Toward an Anonymous Process Mining

Andrea Burattin
Member, IEEE
University of Innsbruck

Mauro Conti
Senior Member, IEEE
University of Padua

Daniele Turato
Siav SpA

***Abstract*— Process mining is a modern family of techniques applied to datasets generated from business processes run in organizations, in order to improve and obtain useful insights and performance measurements on the processes themselves (with clear societal and economical benefits). While these techniques are very promising in understanding business processes, their complete and efficient implementation inside the organizations is often not possible. Hence, in a way similar to what is done for most non core activities, and in particular for most ICT services, companies evaluate the possibility of outsourcing such task. However, the confidentiality of the dataset related to the business processes are often key assets for most of modern companies. Then, in order to avoid threats that might come from disclosing such information, most companies decide not to benefit from these process mining techniques.**

In this work, we propose a possible approach toward a complete solution which allows outsourcing of Process Mining without thwarting the confidentiality of the dataset and processes. Furthermore, we provide a prototype implementation of our proposed approach and run several experiments that confirmed the feasibility of our approach. We believe the one highlighted in this paper is an important direction to work on, in order to remove the obstacles that prevent companies to fully benefit from outsourcing process mining.

***Index Terms*— Data extraction, Anonymization, Process mining, Privacy.**

I. Introduction

A business process [1] is defined as a collection of activities or tasks that have different dependencies each other. Each of these activity can be identified with an atomic event, which is executed by a certain person, or may be an automated task (i.e., it is executed by a system). All business process instances (also called "cases") have some associated data stored in a database, and the execution of a task involves reading, writing or updating one or more of its entries. When a task ends, the dependency relationships with other assets determine the set of additional tasks that can be performed. When there are no more executable tasks for a given instance, we can consider it as terminated.

Not all companies adopt a "process oriented working structure". However, both for legal and practical reasons, almost all of them take advantage of information systems to support production and administration departments. In fact, these activities are often supported by the use of business management and information sharing tools. These systems are based on the recording of a large amount of descriptive information (e.g., who has performed a certain action) and temporal data (e.g., timestamp of the action performed) on different databases. These information could serve as input for techniques able to reconstruct the processes actually taking place within the company.

In more detail, these recordings describe each step of the instances of a certain process. An instance can be seen as a single sequence of events, associated with the corresponding activity of the process: in which the first event is associated with the initial activity of the process, and the last with the final one (unless the instance is incomplete). Each log entry can contain many information such as: the type of recorded event (*event type*); the starting and ending time (*timestamp*) of the task; the person/machine in charge of executing it (*originator*). These logs are then converted into different formats (usually XML-based) in order to be processed by means of process mining techniques [2], [3].

Fig. 1 is an effective summarization of the idea behind process mining [4]: all assets involved in the company business are supported by information systems, producing a series of event logs. We can use them by applying discovery techniques to derive models that describe the company's processes. There are many formalisms useful to represent a process, one of the most widely used is BPMN (Business Process Modeling Notation) [5]. We can use these models to apply conformance checking techniques, enabling us to verify whether a series of logs can be generated from the analyzed model, and to detect deviations from the model itself. The results of analysis can trigger alerts, in order to warn users about abnormal situations, or can be useful to update the models. Additionally, as new data flow into the models through traces execution on the model itself, we can collect and update a series of performance indicators (e.g., task execution times, frequencies of certain events or events series, and associated costs) which can be very useful in order to enrich the process model itself with additional perspectives (extension).

Despite the size of a company, a process oriented organization can help increasing the productivity and minimizing wastes, therefore the development of tools and methods that can be exploited in a real enterprise environment becomes crucial. For years, process mining has been relegated to a purely academic setting, but more recently this trend is changing (see, for example, [7], [8]), thanks to the growth and the strengthening of interest on efficiency and how a process oriented organization can enable a company to achieve it.

a) Problem Statement: By definition, process mining operates on business processes, the "heart of the company",

A. Burattin (*andrea.burattin@uibk.ac.at*) is with the University of Innsbruck, Austria. M. Conti (*conti@math.unipd.it*) is with the Department of Mathematics, University of Padua, Italy. D. Turato (*daniele.turato@siav.it*) is with Siav Spa, Italy.

