and processing providers, and it is necessary to prevent "rotation" of providers (avoid using same provider to encrypt different versions of same data), as one might otherwise risk all providers having all data after a while. It is thus better to accept that each provider has partial, albeit updated, information. However, due to the botnet-like hierarchy, the providers do not know to whom the information belongs, and the value of the information is practically nil.

In a real-life setting, there will be cases where very small units of data carry a significant amount of sensitive information, such as blood type for patients. It will thus be imperative that not only shall it not be possible to match e.g. a blood type to an identity, but in storage it should also not be possible to determine what the data item refers to.

Since we place all our trust in the C&C node, it will remain as a "single point of trust" as long as it is realized as part of the cloud. It would have been desirable to strengthen this by ensuring that the C&C node provider only sees the information as we see it ourselves, preventing it from mining stored information. However, as long as the C&C node is required to keep track of all the data items (or slices, as in the example), there is nothing to prevent it from accessing this information as it pleases.

VII. Further work

As a first step, we will implement a prototype of our divide-and-conquer approach, specifically to gauge performance impacts on typical cloud applications. One particular challenge in this respect is to determine the optimal slicing strategy for arbitrary data. It is likely that a trade-off between security and efficiency will have to be made in order to capitalize on the advantages of the Cloud Computing paradigm. The prototype will be targeted toward a "sensitive but unclassified" application, representing a realistic use case.

VIII. Conclusion

We have described an idea on how to achieve confidentiality in the cloud through dividing data in sufficiently small chunks.

The main contribution of this paper is as follows. Firstly, we identify the need for protecting data confidentiality in clouds. Secondly, we propose an confidentiality protection approach based on RAIN, which provides data confidentiality without the need of encryption/decryption or trust. Thirdly, we formalize the proposed approach to articulate the methods. Lastly, we perform a security analysis, which indicates that the proposed approach is capable of providing data confidentiality.

ACKNOWLEDGEMENTS

This work has been supported by Telenor through the SINTEF-Telenor research agreement, China State Key Lab of Software Engineering [SKLSE2010-08-22], GD Higher Education Association Lab Management Committee [Grant No. 2010060], and China Canton-HK Research Fund [Grant No. TC10-BH07-1].

REFERENCES

[1] G. Zhao, C. Rong, M. G. Jaatun, and F. Sandnes, "Reference deployment models for eliminating user concerns on cloud security," *The Journal of Supercomputing*, pp. 1–16, 2010, 10.1007/s11227-010-0460-9. [Online]. Available: http://dx.doi.org/10.1007/s11227-010-0460-9

[2] Y. Chen, V. Paxson, and R. H. Katz, "Whats new about cloud computing security?" EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-5, Jan 2010. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-5.html

[3] Å. A. Nyre and M. G. Jaatun, "A Probabilistic Approach to Information Control," *Journal of Internet Technology*, vol. 11, no. 3, pp. 407–416, 2010.

[4] E. Parliament., "Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data." pp. 31–50, 1995.

[5] M. G. Jaatun, Å. A. Nyre, S. Alapnes, and G. Zhao, "A Farewell to Trust: An Approach to Confidentiality Control in the Cloud," in *Proceedings of the 2nd International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless Vitae Chennai 2011)*, 2011.

[6] M. Jensen, J. Schwenk, N. Gruschka, and L. L. Iacono, "On Technical Security Issues in Cloud Computing," *Cloud Computing, IEEE International Conference on*, vol. 0, pp. 109–116, 2009.

[7] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 199–212.

[8] P. Wang, L. Wu, B. Aslam, and C. C. Zou, "A systematic study on peer-to-peer botnets," in *ICCCN '09: Proceedings of the 2009 Proceedings of 18th International Conference on Computer Communications and Networks*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–8.

[9] J. Park and R. Sandhu, "The UCON_ABC usage control model," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 128–174, 2004.

[10] F. Krautheim, "Private virtual infrastructure for cloud computing," in *proceedings of the Workshop on Hot Topics in Cloud Computing, HotCloud*, 2009.