UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Maksym Yerokhin

Multi-Level Policy-Aware Privacy Analysis

Master's Thesis (30 ECTS)

Supervisor: Pille Pullonen, MSc
Supervisor: Luciano García-Bañuelos, PhD

Tartu 2019

Multi-level policy-aware privacy analysis

Abstract:

The NAPLES (Novel Tools for Analysing Privacy Leakages) project is a research initiative conducted as a collaboration between Cybernetica AS and the University of Tartu, with funds of the Brandeis program of the Defense Advanced Research Projects Agency (DARPA). The research project has produced the theory and a set of tools for the analysis of privacy-related concerns, to determine the potential leakage of the data from the information systems. Specifically, PLEAK is a tool that takes as input business processes specified with the Business Process Model and Notation (BPMN), where model entities are associated with privacy-enhancing technologies, in order to enable the analysis of privacy concerns at different levels of granularity.

With the time, the NAPLES project has produced several analyzers. Such analyzers target SQL collaborative workflows, that is, BPMN collaborative models that specify the steps of computation that correspond to SQL manipulation statements over the data objects representing the SQL data sources. The simple disclosure analysis performs a high-level data reachability analysis that reveals potential data leakages in the privacy-enhanced model of a business process: it tells whether a data object is visible to a given party. Other analyzers, such as the Leaks-When and the Guessing Advantage ones, provide finer-grained, qualitative and quantitative measures of data leakage to stakeholders.

My work was part of the NAPLES project and my contributions are manifold. First, I added the concept of Global and Local privacy policies in the SQL collaborative workflows, which endow a party of the business process with access rights to the selected SQL entities with defined constraints. Second, I designed an integrated multi-level approach to the disclosure analysis: from the high-level declarative disclosure (What data might leak?) to the conditional disclosure (When does data leak?) and quantitative measure (How much does data leak?). This approach is based on existing tools of PLEAK for privacy analysis. However, I refined these tools to accept more unified set of inputs and integrated the privacy policies with the Leaks-When and Guessing Advantage analyzers. Finally, I developed a case study, which has been used for showcasing the aforementioned integrated multi-level approach to the disclosure analysis, and that has been used as a proof-of-concept for NAPLES tools.

Keywords: BPMN, privacy analysis, SQL collaborative workflow, privacy policy, Leaks-When analysis, Guessing advantage, PLEAK

CERCS: P170 - Computer science, numerical analysis, systems, control

Mitmekihiline käitumisteadlik privaatsuse analüüs

Lühikokkuvõte:

Projekt NAPLES (Novel Tools for Analysing Privacy Leakages – Priivatsuse Lekkimise Analüüsi Uudsed Vahendid) on uurimisalgatus mis on läbiviidud Cybernetica AS ja Tartu Ülikooli koostöös, Kaitsealase Täiustatud Uurimisprojektide Agentuuri (DARPA) Brandeis programmi vahendite abil. Uurimisprojekti raames sai välja väljatöödatud privaatsusega seotud ohtude analüüsi teooria ning vahendidte hulk, mille abil saab tuvastada võimalikud admelekked infosüsteemides. Nimelt PLEAK on vahend, mis võtab sisendiks Äriprotsessi Mudel- ja Märgisüsteemis (BPMN) kirja pandud äriprotsessi, kus mudeli üksused on seotud privaatsust suurendavate tehnoloogiatega eesamärgiga võimaldada erineva granulaarsusega privaatsuse ohtude analüüsi.

Ajaga projekt NAPLES tootis mitu analüüs vahendit. Need keskenduvad SQL koostöövood, mis on BMPN koostöö mudelid, mis omakorda määravad arvutusetappe, mis vastavad SQL manipuleeerimisväidetele üle SQL andmeallikate esindavaid admeobjekte. Lihtne avalikustamise analüüs täidab kõrgetasemelise andmete kättesaadavuse analüüsi, mis tuvastab võimalikke andmete lekke äriprotsessi privaatsust suurendavas mudelis: otsustades kas andmeobjekt on nähtav teatud isikule. Muud analüüsivahendid nagu Leaks-When ja Guessing Advantage pakuvad huvilisele suurema täpsusega andmete lekke kvalitatiivseid ja kvantitatiivseid meetmeid.

Minu töö oli NAPLES projekti osa ning minu panused mitmesugused. Esiteks ma lisasin Globaalse ja Lokaalse privaatsuspoliitika mõisted SQL koostöövoogudes, mis tagab äriprotsessi isiku määratud kitsendusega ligipääsu teatud SQL üksustele. Teisegks ma kavandasin ning integreerisin multikihilist lähenemist avalikustamise analüüsile: alates kõrgtaseme deklaratiivsest avalikustamisest (millised andmed on leekimisvõimekad?) kuni tingimusliku avalikustamise (millal leek osutub?) ja kvantitatiivse meetmeni (kui palju andmeid lekib?). Vastav lähenemine põhineb olemasolevate PLEAK vahendite peal, kuid ma kohendasin neid, võimaldades ühtset sisendit ja integreerides privaatsuspoliitkaid Leks-When ning Guessing Advantage riistade sisse. Lõpuks ma arendasin juhtumiuuringu, mis sai kasutatud eeelkirjeldatud panuste demonstreerimiseks.

Võtmesõnad: BPMN, privaatsuse analüüs, SQL koostööl põhinev töövoog, privaatsuspoliitika, Leaks-When analüüs, Arusaadav eelis, PLEAK

CERCS: P170 - arvutiteadus, arvuline analüüs, süsteemid, kontroll

CONTENTS

1 INTRODUCTION

Events such as Facebook's massive security-breach[1] and, to a lesser extent, Quora's data breach[2] have made us more aware of the risks of privacy-related issues. This has also motivated companies to review their own systems to identify potential security-holes and vulnerabilities that could compromise customer's privacy. Moreover, newly adopted regulations such as the General Data Protection Regulation (GDPR)[3] in the European Union or Brazil's General Data Protection Law (LGPD)[4] have obliged companies to review their internal policies and established processes to comply with these restrictions. Despite its importance, however, there are no tools to automate the analysis of privacy-related issues. In this context, NAPLES (Novel Tools for Analysing Privacy Leakages)[5] is a collaborative effort by Cybernetica AS and the University of Tartu, with funding by Defense Advanced Research Projects Agency (DARPA), that aims at developing such type of tools.

NAPLES is a research project, part of the DARPA's[6] Brandeis program[7]. One of the outcomes of the NAPLES project is the development of PLEAK[8] - a set of tools for detecting and analyzing privacy leakages that can occur during the execution of a business process [TTY+19]. To that end, NAPLES has also proposed an extension to the standard Business Process Model and Notation (BPMN), called Privacy-enhanded BPMN (PE-BPMN), which can be used to specify the use of privacy-preserving technologies in the context of a business process. Using PE-BPMN and plain BPMN with privacy-related annotations as input, PLEAK tools can be used to perform a set of analysis to characterize potential information leaks and, in some cases, quantify the theoretical bounds of the leakage or a measure of the potential advantage a stakeholder can gain by misusing the information disclosed to him/her.

By the time I joined the NAPLES project, PLEAK included already the following set of tools:

- A Simple disclosure analyzer produces a high-level report about what can leak to which party.
- A Leaks-When analyzer that allows an analyst to trace-down what attributes (columns) of a given SQL input of the process are disclosed to a given party, and under what conditions this disclosure occurs.
- A Guessing Advantage analyzer that allows an analyst to determine how much noise should be injected into a given output of a process in order to ensure that after disclosing this output, the probability that an attacker would be able to guess the value of an input, within a given interval, does not increase by more than $g\%$ (where g is the tolerated guessing advantage).

However, the existing analyzers worked in isolation and had different requirements concerning the format of privacy-related annotations added to the process models used as input.

1.1 Problem Statement

As mentioned before, PLEAK provided a set of analyzers that worked mostly in isolation. However, the challenge was not only one of integration, but one to provide an approach to analysing privacy-concerns in business processes in an integrated and consistent manner. Therefore, the aim of my work was to develop a holistic multi-level analysis for the business processes with SQL collaborative workflows as follows:

[1] https://www.facebook.com/help/2687943754764396

[2] https://www.quora.com/How-did-the-Quora-data-breach-in-2018-occur

[3] https://gdpr-info.eu/

[4] http://portaldaprivacidade.com.br/wp-content/uploads/2018/08/LGPD-english-version.pdf

[5] https://cyber.ee/research/projects/#naples

[6] This research was funded 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 author(s) and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

[7] https://www.darpa.mil/program/brandeis

[8] https://pleak.io/

- The top level, the Boolean level, analysis provides us with a high-level report telling what inputs may be disclosed to which parties. We implement an Extended Simple disclosure report, which reveals additional types of disclosure and handovers the analysis of the interesting input to the next level.
- The middle level, the qualitative level, gives an insight about concrete SQL attributes or their derivatives that may leak and the conditions for the leakage to appear. An upgraded Leaks-When analyzer also considers the privacy policy attached to the process model.
- The lower level calculates a quantitative estimation about the attributes disclosures inferred from the middle-level analysis. The outcomes of this low-level analyzer show to what extent a given output leaks data about an input, either in terms of a sensitivity measure or in terms of a guessing advantage and differential privacy analysis, when an attacker gains knowledge by having the outputs.

However, not every disclosure corresponds to every leakage. That is, an information leakage is more precisely the undesired disclosure of sensitive information. That implies that we require a notion of privacy policy, such that, any information disclosure that violates a privacy policy should be considered a leakage.

In light of the above, we postulate the following objectives addressed by this work:

- Introduction of a privacy policy and an extension of the Leaks-When analyzer that takes policy as input. Leaks-When can then distinguish between disclosures that are allowed by the policy versus disclosures that partially or totally violate the policy. To this end, we define a privacy definition language that allows users to state what attributes/functions of a given input can be disclosed to a given party. Rather than defining a completely new policy definition language, we extend the syntax of the SQL 'grant' statement to capture the privacy policy rules. The proposed definition language of the privacy policy also allows to capture policies that can be used by the Guessing Advantage analyzer (in addition to the Leaks-When analyzer).
- Taking the first steps towards the integration of the Leaks-When analyzer with the Guessing Advantage analyzer and the Extended Simple Disclosure analyzer, This represent our effort towards defining an approach to holistic multi-level privacy analysis of business process models.
- As third goal we have designed an extension for the 'Aid Distribution' scenario and used it to demonstrate how the Extended Simple Disclosure analysis, Leaks-When analysis and the Guessing Advantage analysis can be used together in a multi-level flawless paradigm. Since the Guessing Advantage analyzer deals with 'count', 'sum' and other numerical aggregation functions, we added support for aggregation functions to the Leaks-When analyzer to make them work with the uniform SQL input.

All the above has been implemented as extensions of the existing software components (backend, frontend), and also the extension and unification the existing analyzers to enable the multi-level privacy analysis over the same process model and in the same working environment.

1.2 Thesis Organization

The rest of this Master Thesis is structured as following.

Chapter 2 focuses on the theoretical basics of the privacy analyzers and introduces the background for the subsequent work.

Chapter 3 presents the detailed notion of the privacy policy.

Chapter 4 outlines the integration and improvements of the Simple disclosure, Leaks-When and Guessing Advantage analysis.

Chapter 5 provides the design choices and limitations about the current implementation of the SQL-privacy editor.

The final chapter summarizes the advancements reported in this Master Thesis and outlines the future outcomes to be pursued.

2 BACKGROUND

This section provides an overview of the theoretical background available in the literature, on process modeling and concepts regarding disclosure of data. Since the current work is being conducted in the scope of PLEAK (Privacy LEAKage) tool, the main papers analyzed have been published by the research groups interested in discovering and solving of the contemporary privacy-related challenges. In particular, we want to answer the following research questions regarding the data leakage lifecycle:

1. What pieces of data leak, i.e. what specific SQL attributes from the SQL table are shared to the 3rd party?
2. In which conditions does it leak, i.e. what is the sequence of operations leading to disclosure?
3. How much does it leak, i.e. what is the quantitative measure of the leakage?

We analyze some of the theories and methods trying to answer these questions and elaborate on the creation of the environment for the complex analysis uniting different analysis aspects.

2.1 Business Process Model and Notation

Business Process Management (BPM) is a discipline, that aims at providing tools to define, regulate and improve the operation of organizations. Central to this discipline is the notion of business processes which are usually captured in the form of models. To that end, BPM practitioners employs usually a graphical notation to capture business processes, among which the Business Process Model and Notation (BPMN) [RLP16] is probably the most widely used one. Endorsed by the OMG [OMG11], BPMN is a standard which defines a graphical notation and semantics thereof, which serve as an instrument to represent parties of the organization or system, their actions, data, communications and the sequence of the process execution. BPMN comprises a large set of graphic elements but, for the purpose of this thesis, we restrict our attention only to the major elements in the notation, which are summarized if Figure 1.

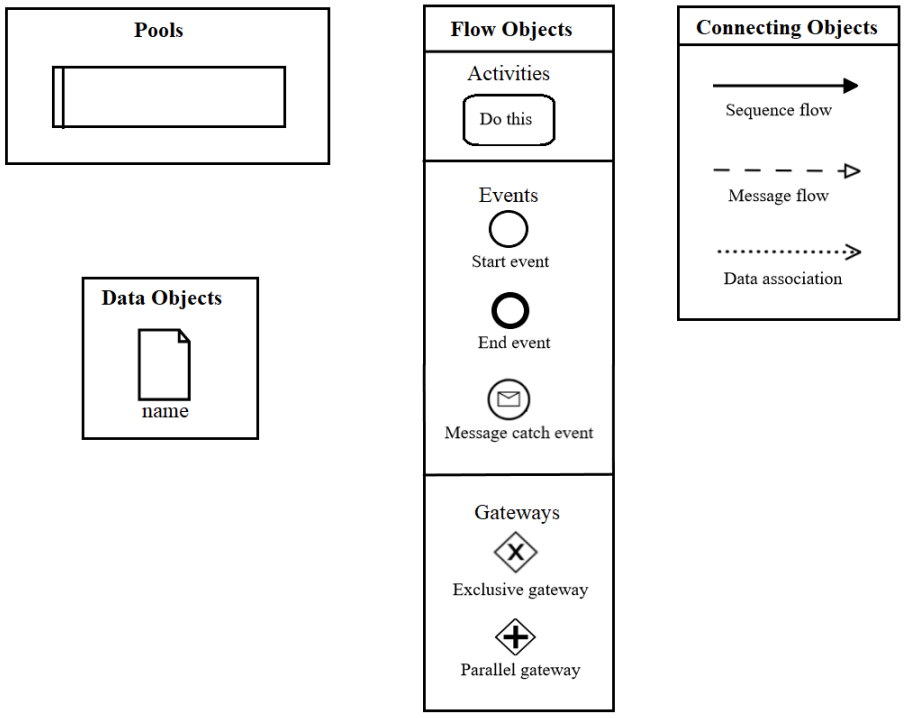


Figure 1: Subset of BPMN elements relevant to this thesis

'Pools' represent participants (stakeholders) of the process. If the process model has only one party, then the pool is usually not used. 'Flow Objects' consist of events, activities and gateways, they define most of the process logic and decisions of execution. We employ three types of BPMN event elements: start event, message catch event and end event.

'Start event' stands for the beginning of the business process. 'Message catch event' designates a point where the process has to wait for a message (a trigger) to proceed. 'End event' stands for the ending of the process. In case of pools usage, each pool possesses an own end event.

'Activities' are the elements with the work to be executed for the process. 'Tasks' represent a granular unit of work that cannot be split into smaller activities. 'Gateways' join and split the flows in a process model. 'Exclusive gateways' corresponds to 'if-then-else' flow, so that the process is split into exclusive branches. The condition attached to the gateway defines which branch must be chosen to follow. On the contrary, 'Parallel gateways' execute all the branches simultaneously without a condition. After splitting the flow into several branches, either exclusive or parallel gateway should be synchronized by the same type of a gateway joining the branches.

'Connecting Objects' build a sequence out of the flow elements. 'Sequence flows' connect tasks, events and gateways. 'Message flows' are required to share the information from one party (pool) to another. 'Data associations' express the flow of information and their direction between activities. 'Data objects' represent the data used, produced and stored by activities of the process model.

Additionally, we consider a BPMN process model as a BPMN collaborative model, if it consists of multiple pools and, therefore, multiple parties. This kind of process model is used in Section 2.5.

2.2 Privacy-Enhanced Business Process Model and Notation

Given that BPMN remains a general purpose notation, the standard graphical notation to capture only control flow, data flow and participants perspectives. Therefore, the standard notation falls short when it comes to represent aspects connected with the domain of privacy and security. To cope with this limitation, the NAPLES project team has defined an extension to the standard BPMN, called the Privacy-Enhanced Business Process Model And Notation (PE-BPMN) [PMB17, PTMT19], to add concepts and notation elements allowing analysts to annotate BPMN models with privacy related information. In addition to the definition, the NAPLES project has implemented some tools around PE-BPMN which includes an graphical editor, which enables analysts to add information about the use of privacy-enhancing technologies (PETs) to process models to address security concerns.

