SHARING SECURE DOCUMENTS IN THE CLOUD
A Secure Layer for Google Docs

Lilian Adkinson-Orellana, Daniel A. Rodríguez-Silva, Francisco J. González-Castaño
GRADIANT, ETSI Telecomunicación, Campus, 36310 Vigo, Spain
{ladkinson, darguez}@gradiant.org, javier@det.uvigo.es

David González-Martínez
University of Vigo, ETSI Telecomunicación, Campus, 36310 Vigo, Spain
david.gonzalez@gti.uvigo.es

Keywords: Cloud security, SaaS privacy, Google Docs, Collaborative work, Document sharing, Document encryption.

Abstract: With the advent of Web 2.0, end users generate and share more and more content. One of the most representative services in this context is the collaborative edition of online documents. This service is commonly provided through Cloud Computing as Software as a Service. However, the Cloud paradigm still requires users to place their trust in Cloud providers with regard to privacy. This is the case of Google Docs, a very popular service without privacy support for the documents stored on its servers. In this paper we present and discuss a secure layer to guarantee privacy of shared Google Docs documents. To our knowledge, this is one of the first approaches to private shared edition with real Cloud tools.

1 INTRODUCTION

In Web 2.0, and social networks in particular, users are active agents who generate content collaboratively (T. O'Reilly, 2007). This is particularly interesting when several users from different locations work together and it has resulted in the development of new mechanisms for sharing documents through the Internet.

Furthermore, new paradigms have emerged following improvements in Internet technologies and computing resources. Among them, the Cloud Computing (or simply "Cloud") paradigm takes application complexity to the server side to simplify clients and client maintenance. Cloud solutions are intrinsically scalable and ubiquitous, and follow a pay-per-use approach at all levels (M. Armbrust et al., 2009). Of the three levels of Cloud Computing, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), we will focus on the third. We introduce a security layer at the SaaS (or application) level to manage shared documents privately. This level is highly familiar to end users because it provides final services such as GMail, Google Docs, Zoho and Microsoft Office Live.

One of the main barriers to the adoption of Cloud Computing is security (CSA, 2010): user data are stored on provider servers and there are no rigorous guarantees that this information will not be accessible to a third party. This can contravene legal requirements when the stored data are sensitive, as occurs in health care or banking environments. For this reason many users do not trust Cloud services, despite the fact that, from an operational perspective, there are optimal solutions for sharing their work.

Although the privacy problem has been adressed in virtualization platforms, Cloud privacy is completely open (T. Ristenpart et al., 2009). Indeed, in the case of Google Docs –one of the most important shared Cloud services– there is no guarantee that the documents will be safe at the server side, as there is evidence that Google has analyzed user information in the past, in particular to personalize advertisements (T. O'Reilly, 2004 and S. Lederer et al., 2004). In a previous paper (L. Adkinson-Orellana et al., 2010), we presented a solution based on the encryption of user data before storing it on Google Cloud servers, but we only dealt with the case of a single user. In this article we address secure on-line document *sharing*. To our knowledge, this is one of the first approaches to private shared edition with real Cloud tools.

The paper is organized as follows: Section 2 discusses the background in on-line document sharing and encryption. Section 3 presents technical insights into Google Docs document handling by external applications, with a particular emphasis on sharing options. Section 4 describes our solution to the problem of encrypted document sharing, focusing on Google Docs. Finally, section 5 concludes the paper and proposes future lines of work.

2 BACKGROUND

In this section we describe some on-line collaborative edition solutions related to our work, emphasizing characteristics such as security and sharing options.

From the point of view of SaaS, popular products such as Zoho, Microsoft Office Live and Google Docs itself support document sharing (H. Park et al., 2009). However, they do not offer document storage privacy. Zoho (www.zoho.com) expects to provide additional features for document encryption, although these have not yet been implemented. The same holds for Microsoft Office Live (workspace.officelive.com) and Google Docs (docs.google.com). The complete API of the latter simplifies the development of extensions such as those described below.