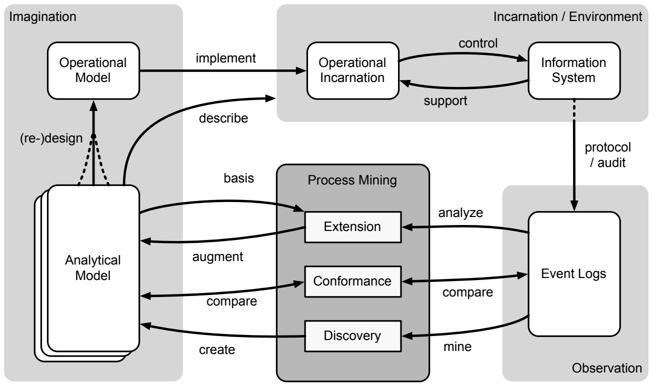

Fig. 1: The process mining "ecosystem" [6].

and, for this reason, it involves the most sensitive data stored in its information systems. During the procedure that goes from data extraction and transformation, through log analysis, to the visualization of results by analysts and managers, the process mining method involves all kinds of people inside a company, with very different responsibilities and data access levels. Since all data connected to process models represent real business assets, the leakage of such information may result in important losses or costs (for example, due to lost competitive advantage) [9], [10]. It is therefore important to protect the data during the analysis phase, in order to conform them to security policies, especially if the process improvement mission is carried on by external consultants.

b) Contribution: The aim of this paper is twofold. On one hand, we aim to rise awareness, on the process analysts community, with respect to the fundamental importance for some kind of data protection policies. On the other hand we present an approach capable of providing some very basic data hiding, without invalidating all process mining approaches already available. An evaluation of this approach, implemented in ProM and tested in real-world scenarios, is proposed as well.

The rest of this paper is structured as follows: Section II reports the fundamental data required for process mining purposes. Section III describes the data encryption approach that we used. Section IV contains the description of our data extraction and encryption architecture. Finally, Section V concludes the paper and draws some possible future work.

II. Purposes of Data for Process Mining

A key point underpinning all process mining techniques is that most of the work is performed with the support of information systems. In fact, these systems keep traces of all the performed activities (together with additional with information) in the so called "event log files". These files are the basic input for most process mining techniques, since they represent a snapshot of the actual performed activities. For example, process mining algorithms use this information to reconstruct the actual control-flow of the work [11], [12], [13], [14], or the social network with interactions among different employees [15]. Although it is not possible to fix *a priori* the minimum set of required information to perform process mining analysis (e.g., control-flow analysis tasks do not require information on event originators) there is one mandatory information that is required for all types of analysis: the *case id*. This field is necessary to group together all events that belong to the same process instance.

Case Id	**Activity**	**Originator**	**Time**	**Cost (€)**
C_1	A	U_1	2015-01-01	160
C_2	A	U_1	2015-01-02	160
C_1	B	U_1	2015-01-02	100
C_2	B	U_1	2015-01-03	100
C_1	C	U_2	2015-01-03	150
C_1	E	U_2	2015-01-04	200
C_2	D	U_3	2015-01-04	175
C_2	E	U_3	2015-01-05	300

(a) Standard event log table. Events are sorted according to their time.

Activity	**Originator**	**Time**	**Cost (€)**
Case Id: C_1			
A	U_1	2015-01-01	160
B	U_1	2015-01-02	100
C	U_2	2015-01-03	150
E	U_2	2015-01-04	200
Case Id: C_2			
A	U_1	2015-01-02	160
B	U_1	2015-01-03	100
D	U_3	2015-01-04	175
E	U_3	2015-01-05	300

(b) Event log table, with events grouped according to their case id.

TABLE I: Small event log fragment with eight events. Table (a) reports the *raw* event table. Please note that, in this case, different process instances may interleave. Table (b) groups events according to their case id.

However, the *case id* field is not enough to do any specific analysis. So, let us assume to focus on control-flow discovery, social network analysis and performance analysis. Then, the fields that our log might store are: *(i)* the name of the activity that the events is referring to; *(ii)* the time when the activity has been performed, and its possible duration (if the event is referring to a non-instantaneous activity registration); *(iii)* the originator, which is the identifier of the person (or system) that performed the work.

For example, let us consider the small event log fragment reported in Table I. This log is made of eight events and, as table (a) presents, it contains two distinct case ids (i.e., C_1 and C_2) and five different activities (i.e., A, B, C, D and E). All these activities are performed by three distinct users (i.e., U_1, U_2 and U_3). This case also reports an additional data field (i.e., *cost*) which contains some cost information associated to each event execution that might be useful for some cost analyses and Service Level Agreements (SLAs) monitoring.