In [Too18] each PET is represented by a programmatic component incorporated into a PE-BPMN editor. This set of tools is operating over bpmn-js modeler in form of a web application. The PET is a common concept to designate any security-aware tool, but in specific BPMN modeling we operate concrete stereotypes to distinguish PETs [TTY+19]. These stereotypes affect what data will be protected and how during the process execution. One can attach PET by selecting data object or task in PE-BPMN editor and navigating through 'Stereotypes menu'. For instance, we want to incorporate the encryption in a business process. We have one data object with a public key, another data object with the data to encrypt and a task that takes two data objects as input, carries out the encryption and outputs the encrypted data. To mark that the data object will hold the public key, we select PKPublic stereotype and assign it to the data object. Then we select task and choose from menu Data protection/Confidentiality protection/PKEncrypt. If the structure of BPMN is compliant (task has an input PKPublic stereotype and has one output) the stereotype will apply to the task.

All of the stereotypes are displayed on the business process diagram and inputs/outputs of stereotyped task are highlighted with different colors, making it visually convenient to observe. In Figure 2 we see a part of some conceptual business process. The tasks which are performing a Multi-Party Computation (MPC) [GGL15] are highlighted with blue color, their respective inputs are highlighted with green and outputs with red. Secure multiparty computation (MPC) techniques are approaches that enable participants to protect their information and process it in a distributed manner, while preserving the

confidentiality of all the inputs. We also see the usage of DifferentialPrivacy stereotype and SKEncrypt (secret key encryption) [Too18].

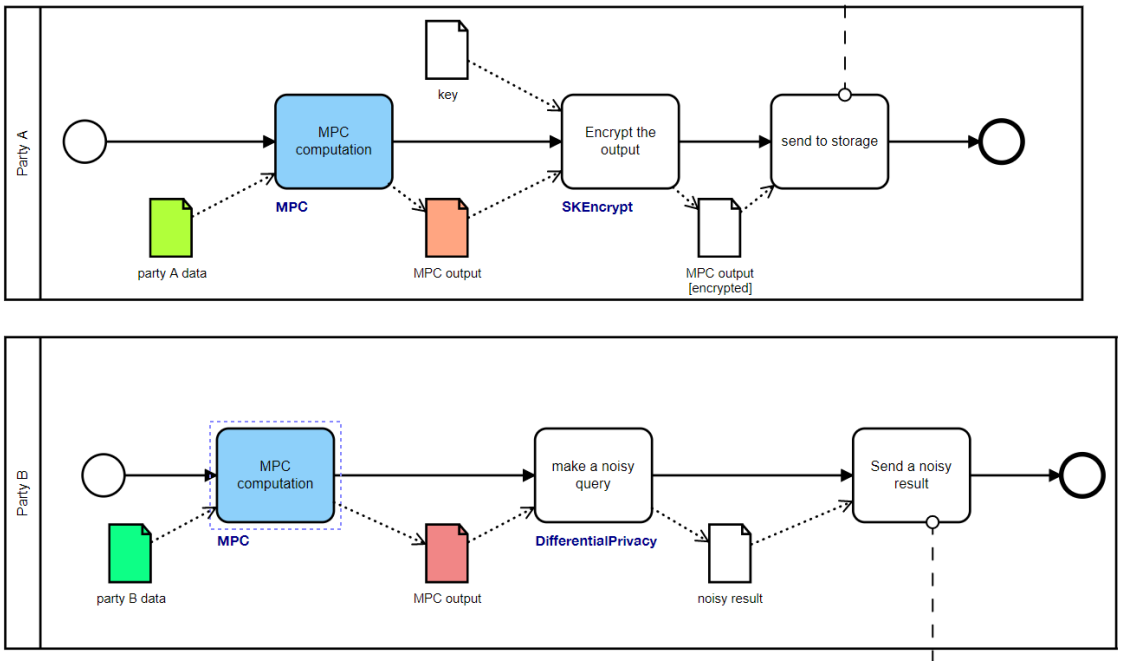


Figure 2: Conceptual example of applied PETs

All of the stereotypes are saved to the BPMN scheme while exporting XML file with BPMN description. This allows to run PET-aware analysis on business process. There are several groups of PETs highlighted in [PMB17] defined according to the goals they pursue and targets that are met in different privacy areas such as data transfer or data encryption. Overall these goals, targets and generic stereotypes with examples form a classification and hierarchy of PETs. For instance, Communication Protection techniques that allow for keeping the content exchanged between two entities secure - not viewable and not modifiable by third party. Therefore, Communication Protection goal includes Security and Anonymity targets. Security can be provided with TLS, IPSec and other protocols and represented by SecureChannel generic stereotype, whereas Anonymity can be provided by VPN and proxies, but not yet covered with concrete stereotype implementation. Data Protection goal ensures purity and confidentiality of the data and it includes digital signatures, authentication tokens, encryption keys etc. By applying stereotype derived from generic ProtectConfidentiality stereotype there is no disclosure during the transfer unless party has decryption key and can apply decryption stereotype or it is special case of secret sharing, that requires an additional constraint of the distributed storing of the data (so only a small random part of data can be disclosed during the transfer).

BPMN can be extended to support each group of PETs and not only involves BPMN tasks and data objects. For example, BPMN Data Flow can be complemented with Communication Protection. This is one of the simplest case since to activate SecureChannel stereotype we only need to select a specific BPMN Data Flow arrow and then select wanted stereotype from Stereotypes menu. Most of the privacy related activities that are currently implemented in [PMB17] deal with BPMN Tasks, so authors complement the standard BPMN elements hierarchy by introducing an abstract PET-Task. It is derived from BPMN Task and it is a sibling for such standard subtypes as User Task and Script Task. The hierarchy extension of the BPMN specific syntax is accomplished using general stereotypes such as ProtectConfidentiality and PetComputation and then going down to more specific such as PKEncryption and PKComputation. Some goals result in a set of tasks, for example, if we choose DataProtection/ConfidentialityProtection in the context menu of the BPMN task, we will see two subgroups of stereotypes – Protect and Open. Protect section incorporates stereotypes to add protection to data such as PKEn-

crypt and SKEncrypt, whereas Open section consists of symmetrical operations such as PKDecrypt and SKDecrypt.

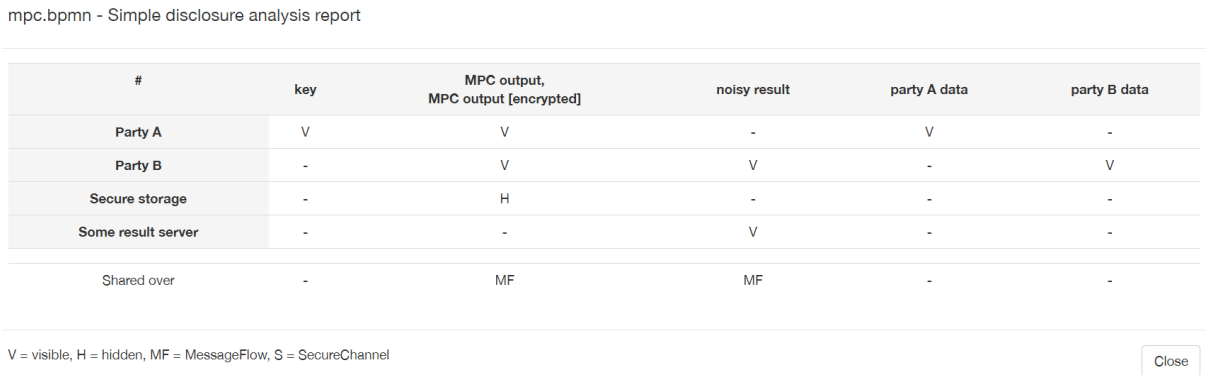
As a remark we should notice that there is another PET-related branch of research with respect to [PMB17] for incorporating privacy into business processes. In [DAKG17] authors make an attempt to highlight several privacy process patterns. These patterns are deemed to bridge the gap between privacy prototypes and implementation by having elicited necessary use cases and goals. Instead of prototyping the security-aware systems in form of business process from scratch we can now choose from the hierarchy of typical privacy problems and add prepared pattern. This approach is supposed to speed up initial prototyping and avoid basic errors by having most of the potential issues considered in the proposed BPMN process pattern. Researchers also introduce a set of the eight privacy areas that they found out to be feasible: authentication, authorisation, anonymity, pseudonymity, unlinkability, undetectability, unobservability and data protection. The main advantage of proposed patterns is their typical behavior used in security-aware business processes, authors claim that they figured out the main security goals, built own PET classification and proposed a solution for each security goal basing on comprehensive literature review that corresponds to research groups needs. In this thesis we focus on the PE-BPMN work that is already a part of PLEAK. The approach proposed in [DAKG17] can be used to extend the analysis offered by PE-BPMN, but it is out of scope of this thesis.

2.3 Simple Disclosure Analysis

The Simple Disclosure Analysis is a high-level process model insight, initially located within the PE-BPMN editor of PLEAK. It takes as an input a BPMN process model and also considers PETs attached to the model. For the purposes of PE-BPMN, an object is considered to be disclosed if it is received or intercepted by another party regardless of intent or policy. The existing version of the PE-BPMN editor is used to gain an understanding of data sharing between pools, their associated data objects and inter-dependencies between inputs and outputs of activities. The main interest of this analysis is revealing what is disclosed when the process is executed as intended.

The current version of PE-BPMN editor consists of two analyzers: Simple Disclosure analyzer and Data dependencies analyzer. In Figure 3 we can see the output of Simple Disclosure analyzer - the visibility matrix.

A visibility matrix provides an overview of the data objects that each party possesses at some point along the process execution. It also gives the extent of data visibility to each of the parties. The parties observe the content of data sent to them or computed by them, but some data objects hide the actual data that they encode (e.g. through encryption or sharing stereotypes). The analyzer considers PETs that provide data confidentiality protection and PET computations that give protected outputs as producing the data objects that hide their underlying content. Such analysis can also be carried out on the general stereotypes that simply specify if a data object is protected or not.



mpc.bpmn - Simple disclosure analysis report

#	key	MPC output, MPC output [encrypted]	noisy result	party A data	party B data
Party A	V	V	-	V	-
Party B	-	V	V	-	V
Secure storage	-	H	-	-	-
Some result server	-	-	V	-	-
Shared over	-	MF	MF	-	-

V = visible, H = hidden, MF = MessageFlow, S = SecureChannel

Close

Figure 3: Sample of the Simple disclosure report

The columns refer to all of the BPMN data objects in the process model, while the rows correspond to the BPMN pools. The possible values in a matrix cell are the following:

1. 'V' - visible, the party has an access to the data object.
2. 'H' - hidden, the party has access only to the protected (cryptographically encrypted or secret shared) version of the data object.
3. '-' - the data object is not disclosed to the party.

In the lower row of the table it is mentioned the channel, used to disclose the information between pools. Here 'MF' stands for Message Flow and 'S' stands for applied SecureChannel stereotype.

In Figure 4 we can see the output of Data dependencies analyzer.

mpc.bpmn - Data dependencies

#	MPC output	MPC output [encrypted]	key	noisy result	party A data	party B data
MPC output	#	-	-	-	D	D
MPC output [encrypted]	D	#	D	-	I	I
key	-	-	#	-	-	-
noisy result	D	-	-	#	I	I
party A data	-	-	-	-	#	-
party B data	-	-	-	-	-	#

How is the element in the row dependent on the element in the column: D = directly, I = indirectly

Toggle transparency Close

Figure 4: Sample of the Data dependencies report

It shows whether and how the data objects are connected through the processing flow of the business process model. For instance, we see that the encrypted version of MPC output is directly dependent (D) on the raw MPC output, which means there is a processing task, which takes the data object with raw MPC output as input and produces encrypted data object. If the task does not produce an output right away but shares the data over the message flow, and the data is produced on the receivind side, it is also considered as a direct dependency. If two data objects are connected via the path comprising multiple direct dependencies, they form an an indirect dependency (I) between the first and the last data object in the sequence. We will use these outcomes of Data dependency report for Extended Simple disclosure analysis, which is described in Chapter 4.

2.4 Information Flow Analysis

In [ALL15] authors implement a method and related tool called Anica for discovering of the information leakages between domains (parties) of the business process model. To that end, they consider the Petri net representation of the business process model as a basis for the analysis.

The proposed approach and tool Anica takes the Petri net representation of a process model as input and produces a report containing a set of places where information leakage is possible. This process consists of two steps.

First, the activities of the process model are labeled as belonging to one of the security domains: 'high' for secret and 'low' for public domain. Authors observe the strategies for the analysis of concurrent process executions or looping Petri nets. We note that tool is working with looping models as they are, without running the unfolding techniques.

Second, Anica takes the labeled Petri net and automatically creates a series of reachability problems, whereas the underlying model-checking tool solves these problems and returns the corresponding witnesses [ALL15]. By narrowing down the analysis to the reachability problems, authors employ a broad variety of verification tools and techniques inherent for the Petri net reasoning.

The advantages stated by authors are the tool support of the complex process models (tested on a dataset of process models) and fast execution. However, the outcomes of the analysis are boolean and high-level. The tool is more oriented on Petri net representation, than on proposing the specific alterations to the BPMN itself. Also there is no insight for the user about the qualitative and quantitative measures of the found leaks and the conditions in which they occur.

2.5 SQL Collaborative Workflows

BPMN collaborative models can possess some sensitive information about one of the parties, that is why the emphasis is put on the inter-pool communication. In the live business represented by the process model, data is usually stored in SQL databases, so in [DGL18] the research group introduced a concept of SQL workflows in BPMN.

In SQL Workflow, each BPMN task or data object corresponds to a SQL statement executed against a database. Each SQL statement either defines a new SQL table or executes a query against a set of the input tables [DGL18]. The query execution always produces new SQL tables, or the output, which can be used as an input by subsequent tasks of the workflow.

The table (or set of tables) that are taken as input by the first SQL statements in the workflow are called the inputs. Conversely, the tables produced by the last SQL statements in the workflow are called the final output(s), while the tables produced by intermediate tasks in the workflow are called intermediate outputs.

The SQL Workflow associated with collaborative business process model becomes SQL collaborative workflow. Each of the BPMN Pools represent a separate SQL Workflow. However, they are dependent on each other because of communication channels: the task from one pool can take the SQL table produced by another pool as an input.

There is a technique proposed for 'flattening' a SQL collaborative workflow into a single-run SQL workflow and 'unfolding' it into a set of acyclic SQL workflows without conditional gateways. The latter set of SQL workflows can then be analyzed using the Leaks-When analysis technique described in section 2.8.

2.6 Differential Privacy Analysis

Differential privacy [Dwo06] is a modern privacy approach for security experts. The idea behind it leans on the guarantee, that including or excluding of any individual entry in a sufficiently large data set does not significantly affect the results of statistical analysis of the given data set [SDSM17]. This is an important notion for privacy-aware (e.g. statistical) databases, where only aggregated results are allowed to be disclosed because of sensitive information protected by law, for instance, medical records or police/criminal records. There has emerged the concept of altering the impact of concrete entry on the whole statistics operation in such way as this entry was not included in the data set. Nevertheless, providing this strict requirement in practice can distort the data significantly and limit data uses, deteriorating the endpoint utility of these differentially private results.

As a simple example, we can use the following scenario. Let's assume we want to execute a query over the database asking 'Does this entry possess an attribute A?'. We could simply output 0 and 1 values to such query and it would yield a one-number statistics about attribute. But if we allowed to run queries on a custom UserId range then the malicious party can figure out the value of exact row. As an attempt to overcome this issue, we could run the following simplified algorithm before giving the value to query. We throw a coin (run a random function). If we got head, then we yield a real value, otherwise we throw a coin again. This time if we get head, we yield 1 value, otherwise 0. Now any single result given to the query is not always true. However, knowing the probability distribution, it is possible to adjust the function (algorithm) so that the query yields the same statistical result, but aggregated from altered single values [SDSM17].

In general, ε-differential privacy is a method designed to preserve the privacy between databases that differ only in one entry. This means that no adversary 3rd party with some general knowledge about database can figure out if one specific participant submitted his information. The differential privacy model enables to quantify and keep under control possible data leakages when accessing sensitive information. Researchers in this area are concentrated on designing algorithms to preserve the output ε-differentially private objects in data sets given some privacy budget (usually it requires certain resources for adversary party to penetrate and analyze the database).

