Obfuscating a Business Process by Splitting Its Logic with Fake Fragments for Securing a Multi-cloud Deployment

Amina Ahmed Nacer, Elio Goettelmann, Samir Youcef, Abdelkamel Tari, Claude Godart

Obfuscating a Business Process by Splitting its Logic with Fake Fragments for Securing a Multi-Cloud Deployment

Amina Ahmed Nacer[1,2], Elio Goettelmann[2,3], Samir Youcef[2], Abdelkamel Tari[1], Claude Godart[2]

[1]LIMED, Faculty of Exact Sciences, University of Bejaia, Algeria
{*amina.ahmed-nacer, samir.youcef, claude.godart*}*@loria.fr*

[2]LORIA, University of Lorraine, Nancy France
elio.goettelmann@list.lu

[3]LIST Luxembourg Institute of Technology Luxembourg
abdelkamel.tari@univ-bejaia.dz

***Abstract*—Companies are ready to outsource their business process to the cloud to enjoy its benefits. However they must be sure that their business know-how is preserved. Although some solutions already exist and consist in splitting the model into a collaboration of BP fragments for a deployment in a muti-cloud context, none of them takes into account the possibility of collusion between the clouds executing the different fragments. To address this issue, we propose in this paper a solution consisting in adding fake BP fragments at specific locations in the process, so as to complicate its structure and hide the direct interaction between clouds executing sensitive fragments. Therefore, the discovery of the process by malicious cloud providers is delayed. The approach is validated against an introduced metric. It demonstrates that our approach is better in the worst case than previous approaches in best cases.**

Index terms*— *Fake Fragments, Security Risk Management, Cloud Collusion.

I. Introduction

Cloud computing has emerged as a technology that has revolutionized the world of the internet. It avoids upfront infrastructure costs, and helps organizations to focus on their core business activities, instead of their system infrastructure.

In this context, companies are ready to outsource their business processes (BP) to the cloud in order to enjoy its benefits. However, the new security risks introduced by the cloud exposes the know-how of business activities, and ensuring the privacy of those organizations is becoming a great challenge.

One way for companies to prevent risks is to deploy their processes in a multi-cloud context by splitting their process models at premises, so that externalized BP fragments do not allow understanding critical parts of the business process and even less the whole process. This is connected to the idea of program obfuscation that makes code harder to understand for preventing code plagiarism.

In [12] we have proposed a design time approach for transforming a BP model into BP fragments, so that these BP fragments externalized in a multi-cloud context do not allow a cloud provider alone to understand a company know-how critical fragment.

However, this work provides mainly active support against one malicious cloud at a time, but does not explicitly address the risk of conspiracy of several malicious cloud providers, which could combine their local knowledge of the process model to discover larger critical know-how (even if nevertheless the simple splitting of a process renders such a conspiracy more difficult).

The objective of the work described in this paper is, based on the work in [12], to go one step further and to provide a more active support for struggling the conspiracy of several malicious cloud providers.

The approach is to increase the obfuscation level of BP models by introducing *fake BP fragments* in the BP model splitting algorithm in such a way that these fragments extend the number and length of paths between sensitive fragments, rendering malicious cloud collusion more difficult.

The rest of the paper is organized as follows: the next section explains our motivations, the context and the objective of this work. Section III presents our obfuscation methodology based on fake fragments. Section IV proposes metrics to take into account the introduced obfuscation artifacts. Section V evaluates our approach using these metrics. Section VI discusses the state of the art and finally section VII concludes and introduces some future work.

II. Motivations, Context and Objective of work

This section starts with a motivating example. Then, as this work comes ahead of previous ones, it situates its context and remind the necessary definitions. Finally, we refine our objective and approach.

A. Motivating example

Fig. 1 depicts a loan process in a bank which objective is to accept or reject a loan request. Depending on the customer history and other parameters (loan amount ...), the loan is treated in different ways. In all cases, the risk of the loan is evaluated, but the loan request can be either directly accepted or rejected. At any point in the process the hierarchy can directly intervene. The final decision is taken depending on the loan request treatment and the hierarchy validation.

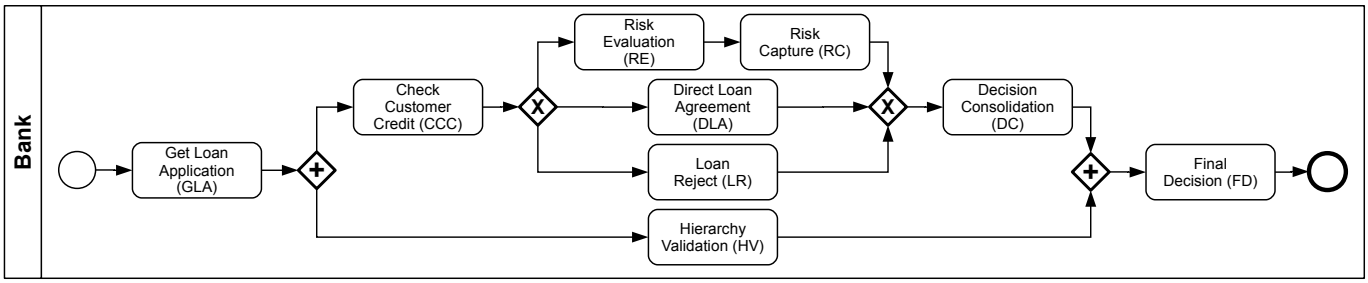

Fig. 1. The *Loan* Process (orchestration)

