Securely Storing and Executing Business Processes in the Cloud

David Martinho[1] and Diogo R. Ferreira[1]

IST – Technical University of Lisbon
{davidmartinho,diogo.ferreira}@ist.utl.pt

Abstract. This paper proposes an architectural solution that allows organizations to rely on cloud-based services to securely operate their business processes. The solution is built upon a thick client and thin server architectural pattern, where security constructs such as public-key and symmetric cryptographic systems are used to maintain confidentiality between the participants while keeping the server unaware of their participations and business process instances.

Key words: Cloud Computing, Security, BPM, Cryptography Systems

1 Introduction

Given the latest advances in IT, the envisioning of computing as an essential utility for the general community led to the proposal of novel computing paradigms such as cloud computing. Cloud computing [1] is presented as being able to transform the IT industry by introducing a delivery model of Software as a Service (SaaS), and shaping the way infrastructural hardware is designed and adopted. Rather than acquiring expensive Business Process Management System (BPMS) software licenses, and install and manage the software within a local hardware infrastructure, an organization signs up to use the application hosted by the company that develops and sells the software as a service.

However, cloud computing means entrusting data to information systems managed by external parties on remote servers "in the cloud". This raises privacy and confidentiality concerns given that the service provider can access all data, and accidentally or deliberately leak it or use it for unauthorized purposes [7]. These threats hinder the adoption of cloud-based solutions by organizations.

This paper proposes an architectural solution that enables organizations to securely store their business processes within cloud-based services, while preserving zero-knowledge of the service provider concerning their business processes. To validate such solution, the following requirements must be preserved: (R1) a process instance must be shared among its participants, allowing concurrent executions to take place; (R2) the service provider must never have access to the business process instance content, and (R3) never know which process instances are associated to which participants; (R4) communication must never be compromised by potential eavesdropping; and (R5) a malicious party cannot deprive authorized process participants to access their business processes.

2 Architectural Solution

The use of cloud-based services is associated with the client-server architectural pattern. This architectural pattern allows to centralize the data manipulated by an application within a server, and distribute the interaction with end-users through a set of multiple clients that communicate with the centralized server. The use of this architectural pattern contributes to the fulfillment of the first requirement (R1) where the process instance must be shared among the process participants, allowing concurrent executions to take place.

However, this client-server architectural pattern creates problems when security requirements are introduced. To ensure that (R4) communication is not compromised by potential eavesdropping, two possible solutions exist: either the client communicates with the server through Secure Socket Layer (SSL) protocols, ensuring the communication is encrypted, or the client encrypts the business process before sending it to the server. Considering also that (R2) the service provider must never have access to the content of business process instances, we must adopt the latter strategy.

By encrypting data beforehand, it becomes impossible for the server to actually make domain-specific computations (e.g. execute the process model, compute work allocation, etc...). Hence, we must adopt a particular variant of the client-server architectural pattern: a thick client and a thin server. With this variant, all the business logic is essentially present in the client, where the business process instance is decrypted and executed, while the server takes the role of a centralized repository, providing access to encrypted process instances.

Now that we justified the decisions on the main architectural pattern, we will focus on the security layer that empowers the communication workflow occurring between the client and server applications. When integrating a cross-cutting concern such as security into our solution, we must consider the following security qualities [6]: *confidentiality* to ensure that the content of a business process instance is only available to authorized parties; *integrity* to ensure that the content of a business process instance has not been tampered and modified by unauthorized parties; and *availability* to ensure that authorized parties cannot be deprived from accessing their business process instances.

In this proposal, we are interested in supporting all three security qualities. However, in what concerns availability, we will not focus on tackling denial-of-service attacks, which can be avoided by using generic infrastructure-based strategies that are outside the scope of this paper. Instead, we are concerned to ensure (R5) that a malicious party cannot deprive authorized process participants to access their business processes, in situations that the service is still available (e.g. data corruption).

To ensure (R2) that a service provider can never access the business process instance's content, and (R3) never know which process instances are associated to which participants, public-key and symmetric cryptography systems are used.

Depending on whether the encryption or decryption key is publicly shared, the public-key cryptography system can be used to address either confidentiality or integrity qualities respectively. In order to achieve both confidentiality and

integrity security requirements, our solution assigns two distinct pairs of asymmetric keys to each user for each purpose. However, in public-key cryptographic systems, the security is handled peer-to-peer, meaning that it is not well-suited for sharing confidential business process instances with more than one participating party at a time, hindering (R1) the sharing of business process instances among the process participants, and for concurrent executions to take place.