However, it is arguable that indistinguishability between any pair of results is required. In [SDSM17] it is proposed that protection towards neighboring subsets is sufficient, but leads to significant accuracy loss. That means significant data distortion, which not only hides a real statistical picture but can form an impression of stable system, because this distortion does not always appear.

So the authors introduce the concept of individual differential privacy, an alternative that allows to adjust the final distortion and thus to control the tradeoff between statistical mutilation and privacy level. The specific mathematical mechanisms in this study include the usage of background concepts such as Laplace noise, Global Sensitivity, Local Sensitivity and Smooth Sensitivity [LPP18, NRS07, AFG16].

In the development of an enterprise software, operations on data are commonly executed within business processes consisting of complicated chains of tasks and data objects and involving a variety of IT and human parties [DGL16]. Preserving privacy in this case needs us to analyze the flows of disclosures to a range of parties. When the input is being used by multiple parties and tasks, this is seen as a special use case and requires specific technique of composition rules to figure out the sensitivities of chained tasks that may produce bulk output of different types. So the main goal there is to calculate end-to-end differential privacy for the provided business process. That means we do not try to deal with single queries, but rather to concatenate them into one SQL workflow and work with the input and output of the entire workflow. The standard technique of making data source release securely is adding noise to the output, which is based on custom randomized algorithms like aforementioned example with coin throwing.

To analyze privacy-aware business processes in terms of differential privacy, researchers in [DGL16] introduce a graphical notation called data processing workflow. This is a simplified process modeling notation focused on representing how data inputs are given to the business process and then translated into intermediate and final outputs. Another concern is multiple parties to which the outputs are disclosed during the execution flow.

To understand the concrete interest in this workflow we also have to observe what is sensitivity. In scope of differential privacy, having some statistical function f, sensitivity defines how much the output of the f will change if we affect specific attribute. It makes sense both from the point of view that, for instance, one attribute can be represented in multiple database entries, or the function can estimate certain attributes with higher weight. In ultimate case, if we have only one entry in the database, then altering this row will change the whole output of almost any function. So in [DGL16] there has been designed an algorithm that calculates an upper bound of the data disclosed when the node is accessed by a user endowed with a defined role in the business process. The algorithm also calculates the differential privacy and sensitivity parameters of each intermediate and output data node in the privacy-enhanced workflow. The aforementioned fundamentals of the workflow analysis technique has been employed in the PLEAK tool, for instance, in Guessing Advantage analyzer.

In [LPP18] authors introduce a concept of derivative sensitivity for continuous functions. They use this concept to determine the amount of noise to be added to the outcome of a database query in order to reach a certain level of differential privacy. The research demonstrates that derivative sensitivity enables to employ methods from calculus to carry out the analysis for a variety of SQL queries. Authors also propose the metrics different from the 'number of changed rows' metric. Their metrics can depend on the location and amount of changes, which makes the approach more flexible for the data owner to determine that the distance between the queries output is not larger than defined acceptable level.

Other studies on privacy analysis of business processes employ Petri net reachability analysis. Translating BPMN process to a Petri net might be non-trivial task in a generic case, because of many interac-

tion cases such as multilane and special occurrences like cycles in BPMN. However, after translation it is much easier to traverse and analyze the workflow since there are only places and transitions instead of complicated hierarchy of BPMN elements. There has been used model checking to reveal data objects that are disclosed to third parties [FGF08]. An advanced multi-level privacy model divides the objects of the system into security levels. Provided techniques attempt to identify the path where the data from an object of a high security level is transferred to an object with lower security level. The drawback of such techniques is boolean nature: the output in form of a table only displays whether the attribute is disclosed in low security level or not. They do not assess the quantitative component of leakages but only reveals the fact of disclosure.

2.7 Sensitivity Analysis of SQL Queries

Differential privacy paradigm often can be supported by employing the aforementioned sensitivity concept. Depending on concrete statistical database, the main step is usually to figure out a function which is supposed to be used for statistics calculation and adding noise to the output. The differential privacy value results in the ratio of the function sensitivity for different attributes and the magnitude of the generated noise [LPR18]. Since the differential privacy and sensitivity analysis are supposed to be carried out on statistical databases, it is deemed to develop and run those techniques on SQL queries and compose those queries to SQL workflows.

Algorithms for differential privacy are mostly suited for SQL queries that return numeric data types, because the addition of noise to numeric variables is more straightforward. Therefore, the focus of investigation has been on queries that eventually return one single number - the number of rows returned by an SQL query for specific dataset. So the result of the sensitivity analysis can be utilized to calculate the amplitude of noise that has to be added to reach the differential privacy for the given query.

In [LPR18], the authors propose an analysis of SQL queries for their sensitivity. In the proposed approach, the inputs are a query Q and the database schema dbs produced by the given set of SQL statements. The internal formula ϕ contains the model for applying the noise to the attribute only if the related sensitivity of the query Q is less or equal to N. Z3 SMT solver[9] is employed to assure that ϕ contains a model given the specific N[dMB08]. By adjusting N, we can figure out the exact sensitivity of the query by checking that model is working. Still there is a limitation in case the Z3 SMT solver does not give us definite response or the sensitivity is too high. In the latter case computation is aborted and the estimation is set to 'infinity'.

The implementation of the analyzer has been done using the Haskell programming language. The front-end part of the analyzer does not allow to use all SQL features, so the queries and schemas are often adjusted to meet this limitation in a non-substantial way. These adaptations do not change the overall logic of the calculations and do not affect queries and their sensitivities. An important aspect of the analyzer is the performance concern. To yield the precise value of the sensitivity, the linear search is executed attempting a sequence starting from 0 and increasing by 1 at each step until it is found that the value N is where the sensitivity reaches the maximum. The linear search is used instead of binary because the execution time grows rapidly as the value increases. In general, it is faster to check all n from 0 to N-1 than a single N value.

The highest finite sensitivity value calculated which terminated in less than 5 mins. was 9. However, for the most of tested SQL queries with the sensitivity of 9, the analyzer did not terminate within 5 mins. Therefore, the highest finite sensitivity value, which is supposed to be confidently computed for a simple query is approximately 8. Once the Z3 solver exceeds 5 mins. span for a single query, we force the process termination and save infinity as an upper bound for the sensitivity. The linear search is terminated not regarding the time once all of the N values up to 10 are checked, because it is not feasible to await the calculation result. In this case we keep an assumption that the sensitivity is greater than 10 and display it as 'infinity'. This makes revelation of the infinite sensitivity feasible and fast.

[9] https://github.com/Z3Prover/z3

As we can see, Sensitivity Analysis has reached some progress with regard to SQL flow analysis. However, due to the current limitations this field is open for improvements from many sides. Along with differential privacy, sensitivities can be employed for the Section 2.9.

2.8 Leaks-When Analysis

Leaks-When analysis is a technique and a related tool within the PLEAK tool that takes as input a SQL workflow and reveals the data and the conditions under which such data would be disclosed during the process execution. In this context, we consider a SQL workflows where BPMN tasks and data objects contain SQL statements. The Leaks-When analysis can be used to provide insights on the conditions over which the data is disclosed to parties that are having access to the intermediate and/or final results in the SQL workflow computation. The tool generates an intermediate process calculus in the field of relational algebra, using OCaml programming language. This intermediate layer is then used for the analysis and building the summary dependency graph. These graphs are being processed and displayed as a Leaks-When report.

In Figure 5 we see the flattened 'Aid Distribution' scenario. In this scenario, a country requests for an aid after a natural disaster and expects the goods to be distributed by the ships, which belong to different Aid providing countries. The disclosure of data about ships to the Aid requesting country is done incrementally.

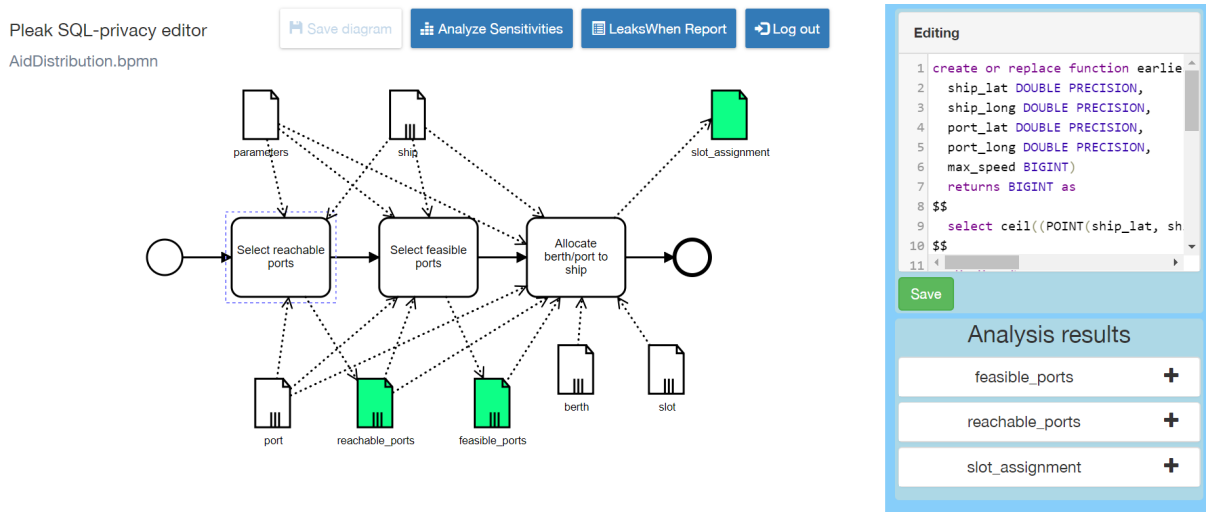


Figure 5: Flattened 'Aid Distribution' scenario

To carry out the Leaks-When analysis, user should select one or more output data objects (the ones produced by the tasks or events) and click the "SQL LeaksWhen" button. The output sidebar shows one tab for each of the selected data objects. In each of the tabs a separate report is generated for each SQL attribute in the output SQL table. The report is generated by applying the dataflow analysis techniques over the SQL workflow in order to build a summary dependency graph [DGL18].

A sample of a Leaks-When report in form of a dependency graph is demonstrated in Figure 6. The figure reveals the conditions in the SQL workflow execution for the SQL query attached to the 'Select reachable ports' task. The operations order is the following: the process calculates the distance between ports and ships and divides the result by ship's maximum speed. This gives the travel time to the port from the current point. Then we filter out those ships, that do not fit into the given deadline, and then we filter out those ships, that are not among allowed. Therefore, a branch marked with '1' and leading to the 'Filter' node is 'What leaks' part, whereas a branch marked with '2' is 'When leaks' part. Each Leaks-When graph has a final 'Filter' node, while the rest of the graph summarizes the computations in the SQL queries.

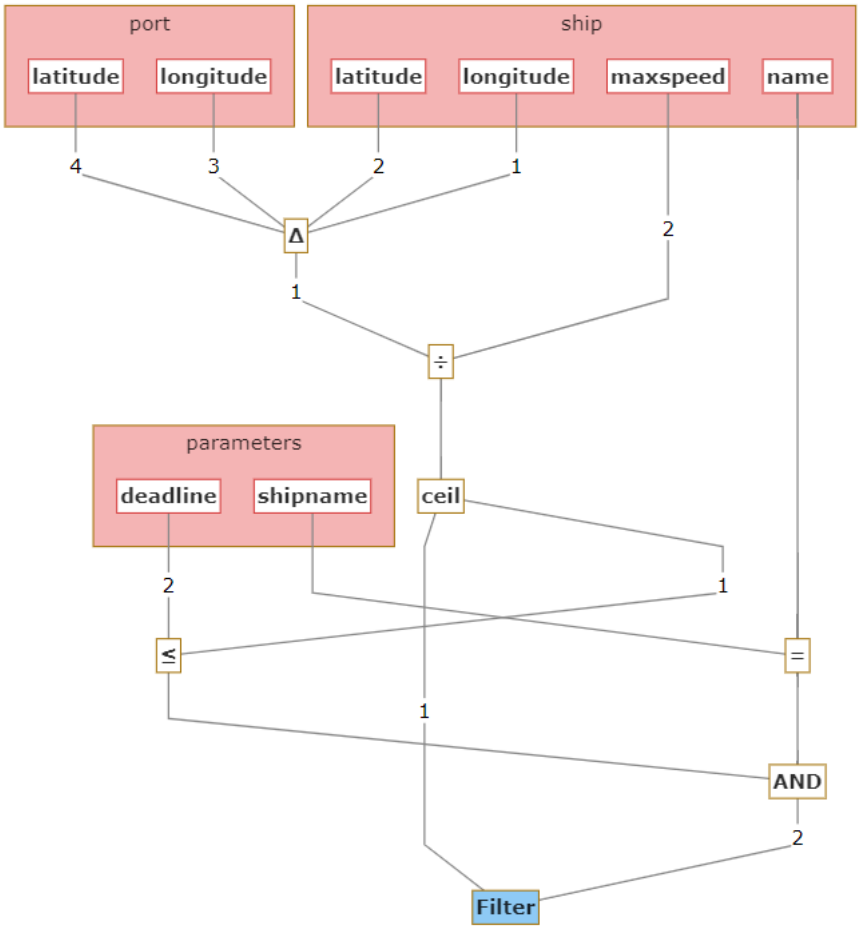


Figure 6: Sample of the Leaks-When report

2.9 Attacker's Guessing Advantage

The sensitivity of a function can be used directly to calculate the noise required to achieve ε-differential privacy as defined in Section 2.7. However, it is in general not clear which ε is sufficient, it's 'goodness' depends on the particular data and the query [TTY+19].

Attacker's guessing advantage is a more standard security measure. Formally, it is defined as a difference between the prior probability (before observing the output) and posterior probability (after observing the output of the SQL query) of adversary party (attacker) to guess the input data.

Such measure gives an insight to the user about how much the adversary party can infer about the input SQL data after observing the output, considering the knowledge it possessed before. In [TTY+19] authors describe that internally, PLEAK tool is still executing the sensitivity analysis of a query, but represents the analysis result in terms of guessing advantage. The main input of the Guessing advantage analyzer is an upper bound on attacker's advantage, which ranges between 0% and 100%. To define the attacker's parameters more precisely, user specifies the prior knowledge of the attacker about selected SQL attribute with defined precision range for each of the attributes. This can be done by selecting an upper and a lower bound on the SQL attribute. The Guessing advantage analyzer automatically converts these values to a suitable ε for differential privacy, and then figures out the amount of noise required to reach the bound on attacker's advantage.

The guessing advantage analysis is enclosed in Guessing Advantage Editor in PLEAK. It accepts simple BPMN models with single task outputting a single value - the result of a SQL aggregation function such as COUNT or SUM.

In Figure 7 we observe a process model that calculates the number of ships that can arrive to the

given port by the given deadline. The sensitive SQL attributes are defined in a separate parameters section. Here we define that ship's latitude and longitude are not allowed to be acknowledged within the radius of 5 units. The prior attacker's knowledge is about ship's coordinates range (area of interest) of [0..300] and possible maximum speed range of [20..80].

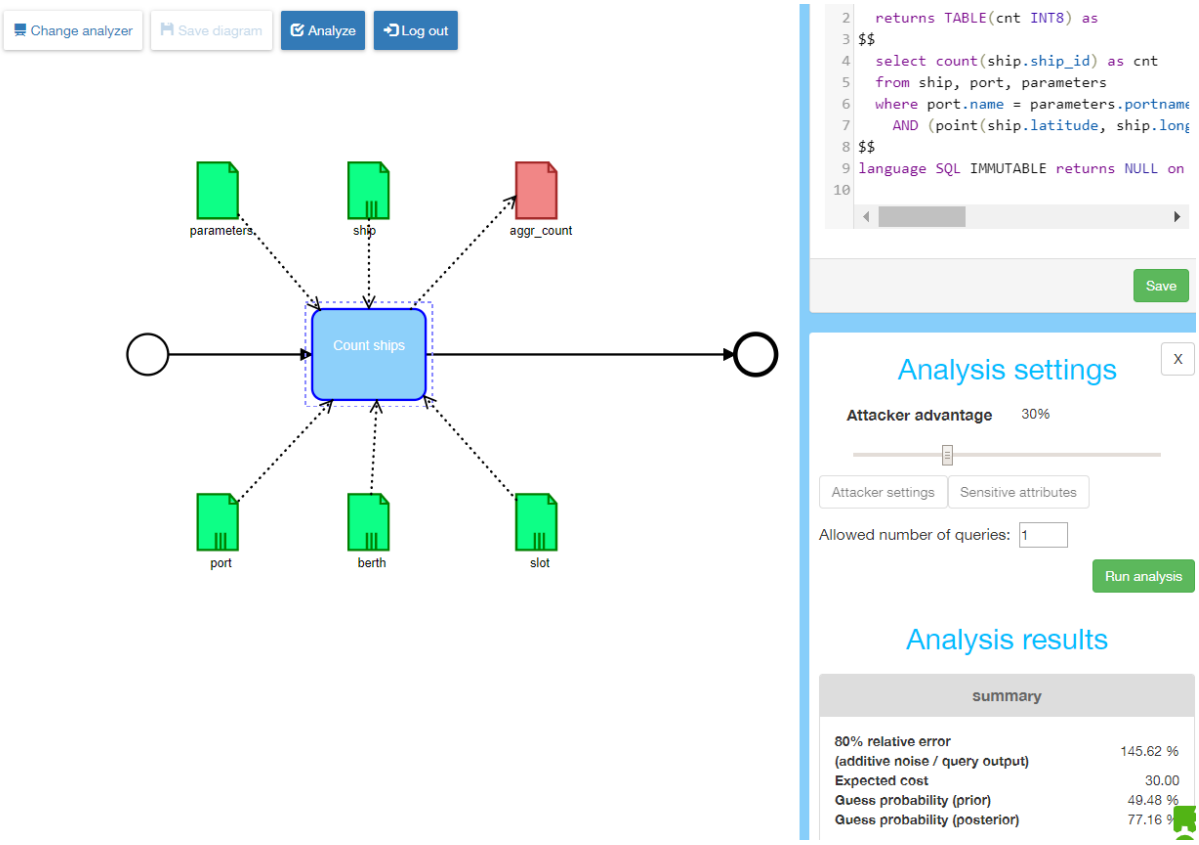


Figure 7: Guessing Advantage editor

The attacker's goal is to figure out the coordinates of the ship with a precision of 5 units. The analysis output says that, if user wants to limit the guessing advantage by 30% using the mechanism of noise addition, than the relative error of the output will be 145.62%.