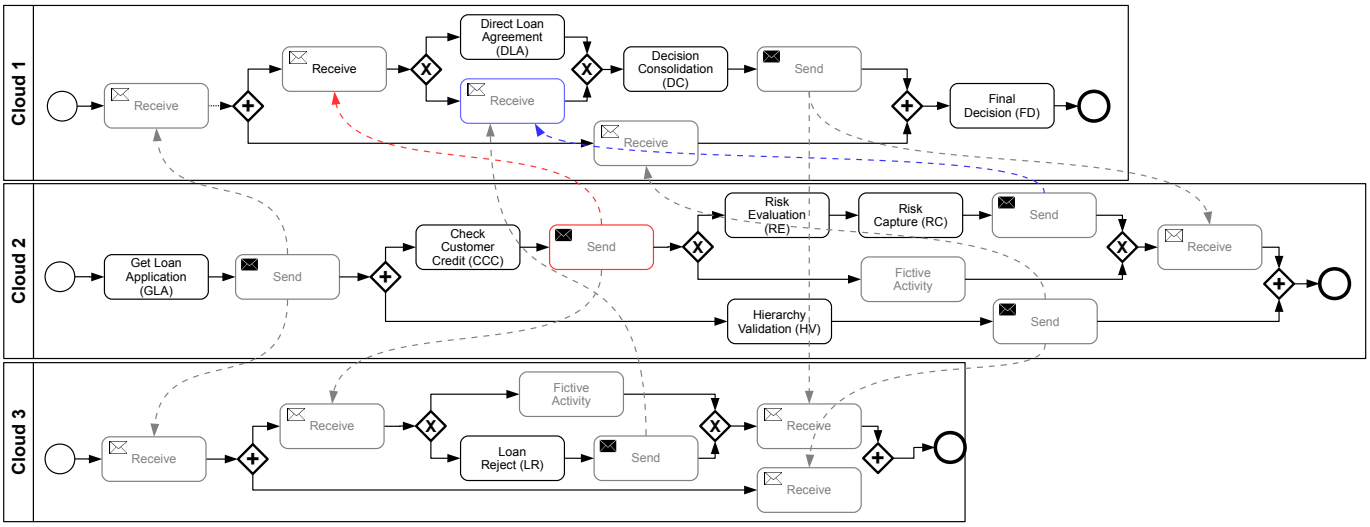

Fig. 2. The *Loan* Process as a collaboration of fragments (taking into account sensitivity of fragments)

The bank is ready to use cloud resources. However, it needs to be in confidence with its cloud providers, and especially to be sure that its strategy for directly accepting or rejecting a loan will not be disclosed. In the same way, it does not want to disclose how the hierarchy intervenes in the process and how the final decision is taken. One way for reaching this objective is to anticipate problems before they occur and the bank is ready to make some preliminary work in this direction.

For mitigating risks, the bank can obfuscate its BP by transforming its BP model using the principles and the methodology introduced in [14] and in [12] overviewed in the next section.

In addition, and it is the purpose of this paper, it can add nonfunctional fake BP fragments which role is to increase BP collaboration model complexity and BP obfuscation level, and to catch malicious behavior of providers.

B. BP model obfuscation

The objective of the obfuscation activity is to preserve the know-how formalized in BP models from the malicious clouds involved in the BP execution

While very few work has been done on this topic, we have proposed in [14] a general methodology for BP model obfuscation and in [12] an automated approach for splitting a BP model in a collaboration of BP fragments with the objective to support managers in the obfuscation task. They are overviewed below.

1) A General methodology for BP obfuscation: The principles are listed below; for more see [14]:

- *Retain sensitive data and logic at premises*. For example, in the *Loan* process example, designers can decide to maintain at premises *Check Customer Credit* and *Final Decision* tasks that contain important know-how.
- *Split BP logic into several BP fragment logics*: Another intuitive mean is to split the BP logic in several fragments and to distribute them to different cloud providers so-that a cloud provider has only a partial view of the whole logic. To maintain the semantic of the initial logic, these fragments are then weaved together to form a BP fragments collaboration (activities *send* and *receive* are used to manage communications between fragments and to support process fragments weaving).
- *Add non-functional logic*: Adding useless code in a program is a mean used by programmers for obfuscating it. In the same objective, useless BP fragments can be added. In this direction, we understood also that some service logic added for non-functional purpose (security, replication for verification, . . .) has also the property to increase the complexity of BP logic understanding.
- *Obfuscate data*: Data obfuscation is also largely used to obfuscate programs. Data cryptography and anonymization can be used to obfuscate process data. In the *Loan* process example, anonymizing customer information account when possible is a way to render the strategy of the bank for loan management more difficult to discover.
- *Separate logic and data*: The idea there is to store the logic and its related data in different places so that a

cloud cannot mine some links between data and logic by analyzing logic and data storage.

