Business Process Privacy Analysis in PLEAK

Aivo Toots[1,2], Reedik Tuuling[1], Maksym Yerokhin[2], Marlon Dumas[2], Luciano García-Bañuelos[2], Peeter Laud[1], Raimundas Matulevičius[2], Alisa Pankova[1], Martin Pettai[1], Pille Pullonen[1,2](✉), and Jake Tom[2]

[1] Cybernetica AS, Tallinn, Estonia
{aivo.toots,reedik.tuuling,peeter.laud,alisa.pankova,martin.pettai,
pille.pullonen}@cyber.ee
[2] University of Tartu, Tartu, Estonia
{aivo.toots,maksym.yerokhin,marlon.dumas,luciano.garcia-banuelos,
raimundas.matulevicius,pille.pullonen,jake.tom}@ut.ee

Abstract. PLEAK is a tool to capture and analyze privacy-enhanced business process models to characterize and quantify to what extent the outputs of a process leak information about its inputs. PLEAK incorporates an extensible set of analysis plugins, which enable users to inspect potential leakages at multiple levels of detail.

1 Introduction

Data minimization is a core tenet of the European General Data Protection Regulation (GDPR) [2]. According to GDPR, usage of private data should be limited to the purpose for which it has been collected. To verify compliance with this principle, privacy analysts need to determine who has access to the data and what private information these data may disclose. Business process models are a rich source of metadata to support this analysis. Indeed, these models capture which tasks are performed by whom, what data are taken as input and output by each task, and what data are exchanged with external actors. Process models are usually captured using the Business Process Model and Notation (BPMN).

This paper introduces PLEAK[1] – the first tool to analyze privacy-enhanced BPMN models in order to characterize and quantify to what extent the outputs of a process leak information about its inputs. The top level (Boolean level, Sect. 2), tell us whether or not a given data in the process may reveal information about a given input. The middle level, the qualitative level (Sect. 3), goes further by indicating which attributes of (or functions over) a given input data object are potentially leaked by each output, and under what conditions this leakage may occur. The lower level quantifies to what extent a given output leaks information about an input, either in terms of a sensitivity measure (Sect. 4) or in terms of the guessing advantage that an attacker gains by having the output (Sect. 5).

[1] https://pleak.io (account: *demo@example.com*, password: *pleakdemo*, manual: https://pleak.io/wiki/, source code: https://github.com/pleak-tools/).

R. Hähnle and W. van der Aalst (Eds.): FASE 2019, LNCS 11424, pp. 306–312, 2019.
https://doi.org/10.1007/978-3-030-16722-6_18

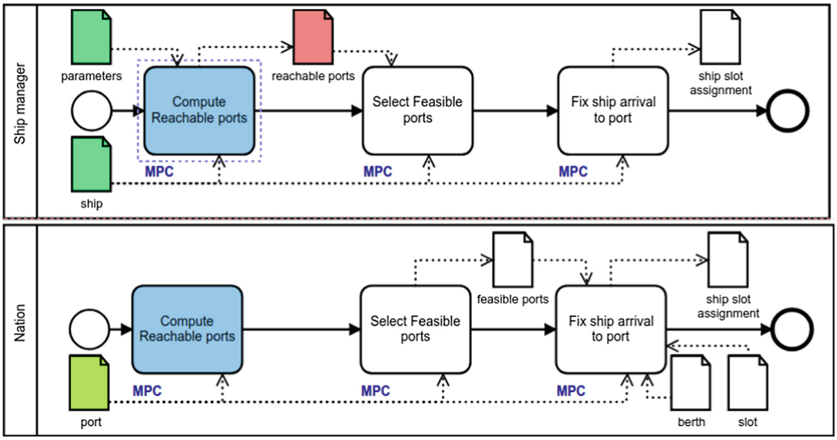


Fig. 1. Aid distribution process

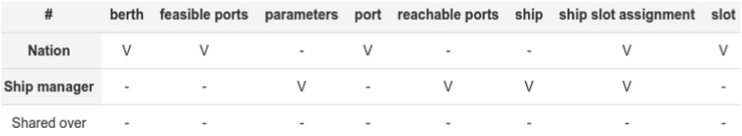
To illustrate the capabilities of PLEAK, we refer to an "aid distribution" process in Fig. 1. This process starts when a nation requests aid from the international community to handle an emergency and a country offers to route a ship to help transport people and/or goods. The goal of the process is to allocate a port and a berth to the ship but not to reveal information about ships that are unable to help or the parameters of the ports. The process uses a type of privacy-enhancing technology (PET) known as secure multiparty computation (MPC). MPC allows participants to perform joint computations such that none of the parties gets to see the data of the other parties, but can learn the output depending on the private inputs. Given a ship, a deadline and the list of ports, task "Compute reachable ports" retrieves the list of ports reachable by the deadline. Tasks with identical names in different pools denote MPC computations carried out jointly by multiple stakeholders. Task "Select feasible ports" retrieves ports with the capacity to host the ship. The third task selects a port, a berth, and a slot for the ship, and discloses them to both participants.