Event logs, as the one reported in the previous example, can now be saved into XML files using the XES Standard

[16] (the OpenXES libraries[1] are Java libraries that allows an easy handling of this type of files). The files generated using OpenXES are standard XML files where all events belonging to the same process instance are grouped into the same `trace` element. Each `event` is allowed to contain several other children: one for each attribute. Each child attribute contains also information on the attribute type.

XES is actually an extensible language and allows for new extensions capturing different perspectives. In this work, we will assume the "standard" set of extensions [16], the most frequently used for process mining purposes.

III. Building Blocks

In this section we introduce two cryptographic building blocks used in our proposal: AES and Paillier. AES cryptosystem [17], [18], the *de facto* encryption standard for commercial transactions, has been chosen for enciphering string attributes of event logs. Widespread use and efficiency, along with an optimal degree of security were the main reasons behind this choice. Concerning the encryption of numerical attributes we have opted for the use of Paillier [19], [20], which is an additive homomorphic cryptosystem. Homomorphic encryption allows to compute operations on encrypted operands and generate a result that, once decrypted, is equal to what we would have obtained by executing the same operation on decrypted operands. In other words, given two numbers $E(A)$ and $E(B)$ (encrypted with the same homomorphic algorithm starting from two plain numbers A and B, respectively) it is possible to calculate the encryption of the sum of A and B by adding $E(A)$ and $E(B)$ directly: the decryption of the obtained result is the sum of A and B. Executing the computations directly on encrypted data permits to execute analysis on data while keeping its security and save time and computational resources involved in encryption/decryption phases.

This section provides a brief description of the two encryption approaches that we adopted on our approach.

A. Advanced Encryption Standard (AES)

The first encryption procedure we are going to use is the Advanced Encryption Standard (AES) [17]. AES is a block-based algorithm which has been adopted as standard by the USA government. It is based on the cryptosystem Rijndael, proposed by Daemen and Rijmen in 1999.

Rijndael is a substitutions and permutation based cryptosystem which implements the Shannon's cryptographic principle "confusion and diffusion". One of the peculiarities of AES is its speed: it is particularly fast both in software and hardware implementations; moreover, it is easy to implement, not eager in memory; and guarantees a good security level. All these reasons allow Rijndael to be preferred with respect to other competitors.

AES uses four different functions:

1) `SubBytes`: non linear replacement of all bytes with values coming from a specific table;
2) `ShiftRows`: bytes shifting of a certain number of positions, depending on each byte row;
3) `MixColumns`: bytes linear combination, one column per time;
4) `AddRoundKey`: each table byte is combined with the session key (which is computed inside this function).

[1]See `http://www.xes-standard.org/openxes/`.

Algorithm 1: AES encryption.

```
Input: $M$ input message
Output: The encrypted message
1  $state \leftarrow M$
2  $state \leftarrow$ AddRoundKey($state$)
3  for $round \leftarrow 0$ to $NumberOfRounds$ do
4      $state \leftarrow$ SubBytes($state$)
5      $state \leftarrow$ ShiftRows($state$)
6      if $round < NumberOfRounds$ then
7          $state \leftarrow$ MixColumns($state$)
8      end
9      $state \leftarrow$ AddRoundKey($state$)
10 end
11 return $state$
```

Algorithm 1 describes the AES encryption procedure. Specifically, the input, which is required to be a 4×4 matrix, is immediately assigned to the *state* variable. This variable is going to be modified several times and, at the end of the procedure, it will contain the encrypted message. AES consists of 10 cycles (i.e., the $NumberOfRounds$ variable) which repeat the application of the abovementioned functions. All cycles are equal, except for the last one, in which the `MixColumns` function is not executed. The value of the $NumberOfRounds$ variable depends on the key length: it is 10 if the key is 128 bits long, 12 if 192 bits, and 14 if the key is 256 bits long.

Since all the functions involved in the encryption phase are invertible, in order to decrypt a text, it is possible to execute the operations described above in reverse order, changing each of them with the corresponding inverse. It is also necessary to reverse the session keys order.