- *Split cases between clouds*: The objective is to split the process cases (instances) between the different clouds so that none of the clouds has enough data to mine the process logic (the number of different cases necessary to mine a BP model is easily calculable [20]).

The work described in this paper is mainly concerned with "Split BP logic into several BP fragment logics" and "Add non-functional logic" principles. Basically, the idea is to add fake BP fragments in the BP splitting activity.

2) BP splitting automation: The work in [14] remains at a methodological level and supposes that splitting decisions are taken "manually" by informed designers. In [12], we go one step further by proposing an algorithm for automating the splitting of a BP logic into several BP fragment logics.

Of course, one way to manage BP logic splitting is to generate aleatory collaborations, but, on the one hand, a first problem is to be able to choose a good collaboration in a large solution set, and on the other hand, if managers want supports in this activity, they want also that the generated collaboration remains readable to them, what risks not to be the case of an aleatory generation.

In [12], we propose an approach for automating the splitting of a BP logic into several BP fragment logics which maintain some readability of the collaboration model as it rests on a BPMN[1] modelling principle: to assign the more sensitive fragments to different clouds.

And the question turned in: *how to automatically identify these more sensitive fragments?* We identified that the more sensitive information of a process are genrally located in some specific BP fragments where important *decisions* and *syntheses* are made (see figure 3). Intuitively, *decisions* are taken in the fragments (blocks) preceding (x)or-split gateways triggering alternatives fragments. Respectively, *syntheses* are done in the fragments succeeding and-join gateways synchronizing several flows executing in parallel. In addition decisions and syntheses fragments are often complemented by an activity in the fragment following the (x)or-join gateway closing the opening (x)or-split (our algorithm supposes well-structured process ([18]), and syntheses are often prepared in the fragment preceding the opening and-split gateway corresponding to the closing and-join gateway: we consider that these *complementing fragments* are also often sensitive. In addition, decisions are implicitly existing in, and thus can be mined from, the flows arising from the corresponding (x)or-split gateway (called *alternative fragments* in the following).

These fragments ($decision$, $decision\ complementing$, $synthesis$ and $synthesis\ complementing$, $alternative\ fragment$), once identified, are assigned to different clouds.

This algorithm is described in detail in [12]. Applied to our motivating example, it generates the collaboration described in figure 2.

3) BP model obfuscation metric: To validate this work by comparing the solution generated by our algorithm to other solutions, we have developed a metric for measuring the level of obfuscation of a collaboration.

In a first approach, we have considered that the level of obfuscation of a BP model is directly correlated to the complexity of its logic: the more it is complex and the more it is obfuscated. As very few complexity metrics of BP models exist and as we were not able to find an existing obfuscation metric for collaborations, we have developed ours, described and justified in [1].

This metric is based on the Interface Complexity metric[2] [5] developed to evaluate the complexity of a BP logic . We calculated the complexity of a collaboration (CCOM) as follows:

$$CCOM = NoC \times \prod_i (mess\_in_i \times mess\_out_i)^2 \quad (1)$$

Where:

- NoC is the number of clouds
- $mess\_in_i$ the number of input messages of the i_th fragment
- $mess\_out_i$ the number of output messages of the i_th fragment

C. Objective and approach of the current work

While our previous work provides active support against one malicious cloud at a time, it does not directly addresses the risk of conspiracy of several malicious cloud providers, which could combine their local knowledge of the process model to discover larger critical know-how (even if nevertheless the splitting of a process renders such a conspiracy more difficult).

The objective of the work described in this paper is to provide a more active support for struggling the conspiracy of several malicious cloud providers.

Specifically, we are concerned with the case of collusion between some of the clouds executing sensitive fragments, i.e. $decision$, $decision\ complementing$, $synthesis$, $synthesis\ complementing$, and $alternative\ fragment$, violating therefore our separation principle. In this objective, our strategy is to increase the difficulty for a cloud provider to understand who the provider(s) with complementing knowledge is (are).

Our approach consists in inserting in strategic points of a BP collaboration fake fragments to:

1) increase the number of possible paths between complementing sensitive fragments
2) increase the length of paths separating complementing sensitive fragments.

This approach implies cloud interactions to build a collusion, and respectively decreases for a malicious cloud, the likelihood of selecting quickly a path towards complementing fragments.

[1]In all this work, we use BPMN (Business Process Modelling Notation) for modelling process (www.bpmn.org)

[2]Interface Complexity Metric: $IC = Length \times (number\ of\ inputs \times number\ of\ outputs)^2$ where *Length* is the number of possible paths of the activity

Before going later, we must underline that this does not completely prevent the collusion of malicious clouds, but it renders much harder and greatly delays such a collusion success.

This approach is deepened in the next section.

III. Using fake BP fragments as an obfuscation and security mechanism

This section is organized around the three following questions:

- when to insert fake fragments?
- where to insert them?
- how to deploy them in a multi-cloud?

