ENCRYPTED DOMAIN PROCESSING FOR CLOUD PRIVACY
Concept and Practical Experience

D. A. Rodríguez-Silva, F. J. González-Castaño, L. Adkinson-Orellana, A. Fernández-Cordeiro
Gradiant, ETSI Telecomunicación, Campus, 36310 Vigo, Spain

J. R. Troncoso-Pastoriza, D. González-Martínez
Universidade de Vigo, ETSI Telecomunicación, Campus, 36310 Vigo, Spain

Keywords: Cloud privacy, Encrypted domain processing, Homomorphic encryption, Trusted computing.

Abstract: Cloud security comprises access control and end-to-end security based on flow or message-level privacy. In some applications, in which all processing takes place at the client side and the Cloud simply handles data storage (e.g. Google Docs), on-line data encryption/decryption guarantees privacy. However, when a service requires server processing (e.g. spreadsheets), privacy must necessarily rely on a dependable entity according to *local* regulations. Summing up, full Cloud privacy has not been achieved so far. In this paper we take a step towards that goal. We propose executing server side operations in the encrypted domain, so that both the operands and the results are opaque to the server, yet clear to the user. We evaluate this concept with a real Google Apps implementation of basic arithmetic operations.

1 INTRODUCTION

As in other service architectures, Cloud security comprises access control and end-to-end security based on flow or message-level privacy. *Access control* is extremely important in the Cloud, to protect its resources against non-authorized access. This is not a challenge, because many traditional solutions for service-oriented architectures can be easily applied to Cloud environments. Regarding *privacy*, current schemas mainly focus on sending user information in a secure way to Cloud servers. This approach is completely valid when the Cloud provider is reliable. Otherwise, sensitive data must be completely protected from the moment it leaves the client. In some applications, in which all processing takes place at the client side and the Cloud simply handles data storage (e.g. Google Docs), on-line data encryption guarantees privacy. However, when a service requires server processing (e.g. spreadsheets), privacy must necessarily rely on a dependable entity according to *local* regulations. Summing up, full Cloud privacy has not been achieved so far.

In most countries there are institutions that regulate personal data processing. For example, the European Commission implemented the Data Protection Directive in 1995. However, an international agreement is extremely difficult to reach due to the deep political differences. This is a complex context for Cloud Computing security due to the need to distribute resources among different locations to maximize performance. There have been privacy problems with Google services in some countries in recent years. For example, in Spain, the Data Protection Agency declared the business model of GMail (inserting advertisements sensitive to e-mail content) (Europasur, 2008) to be illegal, and in Italy, the Milano court accused three Google executives of illegally accessing video postings (G. Sartor, 2010). We conclude that a mechanism to keep user information privacy at the server side without losing any of the advantages of the Cloud paradigm is of great interest.

In this paper we take a step towards that goal. We propose executing server side operations in the encrypted domain, so that both the operands and the results are opaque to the server, yet clear to the user. We evaluate this concept with a real Google Apps implementation of basic arithmetic operations.

The rest of this paper is organized as follows: in section 2 we discuss the background to Cloud security and other technologies related to our research. In section 3 we present the design decisions to implement

A. Rodríguez-Silva D., J. González-Castaño F., Adkinson-Orellana L., Fernández-Cordeiro A., R. Troncoso-Pastoriza J. and González-Martínez D..
ENCRYPTED DOMAIN PROCESSING FOR CLOUD PRIVACY - Concept and Practical Experience.
DOI: 10.5220/0003380905910596
In *Proceedings of the 1st International Conference on Cloud Computing and Services Science* (CLOSER-2011), pages 591-596
ISBN: 978-989-8425-52-2

our concept and the architecture of a Cloud application to test it. In section 4 we describe some numerical results and, finally, section 5 concludes the paper.

2 BACKGROUND

2.1 Cloud Privacy

Despite the many advantages of Cloud Computing such as scalability, flexibility and cost savings, there are no guarantees that stored data will not be accessed by unauthorized entities, such as the Cloud provider itself or malicious attackers. Although the user can deliberately assume the risks of placing information on the Cloud, there exist activities in which law regulates data protection (F. Doelitzscher, 2010). This is the case of government, banking, medical care or accounting.