DocCloak (DocCloak, 2010) is a commercial product under development that encrypts Google Docs documents. It has three options with different features: *individuals* (free), *groups* and *enterprises*. In the simplest free version, only three users can share encrypted documents, but these cannot be modified once shared. Furthermore, the ciphering keys are kept on the user's computer, so SaaS access is not ubiquitous.

SeGoDoc (G. D'Angelo et al., 2010) is an extension for the Firefox browser based on the gDocsBar add-on (www.gdocsbar.com) to protect on-line document privacy and integrity against service providers. It works for Google Docs, but it has few data encryption options (it only includes the AES algorithm). In addition, this plugin is not directly integrated with the Google Docs interface and depends on third-party developers, meaning that it is unstable and not completely transparent for Google Docs users. Moreover, the password used to encrypt the documents is saved locally, at the client side, as in the previous case, and it currently does not support encrypted document sharing.

In Y. Huang and D. Evans, 2010, a "Secure Docs" Firefox add-on to enable private edition of Google Docs is presented. It is a proof-of-concept that allows users to access the Cloud editing service from Google securely (ensuring both data confidentiality and integrity) without the need for trusting the service provider. The content of the document that the user submits is incrementally encrypted using a password that is hidden from the service provider. However, as with the previous tools, it does not yet support collaborative edition.

We conclude that a transparent solution to share and edit secure documents in the Cloud is still missing. For this reason we have focused on improving such a popular Cloud application as Google Docs. We thus offer the possibility of working with personal or shared documents using a public Cloud service, preventing access to third parties.

3 HANDLING DOCUMENTS IN GOOGLE DOCS

3.1 Google Docs API

The Google Docs API (Google, 2010) is intended for programmers who wish to write client applications accessing Google Docs. The protocol used to interact with Google Docs documents is based on XML and HTTP and allows client applications to request a list of the users' documents, query their content, upload/download documents, modify sharing permissions, view the revision history, and organize documents into folders.

To request or upload documents, the client needs an authentication token that can be obtained in different ways depending on the type of client, according to the options available through Google Accounts. For example, to obtain the document list of a particular user, a GET request is sent to the URL:

https://docs.google.com/feeds/default/private/full

The result is an XML file containing a list of all available documents, where each entry represents a user document with a document identifier (docID).

As described, there is a mechanism for accessing Google Docs from an external application, but we need another mechanism to intercept information from and towards the Google Docs interface in order to encrypt/decrypt documents so that users can work with their interface as usual. This mechanism is described in the following section.

3.2 Listeners

To intercept and modify the content of the documents that are sent from the Google Docs interface to Cloud servers and vice versa, we have defined two *listeners*. These listeners should be embedded in the Internet browser (as a plugin or add-on) to facilitate the capturing of data.

Each listener will be in charge of certain HTTP requests:

- *http-on-modify-request*: output requests that are sent from the client
- *http-on-examine-response*: requests that are received from the server

By combining both types of requests, it is possible to intercept all GET/POST requests and responses. Two listeners may respectively capture incoming and outgoing information. It is necessary to filter the packets by IP address using the URI field included in each request.

Therefore, the following combinations help us achieve our target:

- http-on-modify-request + POST: request to save a document on the server
- http-on-modify-request + GET: request to retrieve a document from the server
- http-on-examine-response + POST: data returned after saving a document
- http-on-examine-response + GET: response from a document retrieval request

3.3 Sharing Documents in Google Docs

One of the most interesting features of Google Docs is the document sharing option. This allows a group of users to access a specific document when the owner configures its permissions properly, maximizing productivity.

The Google Docs API allows the management of shared documents through an Access Control List (ACL) feed. ACLs are simply basic lists that show who has access to a given resource. We can find the following roles in the ACL feed for a given document:

- *owner*: the owner of the document. This role can modify the ACL feed, delete or modify the document, etc.
- *writer*: a collaborator. This role can modify but not delete the document.
- *reader*: a viewer (equivalent to read-only access).