Related Work. We are interested in privacy analysis of business processes and in this space Anica [1] is closest to our work. However, PLEAK's analysis is more fine-grained. Anica allows designers to see that a given object O1 may contain information derived from a sensitive data object O2, but it can neither explain how the data in O2 is derived from O1 (cf. Leaks-When analysis) nor to what extent the data in O2 leaks information from O1 (cf. sensitivity and guessing advantage analysis). In addition, they are interested in security levels and our high level analysis looks at PETs deployed in the process.

2 PE-BPMN Editor and Simple Disclosure Analysis

The model in Fig. 1 is captured Privacy-Enhanced BPMN (PE-BPMN) [7,8]. PE-BPMN uses stereotypes to distinguish used PETs, e.g. MPC or homomorphic encryption, that affect which data is protected in the process. The PE-BPMN editor allows users to attach stereotypes to model elements and to enter the stereotype's parameters where applicable. The editor integrates a checker, which verifies stereotype specific restrictions. For example, that: (1) when a task has an MPC stereotype, there is at least one other "twin" task with the same label in another pool, since an MPC computation involves at least two parties; (2) when one of these tasks is enabled, the other twin tasks is eventually enabled; and (3) the joint computation has at least one input and one output.

Given a valid PE-BPMN model, Pleak runs a binary privacy analysis, which produces a *simple disclosure report* and data dependency matrix. The disclosure report in Fig. 2 tells us whether or not a stakeholder gets to see a given data object. In the report "V" indicates that a data object (in columns) is visible to a stakeholder (in rows). Marker "H" (hidden) is used for data with cryptographic protection, e.g. encrypted data. Row "shared over" refers to the network service provider, who may also see some of the data (e.g. unencrypted data objects).



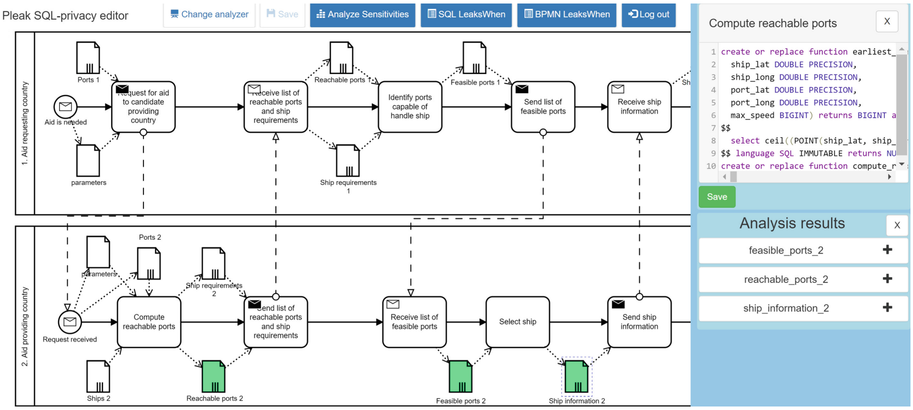
#	berth	feasible ports	parameters	port	reachable ports	ship	ship slot assignment	slot
Nation	V	V	-	V	-	-	V	V
Ship manager	-	-	V	-	V	V	V	-
Shared over	-	-	-	-	-	-	-	-

Fig. 2. Simple disclosure report for the aid distribution process in Fig. 1

3 Qualitative Leaks-When Analysis

Leaks-When analysis [3] is a technique that takes as input a SQL workflow and determines, for each (output, input) pair which attributes, if any, of the input object are disclosed by the output object and under which conditions. A SQL workflow is a BPMN process model in which every data object corresponds to a database table, defined by a table schema, and every task is a SQL query that transforms the input tables of the task into its output tables. Figure 3 shows a sample collaborative SQL workflow – a variant of the "aid distribution" example where the disclosure of information about ships to the aid-requesting country is made incrementally. The figure shows the SQL workflow alongside the query corresponding to task "Select reachable ports". All data processing tasks and input data objects are specified analogously.