In addition to confidentiality, there are other barriers that may dissuade users from migrating their systems to the Cloud, such as the possibility of data integrity threats or leaks (M. Jensen, 2009; CSA, 2010). There are proposals of additional security layers to protect the user from data mishandling by Cloud providers. Most solutions are based on a ciphering schema using private or public-key encryption algorithms, which are applied to the client side or to the Cloud server. Client side solutions encrypt any data that leaves the client: encrypted storage on the Cloud (S. Kamara, 2010); encrypted DaaS (Databases as a Service) (T. Xiuxia, 2009); and encrypted SaaS (Software as a Service) (L. Adkinson-Orellana, 2010). These solutions provide full privacy, but are only valid when all data processing takes place at the client side and the Cloud simply handles data storage. To provide encryption, there are proposals based on introducing new elements to the Cloud architecture. An example is the PasS (Privacy as a Service) model (W. Itani, 2009), in which cryptographic processors perform ciphered operations (R. Anderson, 2006). PasS thus depends on the hypothetical deployment of those processors by Cloud providers.

In this paper we propose a model that relies on *encrypted domain processing* in which the only dependable entity is the user. According to this paradigm, both the operands and the results of the operations are hidden to the servers that execute them. In order to implement this model, Cloud providers must collaborate by installing encrypted domain processing libraries in their servers, but, unlike in PasS, it is not relevant if they are reliable or not.

2.2 Commercial PaaS Environments

The Platform as a Service paradigm provides developers with a high-level managed software environment for building particular classes of applications and services. That environment includes mechanisms to access underlying computing and network resources, and offers additional services such as monitoring and billing. Thus, the infrastructure is abstracted away from the developer, who can create and manage Cloud applications efficiently.

Google App Engine (Google, 2010) is one of the most popular commercial PaaS environments. It provides a single, pre-built solution for constructing very-large-scale web-based applications hosted by Google's infrastructure, using standard Java technologies (and, by extension, other JVM languages), or a dedicated Python runtime environment.

Microsoft Azure (microsoft.com/windowsazure) is a group of Cloud development services that forms a complete PaaS and provides a Windows-based environment for running applications and storing data on Microsoft servers. Software developers may use C# or other .NET languages, along with traditional tools such as Visual Studio 2008. Azure also provides a portal to create, configure and monitor applications.

Another relevant platform is Force.com (www.salesforce.com). It allows the development of custom user interfaces using standard Web technologies, which can be generated and customized automatically with a layout editor. Force.com had the disadvantage of a substantial learning curve, but the VMForce Java development environment has solved this problem.

We do not intend to provide an exhaustive description, since there are many more competitors in this area: GigaSpaces XAP, Heroku, Bungee Connect...

In PaaS, security usually refers to controlling access to resources with enforcement policies, and encrypting generated data once the processing is complete. Therefore, it is assumed that the Cloud provider is reliable. For this reason, we have focused our work on the development of a security schema for PaaS that allows encrypted data processing on Cloud servers using homomorphic encryption techniques.

2.3 Arithmetic Processing in the Encrypted Domain

Informally, data privacy consists of the possibility of hiding certain data from certain parties. Traditional cryptosystems provide only all-or-nothing security, in the sense that they do not allow any access to encrypted data unless the deciphering key is available. Data Processing in the Encrypted Domain (DPED) (Ahituv et al., 1987) was born to satisfy the privacy needs for data that must be handled by third parties. It allows the processing of certain operations without the need to encrypt the data and consequently without the need to access their clear-text version. This concept has been a hot topic in processing theory in the recent years.