The symmetric cryptography system works upon a shared secret strategy, where two or more parties may exchange confidential (i.e. encrypted) business process instances because they secretly share the same symmetric key. This means that a symmetric cryptography system is only valid to support the confidentiality security requirement, since there is no concept of public key with which other users can verify the authorship and integrity of a message.

Our solution makes use of both these public-key and symmetric cryptographic systems within a communication workflow between a thick client where business processes are executed, and a thin server that allows process participants to share encrypted business process instances while keeping the server, and consequently the service provider completely unaware of the business processes' content.

2.1 Client-Server Communication Workflow

Let us assume that we have a participant named Alice, who creates a new business process instance and executes the first activity. To do so, she logs in into the workflow client using her credentials (i.e. username and password). Using her credentials, the workflow computes her respective symmetric key P_K, and uses her username to retrieve her encrypted passport (see Definition 1) from the server provider, and decrypts it using the computed symmetric key P_K.

Definition 1 (Passport). *Given a particular user U working for organization O, let O_i be the organization identity containing information necessary to communicate with the service provider. Let (E_{K^c}, D_{K^c}) be her pair of asymmetric keys respecting to confidentiality, and (E_{K^i}, D_{K^i}) her pair of asymmetric keys concerning integrity. Let P_L be her list of available process instances containing tuples (process-title, ID_R, p_k) which refer to the process title, the server's remote ID of the encrypted process instance, and the symmetric key used to encrypt the process instance. Finally, let S_K be a symmetric key that is shared with all organization members. A passport is defined by the 5-tuple $(O_i, D_{K^c}, E_{K^i}, P_L, S_K)$.*

After Alice logs in, and the workflow client application retrieves her passport from the server, it uses the first element of her passport tuple, the organization identity O_i, to fetch the shared organization's information repository from the server, which is encrypted with S_K. The organization's information repository contains the organization's business process models that can be executed within the thick workflow client, and the list of employees with their respective public confidentiality and integrity keys, as well as their organizational roles to allow the workflow engine to perform work allocation. Then, Alice selects one process model P from the list of available business processes available according to her organizational roles, and begins the workflow depicted in Figure 1.

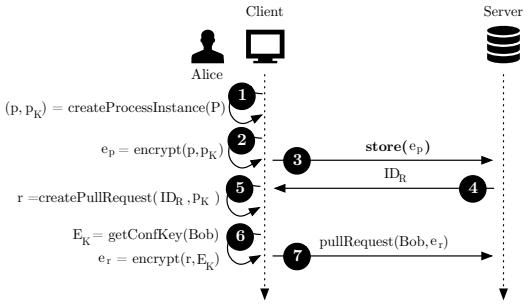

Fig. 1. Alice's Communication Workflow with the Server.

First, (1) Alice creates a new business process instance p from the business process model P, also generating a new symmetric process key p_K to encrypt that process instance. Then, after executing her workflow task, the workflow client application decides, based on the model for organizational roles contained in the organization repository fetched earlier, the process participant that will execute the next task, which is Bob in this case. Afterwards, (2) the workflow client, before sending the process to the server, encrypts it using the generated symmetric key p_K, originating an encrypted process instance e_p. Then, (3) the workflow client sends a new storage request to the server so that it stores the encrypted process instance and (4) respond with the remote identifier ID_R that will be used for future retrievals and re-storages. After the server replies with the remote identifier, the workflow client application can (5) create a pull request (see Definition 2) containing the remote identifier ID_R and the symmetric key p_K that will be needed to decrypt the process instance. In this step, the workflow client also adds the process instance information in Alice's process list P_L. The workflow client application then (6) confidentially addresses the pull request to Bob by encrypting it with his public encryption key E_{K^c}, listed in the previously fetched organization repository, and signing the content with Alice's private integrity key. Finally, the workflow client sends that signed and encrypted pull request to the server (7).

Definition 2 (Pull Request). *Let (ID_R, S_K) be a tuple containing a remote identifier ID_R that identifies a business process instance encrypted with a symmetric key S_K. Sending a pull request to user U, consists on encrypting such tuple with the user's confidentiality public key E_{K^c} and store it in the server, associating it to the respective user U.*

The pull request is encrypted so that (R3) the service provider must never know which process instances are associated to which process participants. Omitted for the sake of space, the pull request is signed by Alice so that Bob can verify the authorship of the request and avoid malicious parties outside his organization to send him pull requests. The communication workflow between Bob's client and the server is depicted in Figure 2.

