Published in Proceedings of the International Conference on Internet of Things and Big Data 2016, ISBN: 978-989-758-183-0, pages 399-402
https://doi.org/10.5220/0005953903990402

I'll Trust You – For Now

Martin Gilje Jaatun
Department of Software Engineering, Safety and Security, SINTEF ICT, Trondheim, Norway
Martin.G.Jaatun@sintef.no

Keywords: Cloud, Security, Privacy, Trust

Abstract: The pervasiveness of cloud computing paired with big data analytics is fueling privacy fears among the more paranoid users. Cryptography-based solutions such as fully homomorphic encryption and secure multiparty computation are trying to address these fears, but still do not seem to be ready for prime time. This paper presents an alternative approach using encrypted cloud storage by one provider, supplemented by cloud processing of cleartext data on one or more different cloud providers.

1 INTRODUCTION

Cloud computing is everywhere, and more and more users are putting their data into the cloud, often without knowing it (Rong et al., 2013). With all this data in the hands of one of the big cloud providers, the emergence of big data analytics (Jaatun et al., 2014) are raising new privacy fears among the more concerned members of the public. According to the privacy paradox, even users who claim to be privacy-conscious frequently behave in a counter-privacy manner (Jaatun et al., 2012a), but there will still be a certain number of users who are paranoid enough to demand a better way.

If we assume the threat model of an honest-but-curious cloud provider (Jaatun et al., 2012b), we have to expect that the provider will index the information we store, and may use this information for various purposes, such as targeted marketing (Vivian, 2015). How can we keep using the cloud without surrendering to privacy invasion? This paper will present a scheme that addresses this challenge.

The remainder of this paper is organized as follows: Section 2 presents relevant background information. The scheme is presented in Section 3, and discussed in Section 4. Section 5 concludes the paper and outlines further work.

2 BACKGROUND

The literature documents numerous approaches to deal with honest-but-curious providers. Fully homomorphic encryption (Gentry, 2009) allows performing computations on encrypted data, in such a manner that the result is the same as the encrypted version of the result from the same operation performed on the corresponding unencrypted input. The main drawback of fully homomorphic encryption has thus far been that it is too processor intensive.

Another variant is Secure Multiparty Computation (Bogetoft et al., 2009; Bogdanov et al., 2008), where multiple providers are involved in the computation of a result, but no one provider has access to cleartext data. A trivial example of SMC is a scheme to calculate the average salary of Alice, Bob, Charlie and Debbie. Alice takes her salary, and adds a random number R to it. She then sends the result to Bob, who adds his salary, and sends the result to Charlie. Charlie add his salary and sends the result to Debbie, who adds here salary and sends the result back to Alice. Alice then subtracts R, and gets the average salary by dividing by four, but none of the four participants have learned the value of anybody else's salary.

Splitting data and dispersing it among multiple providers is an approach that has been used in several contexts (Storer et al., 2009; Adya et al., 2002; Rhea et al., 2003; Jaatun et al., 2012b), where the idea is that no single provider has enough information to make sense of it. The technique has seen application in some special cases, but seems too complicated for widespread use.

Trusted cloud computing (Santos et al., 2009) is another approach where trusted computing principles based on a Trusted Platform Module (TPM) on the physical cloud servers are used to ensure that the provider "stays honest", also to the extent of denying the provider access to cleartext data within users' virtual machines. However, the caveat of this approach is that it moved the single point of trust from the provider to the hardware chip manufacturer, which the PRISM revelations indicate is not necessarily any better.

A totally different approach is using accountability (Pearson, 2011; Pearson and Charlesworth, 2009; Jaatun et al., 2014) to entice the provider to do the right thing, making it a business advantage to offer accountable service. The main disadvantage with this approach is that it depends on the cooperation of the provider.

3 SCHEME DESCRIPTION

The proposed scheme is inspired by RAIN (Jaatun et al., 2012b), but is much simpler.