The first proposals introduced the concept of Secure Multiparty Computation (Yao, 1982), symbolized by the so-called Millionaires Problem, in which two millionaires wish to know who is the richest without disclosing their respective wealths. That solution employed the concept of garbled circuits, which allow the execution of a function, expressed as a combination of binary gates, on certain data (also in a binary representation) that belong to two parties. After the execution of the garbled circuit, both parties can obtain the result of the computation, but neither has access at any time to the data belonging to the other party. In spite of the generality of that solution, its inefficient implementation was the biggest obstacle for the development of arithmetic processing in the encrypted domain. Finding efficient solutions for the secure execution of a generic function is still an open problem. Recently, there have been numerous contributions (Troncoso-Pastoriza et al., 2007; Troncoso-Pastoriza et al., 2009). In particular, arithmetic operations can be performed efficiently in the encrypted domain thanks to the concept of additive and multiplicative privacy homomorphisms (Brickell and Yacobi, 1987), as later explained in Section 3.2.

2.4 Related Work

The idea of homomorphic encryption in Cloud environments is quite novel. There has been some work on this topic, but a complete solution for the Cloud is still missing. In (Chow et al., 2009) some common Cloud security problems are characterized, and the proposed solutions include empowering data with intelligence to protect themselves using trusted computing or privacy-enhanced business intelligence based on homomorphic encryption. In (Pearson et al., 2009) a privacy manager for Cloud Computing, an appliance for secure Cloud access, was presented. To achieve this goal an obfuscation mechanism based on homomorphic encryption is used. The paper describes some application scenarios such as SQL queries or photo tagging. The solution only encrypts part of the information that is sent to the Cloud in order to keep efficiency at reasonable levels. We conclude that previous related work focuses on particular applications or assumes that the service provider is honest, since server side appliances are necessary.

Our contribution consists of applying efficient homomorphic encryption to basic operations in order to provide a development platform for secure Cloud applications, generalizing the concept of CryptoDSPs, presented in (Troncoso-Pastoriza and Pérez-González, 2010) for signal processing operations, and providing a reference implementation as a proof of concept.

3 ENCRYPTED PROCESSING ON PAAS ENVIRONMENTS

3.1 Choosing a PaaS Platform

We have selected Google App Engine as the PaaS for our research. It is free up to a certain resource usage level, so it is convenient for developing a realistic proof-of-concept. Application development is well documented. An active and experienced community provides support in many fora. Availability of open source IDEs as Eclipse or Netbeans saves costs since commercial development tools are not necessary.

Google App Engine is more mature than Azure and more open than Force.com. Platform performance depends on development language, infrastructure load (which is related to server location) and implemented functionality (caches, load balancing). A deeper study (beyond the scope of our research) would be required to compare existing providers. In any case, in our tests, Google App Engine performed satisfactorily. It is worth mentioning that Google offers libraries to integrate the ubiquitous Google Apps (Accounts, Docs, Gmail, IM etc) at zero cost.

The DPED solution we propose allows for a completely secure Cloud. Data privacy is automatically provided, and Cloud services are not hindered by privacy restrictions. In order to show the potential of this approach, we present a proof-of-concept of arithmetic calculations in the encrypted domain, in the form of encrypted processing libraries for a PaaS to be used in the development of secure Cloud applications. To test their performance we have built a simple application consisting of a web calculator, which executes remote calculations with encrypted operands, as described in the following section.

3.2 Arithmetic in the Encrypted Domain

At a conceptual level, arithmetic operations can be performed in the encrypted domain using a privacy homomorphism (Brickell and Yacobi, 1987). This consists of a group homomorphism between clear-text data and encrypted data. A formal description is given in (Rivest et al., 1978).

The practical and provably semantically secure cryptosystems that present a privacy homomorphism typically allow for the execution of one arithmetic operation (either addition or multiplication) directly on ciphertexts, without the need for decryption or interaction with a trusted decryption party. This presents several decisive advantages over other privacy solutions, such as the drastic reduction of overheads in the communication required between parties and the automatic provision of privacy while data are being processed, as they never leave their encrypted state.

For our proof-of-concept, we have implemented a client-server software library in Java that provides transparent encryption and homomorphic operation on data, conforming to the new secure paradigm of *secure remote data processing*. This library is divided into two modules with the following functionality:

- **Client Module.** It provides transparent encryption of the data prior to their transmission to the server, and decryption of the data received from the server prior to their presentation.
- **Server Module.** It provides access to the arithmetic operations on the encrypted data received from the client or stored on the server, applying suitable privacy homomorphisms.