A. When to consider fake fragments?

More precisely, the question is: do we add fake fragments in the initial BP logic before it is split? Or do we add them directly in the BP fragments collaboration resulting from the initial obfuscation of this first model? Back to our example, do we insert them in the model in figure 1 or in its obfuscated version in figure 2?

Our choice is to insert fake fragments as tasks in the initial BP logic, for the following reasons:

- If inserting such fragments can be almost completely automatized, operating at the BP logic level allows designers to maintain some control if they want,
- If, as developed below, preventing a coalition is mainly related to interaction between clouds deploying sensitive fragments, this can be managed at an abstract level with, on the one hand, a kind of design patterns (see section III-B) and, on the other hand, deployment rules imposed to fake and sensitive fragments (see III-C).

B. Where to insert fake fragments?

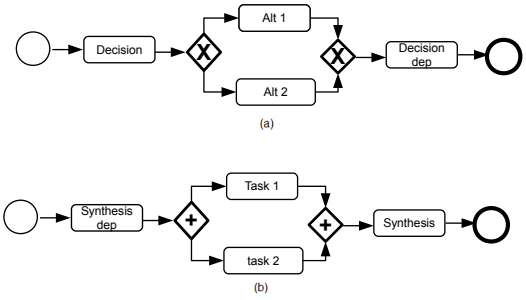

Fig. 3. Decision and Synthesis sensitive activities

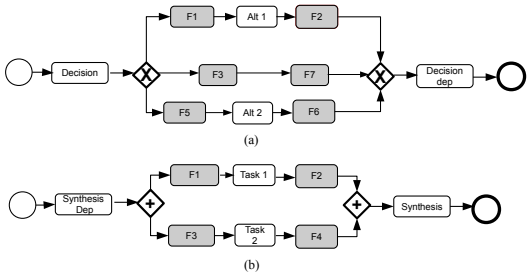

Fig. 4. Decision and Synthesis (sub-)processes with fake fragments

The main principles are:

- The fragments with the more sensitive information are, on the one hand the fragments to protect in first place, but also the fragments to control most actively because the more dangerous if their hosting clouds turn to be malicious.
- As a consequence, we must increase the difficulty for clouds managing complementing sensitive fragments to form a coalition by increasing the length of the paths between them, and the number of such paths.

In this objective, we have abstracted the following patterns:

a) Fake fragment pattern:

Add a fake fragment at the beginning and the end of each path between a (X)OR-split (respectively And-Split) and the closing (X)-OR join (respectively And-join) gateway to increase decisions (respectively synthesis) protection (see figure 4).

b) Fake path pattern:

Add a new fake alternative between each (X)OR-split and its (X)-OR join (see figure 4-b).

Applying these patterns is not mandatory: as fake fragments do not change the semantics of the collaboration with regards to the business, they can be added in any place. However, inserting fake fragments as preconized asserts an optimized distribution with regards to an aleatory distribution with the same number of fragments: in terms of cost, it allows to better cover the different paths and, in term of security, it better reduces risks.

Applying these patterns, figure 5 represents our motivating example of figure 1 after inserting fake fragments in the BP logic.

C. How to deploy fake fragments?

More precisely, the question is: to which clouds to assign fake fragments?

Indeed, our deployment process is governed by functional (service provisioning ...) and non-functional (cost, security risk, performance ...) constraints [12]. Especially, one generic security constraint is to assign sensitive fragments to different clouds. In fact, taking into account the management of fake fragments generates additional constraints which must be consistent with the previous constraints.

Taking into account this context, we recommend:

- To separate fake fragments and sensitive fragments in different clouds
- In the ideal (no other functional and nonfunctional constraints), to assign the different fake fragments to different clouds.
- In case of functional and nonfunctional constraints, to priviledge as much as possible
  - the assignment to different clouds of the fake fragments on the same path between two sensitive fragments ,
  - the assignment to different clouds of a sensitive fragment and the fake fragment which directly precedes or follows it