First, choose any reputable cloud provider offering cloud storage, putting highest emphasis on availability of your information. We will call this "Provider A". Choose a symmetric key k of appropriate key length[1]. Select which files are to be uploaded, and encrypt these using key k. Upload the encrypted files to A.

Encrypted cloud storage is of course not novel, and the next step is therefore essential. A second cloud provider ("Provider B") which offers cloud processing services is chosen. The simplest case would be a Software-as-a-Service (SaaS) document editing application. When documents are to be transferred to provider B, they are decrypted using key k and placed in (temporary) storage associated with provider B's editing application. The plaintext version of the file can then be viewed (and if desired edited).

If the file has been changed, it must be re-encrypted before being transferred back to provider A. After the user is finished interacting with the file, it is deleted from provider B. The scheme is illustrated in Figure 1.

3.1 Custom VM image

The main flaw with the scheme described thus far is that commodity SaaS applications do not incorporate a decryption function. This would seem to require the user to first download the encrypted file from provider A, decrypt it locally, and then upload the plaintext version to provider B. However, this would pretty much invalidate the whole point of a cloud solution, in addition to requiring lots of unnecessary file transfers to the edge network (which presumably has lower capacity and higher costs).

[1]Minimum key lengths (Dent, 2010) keep growing as processing power of commodity computers keeps increasing. Current conventional wisdom is that a 128 bit AES key offers sufficient protection, but your mileage may vary.

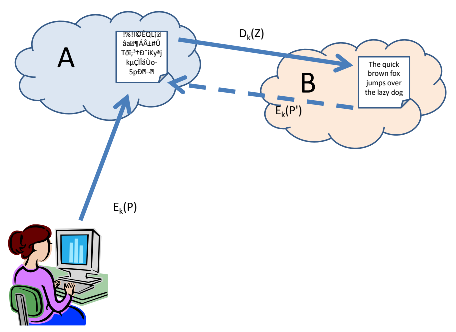

Figure 1: Scheme illustration

A solution to this problem could be to create a custom Virtual Machine (VM) image, complete with decryption software and information viewing/processing/modification software. A practical choice would be an OpenStack (TaheriMonfared and Jaatun, 2012) Linux image with (e.g.) OpenOffice[2] as document editing software. The encryption/decryption software must be custom built, based on proven cryptographic components[3].

When a file from provider A needs to be accessed, a custom VM at provider B is launched, and the encrypted file is transferred to the VM instance. At the same time, the user securely transfers the required decryption key k to the VM instance. The encrypted file is then decrypted, and further processed by the utilities that are already provided in the VM image.

For the highest level of security, the VM instance can be destroyed right after use, but could for convenience also be kept running. In the latter case, the mechanism presented in the next subsection comes into relevance.

3.2 Chaff

Many years ago, Rivest (Rivest, 1998) proposed chaffing (padding sensitive information with irrelevant nonsense) as an approach to improved confidentiality, in his case without using cryptography due to the then draconian export restrictions on cryptographic hardware and software from the US.

Chaffing can be used as an alternative to deleting files from provider B after use, or possibly in addition to it. In the simplest case, the user could make irrelevant changes to the plaintext after an updated document has been returned to encrypted to provider A, and then delete it. This could also be done in the case the document has not been edited, thus keeping Provider B in the dark regarding which edits are real, and which are not. To make this deception complete, it would be necessary to also make a dummy transfer of an encrypted file as well. If, on the other hand, the document is not deleted, a process could periodically add random garbage from various sources to the document, further

[2]`http://www.openoffice.org`

[3]E.g., AES. Under no circumstances should you ever consider creating a new cryptographic algorithm (Ferguson and Schneier, 2003)

obscuring or even replacing the original content, such that any indexing performed by provider B is worthless.

By employing yet another cloud service (provider C), it would be possible to create a bot (Chu et al., 2010) to edit your chaff documents, further confusing the curious provider B.