We have chosen Java to implement these libraries due to its portability, compatibility with Google App Engine APIs and easy integration in an applet. This choice also has some drawbacks, mainly low computational efficiency, which we have overcome through concurrent precomputation of key values at the client side, and efficient modular arithmetic algorithms.

3.3 Proof-of-concept

In order to demonstrate our concept, we developed an application that employs Web services hosted by Google Cloud which rely on the encrypted domain processing libraries described in the previous section. The application has two parts: the server side, implemented with Java servlets, and the client side, a Java applet embedded in a JSP page. As a toy example, the applet has a GUI that resembles a calculator written with the Swing Java library and the NetBeans IDE development environment by Oracle. The servlet was created with the Google App Engine SDK for Java, installed as an Eclipse SDK plugin. That SDK also allows the integration of the Google Accounts service into our application to manage user sessions. This is important to store information from simultaneous users and keep it separate. The application is currently deployed in Google App Engine and available at http://calcgrad.appspot.com. The client encrypts the data sent to the server and decrypts the results. These procedures work as follows (Figure 1):

Once the operands are available, when the operation takes place for the first time, the client initializes the object module *EncryptedProcessClient* for the execution of that operation and subsequent ones (with the *initialize* method). The returned code must be passed to the server to initialize its module. Then, the operands are ciphered (with the *performOperation* method, which returns a wrapper) and the client uses HTTP tunneling with POST requests to exchange ciphered data with the server. An operation may need several client-server communication cycles to obtain the final result. The client calls the *isContinueOperating* method to determine if it must communicate with the server again. In the response, the client receives a ciphered result. This result is needed to call the *continueOperation* method, which modifies the wrapper returned by *performOperation*. Using this wrapper, the client checks if it must perform additional communication cycles with the *isContinueOperating* method. If not, the final result is available and the client deciphers it (using the *finalDecode* method), to display the solution locally.

The server operates with the ciphered data received from the client. For the server, we have written a Java servlet that processes client requests and returns the results. When a petition arrives, the server checks the requested action from the corresponding object field and obtains the ciphered data. If the client requests a server object module initialization (*EncryptedProcessServer*), the server must call the *initialize* method with the received data as an argument. On the contrary, if an operation is requested, the server takes the ciphered operands and, with the initialized module, calls the *operate* method to execute the operation. The resulting object is kept on the server for further usage. Once *operate* is applied, the result is sent to the client for it to check if the operation has finished or if additional communication cycles are necessary. In any of these cases, the behavior of the servlet is the same: it gets the received ciphered data, calls the appropriate method according to the operation the client requests, stores the resulting object and returns the ciphered result to the client.

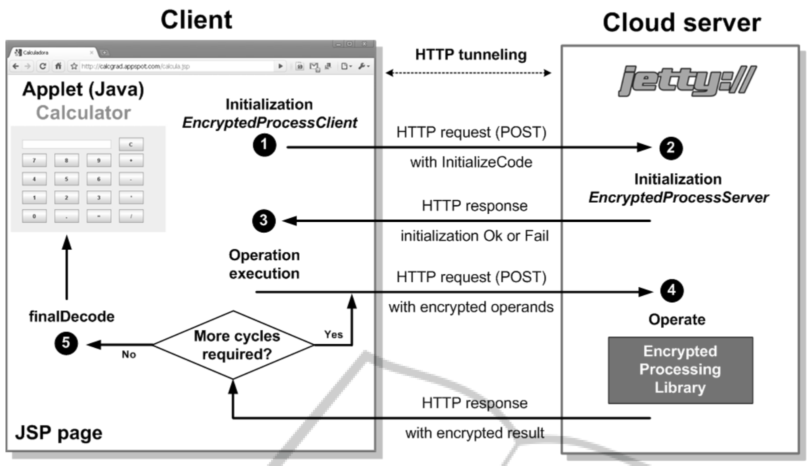

Figure 1: Application workflow.