The API supports multiple levels of sharing permissions. It describes four levels (*user*, *group*, *domain* and *default*), which refer to the different scopes that the shared document may have. Documents containing the *user* attribute are private while those containing the *default* attribute are public.

As the ACL feed is the list that allows to share documents, the API supports several operations with the feed using the common HTTP operations GET, POST, PUT and DELETE through https://docs.google.com/feeds/default/private/full/docID/acl.

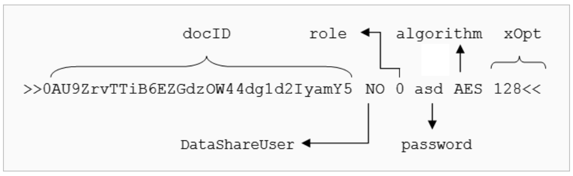
In the context of our work, the most relevant interactions with the ACL feed are related to retrieval, modification, update and removal of the permissions assigned to a document.

4 "SECURE GDOCS": SECURING SHARED AND PRIVATE CLOUD DOCUMENTS

4.1 Encrypting Documents

The Secure GDocs add-on for the Firefox browser allows the encryption of Google Docs documents, meaning that the encrypting process is transparent to users. As explained in our previous work (L. Adkinson-Orellana et al., 2010), Secure GDocs provides seven symmetric key algorithms for data ciphering: AES, DES, Triple DES, Blowfish, RC4, TEA, xxTEA (S. Rinne et al., 2007 and A. Nadeem and M.Y. Javed, 2005), each of which has different characteristics of speed and security. Thus, users can choose one or another depending on their particular needs.

In order to encrypt and decrypt the content of documents, certain information must be stored, such as the password used to cipher each document. For this purpose, a hidden index document containing this critical information is created through the Google Docs API. In this way, Cloud Computing ubiquity is respected, as the information required for deciphering the documents is preserved on-line. Figure 1 shows a line of an index document. This structure is repeated for every encrypted document.



```
>>0AU9ZrvTTiB6EZGdzOW44dg1d2IyamY5 NO 0 asd AES 128<<
```

Figure 1: Example of index structure.

To strengthen the security of the process, the content of the index document is ciphered with the AES algorithm using a 128-bit key, meaning that its sensitive information will not be exposed when stored on Google servers. Thus, an additional security layer for the ciphered documents is provided, since the password needed to access the information for deciphering the documents is different from the Google account password or the password used to encrypt each document.

Thanks to the listeners explained in the previous section, combined with the selected algorithm and a master password, the documents are decrypted and presented to the user as plain text when requested. In most cases, the document will be received in packets, meaning that it will have to be stored temporarily in a buffer until it is ready to be deciphered.

As expected, the size of the documents increases after encryption. Depending on the algorithm used, this size can double. The documents are thus compressed before they reach the servers as the current storage limit of Google Docs is 512 KB. Thus, we can assume that the ciphering process does not have a significant effect on typical editing in terms of size. Furthermore, there are even cases in which the compressed ciphered document is smaller than the original, unciphered version.

4.2 Encrypted Document Sharing

To address the encrypted document-sharing problem, we have extended our previous work and proposed a new solution based on the creation of a new index for each new document to be shared. This new shared index document must contain a unique line with data related to the encryption of the shared document, as it appears in the general index. This data will be created as a hidden file.

The names of the different indices follow a particular notation to identify them:

- plugin.doc: the general index. This contains data related to all ciphered documents, regardless of their status (shared or private).
- plugin_<docID>.doc: shared index for the shared document with the identifier docID. Its content is also included in the general index.

For the owner, sharing a ciphered document with other users involves the same process as in the case of unciphered documents; there is no difference in the use of the Google Docs interface.