To perform a Leaks-When analysis, the user selects one or more output data objects and clicks the "SQL LeaksWhen" button. The Leaks-When analysis shows one tab for each output data object and one report for each column in the output table. The report is generated by extracting all runs of the workflow and applying dataflow analysis techniques to each run in order to infer all relevant data dependencies. An example of a leaks-when report (in graphical form) is shown in Fig. 4. The first input to *Filter* is the disclosed value (leaks branch), e.g. the arrival time. The second input (when branch) is the condition of outputting the first input, e.g. that the arrival time is less than the deadline and the ship has the required name. Each Leaks-When report ends with such filter but the rest of the graph aggregates the computations described in SQL.



```
create or replace function earliest_
  ship_lat DOUBLE PRECISION,
  ship_long DOUBLE PRECISION,
  port_lat DOUBLE PRECISION,
  port_long DOUBLE PRECISION,
  max_speed BIGINT) returns BIGINT a
$$
  select ceil((POINT(ship_lat, ship_
$$ language SQL IMMUTABLE returns NU
create or replace function compute_r
```

Fig. 3. Aid distribution SQL workflow in Pleak SQL editor

4 Sensitivity Analysis and Differential Privacy

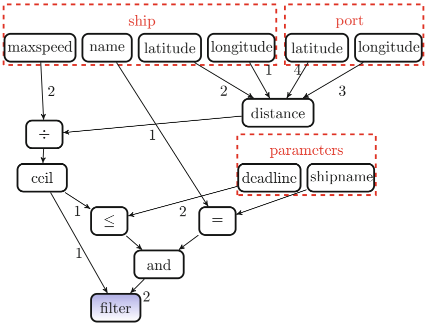


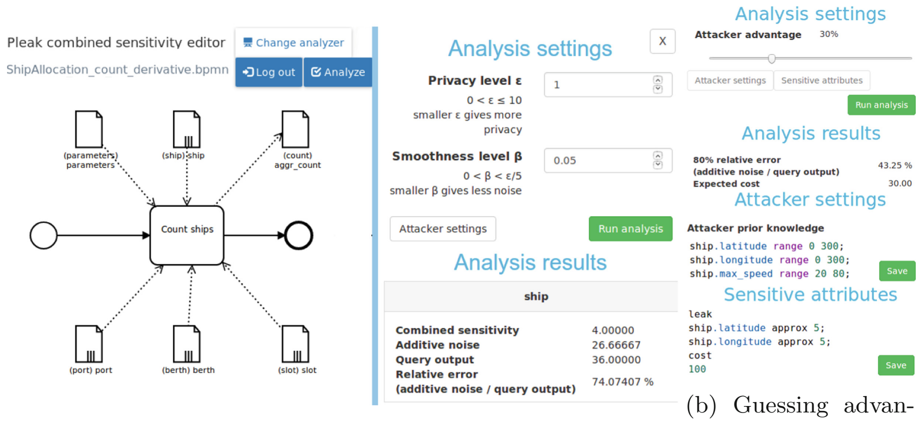
Fig. 4. Sample leaks-when report

The *sensitivity of a function* is the expected maximum change in the output, given a change in the input of the function. Sensitivity is the basis for calibrating the amount of noise to be added to prevent leakages on statistical database queries using a differential privacy mechanism [6]. Differential privacy ensures that it is difficult for an attacker, who observes the query output, to distinguish between two input databases that are sufficiently "close" to each other, e.g. differ in one row. Pleak tells the user how to sample noise to achieve differential privacy, and how this affects the correctness of the output. Pleak provides two methods – global and local – to quantify sensitivity of a task in a SQL workflow or of an entire SQL workflow. These methods can be applied to queries that output aggregations (e.g. count, sum, min, max).

Global sensitivity analysis [5] takes as input a database schema and a query, and computes the theoretical bounds for sensitivity, which are suitable for any instance of the database. This shows how the output changes if we add (remove) a row to (from) some input table. The analysis output is a matrix that shows the sensitivity w.r.t. each input table separately. It supports only COUNT queries.

Sometimes, the global sensitivity may be very large or even infinite. *Local sensitivity* analysis is an alternative approach, which requires as input not only a schema and a query, but also a particular instance of the underlying database, and it tells how the output changes with the change *from the given input*. Using the database instance improves the amount of noise needed to ensure differential privacy w.r.t. the number of rows. Moreover, it supports COUNT, SUM, MIN, MAX aggregations, and allows to capture more interesting distances between input tables, such as change in a particular attribute of some row. In PLEAK, we have investigated a particular type of local sensitivity, called *derivative sensitivity* [4], which is in first place adapted to continuous functions, and is closely related to function derivative. PLEAK uses derivative sensitivity to quantify the required amount of noise as described in [4].