B. Paillier Cryptosystem

The Paillier cryptosystem is a public key cryptosystem, based on the assumption that the problem of computing n-th residue classes is computationally difficult (decisional composite residuosity assumption or DCRA). In other words, given a composite n and an integer $y \in \mathbb{Z}^*_{n^2}$, it is hard to decide whether z is a n-residue modulo n^2 or not, i.e., whether there exists y such that: $z \equiv y^n \pmod{n^2}$.

The public/private key generation procedure, necessary to encrypt any value, is reported in Algorithm 2. It can be summarized by randomly choosing two large prime numbers p and q, independent of each other such that $\gcd(pq, (p-1)(q-1)) = 1$ (for practical adoption, a 160 bit size is recommended , in order to avoid Baby-step giant-step attacks). After that, it is computed $n = pq$ and $\lambda = \operatorname{lcm}(p-1, q-1)$

Algorithm 2: Paillier keys generation.

```
1 repeat
2     Generate $p$ and $q$ be large prime numbers s.t. $\gcd(pq, (p-1)(q-1)) = 1$
3     $n \leftarrow pq$
4     $\lambda \leftarrow \mathrm{lcm}(p-1, q-1)$
5     Randomly select $g$ with $g \in \mathbb{Z}^*_{n^2}$
6 until $\gcd\left(L(g^\lambda \mod n^2), n\right) \neq 1$    /* Where $L(u) = \frac{u-1}{n}$ */
7 $publicKey \leftarrow (n, g)$
8 $privateKey \leftarrow (\lambda, \mu)$
9 return $publicKey$, $privateKey$
```

(required for the fast decryption variant of the algorithm) and a random integer g is selected. This procedure must be repeated until it is ensured that n divides the order of g. This is done by checking the existence of the following modular multiplicative inverse: $\mu = (L(g^\lambda \mod n^2))^{-1} \mod n$, where the function L is defined as $L(u) = \frac{u-1}{n}$. The public key is the pair (n, g) while the private key is the pair (λ, μ).

Let $m \in \mathbb{Z}_n$ be a plaintext message, it is possible to encrypt it by selecting a random number $r \in \mathbb{Z}^*_n$, and then computing the ciphertext c as: $E(m) = g^m \cdot r^n \mod n^2$. A ciphertext $c \in \mathbb{Z}^*_{n^2}$ can be decrypted, to get the plaintext m, by computing: $D(c) = L(c^\lambda \mod n^2) \cdot \mu \mod n$.

We have chosen the Paillier cryptosystem for our work because of its homomorphic properties, which allow to execute calculations on event log data without decrypting them in advance, while at the same time it keeps the results semantically meaningful. This is an additively homomorphic cryptosystem, so we can only execute the addition of two ciphertexts without the decryption phase. These are the main properties guaranteed:

c) Property: Homomorphic addition of plaintexts: The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts:

$$D\left(E(m_1, r_1) \cdot E(m_2, r_2) \mod n^2\right) = m_1 + m_2 \mod n.$$

d) Property: Homomorphic multiplication of plaintexts: An encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts,

$$D\left(E(m_1, r_1)^{m_2} \mod n^2\right) = m_1 m_2 \mod n$$

and

$$D(E(m_2, r_2)^{m_1} \mod n^2) = m_1 m_2 \mod n.$$

We can see that given two Paillier encrypted ciphertexts we cannot compute an encryption of the product of these messages without first decrypt both.

IV. Our Proposal: an Anonymous Data Extraction Architecture

Many companies, in order to prepare logs for the process mining analyses, need to extract the data from existing information systems. Our anonymization framework proposal sits in this extraction component. The same goal is also shared with the PROMPT Project[2].

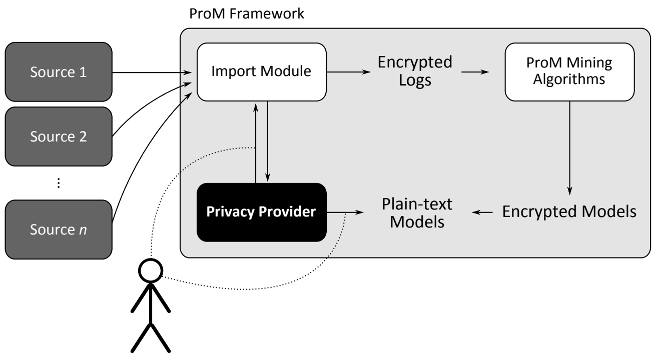

Fig. 2: Architecture diagram of the main components involved in our import framework.