The '80% relative error' of 145% would tell the user that, with the probability of 80%, the outputs of task aggregations due to the added noise will be between $-0.45 \cdot y$ and $2.45 \cdot y$ for an aggregation y. The cost becomes more interesting if we define different levels of the privacy violations, assigning different risks to different leakages.

To observe the full output of the analyzer we have to click 'View more'. The 'prior' probability of 49.48% says that by assumption the attacker would already have guessed a ship's location with the probability of 49.48% before he even has seen the aggregation outputs. Where does this number come from? Since the user has not specified any prior distribution of ship's locations himself, the Guessing Advantage analyzer assumes the worst-case prior distribution, where the advantage increases the most, so that the analysis result is correct for any possible prior distribution. E.g. if the prior distribution was actually less than 1%, then the posterior would be less than 31% with the same amount of noise.

The 'posterior' probability is a value that ranges between 'prior' and 'prior + advantage'. While the user is interested in keeping the guessing advantage below 30% after observing counts, observing counts results in posterior probability $77.16\% < 49.48 + 30\%$.

2.10 Summary

Summarizing the related works, we can notice that there are different approaches to enhance the business process models with regard to privacy. There are also analyzers and approaches to analyze the sensitive data in terms of the SQL workflows. However, we do not observe the privacy policies regarding the sensitive information, mostly because the analyzers are dissociated and require different input. In the following chapters, we will take the Simple disclosure analyzer, the Leaks-When analyzer and the Guessing Advantage analyzer as relevant existing tools for comprehensive multi-level privacy analysis and analysis of the collaborative process models. Therefore, the main focus in this thesis is made on incorporating the privacy policies into the business process models, integration and extension of existing analysis tools.

3 PRIVACY POLICIES IN SQL COLLABORATIVE WORKFLOWS

In this chapter, we provide the outline for an extension of a disclosure analysis technique. In Section 2.5 we described the principles of SQL collaborative workflows. In such cases the business process is executed over the SQL tables and SQL queries. BPMN data objects are associated with data definition SQL statements, such that they define a table schemas, whereas BPMN tasks are annotated with SQL queries. Each task takes the input data objects, executes the query against input tables (data objects) and produces an output/intermediate tables, represented graphically as output data objects. An output table can be then used in subsequent tasks of the workflow. In a SQL collaboration, each one of the parties involved in the collaboration is represented using the a BPMN pool.

In the previous version of the Leaks-When analyzer, the analysis is performed without taking into account the set of roles that are participating in the collaboration. Even if a SQL collaboration was specified with multiple pools, the analysis disregards the pools and there was one single Leaks-When report with all of the SQL operations belonging to all of the parties in the SQL collaborative workflow.

In this version, we have lifted up the aforementioned constraints. Therefore, we improved PLEAK's SQL-privacy editor such that it is possible now to specify privacy-related policies, to analyze and report any eventual undesired/unexpected disclosure of sensitive data. During the Leaks-When analysis, we run the policy compliance checking and highlight the potential disclosures in the Leaks-When report with respect to selected party.

As a basis for running example, we adapted an 'Aid Distribution' scenario demonstrated in Figure 8. This business process encompasses a set of activities split between Aid providing country and Aid requesting country. The scenario implies an incremental disclosure of the sensitive information, such as coordinates of the marine ships, which are transporting the goods provided by international community. The goal is to disclose certain data only when it becomes allowed by the collaborative process constraints.

In the process model the Aid requesting country first shares the available ports and the deadline for the goods delivery using the 'parameters' table. Then Aid providing country checks what ships can reach the available ports by the given deadline and shares the requirements for available ships to be able to allocate it to the ports. These requirements are cargo and length of the ship. For convenience we call this part of the SQL workflow a 'Reachable ports' step. To see the definitions of the SQL tables and SQL queries in this step check out Appendix 1.

After that Aid requesting country selects feasible ports, depending on the ports' harbor depth and offload capacity, it sends this filtered list to Aid providing country and then Aid providing country eventually can select the most fit ship for the most feasible port (e.g. the closest) with right dimensions to allocate it to the free berth of the selected port. We call this step 'Feasible ports'. Its definitions and related SQL statements are located in Appendix 2.

In the next phase the Aid Providing country selects the most feasible ship (which will arrive sooner than others) and discloses all of the data about the selected ship to the Aid requesting country, including ship name, coordinates, id, cargo, draft and maximum speed. See the Appendix 3 to check the related queries.

In the final step the Aid Requesting country schedules the arrival of the selected ship to the port. To that end, it orders the slots of a specific berth by availability and size and assigns the ship to the first available slot. The full logic can be observed in Appendix 4. All aforementioned disclosures are supposed to be made via BPMN message flow.

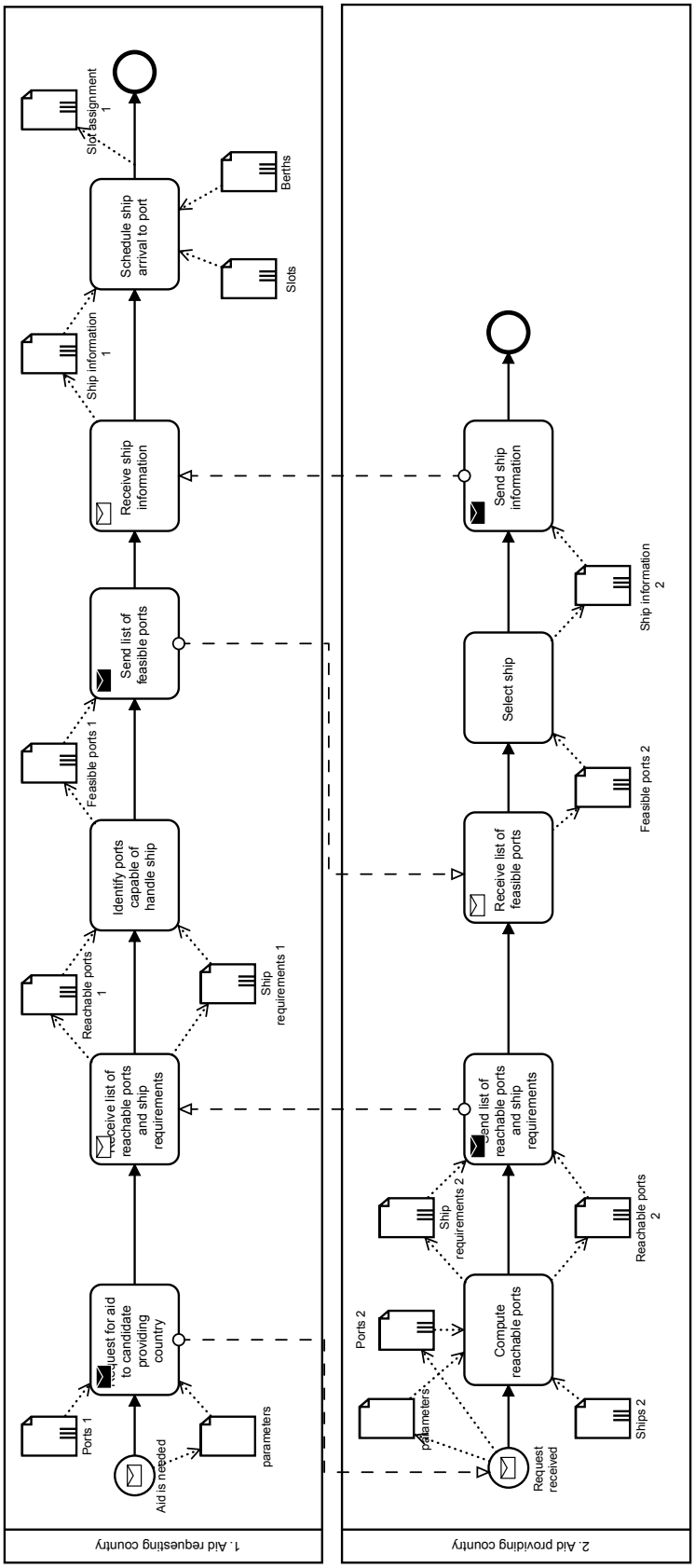


Figure 8: 'Aid Distribution' scenario captured as BPMN collaborative model

3.1 Privacy Policies

Privacy policy is a statement that designates the rights of a specific party regarding the access to the selected data sources. Privacy policy allows to distinguish whether a given data disclosure should be considered a leakage. Also we suppose that it can define the rights of interaction between parties of the business process and, in particular, enhance the Leaks-When analysis. The main issue of the final report of the Leaks-When analysis described in Section 2.8 is a necessity to manually discover the potential process violations coming from the permissions of the parties. If anything in the SQL workflow has been disclosed from one BPMN pool to another by BPMN message flow, user had to figure it out by him/herself. To handover more control and automation to the analyzer, we added a support of the privacy policy extension.

3.1.1 Privacy Policy Notation and Binding in PLEAK

We implemented an extension to the SQL policy specification, so the privacy policy is represented by the SQL 'grant' statement and serves as an established constraint for the data sharing during the process execution. In the SQL collaborative workflows we assume that the concept of a party corresponds to the BPMN pool. The parties to be used in the SQL 'grant' statement are extracted from the BPMN pools' names of the business process.

Since all of the disclosures in the SQL workflows are currently occurring through the 'select into' statements, the policy gives a permission for the selected party to run 'select into' queries over the specific attributes of the SQL table or the entire table. Such approach is equal to the 'read' access right. By default we assume that all of the SQL tables are prohibited for the read access.

As a starting point of building up the policy for specific party we can use the context menu of the BPMN pool and attach an SQL script. In the 'Aid Distribution' scenario we acknowledge that the 'parameters' table is fully accessible from the beginning of the process execution and to the end event. We call such policy rules 'Global' and attach them to the BPMN pools.

Figure 9 demonstrates such a Global privacy policy rule.

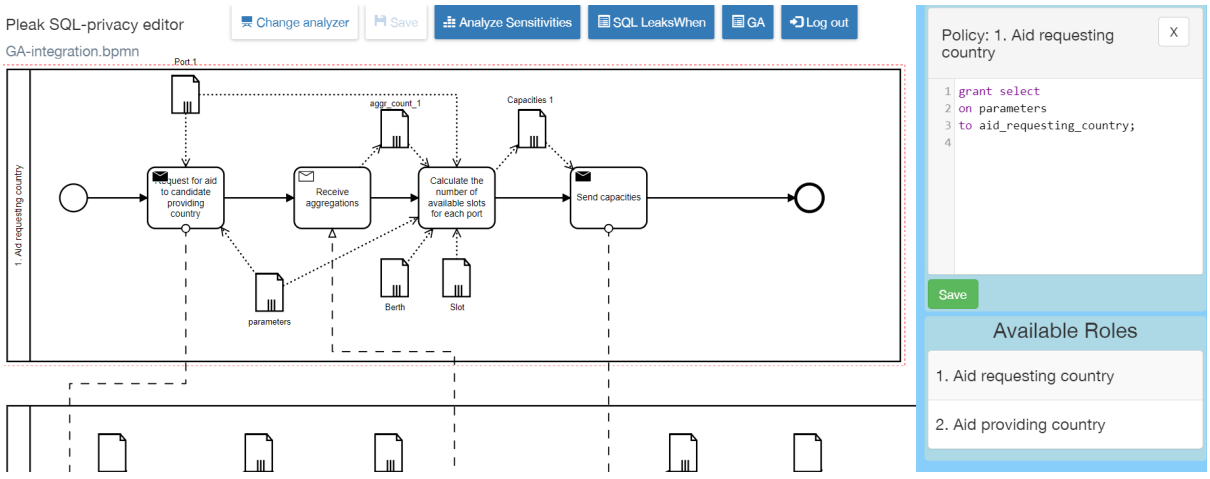


Figure 9: Global policy rule

Since the names of the BPMN pool may appear too diverse and contain the undesired symbols, we assume that the inner representation of the party name is a subset of literal words in a lowercase joined with an underscore. For instance, the '1. Aid requesting country party' must be mentioned as an 'aid_requesting_country' in the policy rule.

In the 'Aid Distribution' scenario it is not secure to disclose all of the sensitive information at once, that is why an incremental approach has been chosen. This approach also implies that during the process execution the permissions of the party may extend. Hence, additional the policy rules can be attached dynamically. To achieve this the process creator can select an output data object of a task and assign

an SQL script with the 'grant' statement of a policy rule. By doing this, we infer that to produce the selected output without violations a party is endowed with this additional policy rule which will hold until the process end event. We call such a policy rule 'Local' and assign it to the output data objects of the BPMN tasks.

Figure 10 demonstrates such a Local policy rule.

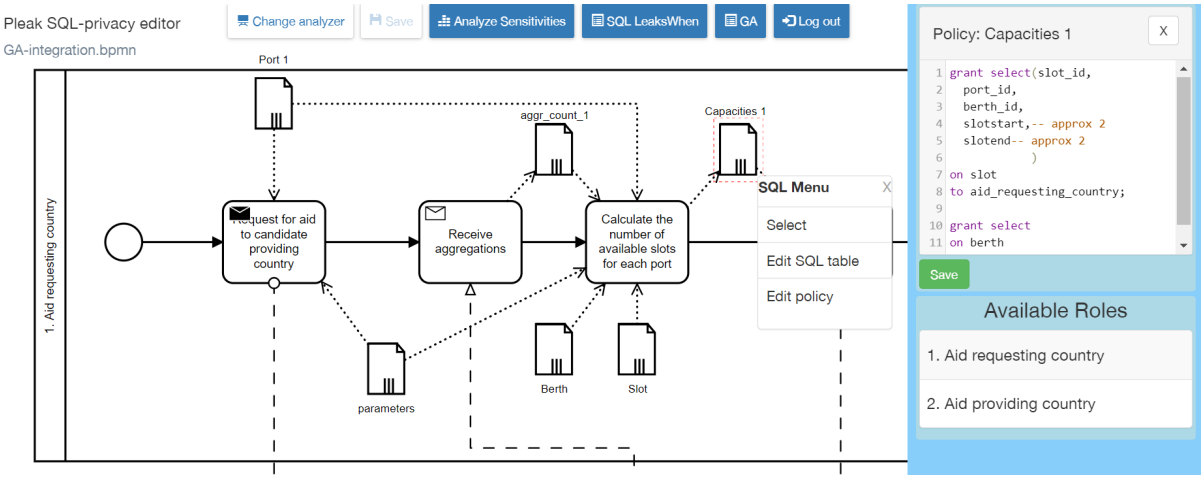


Figure 10: Local policy rule

Hence, in addition to the incremental disclosure of the sensitive attributes we implemented incremental extension of the policy ruleset for the party during the process execution. Such rules are allocated after some point of the process passed and hold until the process end. For instance, to produce the 'Capacities 1' data object, the task 'Calculate the number of available slots for each port' is given the Local policy, which provides the 'Aid requesting country' with a legal access to 'Port' and 'Berth' SQL tables. Before that point (e.g. in 'Receive aggregations' task) the usage of anything related to 'Berth' or 'Slot' would be considered a policy violation.