4 NUMERICAL RESULTS

We evaluated the performance of the encrypted domain libraries by measuring the elapsed times of arithmetic operations that employ them in our Cloud application. The test layout comprised a client computer (Intel Pentium 4 @ 2.80GHz, 1 GB RAM with Ubuntu 9.10), where the Cloud application applet was executed through an Internet browser (Firefox 3.6); and a local server (Intel Core 2 Quad Q8400 @ 2.66GHz, 4GB RAM with Ubuntu 9.10), to perform the encrypted operations under Jetty remotely. Jetty is the light Web server Google has chosen, which supports Java servlets. However, we do not have detailed information about the Google Cloud servers of Google App Engine. All machines used the Java 6 runtime environment.

There were two test scenarios: local, in which we executed the encrypted operations directly on the local server, and remote, relying on Google App Engine. In both cases the client machine was the same. Therefore we had a client-server architecture where the client (the Java applet) issued several requests for arithmetic operations (addition and multiplication). In the tests we measured the time from the moment at which an operation was requested by the user (by clicking on the equal symbol) to the moment at which the application displayed the result (i.e. communication time was taken into account).

We made 2000 runs for each operation in each scenario, using the encrypted domain libraries or not. The time to calculate ciphered operations does not depend on the length of the original operands, since their lengths coincide once encrypted. Table 1 shows our results. Each pair of values represents the average value of the elapsed execution time of a given operation and its standard deviation in milliseconds. The elapsed times of ciphered operations include encryption/decryption at the client side and encrypted processing at the server side.

Table 1: Tests results: mean and standard deviation (ms).

Test	Addition	Multiplic.
Local (clear)	$\bar{x} = 8.02$ $\sigma = 4.19$	$\bar{x} = 7.73$ $\sigma = 4.32$
Local (encrypted domain)	$\bar{x} = 80.66$ $\sigma = 16.31$	$\bar{x} = 207.88$ $\sigma = 15.24$
Google App Engine (clear)	$\bar{x} = 225.40$ $\sigma = 135.12$	$\bar{x} = 225.99$ $\sigma = 133.67$
Google App Engine (encrypted domain)	$\bar{x} = 381.24$ $\sigma = 19.57$	$\bar{x} = 795.14$ $\sigma = 28.97$

If the encrypted domain libraries are not used, the standard deviation of the elapsed times is quite high, because the influence of operating system background processes and network communication times is rather variable. The elapsed times with Google App Engine are much higher than in the local case, mainly due to Internet delays. Ciphered operations are slower than clear ones, as expected. Indeed, although the average times for additions and multiplications are quite similar for clear operations, they differ significantly if ciphered (the multiplications are up to twice as long). This is due to the complexity of encrypted multiplications, which may require several cycles for a result to be obtained, as described in the previous section. However, although the elapsed times for ciphered operations are quite long, they are still satisfactory for user applications with light processing load, such as spreadsheets, and encrypted domain processing technology is expected to evolve rapidly. Let us recall that the encrypted domain libraries were developed with Java, as requested by Google App Engine, so they are executed on a Java virtual machine.

5 CONCLUSIONS

In this paper we have presented a new model for secure Cloud Computing, without any reliable entity but the client itself, based on encrypted domain processing. We have also presented a real implementation as a proof-of-concept, relying on Google App Engine. Our results with basic arithmetic operations indicate that the model is valid for user applications in the Cloud with light server processing. As future work we will develop a more complete user application such as a spreadsheet to perform more complex encrypted operations including vectorial ones. To improve performance, we will consider the use of threads, and remote ciphered operations will be executed with a C version of the libraries through a JNI interface.

ACKNOWLEDGEMENTS

This research was supported by the ACETIC consortium and the SAFECLOUD grant (09TIC014CT), funded by Xunta de Galicia, Spain. The work was also partially supported by projects "Consolidation of Research Units" (2010/85), SCALLOPS (10PXIB322231PR) and VISAGE (10TIC008CT), funded by Xunta de Galicia; by project COMONSENS (CSD2008-00010) of the CONSOLIDER-INGENIO 2010 Program; and by the Spanish Ministry of Science and Innovation FPU grant AP2006-02580.