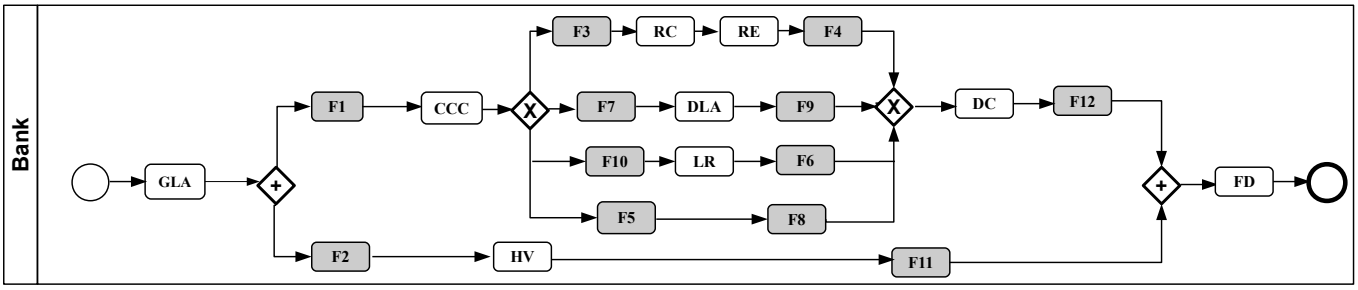

Fig. 5. BP collaboration corresponding to *loan process* with fake fragments

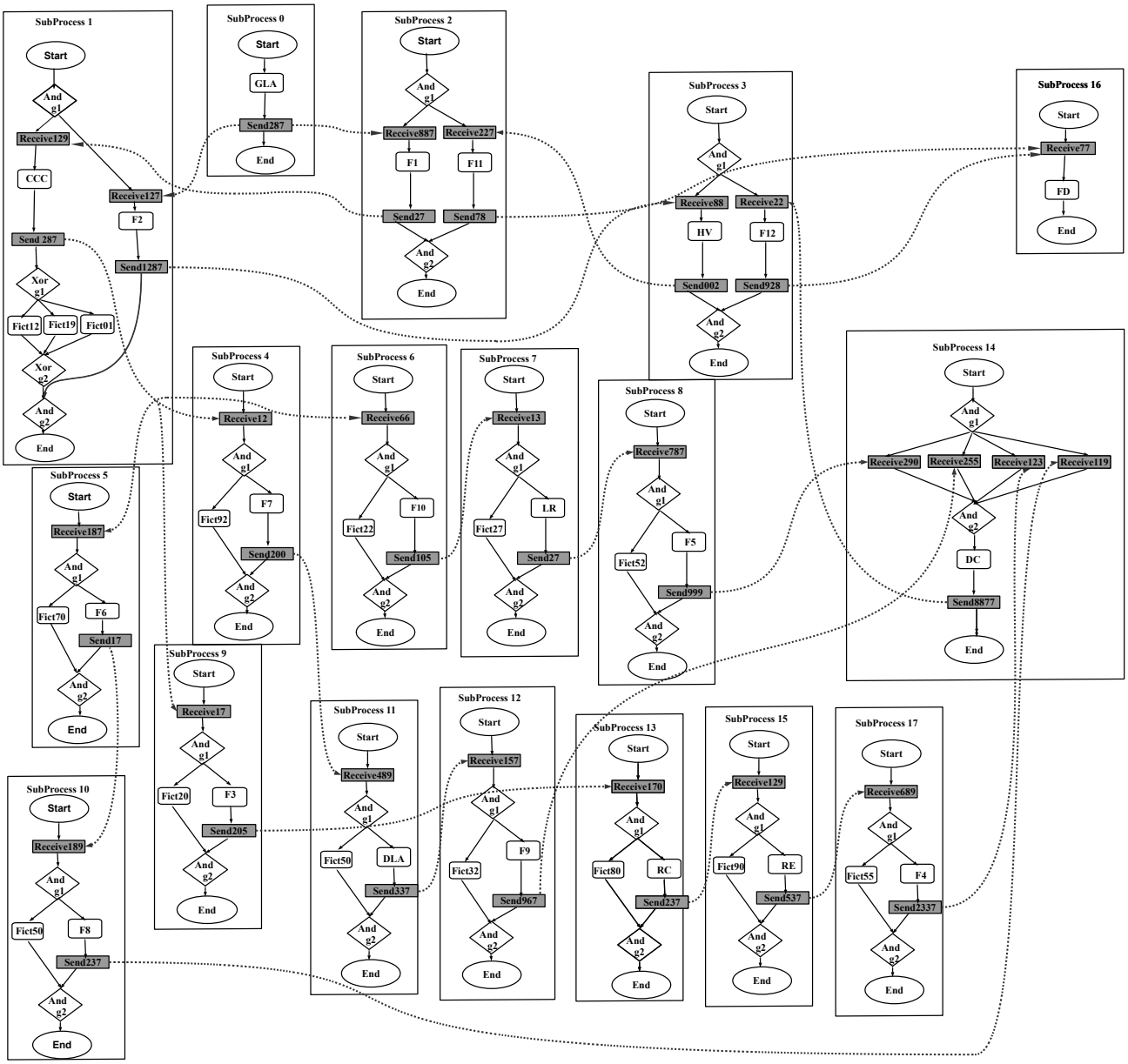

Fig. 6. The BP collaboration corresponding to *the loan process with fake fragments* deployment

Figure 6 represents a deployment of our motivating example after applying these rules. As seen, compared to the deployment of figure 2, it allows to complexify the interaction linking sensitive fragments, resulting in longer paths separating them. Therefore, if a cloud deploying a sensitive fragment turns to be malicious, it needs more cloud interactions to collude.

IV. Obfuscation Metric

A. Introduction

The objective of this section is to evaluate the impact of introducing fake fragments on business process obfuscation.

A first question is which metric to use? We have decided:

- to hold our initial metric measuring a *BP collaboration model complexity* introduced in section II-B3,
- to add a new metric for measuring the *collusion complexity* of a BP, or how it is complex for cloud providers to form a coalition (see section IV-B).

Of course a question that arises at this point is why two metrics and not just one? And what is the position of each metric with regards to the other?