3.1.2 Privacy Policy Syntax and Limitations

In Figure 11 we provide a formal representation of the privacy policy syntax in the Backus-Naur form, demonstrating a full set of capabilities of the privacy policy.

$$
\begin{array}{rcl}
\textit{privacyPolicy} & ::= & \textit{privacyPolicyRule EOL} \\
& | & \textit{privacyPolicyRule EOL privacyPolicy} \\
\textit{privacyPolicyRule} & ::= & \textit{grant select EOL on } \{\textit{tableName}\}\ \textit{EOL to } \{\textit{partyName}\}\texttt{;}\ \textit{EOL} \\
& | & \textit{grant select}(\{\textit{sqlAttributeName}\}\ \textit{sqlAttributesListTail EOL})\textit{EOL} \\
& & \textit{on } \{\textit{tableName}\}\ \textit{EOL to } \{\textit{partyName}\}\texttt{;}\ \textit{EOL} \\
& | & \text{""} \\
\textit{sqlAttributesListTail} & ::= & \texttt{,}\ \textit{EOL } \{\textit{sqlAttributeName}\}\ \textit{sqlAttributesListTail} \\
& | & \texttt{,}\ \textit{sensitivityParameter EOL } \{\textit{sqlAttributeName}\}\ \textit{sqlAttributesListTail} \\
& | & \textit{sensitivityParameter} \\
& | & \text{""} \\
\textit{sensitivityParameter} & ::= & \texttt{--}\ \texttt{approx}\ \{\textit{decimalChar}\} \\
& | & \texttt{--}\ \texttt{exact}\ \textit{EOL}
\end{array}
$$

Figure 11: Privacy Policy syntax

In order to be parsed and applied correctly, the privacy policy must use a SQL 'grant' script in specific manner demonstrated in the listing 3.1.

Listing 3.1: Example of the privacy policy rule using SQL "grant" statement

```
grant select(ship_id,
             latitude, -- approx 5
             longitude -- approx 5
)
on ship_2
to aid_requesting_country;
```

In case of the specification of concrete SQL attributes the first attribute must be placed on the same line with the 'select' keyword following other attributes each on the new line. Additionally, we added a support of the sensitivity parameters as a part of the privacy policy. They can be added in the comment section and do not break the SQL structure. In this example, ship's latitude and longitude are not allowed to be disclosed with less than 5 radial noise. This sensitivity setting is specified by the 'approx' parameter followed by the number of the approximation units. Another option is 'exact', which implies that only a successful guess of the exact value is supposed to be a disclosure. Only integer (INT8), floating point (FLOAT8), text (TEXT) and boolean (BOOL) data types are currently allowed for an attribute which is used with the sensitivity parameters. The syntax used in the script is that of PostgreSQL.

3.2 Changes in the Leaks-When Analyzer

We have implemented a set of changes in the Leaks-When analysis to make it work with policies:

1. The Leaks-When analysis is executed with respect to the party-holder of the selected output data object. If we want to perform the Leaks-When analysis for the output data object A, the Leaks-When will assume that the owner of the SQL collaborative workflow is one attached to the BPMN pool possessing A.
2. The policy rules are aggregated from the BPMN pool of the party-holder up to the selected data object. To have the right ordering of the privacy policy rules we employ the Petri net representation generated for the business process using techniques described in Section 4.2.
3. The Leaks-When report highlights potential disclosures with colors.
4. We added the support of the SQL aggregation functions.

The Leaks-When analyzer uses the specification of the privacy policy rules to carry out the verification over the SQL collaborative workflow. If several data objects are selected, then the analyzer will still execute role-wise analysis for each of the data objects. In the Leaks-When report the edges are colored with a red color in case of 'Direct' disclosure, when the attribute is referred explicitly in the 'select' statement. Orange color is used for 'Indirect' disclosure, when attribute is used in the calculations, over the SQL aggregation functions or as a parameter of a user-defined function. In such case the attribute still could be inferred employing differential privacy technique.

Figure 12 demonstrates the Leaks-When report with the privacy policy highlights.

As we can see, all of the successors of the operation that might disclose the attribute are marked orange up to the root 'filter' operation. The attributes 'latitude', 'longitude' and 'maxspeed' from the SQL table 'ship_2' are not directly used in the final select (a.k.a. Filter in the Leaks-When report). The they are used in the calculation of the distance between the 'port_1' and 'ship_2', so there is a relation which can be employed by the adversary.

3.3 Extension of the Aid Distribution Scenario with SQL Aggregation Functions

The Guessing Advantage analysis described in Section 2.9 works exclusively with a single-value output produced by a SQL aggregation function. However, the 'Aid Distribution' scenario contains the state-

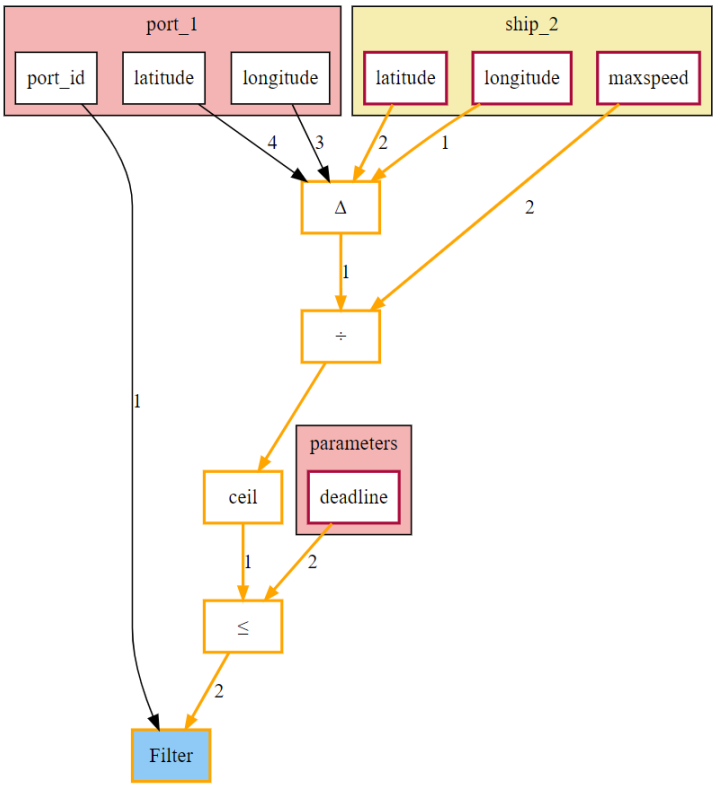


Figure 12: Leaks-When report with applied privacy policy

ments of different type. So we have to elaborate on building some scenario that will be applicable for both Guessing Advantage and Leaks-When analyzers.

In the 'Aid Distribution' scenario during the first step of the workflow called 'Compute reachable ports' we select a subset of the ships that can reach the requested port on time according to the 'parameters' SQL table. We send the information about their arrival times and cargoes to the Aid requesting country. However, if the port capacities are not big enough, some of the ships can be rejected because port cannot allocate so many of the proposed ships. Hence most of the disclosed information about ships is excessive.

For instance, Aid providing country possesses 15 ships that are ready and sufficiently close to the given port. However, the port can match as much as 5 slots available for allocation, while Aid providing country is unaware about this limitation and still discloses arrival times of all of the 15 ships. These arrival times are narrowing down the area where the ship can be present, especially if adversary party tries to match arrival times of the ships with their cargos. In case of only one port adversary can infer only the radius of the ship location, but using multiple queries with different ports it is possible to carry out the triangulation and guess the location more precisely.

We can diminish this issue by inserting another interaction step in the beginning of the process model. We decided to organize it as a separate business process. Figure 13 demonstrates an extension for the main model.

The aforementioned process model is also different from the main 'Aid Distribution' model because of presence of the SQL aggregation functions. The tasks 'Count the number of reachable ships for each port' and 'Count the sum cargo of reachable ships for each port' are using aggregation functions and does not disclose any single ship arrival date or cargo. We call this step of the process 'Calculate ship aggregations', the full SQL statements can be observed in Appendix 5.

After this step the Aid requesting country receives only aggregated count of the ships and their aggregated cargo for each port. Henceforth, Aid requesting country can match them with own port capabilities. All the main logic of this step is stored in 'Count the number of available slots for each

port' task. We call this step 'Calculate slot aggregations', its full SQL statements can be observed in Appendix 6.

After the task 'Count the number of available slots for each port' Aid providing country receives the capabilities of the ports and can now narrow down the list of the ships required to satisfy maximum of the the port slots. On the other hand, the information about currently available slots is also sensitive data, so if the number of available ships is smaller than the number of slots, the potential adversary party will know only about the matched number of slots. The ships entries are then reassigned to the new IDs to have them subsequent and ordered. It allows to filter out the redundant ships before disclosing more precise information about arrivals and handovering the result to the inital 'Aid Distribution' process. We call this step 'Match ships' nad its full SQL statements can be observed in Appendix 7.

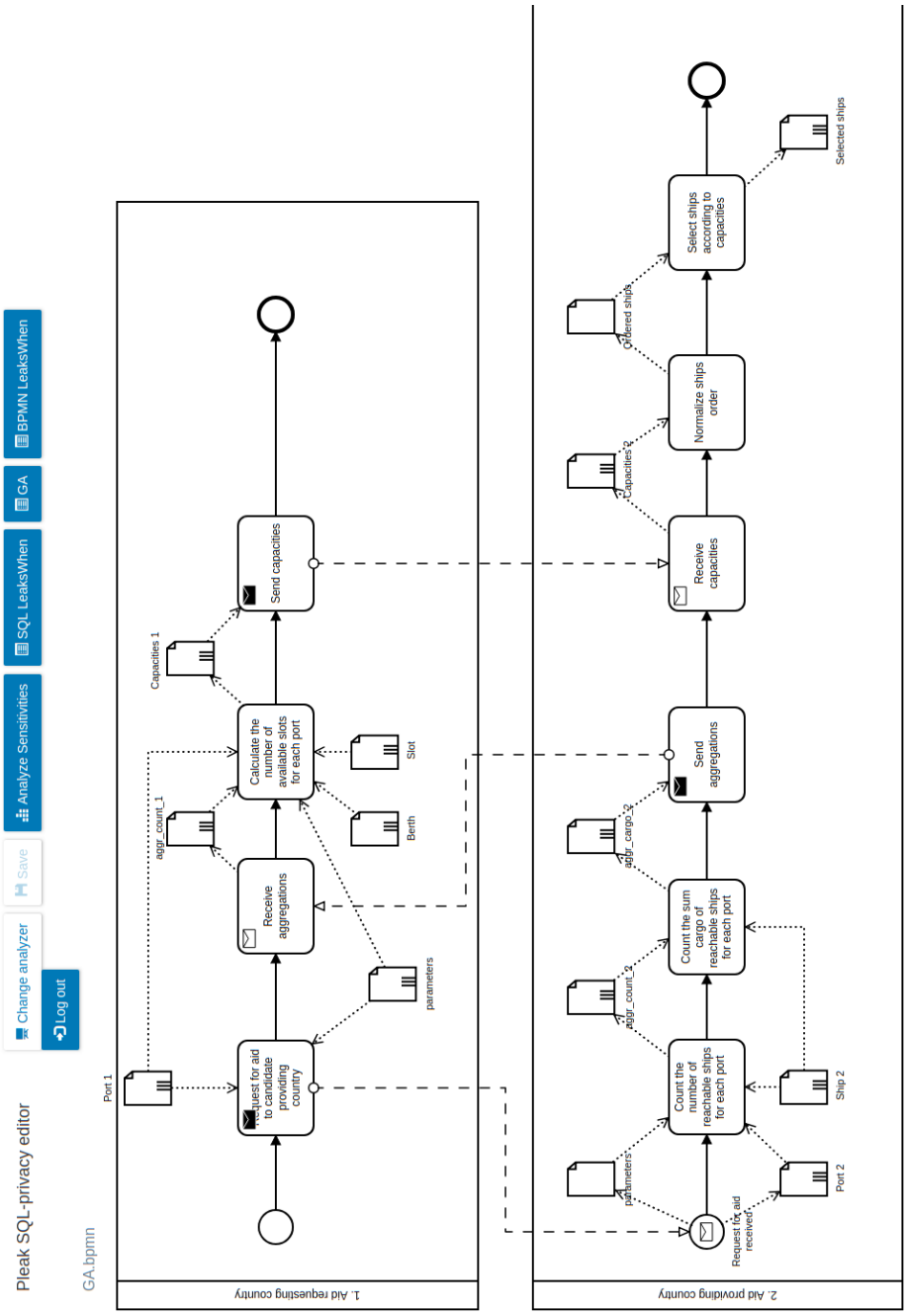


Figure 13: Extension for 'Aid Distribution' scenario

Let us observe the first step more detailed. Listing 3.2 presents the SQL script assigned to the task 'Count the number of reachable ships for each port'.

Listing 3.2: Task from step 1

```
1  create or replace function aggr_count(portname TEXT)
2    returns TABLE(cnt INT8) as
3  $$
4    select count(ship_2.ship_id) as cnt
5    from ship_2, port_2, parameters
6    where port_2.name = parameters.portname
7          AND (point(ship_2.latitude, ship_2.longitude) <@> point(port_2.latitude, port_2.longitude))
8          / ship_2.max_speed <= parameters.deadline
9  $$
10 language SQL IMMUTABLE returns NULL on NULL INPUT;
11
12 select p.name as name, res.cnt as cnt
13 into aggr_count_2
14 from port_2 as p cross join aggr_count(p.name) as res;
```

In this example, the script includes a user function `aggr_count` defined in lines 1-9 that computes the number of ships that can reach a given port by the given deadline having their coordinates and maximum speed. Since every task must be associated with at least one 'select into' statement the task writes the outcome of the computation to the temporary table `aggr_count_2` in the lines 11-13.

Currently supported group operations are following:

1. count - returns a number of rows with a non-empty given attribute; count(*) computes a number of rows with no respect to specific attribute, in current implementation takes an 'id' attribute of the SQL table.
2. sum - returns a sum of a given attribute of all rows.
3. avg - returns an average value of a given attribute of all rows.
4. max - returns a maximum value of a given attribute of all rows.
5. min - returns a minimum value of a given attribute of all rows.

In Figure 14 we observe the Leaks-When report for the task 'Count the number of reachable ships for each port'. It is very similar to the report depicted in Figure 6, but the result dataset is passed on to the SQL count function.

On this graph we select a subset of the records from the 'port_1' SQL table, which match the given 'portname' from the parameters. Then we match each of the selected ports with all of the rows from the 'ship_2' table and calculate the distance between the port and the ship. We divide each distance by the respective ship 'max_speed' and filter out those pairs, whose travel time exceeds the given deadline from the 'parameters' SQL table. The 'Filter' operation for the 'ship_id' attribute which precedes the final 'count' aggregation function is a special way of handling 'count(*)'. That is why it is marked with red color - it started to be used internally, without user's intention and policy rule. The 'GROUP BY' function is not explicitly used in the SQL query, but it is depicted in the Leaks-When graph because of the limitation of the internal tool for building summary dependency graph, it always requires 'GROUP BY' along with SQL aggregation functions like 'count'. The operations related to the 'latitude', 'longitude' and 'max_speed' attributes are marked with orange color, because they are not directly disclosed. But they still can be guessed using multiple queries with the different ports input, which stands for employing the differential privacy algorithms.