3.3 Improved key management

Relying on a single key to protect all documents is not satisfactory in the long run, also because provider B necessarily will have access to the key when decrypting files. A better solution would be if the user is able to run a key management application locally, and create session keys for each file. For further security, each key should only be used for encryption once; when a file is re-encrypted after modification, it is assigned a new session key, which must be transferred securely back to the user.

A variant could be to introduce another cloud provider to store the key management database, but this does of course place a lot of trust on that provider, and may not be acceptable to the most paranoid users.

4 DISCUSSION

The threat model of an honest-but-curious provider implies that it should be sufficient to make it difficult enough for a provider to aggregate the user's information. This means that we accept to trust the provider "for the moment", but make sure to remove any accessible information at the first opportunity, and also muddy the waters by introducing chaff in such a manner that the provider cannot discern it from wheat.

The security of the scheme presented in this paper relies on providers being less interested in IaaS data ("bag of bits" (Jaatun et al., 2014)) than structured files (information) at the SaaS level. It is true that Virtual Machine Introspection (VMI) techniques (More and Tapaswi, 2014) makes it possible for a provider to eavesdrop on users' VMs, but it is debatable whether a provider by doing so crosses the boundary between being an honest-but-curious provider and being a dishonest provider.

This scheme is vulnerable to collusion between the different providers; in particular, if only "real" data is stored by provider A, provider A would be able to inform provider B which files are genuine, and which are chaff. Collusion could be made more difficult by increasing the number of cloud providers in use, e.g., introduce a set of providers $\{A_1, A_2, \ldots, A_n\}$ for storing encrypted data, and a different set of providers $\{B_1, B_2, \ldots, B_m\}$ to run the VMs that process plaintext data. At a certain point, however, it will be difficult for the user to manage all the different providers, and an intelligent (non-cloud) client will be needed to use the system.

Note that the sets of providers A and B should be disjoint, and could also be quite different; if provider B is chosen to be "throwaway", i.e., the VM is destroyed after each use, reliability and availability of any given provider is less of an issue. If a given provider B_i is unavailable, the user can simply conjure up a new provider B_{i+1}.

The expanded scheme is illustrated in Figure 2.

5 CONCLUSION AND FURTHER WORK

It is important to acknowledge that just because you are paranoid, that does not mean that they are *not* out to get you (Heller, 1961). Like many security solutions, the one presented in this paper may seem cumbersome and possibly unnecessary, but it could still be useful for some users in the right circumstances.

The next phase of this project is to implement a demonstrator for field testing, and perform a formal security analysis of the resulting system.

Acknowledgment

The research reported in this paper has been supported in part by the European Commission through the EU FP7 project A4Cloud, grant nr. 317550.

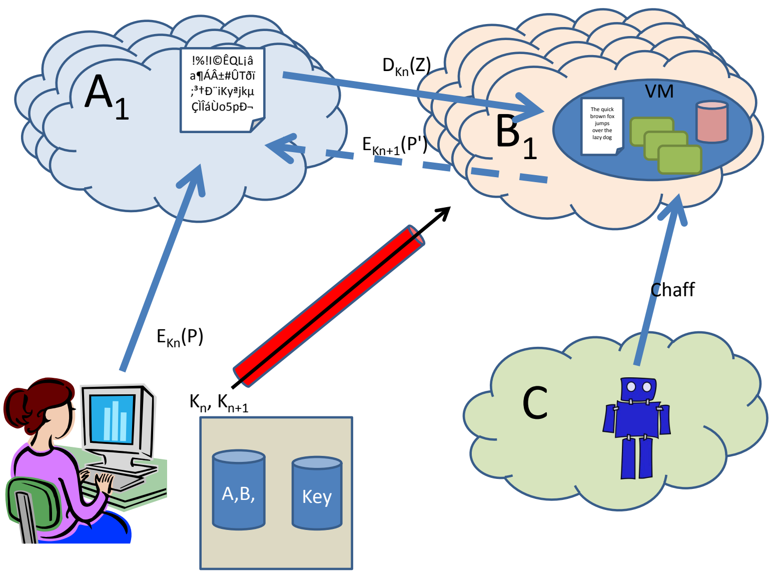

Figure 2: Expanded scheme

REFERENCES

Adya, A., Bolosky, W. J., Castro, M., Cermak, G., Chaiken, R., Douceur, J. R., Jon, Howell, J., Lorch, J. R., Theimer, M., and Wattenhofer, R. P. (2002). FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment. In *In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI*, pages 1–14.