Clearly, adding fake fragments increases the *BP collaboration model* as it increases the NoC (number of clouds), $mess\_in_i$ (number of receive messages) and $mess\_out_i$ (number of send messages) factors.

As such one could content itself with this only metric. However, *BP model complexity* does not take into account that we want to increase this complexity on special paths, i.e. between sensitive fragments. It is why we have introduced the *collusion complexity* metric measuring the length of paths between sensitive fragments as a quantity of work to do by cloud providers to collude.

We had also a reflection about how to combine these two metrics in one, but on the one hand it was not clear how to integrate the notion of a sensitive fragment in our initial *BP model complexity* metric, and on the other hand, given the context, one could be interested in comparing BP deployment solution with this two metrics, in a Pareto style (just comparing solutions that dominate others with regards to the first or the second metric [6]).

B. Collusion complexity metric

In fact, the *Collusion complexity metric* is a collection of three metrics: $AverageLength_{Coalition}$, $LongestLength_{Coalition}$ and $ShortestLength_{Coalition}$.

It allows also to refine the comparison of different solutions while giving to designers more decision information.

a) definition 1 $AverageLength_{Coalition}$*:* is a metric of the average effort a malicious cloud has to do for establishing a collusion between two sensitive fragments of the deployed BP.

$$AverageLength_{Coalition} = AVG(LPC_i)_{\forall i \in P} \quad (2)$$

with:

P: the set of all paths linking sensitive fragments.

AVG: The function average that returns the average length of a coalition.

LPC_i: the number of clouds on path i of P.

b) definition 2 $LongestLength_{Coalition}$*:* is a metric of the more important effort a malicious cloud has to do for establishing a collusion between two sensitive fragments of the deployed BP.

$$LongestLength_{Coalition} = Max(LPC_i)_{\forall i \in P} \quad (3)$$

Where the Max function returns the maximum of LPC_i, i.e. the longest path between sensitive fragments.

c) definition 3 $ShortestLength_{Coalition}$*:* is a metric of the less important effort a malicious cloud has to do for establishing a collusion between two sensitive fragments of the deployed BP.

$$Shortest_{Coalition} = Min(LPC_i)_{\forall i \in P} \quad (4)$$

Where the Min function returns the minimum of LPC_i, i.e. the shortest path between sensitive fragments.

V. Evaluation

This section validates our results with regards to the *collaboration model complexity* metric and the *collusion complexity metric*

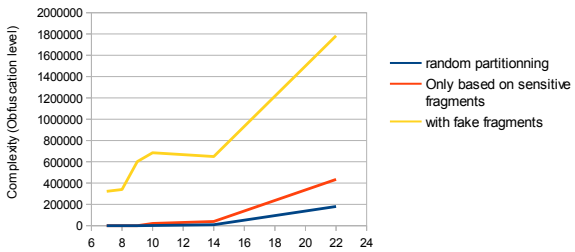

Fig. 7. Complexity regarding our three approaches

A. Experimental environment

All algorithms were coded in java language. All experiments were performed on an intel (R) Core(TM) i3- 2310M 2.10 GHz running Windows Seven.

B. Experimental settings

To evaluate our approach, we have compared BP collaborations generated in three different ways:

- an aleatory distribution
- by splitting a BP model based on sensitive fragments separation only
- by splitting a BP model based on sensitive fragments separation extended with fake fragments (the approach developed in this paper).

We have supposed that we have no functional and nonfunctional constraints, as their impact is orthogonal to our splitting nodes (or impact in the same way the three approaches).

In order to validate the correctness of our assumptions, we generate until 1000 instances of collaborations for each used business process and for each way of distribution.

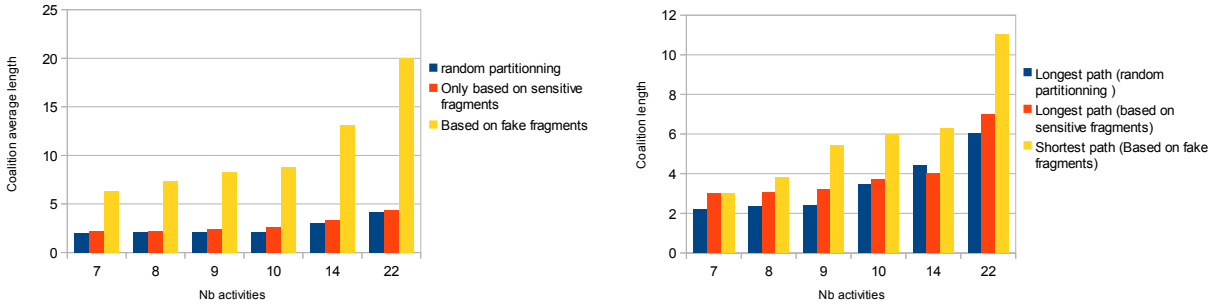

Fig. 8. The resulting obfuscated levels based on Collusion Complexity Metric

C. Collaboration model complexity

Figure 7 illustrates the complexity of collaboration models with regards to our three experimentation sets. As forecasted, the fake mechanism increases model complexity for processes of more than 14 activities in a linear way.