The main components of our extraction architecture are "data connectors". All these different connectors require to adapt the data in order to convert them, from the format used by a particular information system, into an event log. The importer, responsible for the execution of all the connectors, is quite straightforward and does not imply any peculiar property. However, in order to produce anonymised data, we introduced a new layer, called *Privacy Provider*. Each import module is required to pass all the data through such component. Fig. 2 proposes a graphical representation of the architecture of our solution. Pragmatically, a user of our module, before starting the actual importing phase, must specify whether he or she wants to anonymise the data and, if this is the case, random keys are generated. The user has the ability to save such keys on a file which will be required for the decryption.

A. Privacy Provider Usage

Let us consider the small log fragment reported on Table IIa. If we need to extract an anonymised version of the same data we will obtain something similar to Table IIb. Although the fundamental structure of the data is preserved, the data contained into the encrypted log are anonymised and, therefore, it is not possible to easily understand the main content.

The two most important properties of our approach are that the encryption preserves the possibility to group events into traces. Therefore, it is possible to identify which events belong to the same process instance. The second important characteristic is that we apply a numerical encoding to the timestamp of each events (i.e., the number of seconds since the Unix Epoch Time). The consequence of such characteristic is that it is still possible to sort events according to their execution time. The two properties just described allow us to perform several process mining algorithms. It is possible, for example, to execute most of the control-flow discovery

[2]PROMPT stands for "Process Mining for Business Process Improvement". It is a European project belonging to the Eureka Eurostars Program.

Case Id	Time	Activity	Originator	Cost
C_{42}	2015-01-01	Register Request	Mike	50
C_{42}	2015-01-02	Examine Casually	Ellen	400
C_{42}	2015-01-03	Check Ticket	Mike	100
C_{42}	2015-01-04	Decide	Sara	200
C_{42}	2015-01-05	Pay Compensation	Mike	200

(a) Plain event log.

Case Id	Time	Activity	Originator	Cost
2sbl29	21985910	wi93q4	6m4zfq	22939470
2sbl29	37482940	9zykl2	sfydpc	93251889
2sbl29	47456240	vrvk3j	6m4zfq	30399776
2sbl29	59493636	gpude9	x5usw4	69082740
2sbl29	64937336	487an2	6m4zfq	69082740

(b) Enciphered event log.

TABLE II: Example of two log fragments: plain and enciphered.

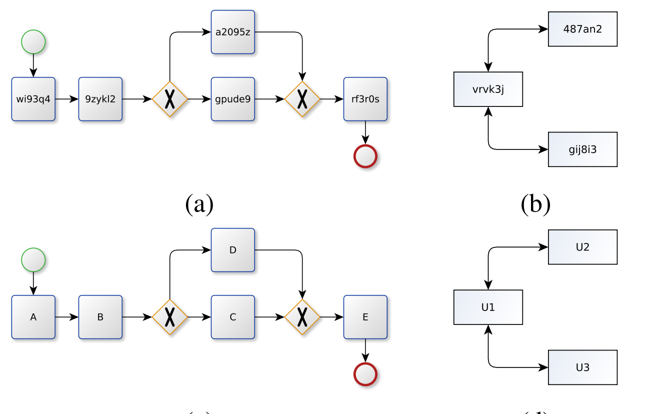

Fig. 3: Models (a) and (c) are the result, expressed as BPMN models, of a control-flow discovery algorithm. Models (b) and (d) are the social networks, extracted using the "working together" measure. Models (a) and (b) have been extracted from the anonymized version of the event log (Table IIb). Models (c) and (d) are the decrypted version of (a) and (b).

algorithms, and the social network analysis tools already available. The resulting models will be structurally correct from a control flow perspective but will contain activity names, or originator names, with no correlation to real events. Decryption procedures can be applied on such model in order to obtain the correct labels of each activity/originator nodes.

The two models, on the top row of Fig. 3, provide a control-flow model and a social network model extracted from the data of Table IIb. Although the graph structure is available, it is not possible to understand the content of each node. The two models at the bottom, instead, are the models obtained after the decryption of the mined models.