When an owner shares a document, the request is intercepted by the listeners. If this is the first time the owner is sharing a ciphered document, a new popup will appear asking for a new password: the shared key. Once this value has been set, it will be written in the general index for future operations. Moreover, the first time a specific encrypted document is shared, a new associated shared index will be created using the Google Docs API.

The new shared index will contain the information associated with the encryption of the document, copied from the general index. The content of this index is also ciphered using AES with a 128-bit key, as occurs with the general index. The password to encrypt the index will be the shared key, which will preferably be different from the master key. Accordingly, when an owner shares a document he will simply have to give the shared key to the rest of the users (using a secure channel), meaning that the master key will remain private and safe.

It is important to highlight that the information relating to the shared and general index must be synchronized at all times. If any kind of incoherence occurs, the data from the owner's general index will be prioritized, and the shared index information will be updated accordingly.

4.2.1 Sharing the New Index

After the sharing of the ciphered document and the creation of its shared index, the index itself needs to be shared with the users who have access to the corresponding document. To share this index, a new petition will be made through the Google Docs API with all the necessary information. This petition will be repeated for every user who has access to the document, keeping the owner's general index with its critical information private.

As shown in Figure 2, the owner of the documents will have as many shared indices as documents that are being shared, each distinguishable by its name, which contains the document identifier. In the example, the owner has four encrypted documents, two of which (UD1 and UD2) are private; their ciphering information is stored in the hidden index file plugin.doc. The other two documents, with docID values SD1 and SD2, are shared with three different users. For each document a new hidden document is created, plugin_<docID>.doc, containing the same ciphering information as in the general index, plugin.doc. The general index is ciphered with the private master key (MK), while the shared indices use the same shared key (SK). If the owner decides to assign a different encryption key to SD1 or SD2, it would be transparent to the rest of the users, who would only need to know the

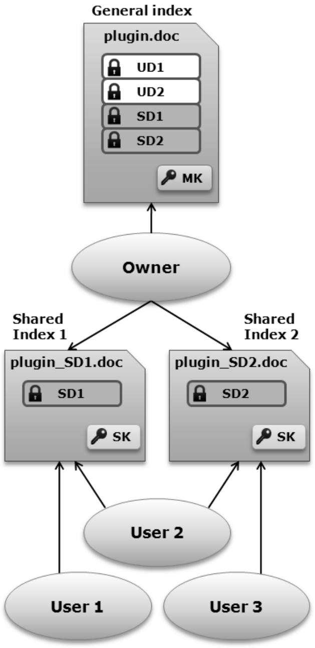


Figure 2: Private and shared indices of the owner of the documents. MK indicates master key; SK, shared key.

shared key to decipher their shared indices. Thus, the users are unaware of the options used to encrypt each document.

4.2.2 Modifying Document Permissions

The plugin extends some Google document-sharing features for encrypted documents, namely adding new permissions to a previously shared document and making a shared document private again for its owner.

Adding new permissions to a document. Providing access for additional users to an already shared document is similar to the first time the owner decides to share it. The main difference is that the shared index with the necessary data for decrypting the document already exists, meaning that this step is skipped.

When the owner of the document shares it through the Google Docs interface, the listeners will detect the petition; a new request will thus be issued to the Google Docs API, modifying the document's ACL feed and offering the new user access to the shared index content.

In this case, the shared key must be distributed using methods that are external to the add-on (i.e. a secure channel). A remarkable aspect in which our implementation outperforms others is that the number of users accessing a shared document is limited only by Google Docs, as the add-on does not impose any restrictions on the number of users accessing a shared document.

Removing Sharing Permissions. If the owner of a document decides to no longer share it with other users, it would suffice to remove their permissions in the corresponding shared index through the appropriate request to the API. The rest of the users would still have the original document on their list, but it would be undecipherable, since the data needed to decrypt it would be restricted.

When the owner removes the document permissions via the Google Docs interface, this action is detected by the listeners and a new request is sent to the API to remove public access from the shared index. If the number of users with access to the document drops to zero, the shared index will be deleted from the list of the owner's documents. However, the owner will still be able to work on the encrypted document because the corresponding ciphering information will still be stored in the general index.