This confirms as expected that collaboration complexity, and the obfuscation level obtained by our new approach is always better than the obfuscation level obtained by a random partitioning, or a splitting only based on sensitive fragments.

In fact, fake fragments increase, on the one hand, the number of cloud providers and on the other hand generate a more important number of exchange messages between the different fragments of the collaboration, i.e. the factors of the collaboration model complexity.

We can note an anomaly in figure 7 for a process with 14 activties which has a very simple logic with few decisions and synthesis, resulting then in a lower obfuscation level.

In fact the obfuscation level of a process does not depend only on the process length, but also on the structure of the process (number of gateways, type of the gateways...etc). Indeed, as our solution is mainly based on protecting against cloud deloying sensitive fragments (generally preceding or following a gateway), we add more fake fragments on process containing more gateways, thereby increasing the number of cloud providers and exchange messages, resulting then in a more obfuscated level according to the collaboration complexity metric.

D. Collusion complexity metric

We report here on the *Collusion complexity*, i.e. the difficulty for cloud providers to form a coalition, using the $AverageLength_{Coalition}$ metric, the $ShortestLength_{Coalition}$, and the $LongestLength_{Coalition}$ metrics introduced above.

Figure 8-a depicts the results of our experiments for the $AverageLength_{Coalition}$ metric, which calculates the average length of the paths between the sensitive fragments of a BP collaboration model, for the three sets of collaboration models.

Figure 8-b depicts the results of our experiments for the $ShortestLength_{Coalition}$ and the $LongestLength_{Coalition}$ metrics, which calculate respectively the shortest path for our new mechanism and the longest path between the sensitive fragments of a BP collaboration model for the two other sets of collaboration models.

These results also confirm experimentally our hypotheses. As seen in figure 8-a, our mechanisms offers always a longer path seperating sensitive fragments (proportional to the obfuscation level) than aleatory distribution and distributions based on sensitive fragments.

In addition, according to the results of figure 8-b, the shortest path for our mechanism offers a better obfuscation level than the longest path of the other ways of distribution. In other words, the worst case (the more dangerous) with fake fragments is better than the best case (the less dangerous) without fake fragments.

VI. State of the art

As introduced above, this work is developed in the vein of [15], [14], but these propositions remain at the level of principles and methodologies. In [12], we go one step further by providing a semi-automatic approach for splitting a BP model into a BP fragments collaboration. However, we do not directly address the risk of conspiracy of several malicious cloud providers. This paper addresses this topic by introducing fake BP fragments in the splitting process.

Directly addressing BP model obfuscation, [11] generates obfuscated BP models, but with a different purpose (model sharing and analysis) and techniques which are not adapted to our context. More related to our work, [3] is interested in privacy preserving of BP fragment model provenance in the cloud using statistical techniques, but it is not concerned with the splitting of BP models into fragments phase.

More closely related to our topic, several works [7], [8], [13] are concerned with privacy preserving of different properties of a BP but they apply at the level of BP configuration not at the design level. Other works are concerned with process model splitting and process fragments weaving [16], [22], [9] but in the objective of meeting functional needs, not considering the privacy dimension.

The idea of a fake fragment is directly inspired from useless code for obfuscating a software [2], but we are, to our knowledge, the first using it for supporting the privacy of a BP collaboration.

Concerning metrics, our reflection has been influenced by previous work in software engineering for measuring software components complexity, with a focus on control flow oriented metrics considering programming in the large, characteristic of BP models (typically the Mac Cabe Cyclomatic Complexity [17] and the Henry and Kafura Information Flow

metric [19]). Based on this metric and BP oriented, the Interface Complexity metric [5] has also founded our reflection and we reuse it as a component of our metric. Different from the software engineering approach, concerned with software structure, the cognitive approach is centered on human cognitive processes [4]. In this vein, [10] introduces some measures of BP mode comprehensibility of a BP model using the Process Structure Tree (PST) representation [21], the same representation we use in our proper work [12]

VII. Conclusion

Preserving the know-how implemented in its processes is of a major importance before an entreprise can accept to deploy them in the cloud. In this context, we proposed in this paper a solution which is considered as being an extension of our previous work in [12] which provides mainly active support against one malicious cloud at a time, but does not completely addresses the risk of conspiracy of several malicious cloud providers.

As a contribution to this topic, our new mechanism consists in adding fake BP fragments at specific location in the process, so as to complicate its structure and hide the direct interaction between clouds executing sensitive fragments, delaying therefore the discovery of the process by malicious cloud providers.

One can consider that adding fake fragments can be a costly solution. But, on the one hand this can be the price to pay for the preservation of know-how, and on the other hand this can be reduced by our recommendations, which managed strategically fragments deployment.

Regarding future works, we are enhancing our optimization algorithm to better select a cloud configuration, taking into consideration the introduction of fake fragments, but in a complete context, with cost (how fake fragments increase cost?), performances (how fake fragments decrease performance?), risks (how fake fragments decrease security risks? What requests obfuscation to be quantified and be introduced in our risk model)... constraints. We are also developing a mechanism for tracing responsibilities and limiting impact in case of attack.