An example of derivative sensitivity analysis output is shown in Fig. 5a. It tells that the derivative sensitivity w.r.t. the *Ship* table is 4, and that a differential privacy level of $\varepsilon = 1$ can be achieved using smoothness parameter $\beta = 0.05$. To this end, we would have to add an amount of (Laplacian) noise such that the relative error of the output is 74%. More precisely, if the correct output is y, the noised answer will be between $0.26y$ and $1.74y$ with probability 80%. A tutorial on sensitivity analyzer can be found at https://pleak.io/wiki/sql-derivative-sensitivity-analyser. More examples can be found in the full version of this paper [9].



Analysis settings

Privacy level ε: 1
0 < ε ≤ 10
smaller ε gives more privacy

Smoothness level β: 0.05
0 < β < ε/5
smaller β gives less noise

Analysis results

	ship
Combined sensitivity	4.00000
Additive noise	26.66667
Query output	36.00000
Relative error (additive noise / query output)	74.07407 %

(a) Derivative sensitivity analysis

Analysis settings

Attacker advantage 30%

Analysis results

80% relative error (additive noise / query output)	43.25 %
Expected cost	30.00

Attacker settings

Attacker prior knowledge

```
ship.latitude range 0 300;
ship.longitude range 0 300;
ship.max_speed range 20 80;
```

Sensitive attributes

```
leak
ship.latitude approx 5;
ship.longitude approx 5;
cost
100
```

(b) Guessing advantage analysis

Fig. 5. Examples of quantitative analysis

5 Attacker's Guessing Advantage

While function sensitivity as defined in Sect. 4 can be used directly to compute the noise required to achieve ε-differential privacy, it is in general not clear which ε is good enough, and the "goodness" depends on the data and the query [6]. We want a more standard security measure, such as guessing advantage, defined as the difference between the posterior (after observing the output) and prior (before observing the output) probabilities of attacker guessing the input.

The *guessing advantage* analysis of PLEAK takes as input the desired upper bound on attacker's advantage, which ranges between 0% and 100%. The user specifies particular subset of attributes that the attacker is trying to guess for some data table record, within given precision range. The user may define prior knowledge of the attacker, which is currently expressed as an upper and a lower bound on an attribute. The analyzer internally converts these values to a suitable ε, and computes the noise required to achieve the bound on attacker's advantage.

Figure 5b shows an example parameters and output of this analysis. The attacker already knows that the longitude and latitude of a ship are in the range [0...300] while the speed is in [20...80]. His goal is to learn the location of any ship with a precision of 5 units. If we want to bound the guessing advantage by 30% using differential privacy, the relative error of the output will be 43.25%. For a tutorial see https://pleak.io/wiki/sql-guessing-advantage-analyser.

Acknowledgements. The research was funded by Estonian Research Council under IUT27-1 and IUT20-55 and by the Air Force Research laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under contract FA8750-16-C-0011. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

1. Accorsi, R., Lehmann, A.: Automatic information flow analysis of business process models. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 172–187. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32885-5_13
2. Colesky, M., Hoepman, J., Hillen, C.: A critical analysis of privacy design strategies. In: IEEE Security and Privacy Workshops (SP), pp. 33–40. IEEE (2016)
3. Dumas, M., García-Bañuelos, L., Laud, P.: Disclosure analysis of SQL workows. In: 5th International Workshop on Graphical Models for Security. Springer, Heidelberg (2018)
4. Laud, P., Pankova, A., Pettai, M.: Achieving differential privacy using methods from calculus (2018). http://arxiv.org/abs/1811.06343
5. Laud, P., Pettai, M., Randmets, J.: Sensitivity analysis of SQL queries. In: Proceedings of the 13th Workshop on Programming Languages and Analysis for Security, PLAS 2018, pp. 2–12. ACM, New York (2018)
6. Lee, J., Clifton, C.: How much is enough? Choosing ϵ for differential privacy. In: Lai, X., Zhou, J., Li, H. (eds.) ISC 2011. LNCS, vol. 7001, pp. 325–340. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24861-0_22

7. Pullonen, P., Matulevičius, R., Bogdanov, D.: PE-BPMN: privacy-enhanced business process model and notation. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNCS, vol. 10445, pp. 40–56. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_3
8. Pullonen, P., Tom, J., Matulevičius, R., Toots, A.: Privacy-enhanced BPMN: enabling data privacy analysis in business processes models. Softw. Syst. Model. (2019). https://link.springer.com/article/10.1007/s10270-019-00718-z
9. Toots, A., et al.: Business process privacy analysis in pleak (2019). http://arxiv.org/abs/1902.05052