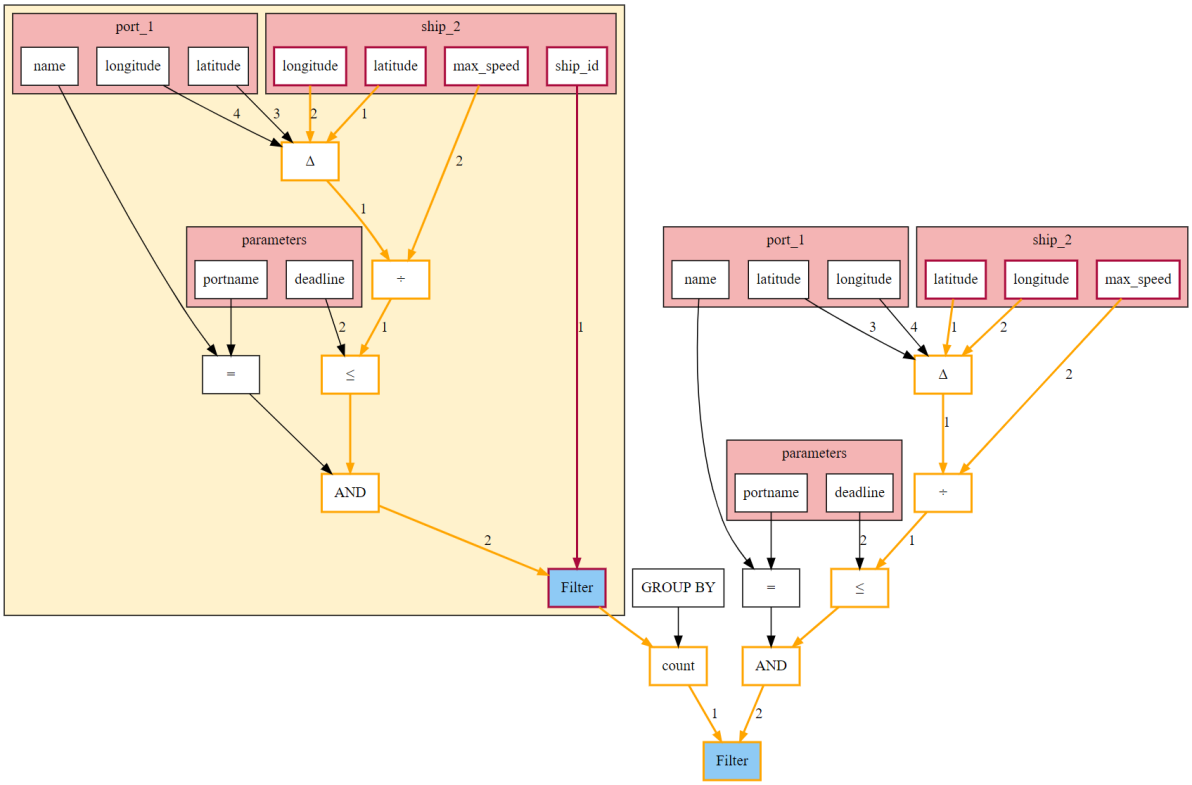


Figure 14: Leaks-When report for aggregation function with privacy policy applied

4 MULTI-LEVEL PRIVACY ANALYSIS

After introducing the privacy policies in the previous chapter, we proceed with an approach of three-level privacy analysis to be able to deal with privacy issues on different levels of granularity. We make a first step towards integrating of the existing PLEAK tools by extending them and establishing the different kinds of interaction between the levels of granularity in holistic privacy analysis.

4.1 Extended Simple Disclosure Analysis

The Extended Simple disclosure analysis is an upgraded finer-grained version of the Simple disclosure analysis, existing in scope of PE-BPMN editor. A Simple disclosure report reveals only the information about visibility of the data objects to the certain party. It also considers the encryption applied to the data during the data sharing. We want to extend the supported types of disclosure in the report. Also it has been decided to integrate the Extended Simple disclosure report into the SQL-privacy editor and with Leaks-When analysis as a first interaction of multi-level privacy analysis. The main implemented changes related to the Extended Simple Disclosure analysis are the following:

1. Integrated with a SQL-privacy editor and applied the analyzer to the 'Aid Distribution' process model with a SQL workflow described in Chapter 2.
2. Make a Simple Disclosure report more fine-grained by extending it with 'Owner', 'Indirect' and 'Direct' values.
3. Integrate with the Leaks-When analysis, producing a Simple Leaks-When report, built with respect to selected data object.

First, we describe the types of disclosure that can be displayed in the Extended Leaks-When report. The disclosure types of 'O' (Owner) and 'V' (Visible) are supposed to be a part of secure non-disclosive workflow, because they are clearly inferred from the process model pools and their communications. However, 'V' can still imply some violation, because the data object may be unintentionally visible to some party if there was an error during the process modeling. Conversely, 'D' (Direct) and 'I' (Indirect) can reveal a potential violation of the privacy policy.

Here we define the concept of Direct and Indirect disclosures. The data object O1 is supposed to be directly disclosed to the party B when it is used as an input for the computation of task T1 of the party A and the output data object O2 is shared between the pools. For the receiving party B we mark O2 data object as V (visible), while for the issuing party A we mark it as O (owner). We also mark O1 as D (directly disclosed). If party B has some direct computation over the shared O2 data object, then the output O2' is indirectly derived from O1, so we mark O1 as I (indirect disclosure) for the party B.

To address the user interest in a specific data object marked as 'D' or 'I', we integrated the Extended Simple Disclosure report with a Leaks-When analysis. User can select a specific cell in the Extended Simple Disclosure report and run the 'Simple Leaks-When analysis'. The purpose of the Leaks-When report remains the same, but the content is pruned to hide most of the details about the operations, which are not directly connected to the selected data object. It especially makes sense for large graphs, which are diffucult to interprete for users. For convenience, user does not select the output data object for the Leaks-When analysis as in the normal workflow. We automatically figure out the message flow following the selected data object and take the output data object of the receiving task or event as an input for the Leaks-When analyzer.

We call such pruned Leaks-When report a Simple Leaks-When report. The Simple Leaks-When report might look much more concise for the large graphs, however, it can remain the same for the small graphs. A simplifications is carried out on the full Leaks-When report, so the 'dot' containing the dependency graph is an input for simplification algorithm. The algorithm is the following:

- We find the root nodes, matching by the selected data object name (e.g. 'Slot'). If nothing is found (e.g. in case of the intermediate data objects), then the graphs remains the same.

- We traverse all the subgraphs from the root nodes to the local 'Filter' and collect all of the operations that are 1-step far from the traverse path. So only the operations and tables that directly connected with a selected table remain. We can call such set a connected component.

- We prune all of the operations and attributes that has not been collected. We still keep all of the tables to demonstrate that some calculations were there, and user can observe them in the full Leaks-When report.

To showcase the Extended Simple disclosure report, we had to adjust an extension of the 'Aid Distribution' scenario to address the limitations and assumptions of the Simple disclosure analysis. The data object of the source party shared via the message flow is deemed to be copied with the same name as an output in the receiving party. Also the receiving entity must be event and not a task, so we have to insert an additional task that will extract only the necessary SQL attributes from the copied data object. On the Figure 16 we can see the adjusted process model. We inserted additional receiving events for 'Ports 1', 'aggr_count_1' and 'Capacities_1' tables. This showcase is publicly available[10] and does not require a system account. A public visitor is allowed to navigate through different editors such as PE-BPMN editor and SQL-privacy editor.

On the Figure 15 we see an Extended Simple Disclosure report, which includes the new types of disclosure. It is generated for the diagram displayed on the Figure 16.

GA.bpmn - Simple disclosure analysis report

#	aggr_cargo_2	aggr_count_1	aggr_count_2	Berth	Capacities 1	Capacities 2	Ordered ships	parameters	Port 1	Port 2	Selected ships	Ship 2	Slot
1. Aid requesting country	I	O	V	O	O	-	-	O	O	I	-	I	O
2. Aid providing country	O	D, I	O	D, I	V	O	O	V	V	O	O	O	D, I
Shared over	-	-	MF	-	MF	-	-	MF	MF	-	-	-	-

V = visible, H = hidden, MF = MessageFlow, S = SecureChannel, D = direct, I = indirect

Close

Figure 15: Sample of Extended Simple disclosure report

As we can see, the 'Capacities 1' data object is shared to the '2. Aid Providing Country' via the message flow. Hence, it is marked as O (Owner) for the '1. Aid Requesting Country' and V (visible) for the '2. Aid Providing Country'. The first party posesses the Slot table and uses it in the task 'Calculate the number of available slots for each port'. The direct output of this computation, namely 'Capacities 1' data object, is shared right away, so we add the mark D (direct) to the Extended Simple Disclosure report. But with respect to the data object 'Capacities 2', which is a derivative of 'Capacities 1', the 'Slot' table data can indirectly appear in 'Capacities 2', so we add the mark I (indirect) to the Extended Simple Disclosure report.

On the Figure 17 we can observe a Simple Leaks-When report for the selected 'Slot' disclosure to the '2. Aid providing country'. We see that the full calculations about ship reachibility are not present, because they are not directly connected to the slot assignment. However, since berths consist of slots, we have to match them in the calculations in a direct way, that is why the 'Berth' node remains with the attributes in the graph. However, we keep other tables in the graph, because they are still present in calculations, so we shrink all intermediate operations and attributes into the black box designation of '...'. If user wants to observe the entire hierarchy of operations, he can run the full Leaks-When analysis with the same selected output data object.

[10] https://pleak.io/app/#/view/JvuqfCYXMbjl02nsvWLS

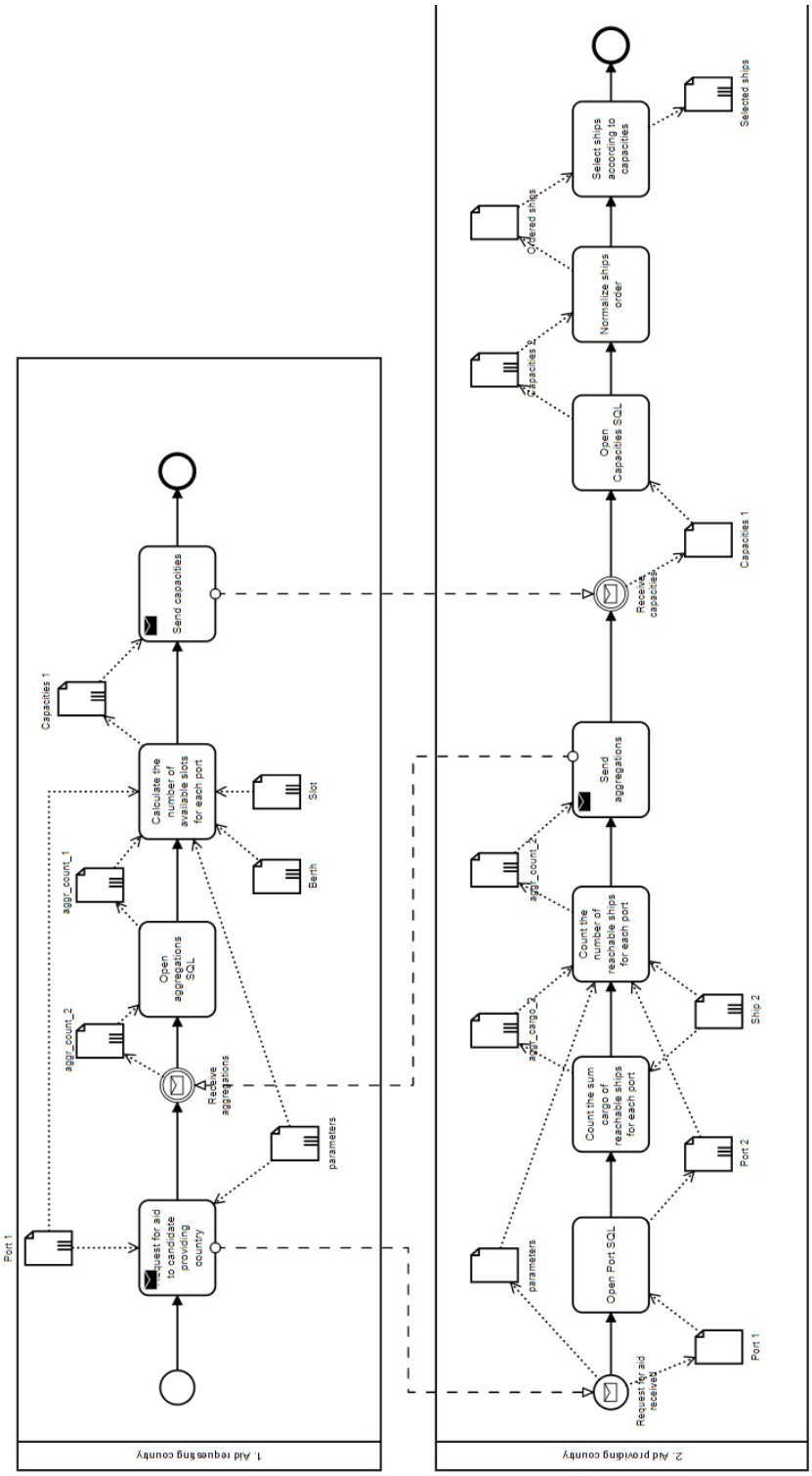


Figure 16: Adjusted extension of the ’Aid Distribution’ scenario

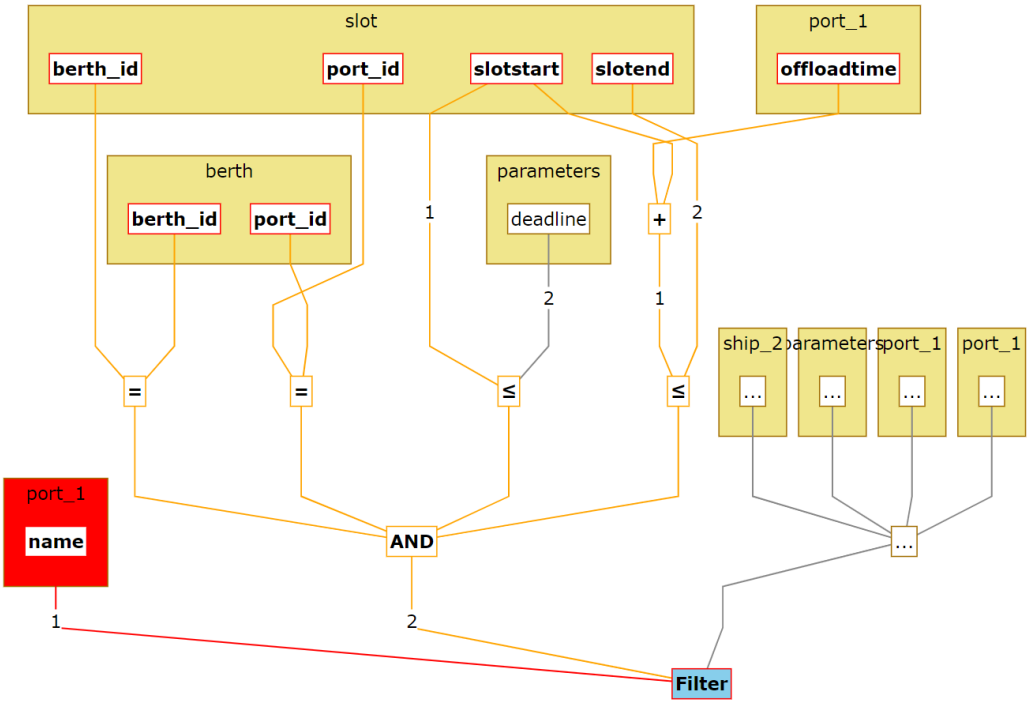


Figure 17: Simple Leaks-When report with respect to Slot table

4.2 Leaks-When Analysis

After assessing the high-level picture of disclosures through the Extended Simple disclosure report, user can handover the analysis to the qualitative level. As we mentioned in Section 2.8 about Leaks-When, the Leaks-When report is used to reveal the conditions (SQL operations) which can lead to the disclosure of a specific attribute. In Chapter 3 we also described how we refined and extended the Leaks-When report and analysis in order to introduce the collaborative compound and multi-party communication into a process model. We focused on the limitations of the support of SQL syntax and on the support of multiple BPMN pools. However, there are also other fundamental limitations related to the process model which are needed to resolve.

The preparatory stage that we carry out in the Leaks-When analysis is collecting all of the SQL scripts attached to the BPMN events, tasks and data objects and building a SQL collaborative workflow. In the listing 4.1 we see the example of the SQL script attached to the 'Compute reachable ports' task from the process model on the Figure 8.

Listing 4.1: Example of a SQL script associated with a task

```
create or replace function earliest_arrival(
  ship_lat DOUBLE PRECISION,
  ship_long DOUBLE PRECISION,
  port_lat DOUBLE PRECISION,
  port_long DOUBLE PRECISION,
  max_speed BIGINT)
  returns BIGINT as
$$
  select ceil((POINT(ship_lat, ship_long) <@> POINT(port_lat, port_long)) / max_speed)::BIGINT
$$
language SQL IMMUTABLE returns NULL on NULL INPUT;

create or replace function compute_reachable_ports(deadline BIGINT, shipname TEXT)
  returns TABLE (port_id BIGINT, arrival BIGINT) as
  $$
  select port_2.port_id as port_id, ship_2.cargo as cargo, ship_2.draft as draft,
      earliest_arrival(ship_2.longitude, ship_2.latitude,
                       port_2.longitude, port_2.latitude, ship_2.maxspeed) as arrival
  from port_2, ship_2
```

```
20    where earliest_arrival(ship_2.longitude, ship_2.latitude,
21        port_2.longitude, port_2.latitude, ship_2.maxspeed) <= deadline;
22    $$
23    language SQL;
24
25  SELECT rports.port_id as port_id,
26    rports.cargo as cargo,
27    rports.draft as draft
28    INTO ship_requirements_2
29    FROM parameters as p cross join lateral compute_reachable_ports(p.deadline, p.shipname) as rports;
30
31  SELECT rports.port_id as port_id,
32    rports.arrival as arrival
33    INTO reachable_ports_2
34    FROM parameters as p cross join lateral compute_reachable_ports(p.deadline, p.shipname) as rports;
```

The syntax used in the script is that of PostgreSQL. In this example, the script includes a user defined functions 'earliest_arrival' and 'compute_reachable_ports' that computes the time for a ship to reach a port and writes the selected ships' data accordingly. Each task can be associated with any number of user-defined functions and at least one select-into statement that stores the outcome of the computation on a output table (attacjed to the output data object). In the aforementioned listing, the first select-into statement defined in lines 11-17 takes the cargo and draft of the ships that are reachable to the port and stores such tuples in table reachable_ports_2 (line 14). Similarly, the select-into statement defined in lines 19-25 takes the same input and stores its result in table ship_requirements_2 (line 22). Both statements, in turn, call the function compute_reachable_ports, which is defined in lines 1-9. We do not unite these result not to disclose the cargo and draft of the ship with its location, because having the ship parameters and arrival time it is possible to narrow down the ship coordinates.