REFERENCES

Adkinson-Orellana, L., Rodríguez-Silva, Gil-Castiñeira, F. and Burguillo-Rial J. C. (2010). Privacy for Google Docs: Implementing a transparent encryption layer. In *Proc. CloudViews 2010*, p. 21–22, Porto, Portugal.

Ahituv, N., Lapid, Y., and Neumann, S. (1987). Processing encrypted data. *Communications of the ACM*, 30(9):777–780.

Anderson, R., Bond, M., Clulow, J. and Skorobogatov, S. (2006). Cryptographic processors - a survey. *Proceedings of the IEEE 94 (2)*: p. 357–369.

Brickell, E. F. and Yacobi, Y. (1987). On privacy homomorphisms (extended abstract). In *Proc. EUROCRYPT 87*, LNCS 304, p. 117–125.

Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., and Molina, J. (2009). Controlling data in the Cloud: outsourcing computation without outsourcing control. In *Proc. CCSW '09*, p. 85–90, NY, USA.

CSA (2010). *Top Threats to Cloud Computing V1.0.* Retrieved August 16, 2010 from http://www.cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf.

Doelitzscher, F., Reich, C. and Sulistio, A. (2010). Designing Cloud services adhering to government privacy laws. In *Proc. TSP'10*, Bradford, UK.

Europasur (2008). *Gmail, el correo espía de Google, ilegal en Europa*. Retrieved August 16, 2010 from http://www.europasur.es/article/sociedad/38601/gmail/correo/espia/google/ilegal/europa.html.

Google (2010). *Google App engine*. Retrieved July 7, 2010 from http://code.google.com/appengine.

Itani, W., Kayssi, A. and Chehab, A. (2009). Privacy as a service: Privacy-aware data storage and processing in Cloud Computing architectures. In *Proc. Int. Workshop on SCC'09*, p. 12–14, Chengdu, China.

Jensen, M., Schwenk, J., Gruschka, N. and Lo Iacono, L. (2009). On technical security issues in Cloud Computing. In *Proc. CLOUD '09*, p. 109–116, Bangalore, India.

Kamara, S. and Lauter, K. (2010). Cryptographic Cloud storage. In *Workshop on Real-Life Cryptographic Protocols and Standardization 2010*.

Pearson, S., Shen, Y., and Mowbray, M. (2009). A privacy manager for Cloud Computing. In *Cloud Computing*, LNCS 5931, p. 90–106..

Rivest, R., Adleman, L. and Dertouzos, M. (1978). On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, p. 169–177. Academic Press.

Sartor, G. and Viola de Azevedo Cunha, M. (2010). *The Italian Google-Case: Privacy, Freedom of Speech and Responsibility of Providers for User-Generated Contents*. Oxford University Press.

Tian, X., Wang, X. and Zhou, A. (2009). DSP re-encryption: A flexible mechanism for access control enforcement management in DaaS. In *Proc. CLOUD '09*, p. 25–32, Bangalore, India.

Troncoso-Pastoriza, J. R., Comesaña, P., and Pérez-González, F. (2009). Secure direct and iterative protocols for solving systems of linear equations. In *Proc. SPEED Workshop 2009*, p. 122–141, Lausanne, Switzerland.

Troncoso-Pastoriza, J. R., Katzenbeisser, S., and Celik, M. (2007). Privacy preserving error resilient DNA searching through oblivious automata. In *Proc. 14th ACM Conference on Computer and Communications Security*, p. 519–528, Alexandria, Virginia, USA.

Troncoso-Pastoriza, J. R., Pérez-González, F. (2010). CryptoDSPs for Cloud Privacy. In *Proc. Int. Workshop on CISE 2010*, LNCS, Hong Kong, China.

Yao, A. C. (1982). Protocols for secure computations. In *Proc. IEEE Symposium on Foundations of Computer Science*, p. 160–164.