After Bob logs in into the system, (1) the client also fetches the server for any new pull requests addressed to Bob. The server will then (2) reply to the client

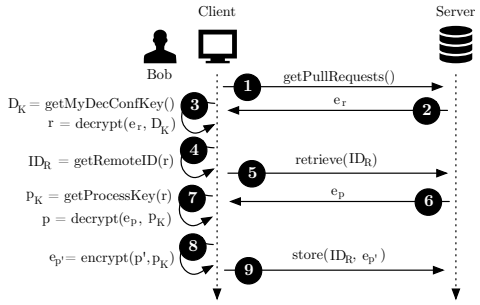

Fig. 2. Bob's Communication Workflow with the Server.

with the encrypted pull request e_r earlier originated from Alice's task execution. Then, (3) the client decrypts the request using Bob's private confidentiality decryption key D_{K^c} available in his passport, obtaining the decrypted pull request r. With the pull request decrypted, (4) the client may use the remote id ID_R to (5) retrieve the encrypted process e_p from the server. After the server replies with the encrypted process (6), the client can (7) use the process symmetric key p_k, also contained in the pull request r, to decrypt the encrypted process e_p into p. Finally, after Bob executes his task, the workflow client application repeats all the steps, storing the process instance information in Bob's process list P_L, and (8) re-encrypting the new version of the process instance p' with the same encryption key p_K , but with the particularity of (9) re-storing it within the same ID_R: this allows previous participants to continually access the business process instance.

In fact, the server does not overwrite the process instance, it rather creates a new version of it. We consider versioning for the situation where a malicious party discovers the remote id ID_R, and attempts to overwrite the currently stored business process instance with the intention of depriving authorized process participants to access their business processes. As the server versions the encrypted business process instances stored under the same ID_R, if the workflow client fails to decrypt the business process instance using the process symmetric key present in the pull request, the server flags that version and responds with the previous version, repeating the process until a version is accepted or there are no versions left. This strategy contributes directly to ensure the delivery of requirement R5.

The solution proposed can be easily extended to support choreographies among organizations. Nevertheless, such extensibility presumes that the choreography actors, i.e. the organizations, would have compatible business process models [8], and their workflow clients implementing the same architectural solution proposed here.

3 Related Work

There is previous work [2, 4] focusing on adapting access control and authorization techniques used in database and operating systems to the business process

management scope. Nevertheless, such fields envision security within relatively narrow applications. Security is often integrated into business process management systems in an ad-hoc manner, during the implementation process [3], disregarding the specificities of its domain.

In [5], a novel architecture of cloud-based BPM is proposed and analyzed, supporting end-user distribution of non-compute-intensive activities and sensitive data. Nevertheless, they do not approach aspects concerning privacy and confidentiality of business processes in the cloud as proposed in this paper.

4 Conclusions

In this paper, we introduced an architectural solution that establishes a workflow of interactions between a thick client that executes the business process and a thin server, subscribed on a cloud-based delivery model, that is confined to simple services of data storage and retrieval. The solution is based on a client application that manages both the execution of business process instances and the participant's access control lists while leveraging on untrusted data storage and retrieval cloud services through a well-defined layer of security based on symmetric and asymmetric cryptographic constructs.

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[2] V. Atluri, W. Huang, and E. Bertino. An execution model for multilevel secure workflows. *Database Security*, 11:151–165, 1998.

[3] M. Backes, B. Pfitzmann, and M. Waidner. Security in business process engineering. In *Busines Process Management*, volume 2678 of *LNCS*, pages 1019–1019. Springer, 2003.

[4] E. Bertino, E. Ferrari, and V. Atluri. A flexible model supporting the specification and enforcement of role-based authorization in workflow management systems. In *2nd ACM workshop on RBAC*, pages 1–12, 1997.

[5] Y.B. Han, J.Y. Sun, G.L. Wang, and H.F. Li. A cloud-based bpm architecture with user-end distribution of non-compute-intensive activities and sensitive data. *JCST*, 25(6):1157–1167, 2010.

[6] P. Herrmann and G. Herrmann. Security requirement analysis of business processes. *Electronic Commerce Research*, 6(3):305–335, 2006.

[7] Mark D. Ryan. Cloud computing privacy concerns on our doorstep. *Communications of the ACM*, 54(1), 2011.

[8] M. Weske. *Business process management: concepts, languages, architectures*. Springer, 2010.