The process of extracting the SQL workflow looks straightforward for simple models, however, the ordering of the SQL scripts is crucial for building the right SQL workflow, so to address process models with loops and exclusive gateways, we applied a well-known formalism of Petri nets.

We perform the Leaks-When analysis of a SQL collaborative workflow in several stages as sketched on Figure 18. Assuming that a SQL collaborative workflow is provided, the first stage consists of translating the BPMN process model into a Petri net. For completeness, we include the transformation rules in Figure 19. In the second stage, an unfolding of the Petri net is computed which explicitly represents all the possible executions of the Petri net including conditional paths and loops. We call each of these executions a run. The unfolding is acyclic, and it explicitly captures one iteration of every loop in the original model. Then, the algorithm concatenates all the SQL statements attached to each of the nodes in the SQL collaborative workflow to generate an aggregated SQL script for each run. Finally, the technique generates a Leaks-When report for each output data object in the SQL collaborative workflow.

The top-level goal of the transformation of BPMN to a Petri net is to acquire a model that embraces all SQL statements of the single workflows in it in correct order and works with a SQL collaborative workflow. Petri net is unaware of the differect actors of the process, so the resulting net stands unified and flat. On the Figure 20 we observe a simple process model to showcase the transformation in details.

The Leaks-When analysis takes an input model specified as BPMN collaborative model that we need to translate into a Petri net. Note that the transformation rules cover only the mapping of BPMN process models (SQL workflows). However, the mapping can be straightforwardly adapted to our context as follows.

We implemented the mapping of the SQL workflow from each one of the pools in the collaborative process model according to the transformation rules. Then, we add a place for each message flow exchange between the pools. Such a place is connected with a source arc to the transition that represents a sending task in one of the pools. Similarly, the place is connected with an arc that stems from itself and targets the receiving task in the target pool. Figure 20 demonstrates side-by-side a sample SQL collaborative workflow and its corresponding Petri net. The places corresponding to the message flows are filled with blue color to help with their identication.

The developed transformation stores the information of the mapping such that, given a task in the process model it is possible to match it with the transition in the Petri net and vice versa. It is worth

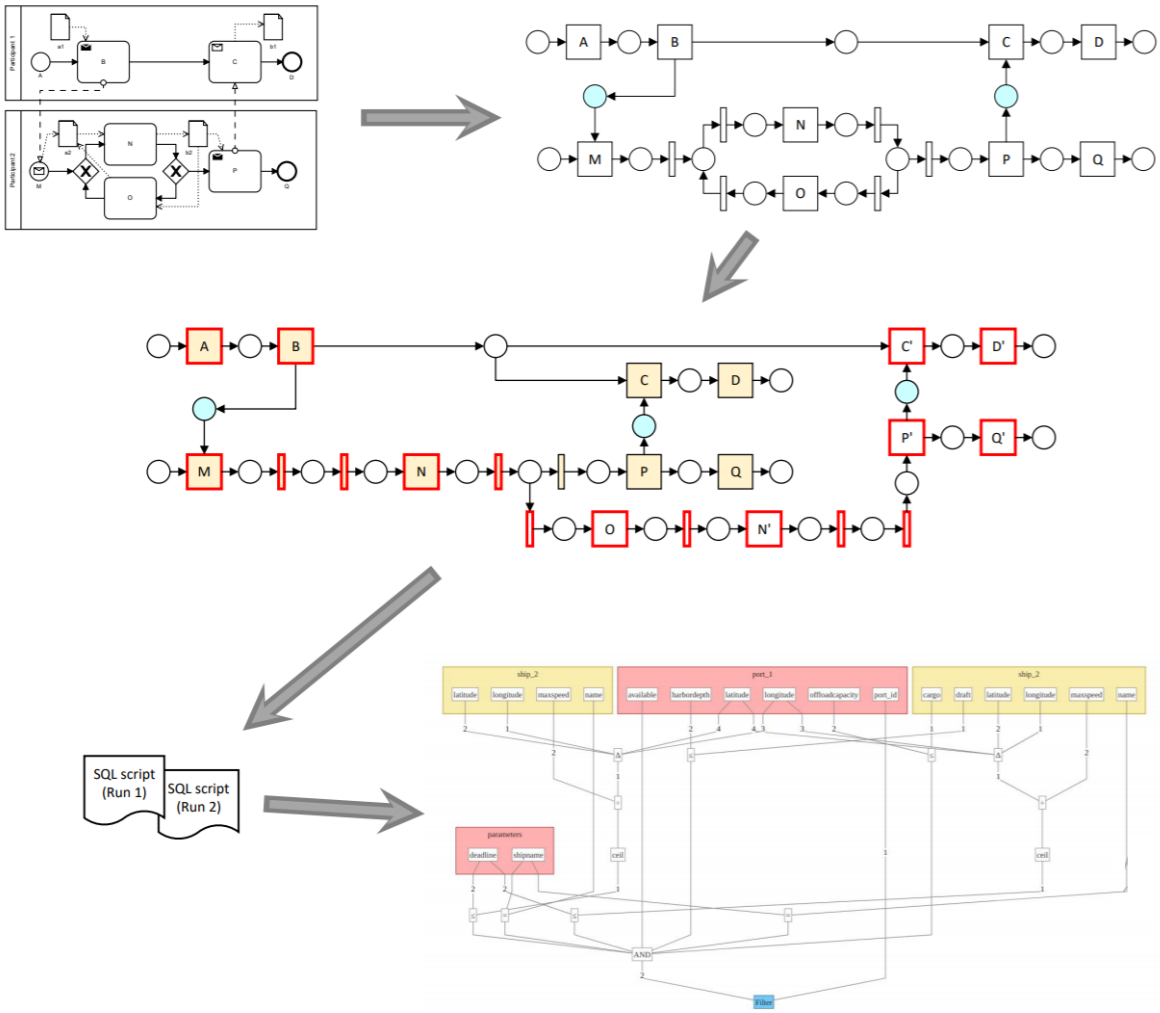


Figure 18: Stages of the Leaks-When analysis

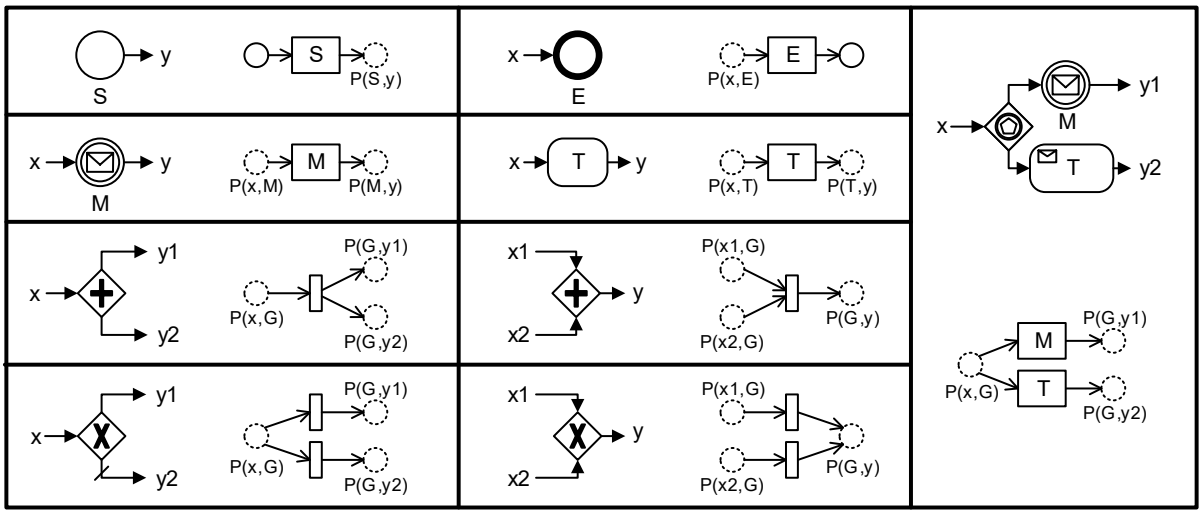


Figure 19: Transformation rules of BPMN to Petri net

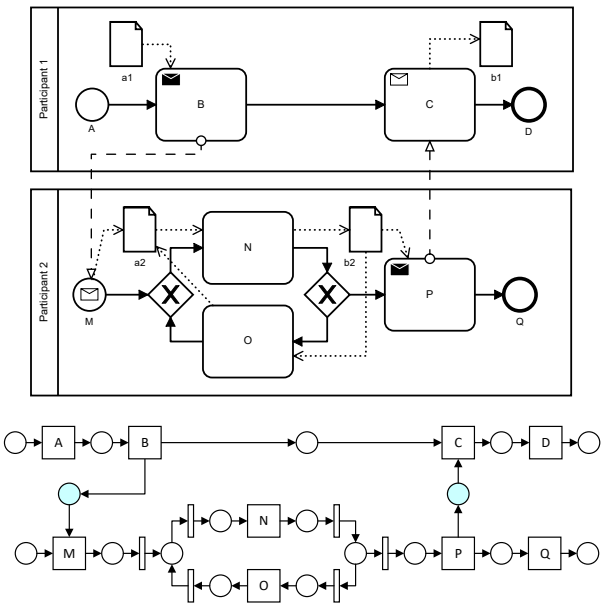


Figure 20: An example of the mapping of a SQL collaborative workflow to Petri nets

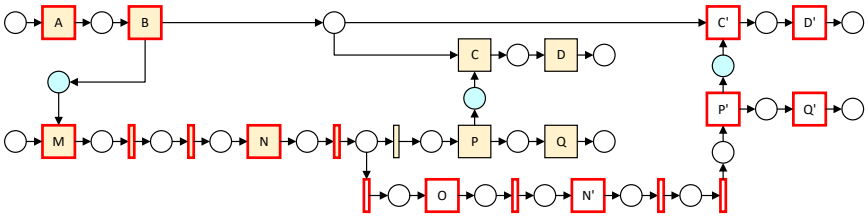


Figure 21: Runs of the SQL collaborative workflow, represented as the unfolding of its Petri net

noting that the transformation covers only the control flow perspective of the collaboration. However, by using the bidirectional mapping of BPMN tasks into Petri net transitions, it is possible to determine the input data objects (e.g. 'a1' in the case of task 'B') and also the output data objects (e.g. 'b1' in the case of task 'C') whenever required.

Once we built a Petri net, we compute a set of runs in for the model. To that end, we apply the technique of the unfolding of the Petri net. We adjusted and employed an existing tool to perform the unfolding. An unfolding of a Petri net is another net that represents the computation specified by an input Petri net. Depending on the number of iterations, the unfolding incrementally adds transitions and places to represent the execution of the original Petri net loops. Hence, an unfolding of a Petri net can be potentially infinite if the input net has loops. However, we limit the unfolding to the point where the behavior of the original Petri net is explicitly represented by a prefix of the unfolding.

The unfolding generated using such truncation criterion for our running example is shown on Figure 21. We can see that the SQL collaborative workflow has two runs. To help in identifying the tasks involved in each one of the runs, we have colored them: the first run includes all the tasks (transitions) filled in yellow, the second one includes all the tasks with the borders in red. As hinted by the labels in the unfolding, it is also possible to track back the transitions to the corresponding tasks in the original SQL collaborative workflow. Indeed, we can keep such mapping in a data structure as required.

For each of the runs, we can straightforwardly collect an aggregated SQL script by concatenating all the SQL statements of the tasks and data objects in the SQL collaborative workflow matched to the Petri net places and transitions. Since some of the tasks in a run can be concurrent, a topological order can be used for deciding an order in the SQL statements added to the SQL script. Each one of these aggregated scripts is then passed to the last step for completing the Leaks-When analysis.

The last stage of the Leaks-When analysis is running an existing tool that executes the translation of the SQL scripts into Summary Dependency Graphs (SDG), which are then simplified, removing redundant information. Then these graphs are properly colored and structured. The similar graph parts can be interpreted as follows. The operations in the box is deemed to describe the 'What leaks' part, while the remaining graph tells about 'When leaks' conditions. The colors of the edges are inferred from the current policy applied up to the selected data object, which is described in Chapter 3.

4.3 Guessing Advantage Analysis

After exploring the insight from the Leaks-When report, user can go further and use a quantitative analysis to assess how much the given input leaks in terms of attacker's guessing advantage described in Section 2.9.

In scope of PLEAK tool the Guessing Advantage analysis was enclosed in the Guessing Advantage editor, which has a number of limitations:

1. It accepts only the process models with one input BPMN Task.
2. It does not allow to specify who is adversary and who is target party.
3. The underlying analyzer can handle only a single workflow.
4. The sample scenarios are not connected with other analysis results because of their simplicity and necessity to adjust SQL queries to other analyzers.

4.3.1 Guessing Advantage Integration and Constraints

We described some of the principles of Guessing Advantage in PLEAK in Section 2.9. We employ the existing analyzer rather as a black box function that takes some input and priduces an output. We provided a concrete example and demonstrated how to interpret the outputs. The analyzer has been operating in scope of the Guessing Advantage editor and had a limited support of process models input. We implemented required changes to integrate the Guessing Advantage editor into the SQL-privacy editor. As a running example, we use the extension of the 'Aid Distribution' scenario described in Chapter 3.

Therefore, our main developments are focused on adjusting the front-end of the SQL-privacy editor to refine it with the inputs required got the Guessing Advantage analyzer. Accordingly, since the SQL-privacy editor supports much larger subset of the BPMN models than the Guessing Advantage editor, we had to decide how to correctly collect and build the inputs for the Guessing Advantage tool.

The first change we implemented concerns multiple parties in the BPMN model, which is now possible in scope of the SQL-privacy editor. We do not select the target data object with a SQL table inside textually, but we select the target party to attack and target data object through the sidebar interface. The desired attacker's advantage is selected within the same sidebar.

Second, we specify the prior knowledge of the attacker about the atributes of the SQL table along with its SQL definition and SQL data and not in some detached popup as it used to be in the Guessing Advantage editor.

Third, we moved the constraints definitions to the privacy policies, whoch is described in Chapter 3.

Additionally, beside these front-end changes we also implemented logic to extract the proper input from the process model to proceed with the execution of the Guessing Advantage analysis. Instead of taking one single task with the SQL statements, we find all of the tasks attached to the target data object. Moreover, since the tasks are used not only for the Guessing Advantage analysis, they may contain SQL queries that we have to ignore and leave them to the Leaks-When analysis. We are interested only in user-defined SQL functions that return a numerical outcome by using the SQL aggregation functions. After introducing the inputs, we should explain a concept of what is considered to be an output. The output is considered to be more abstract, in terms that we have the message flow in the BPMN model, and the outcome of the SQL function can be disclosed through this channel. So by using the Guessing

Advantage analysis we try to estimate how likely it is for attacker to guess the specific SQL entry by observing only a numeric output of the SQL aggregation function.

Since the developed changes are mostly input-based and interface-based, let us introduce a specific showcase to observe the aforementioned adjustments. Because the Guessing Advantage analyzer and the Leaks-When analyzer require different input formats, we had to reconstruct the editor. Fig. 22 depicts the sidebar for editing the SQL table in the selected data object. In order to open this sidebar, user has to click on the data object (for an input data object) or click 'Edit SQL table' in the context menu (for an intermediate data object).