We are aware that this kind of anonymization could be prone to advanced de-anonymization techniques, like the ones presented in [21], [22], [23], [24]. However, we consider this as preliminary work, and our aim is to rise privacy anonymity problems as "first class" citizens in the business

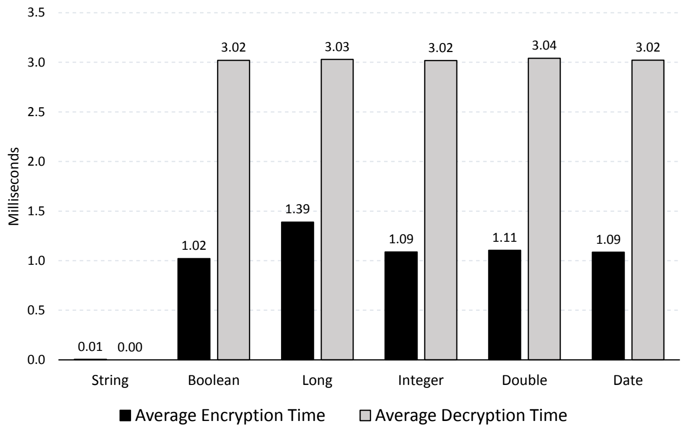

Fig. 4: Average time required to encrypt and decrypt variables with different types.

process mining community.

B. *Implementation and Performance*

In order to retrieve event logs from different sources we wrote a ProM [25] plugin (released as open source[3]) as part of the PROMPT project, which is able to connect to the following information systems: *(i) SAP ERP*, in order to retrieve the *Purchase to Pay* process; *(ii) Microsoft Dynamics*, in order to retrieve the customer management process; *(iii) Siav Archiflow*, in order to extract document management and document workflows processes; *(iv) Microsoft Exchange and MIME email files*, in order to extract the communication process.

In order to implement our procedures as ProM plugins, we needed to use Java as implementation language. Concerning the Privacy Provider, we used the AES implementation already available in the Java Security package. Instead, the code of the Paillier cryptosystem has been written almost from scratch.

Since we are going to need keys for the Paillier system, we implemented a procedure to randomly generate the required prime numbers. Such numbers are also used as a passphrase of the AES system.

We tested our implementation in order to evaluate the performance of our Privacy Provider. In particular we checked the encryption and the decryption procedures, on different data types, by generating 10000 random values for each type. Fig. 4 reports the main outcomes of such tests. We performed all our experiments on a machine with an Intel(R) Core(TM) i7-2670QM processor (2.20GHz) and Oracle Java 1.7.0_55. The string encryption is based on the standard AES implementation available in Java[4]). Instead, all other data types are encrypted using our own implementation of Paillier. The average encryption time for non-string data is about 1 millisecond, while the decryption took a little

[3]At the current stage of the PROMPT project, sources are available upon request. Once the project will be completed, the entire code will be publicly available.

[4]See `http://docs.oracle.com/javase/7/docs/technotes/guides/security/`.

more than 3 millisecond. These times are higher than string data encryption and decryption times: this is probably due to the symmetric nature of AES and its optimization both in the architecture and implementation. We also tested the performance of the random public/private keys generation, producing 10000 keys. The average time required for the generation of a key is 8.65 milliseconds. The machine we used for the key generation test is the same machine mentioned above.

Let us consider a real-world log, with 200000 events, and each of them with 5 string attributes and 5 numeric attributes (10 attributes in total), the encryption time for such log is of about 18 minutes and 20 seconds. We believe that this time result is absolutely acceptable for business application, and hence shows that our approach can be applied in industrial settings.

V. Conclusion and Future Work

In this paper we exposed some privacy issues that arise when extracting information systems data from real scenarios for process mining purposes. In particular we propose to solve the problem with a framework which is able to encrypt both strings and numerical data. We propose the adoption of AES as a symmetric cryptosystem for strings, and Paillier for the homomorphic encryption of numerical values. The entire approach has been implemented in ProM, and a time performance evaluation has been carried out.

For the time being there are no analysis plugins involving numerical data attributes supporting our framework: only conformance checking plugins, which need case id and events order, can be executed successfully. In order to allow numerical data attributes to be exploited by existing analysis plugins, we should properly modify each plugin, and adopt a fully homomorphic cryptosystem (and possibly symmetric) in the encryption of numeric data. Another problem is that our current implementation forces the user to download a generated encryption key, so he is not able to specify a passphrase, compromising the overall usability of the software.

Acknowledgment: The work by A. Burattin has been funded by the Eureka-Eurostars PROMPT project (E!6696). M. Conti is supported by a Marie Curie Fellowship, for the project "PRISM-CODE: Privacy and Security for Mobile Cooperative Devices" funded by the European Commission (grant n. PCIG11-GA-2012-321980), and by the PRIN project "TENACE: Protecting National Critical Infrastructures From Cyber Threats" (grant n. 20103P34XC) funded by the Italian MIUR.