References

[1] A. Ahmed Nacer, E. Goettelmann, S. Youcef, A. Tari, and C. Godart. A design-time semi-automatic approach for obfuscating a business process model in a trusted multi-cloud deployment. In *Submitted to the international journal of web services research*, page 13. IEEE, 2015.

[2] P. Beaucamps and E. Filiol. On the possibility of practically obfuscating programs towards a unified perspective of code protection. *Journal in Computer Virology*, 3(1):3–21, 2007.

[3] M. Bentounsi, S. Benbernou, C. S. Deme, and M. J. Atallah. Anonyfrag: an anonymization-based approach for privacy-preserving bpaas. In *1st International Workshop on Cloud Intelligence (colocated with VLDB 2012), Cloud-I '12, Istanbul, Turkey, August 31, 2012*, page 9, 2012.

[4] S. N. Cant, D. R. Jeffery, and B. Henderson-Sellers. A conceptual model of cognitive complexity of elements of the programming process. *Information & Software Technology*, 37(7):351–362, 1995.

[5] J. Cardoso, J. Mendling, G. Neumann, and H. A. Reijers. A discourse on complexity of process models. In *Business Process Management Workshops, BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, Vienna, Austria, September 4-7, 2006, Proceedings*, pages 117–128, 2006.

[6] Y. Censor. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization*, 4(1):41–59, 1977.

[7] R. Conforti, M. D. Leoni, M. L. Rosa, and W. M. V. D. Aalst. Supporting risk-informed decisions during business process execution. In C. Salinesi, M. Norrie, and O. Pastor, editors, *Advanced Information Systems Engineering*, volume 7908 of *Lecture Notes in Computer Science*, pages 116–132. Springer Berlin Heidelberg, 2013.

[8] E. F. Duipmans, L. F. Pires, and L. O. B. da Silva Santos. A transformation-based approach to business process management in the cloud. *J. Grid Comput.*, 12(2):191–219, 2014.

[9] W. Fdhila, U. Yildiz, and C. Godart. A flexible approach for automatic process decentralization using dependency tables. In *IEEE International Conference on Web Services, ICWS 2009, Los Angeles, CA, USA, 6-10 July 2009*, pages 847–855, 2009.

[10] K. Figl and R. Laue. Cognitive complexity in business process modeling. In *Advanced Information Systems Engineering - 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings*, pages 452–466, 2011.

[11] H.-G. Fill. Using obfuscating transformations for supporting the sharing and analysis of conceptual models. In S. Robra-Bissantz and D. Mattfeld, editors, *Multikonferenz Wirtschaftsinformatik 2012 - Teilkonferenz Modellierung betrieblicher Informationssysteme*, Braunschweig, 2012. GITO Verlag.

[12] E. Goettelmann, A. Ahmed-Nacer, S. Youcef, and C. Godart. Paving the way towards semi-automatic design-time business process model obfuscation. In *Web Services (ICWS), 2015 IEEE International Conference on*, pages 559–566. IEEE, 2015.

[13] E. Goettelmann, K. Dahman, B. Gâteau, E. Dubois, and C. Godart. A security risk assessment model for business process deployment in the cloud. In *IEEE International Conference on Services Computing, SCC 2014, Anchorage, AK, USA, June 27 - July 2, 2014*, pages 307–314, 2014.

[14] E. Goettelmann, N. Mayer, and C. Godart. A general approach for a trusted deployment of a business process in clouds. In *Fifth International Conference on Management of Emergent Digital EcoSystems, MEDES '13, Luxembourg, Luxembourg, October 29-31, 2013*, pages 92–99, 2013.

[15] M. Jensen, J. Schwenk, J. Bohli, N. Gruschka, and L. Iacono. Security prospects through cloud computing by adopting multiple clouds. In *CLOUD'11*, pages 565 –572, 2011.

[16] R. Khalaf, O. Kopp, and F. Leymann. Maintaining data dependencies across BPEL process fragments. In *Service-Oriented Computing - ICSOC 2007, Fifth International Conference, Vienna, Austria, September 17-20, 2007, Proceedings*, pages 207–219, 2007.

[17] T. J. McCabe. A complexity measure. *IEEE Trans. Software Eng.*, 2(4):308–320, 1976.

[18] A. Polyvyanyy, L. Garca-Bauelos, and M. Dumas. Structuring acyclic process models. *Information Systems*, 37(6):518 – 538, 2012. BPM 2010.

[19] H. Sallie and K. Dennis. Software structure metrics based on information flow. *IEE Trans. on Software Engineering*, 7(5), 1981.

[20] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: A survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267, 2003.

[21] J. Vanhatalo, H. Völzer, and J. Koehler. The refined process structure tree. *Data Knowl. Eng.*, 68(9):793–818, 2009.

[22] U. Yildiz and C. Godart. Dynamic decentralized service orchestrations. In *WEBIST 2007 - Proceedings of the Third International Conference on Web Information Systems and Technologies, Volume IT, Barcelona, Spain, March 3-6, 2007.*, pages 36–45, 2007.