5 CONCLUSIONS

Google Docs is one of the main tools for document edition in the Cloud. Moreover, the Google API simplifies its extension.

In this context, Cloud privacy can be guaranteed by ciphering the documents at the client side. A system with this functionality is conceptually simple. It simply requires the user to enter an initial password. In L. Adkinson-Orellana et al., 2010 we presented one of the first on-line encryption plugins based on the Google API.

In this paper, we go a step further and propose a solution that guarantees the privacy of shared documents. Document sharing is one of the most demanded SaaS features, with many applications in business, e-learning and professional Web 2.0 applications in general.

As future work we plan to provide new features such as document ownership changes and support for public or asymmetric cryptography, which would eliminate the need for password exchanging between users. In our current implementation, *real-time collaborative edition* with Google Docs encrypted documents works, but has not been fully tested. We are currently working on a new stable version including this feature.

ACKNOWLEDGEMENTS

This research has been supported by the ACETIC consortium and the SAFECLOUD grant (09TIC014CT), funded by Xunta de Galicia, Spain.

REFERENCES

Adkinson-Orellana, L., Rodríguez-Silva, D., Gil-Castiñeira, F. and Burguillo-Rial J. C. (2010). Privacy for google docs: Implementing a transparent encryption layer. In *Proceedings of the CloudViews 2010 - 2nd Cloud Computing International Conference*, pages 21–22, Porto, Portugal.

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I. and Zaharia, M. (2009). *Above the Clouds: A Berkeley View of Cloud Computing*, Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley, USA.

CSA (2010). *Domain 10: Guidance for Application Security V2.1*. Cloud Security Alliance Whitepaper. Retrieved December 16, 2010 from http://www.cloudsecurityalliance.org/guidance/csaguide-dom10-v2.10.pdf.

D'Angelo, G., Vitali, F. and Zacchiroli, S. (2010). Content cloaking: Preserving privacy with Google Docs and other web applications. In *2010 ACM Symposium on Applied Computing*, pages 22–26, Sierre, Switzerland.

DocCloak (2010). *DocCloak website*. Retrieved December 16, 2010 from http://www.gwebs.com/doccloak.html.

Google (2010). *Google Docs API website*. Retrieved December 16, 2010 from http://code.google.com/intl/es-ES/apis/documents/.

Huang Y. and Evans D. (2010). *Secure Google Docs Extension website*. Retrieved December 16, 2010 from http://www.mightbeevil.org/securedocs/.

Lederer, S., Hong, J. I., Dey, A. K. and Landay, J. A (2004). Personal Privacy through Understanding and Action: Five Pitfalls for Designers. *Personal and Ubiquitous Computing* 8/ 6, pp. 440-54.

Nadeem, A. and Javed, M.Y. (2005). A Performance Comparison of Data Encryption Algorithms. In *Information and Communication Technologies 2005, ICICT 2005*, Cairo, Egypt

O'Reilly, T. (2004). *The Fuss About Gmail and Privacy: Nine Reasons Why It's Bogus*. Retrieved December 16, 2010 from http://oreillynet.com/pub/wlg/4707.

O'Reilly, T. (2007). What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *Communications & Strategies*, 1st Quarter 2007, p. 17.

Park H., Lee K., Lee Y., Kim S. and Won D (2009). Security Analysis on the Online Office and Proposal of the Evaluation Criteria. *World Academy of Science, Engineering and Technology*. Issue 59, pp. 198-204.

Rinne, S., Eisenbarth, T. and Paar, C. (2007). Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers. In *ECRYPT Workshop SPEED - Software Performance Enhancement for Encryption and Decryption*, Amsterdam, Netherlands.

Ristenpart, T., Tromer. E., Shacham, H. and Savage S. (2009). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *CCS'09: Proceedings of, the 16th ACM Conf. on Computer and Comm. Security*, pp. 199–212.