References

[1] W. M. P. van der Aalst, K. van Hee, J. M. van der Werf, A. Kumar, and M. Verdonk, "Conceptual model for online auditing," *Decision Support Systems*, vol. 50, no. 3, pp. 636–647, 2011.

[2] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[3] A. Burattin, *Process Mining Techniques in Business Environments*. Springer International Publishing, 2015.

[4] W. M. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. Weijters, "Workflow mining: A survey of issues and approaches," *Data & knowledge engineering*, vol. 47, no. 2, pp. 237–267, 2003.

[5] S. A. White, "Introduction to BPMN," *IBM Cooperation*, vol. 2, 2004.

[6] C. W. Günther, "Process mining in Flexible Environments," PhD Thesis, Technische Universiteit Eindhoven, Eindhoven, 2009.

[7] M. Jans, M. Alles, and M. Vasarhelyi, "The case for process mining in auditing: Sources of value added and areas of application," *International Journal of Accounting Information Systems*, vol. 14, no. 1, pp. 1 – 20, 2013.

[8] R. Mans, W. van der Aalst, R. Vanwersch, and A. Moleman, "Process mining in healthcare: Data challenges when answering frequently posed questions," in *Process Support and Knowledge Representation in Health Care*, ser. Lecture Notes in Computer Science, R. Lenz, S. Miksch, M. Peleg, M. Reichert, D. Riao, and A. ten Teije, Eds. Springer Berlin Heidelberg, 2013, vol. 7738, pp. 140–153.

[9] A. Hoecht and P. Trott, "Outsourcing, information leakage and the risk of losing technology-based competencies," *European business review*, vol. 18, no. 5, pp. 395–412, 2006.

[10] ——, "Innovation risks of strategic outsourcing," *Technovation*, vol. 26, no. 56, pp. 672 – 681, 2006.

[11] W. M. P. van der Aalst, T. A. J. M. M. Weijters, and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, p. 2004, 2004.

[12] W. M. P. van der Aalst, T. A. J. M. M. Weijters, and A. K. A. de Medeiros, "Process Mining with the Heuristics Miner-algorithm," BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven, 2006.

[13] A. Burattin and A. Sperduti, "Heuristics Miner for Time Intervals," in *European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, 2010.

[14] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering Block-Structured Process Models from Event Logs - A Constructive Approach," in *Proceedings of Petri Nets*. Springer Berlin Heidelberg, 2013, pp. 311–329.

[15] W. M. P. van der Aalst, H. A. Reijers, and M. Song, "Discovering Social Networks from Event Logs," *Computer Supported Cooperative Work (CSCW)*, vol. 14, no. 6, pp. 549–593, Oct. 2005.

[16] C. W. Günther and E. H. M. W. Verbeek, "XES Standard Definition," www.xes-standard.org, 2009.

[17] J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1999.

[18] ——, *The Design of Rijndael*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.

[19] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99*, ser. Lecture Notes in Computer Science, J. Stern, Ed. Springer Berlin Heidelberg, 1999, vol. 1592, pp. 223–238.

[20] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *EURASIP J. Inf. Secur.*, vol. 2007, pp. 15:1–15:15, Jan. 2007.

[21] G. Danezis and C. Troncoso, "You cannot hide for long: De-anonymization of real-world dynamic behaviour," in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, ser. WPES '13. New York, NY, USA: ACM, 2013, pp. 49–60.

[22] ——, "Vida: How to use bayesian inference to de-anonymize persistent communications," in *Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, I. Goldberg and M. Atallah, Eds. Springer Berlin Heidelberg, 2009, vol. 5672, pp. 56–72.

[23] A. Datta, D. Sharma, and A. Sinha, "Provable de-anonymization of large datasets with sparse dimensions," in *Principles of Security and Trust*, ser. Lecture Notes in Computer Science, P. Degano and J. Guttman, Eds. Springer Berlin Heidelberg, 2012, vol. 7215, pp. 229–248.

[24] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, May 2008, pp. 111–125.

[25] E. H. M. W. Verbeek, J. Buijs, B. van Dongen, and W. M. P. van der Aalst, "ProM 6: The Process Mining Toolkit," in *BPM 2010 Demo*, 2010, pp. 34–39.