Bogdanov, D., Laur, S., and Willemson, J. (2008). Sharemind: a framework for fast privacy-preserving computations. Cryptology ePrint Archive, Report 2008/289. http://eprint.iacr.org/.

Bogetoft, P., Christensen, D., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J., Nielsen, J., Nielsen, K., Pagter, J., Schwartzbach, M., and Toft, T. (2009). Secure multiparty computation goes live. In Dingledine, R. and Golle, P., editors, *Financial Cryptography and Data Security*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer Berlin / Heidelberg. 10.1007/978-3-642-03549-4_20.

Chu, Z., Gianvecchio, S., Wang, H., and Jajodia, S. (2010). Who is tweeting on twitter: Human, bot, or cyborg? In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 21–30, New York, NY, USA. ACM.

Dent, A. W. (2010). Choosing key sizes for cryptography. *Inf. Secur. Tech. Rep.*, 15(1):21–27.

Ferguson, N. and Schneier, B. (2003). *Practical Cryptography*. John Wiley & Sons, Inc., New York, NY, USA, 1 edition.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178. ACM.

Heller, J. (1961). *Catch-22*. Simon & Schuster.

Jaatun, M. G., Nyre, Å. A., Tøndel, I. A., and Bernsmed, K. (2012a). Privacy Enhancing Technologies for Information Control. In Yee, G. M., editor, *Privacy Protection Measures and Technologies in Business Organizations: Aspects and Standards*.

Jaatun, M. G., Pearson, S., Gittler, F., and Leenes, R. (2014). Towards strong accountability for cloud service providers. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 1001–1006.

Jaatun, M. G., Zhao, G., Vasilakos, A., Nyre, Å. A., Alapnes, S., and Tang, Y. (2012b). The design of a redundant array of independent net-storages for improved confidentiality in cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, 1(1):13.

More, A. and Tapaswi, S. (2014). Virtual machine introspection: towards bridging the semantic gap. *Journal of Cloud Computing*, 3(1).

Pearson, S. (2011). Toward accountability in the cloud. *Internet Computing, IEEE*, 15(4):64–69.

Pearson, S. and Charlesworth, A. (2009). Accountability as a way forward for privacy protection in the cloud. In *Cloud Computing*, pages 131–144.

Rhea, S., Eaton, P., Geels, D., Weatherspoon, H., Zhao, B., and Kubiatowicz, J. (2003). Pond: the OceanStore Prototype. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*.

Rivest, R. L. (1998). Chaffing and winnowing: Confidentiality without encryption. *CryptoBytes (RSA laboratories)*, 4(1):12–17.

Rong, C., Nguyen, S. T., and Jaatun, M. G. (2013). Beyond lightning: A survey on security challenges in cloud computing. *Computers & Electrical Engineering*, 39(1).

Santos, N., Gummadi, K. P., and Rodrigues, R. (2009). Towards trusted cloud computing. In *HOTCLOUD*. USENIX.

Storer, M. W., Greenan, K. M., Miller, E. L., and Voruganti, K. (2009). Potshards a secure, recoverable, long-term archival storage system. *Trans. Storage*, 5:5:1–5:35.

TaheriMonfared, A. and Jaatun, M. G. (2012). Handling compromised components in an IaaS cloud installation. *Journal of Cloud Computing*, 1(1).

Vivian (2015). Ads in gmail. `https://support.google.com/mail/answer/6603?hl=en`.