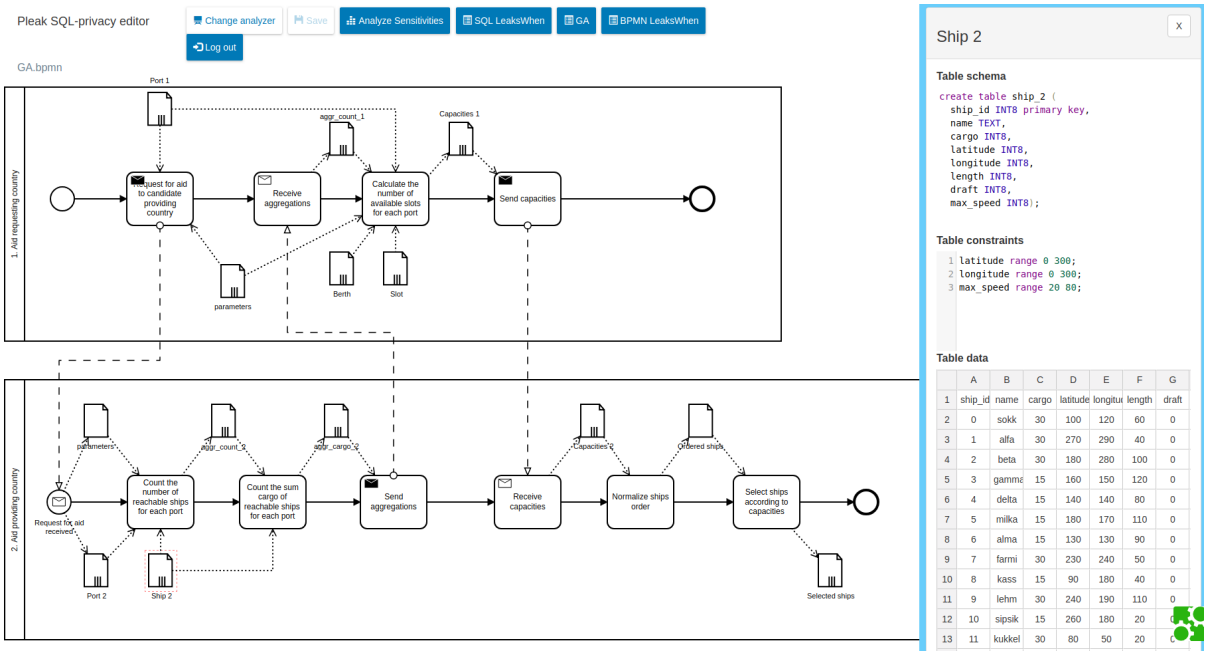


Figure 22: Current parameters of the SQL table

There are three inputs attached to every data object. The 'Table schema' is the same SQL script that we use for the Leaks-When analyzer, with the difference that the Leaks-When analyzer takes only the schemas from the input data objects, and not intermediate. This comes from the assumption that Leaks-When analysis fills in the intermediate data objects using the 'select-into' statements in the BPMN tasks. The only supported data types for the table schemas are INT8, FLOAT8, BOOL, and TEXT.

The 'Table constraints' input is a form of public knowledge about the target SQL table. On the screenshot (Fig. 22) we see an example of adversary's prior knowledge about the ship. The adversary knows that each ship's coordinate varies from 0 to 300 units, whereas the maximum speed varies from 20 to 80 units. These constraints will be used by the Guessing Advantage analyzer, and ignored by the Leaks-When analyzer.

The 'Table data' input is a user-defined dataset which is used for instantiation of the guessing process. The first row must contain the names of table columns, which should be the same columns that are declared in the 'Table schema'. This input is also ignored by the Leaks-When analyzer.

Another input that the Guessing Advantage analyzer extracts are the sensitivity parameters defined in the privacy policies. We described them in Chapter 3. For floating-point attributes, we state privacy policy as an approximation. This is because guessing the exact location of the ship is as bad as guessing in 'almost precisely', so we define the acceptable bounds. If the attacker comes up with a value within this bound, it counts as a violation. Guessing Advantage analysis will quantify the additive noise that will be enough to avoid policy violation.

4.3.2 Guessing Advantage Analysis

To demonstrate how the Guessing Advantage analysis can contemplate the Leaks-When analysis, we provide a typical use case for the given process model:

1. User selects the data object ‘aggr_count_1’ for the Leaks-When analysis in the context menu of the data object. The goal is to figure out what are the potential privacy policy violations by the time when the aid requesting country received the aggregations of reachable ships’ count and cargoes for each port.
2. User clicks on the Leaks-When report button.
3. User opens the Leaks-When report and observes that the ship coordinates may be disclosed (orange color), because they are used in the distance calculations.
4. User poses a question: “How much the received aggregations will help the aid requesting country in guessing the ship coordinates with the approximation defined in the privacy policy?” This question is in general difficult to answer, and the answer would not be helpful in decreasing the guessing probability, so let us pose an alternative question: "If we apply a differential privacy mechanism, how much noise we need to add to the outputs to keep the guessing advantage below 30%"?
5. User opens the Guessing Advantage panel, chooses aid requesting country as an adversary, aid providing country a a target party, and ‘Ship_2’ as a target data object. User runs the analysis.
6. User observes the output.

The Guessing Advantage analyzer takes all of the tasks that have the target data object as input. Hence, in case of ’Ship_2’ analyzer takes the tasks ’Count the number of reachable ships for each port’ and ’Count the sum of cargo of reachable ships for each port’. The integrated Guessing Advantage analysis has a number of limitations which required reworking SQL queries and table schemas.

Fig. 23 demonstrates the Guessing Advantage analysis results for the ’Ship’ target data object. We described how to interpret the outputs of the analyzer in Section 2.9.

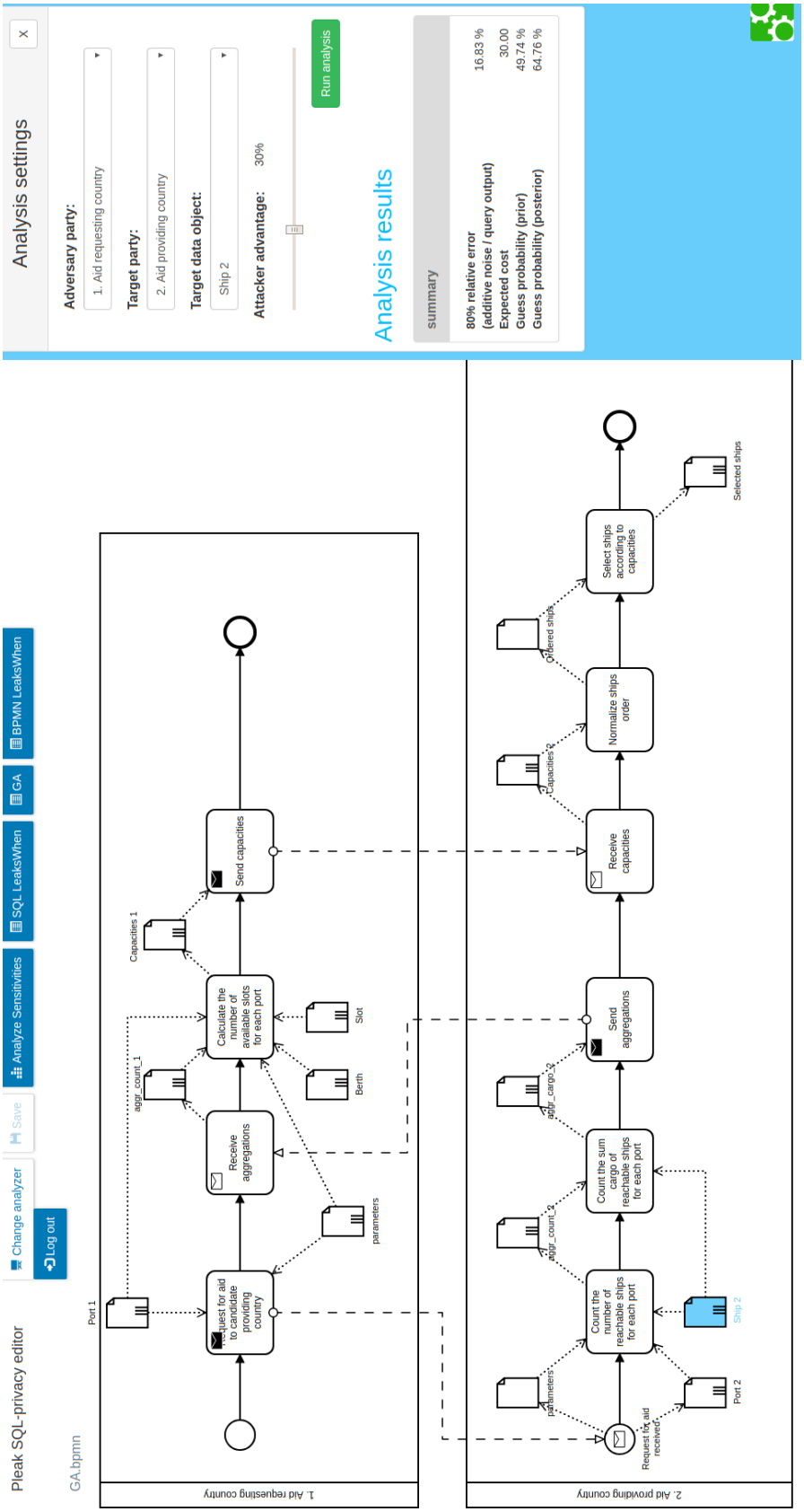


Figure 23: Guessing Advantage analysis results

5 TOOL IMPLEMENTATION

All of the features of this Master Thesis has been developed in scope of the PLEAK tools. The source code is public and can be accessed via *PLEAK tools repository*. The Extended Simple Disclosure analyzer, Leaks-When analyzer and Guessing Advantage analyzer are integrated and can be used via PLEAK SQL-privacy editor. Specifically, one can use PLEAK's BPMN modeler to build a business process model and then switch to the SQL-privacy editor. In the SQL-privacy editor user can incorporate the SQL Workflows, privacy policies and apply a multi-level process model analysis. In this chapter we provide details about the major features and the description of the system.

5.1 Architecture

The user communicates analyzers and all of the pages through the front-end of the *PLEAK website*. The illustrative structure of the system components is shown on Figure 24. The high-level components of the system are designated with plain rectangular-shaped nodes. Grey nodes stand for ones I did not develop or modify in scope of this Master Thesis. Green nodes are the ones I mostly worked on.

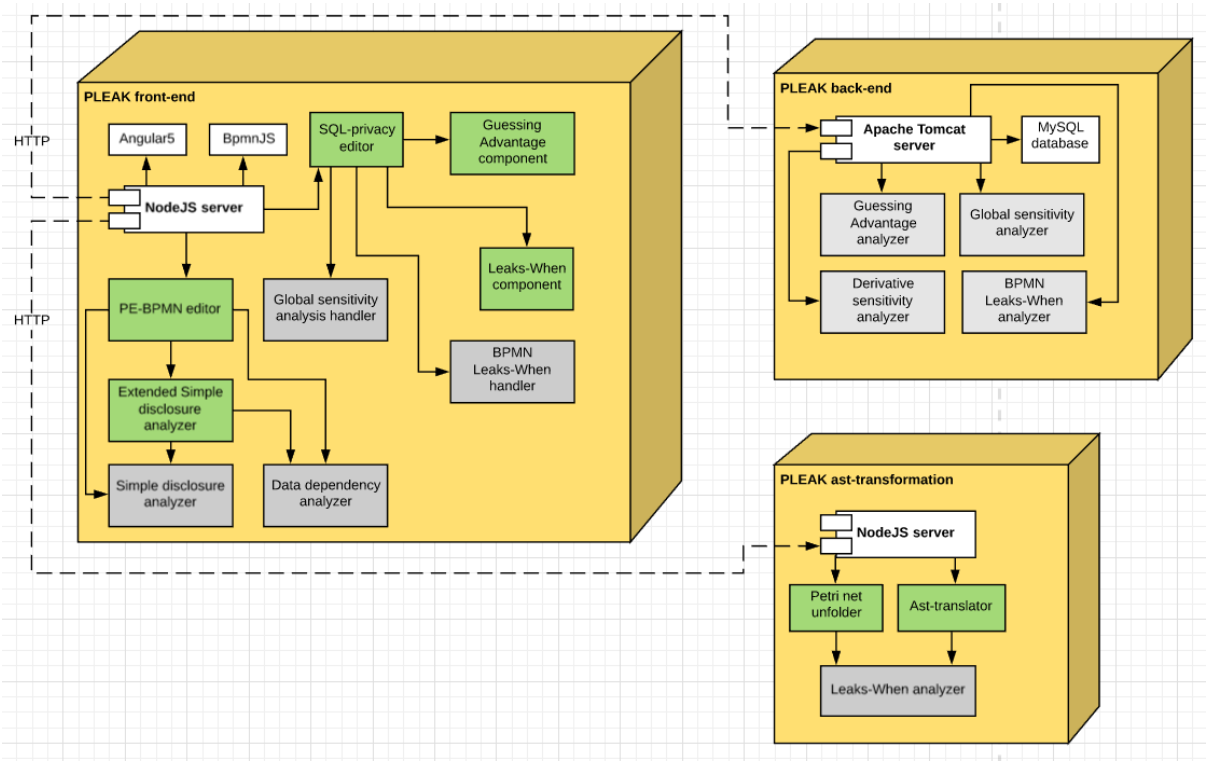


Figure 24: PLEAK components

The front-end is served by a NodeJS server, which is yielding the separate assets for each of the editors and the main page. Each of the pages is managed by an Angular5 application, using the TypeScript language. Hence, the programmatic handler for each of the analyzers is stored in a separate Angular component. From the side of PE-BPMN I developed the component for performing an Extended Simple disclosure analysis on top of existing components for Simple disclosure analysis and Data dependency analysis. In SQL-privacy editor I carried out major changes required to support the various inputs of the Guessing Advantage analyzer, Leaks-When analyzer, to organize the interaction with the 'back-end' server and the 'ast-transformation' server.

All of the user data, process models, sharing information and related data are stored in MySQL database on the back-end node. This node contains an API written with Java 8 and managed by Tomcat Server and Maven. The PLEAK's back-end is also responsible for running guessing-advantage analysis,

derivative sensitivity analysis and other analyzers. It is crucial for the functioning of the system, but I did not participate in its development, because it was out of the scope of this thesis.

The third server called 'ast-transformation' is implemented using NodeJS and dedicated to the Leaks-When analysis. It includes an analyzer for building and optimizing the summary dependency graphs, written in OCaml programming language, and a tool for unfolding the Petri nets, written in Java 8. It also contains an existing tool for translating the aggregated SQL collaborative workflow into the internal format of relational algebra in OCaml, needed for the analyzer. This OCaml tool is being supported and extended by the research group of the NAPLES project. One of my contributions was extending the 'Ast-translator' component, which serves to transform a SQL workflow to the OCaml notation acceptable by the analyzer for building the summary dependency graphs. I also added a sub-component for pruning the resulting Leaks-When report in terms of Simple Leaks-When analysis.

5.2 Design Choices

In this section we will describe some details of the PLEAK components and explain some of the programmatic decisions developed to fulfill the features described in chapter 3 and 4.

All implemented changes are related mainly to SQL-privacy editor, which can be opened by clicking the 'Open in SQL-editor' button, located in a dropdown menu of the model. The improvements of the Extended Simple disclosure analysis are carried out in PE-BPMN editor and reused in SQL-privacy editor.

On the Figure 25 we can see a SQL-privacy editor. On the top bar we removed all of the separate buttons for running the analyzers, because they started taking too much of the work space. So we added an 'Analysis' dropdown button with all of the analyzers listed. In scope of the multi-level privacy analysis they are placed subsequently: first user is supposed to run Extended Simple disclosure analysis, then the Leaks-When analysis and then move on to the Guessing Advantage analysis. This way, the user moves deeper from the high-level analysis to the lower-level disclosure analysis.

We described an Extended Simple disclosure analysis in Section 4.1. In PE-BPMN editor we added a new type of report - Extended Simple disclosure report, which includes Direct, Indirect and Owner types of disclosure. This report is generated using the Angular component and also needed in SQL-privacy editor to connect it with the Simple Leaks-When report. To avoid duplication and strong coherence in the code, we inserted an iframe in the SQL-privacy editor, which loads the PE-BPMN editor on the same model in the background, calls the component for performing the analysis, obtains the report and pulls it to the SQL-privacy editor. There user can select one of the cells containing D or I and run the Simple Leaks-When analysis with regard to the selected data object (stated in the table column) by clicking 'Run Simple Leaks-When' button. This way we integrated an Extended Simple disclosure analysis with the Leaks-When analysis.

To launch the full Leaks-When analysis, the user has to select one or several output data objects from the context menu and click on the 'SQL Leaks-When' in the 'Analysis' dropdown button. The SQL privacy editor then invokes the Leaks-When analyzer on the backend and displays a link to the resulting Leaks-When report(s) in a separate tab of the sidebar. The more detailed process of the Leaks-When analysis is the following:

- We use an Angular component to translate our BPMN to the Petri net.
- We send a HTTP request to the 'ast-transformation' server, to the unfolder API and get the unfolded version of the Petri net in the response.
- We use a Petri net to infer the right ordering of the activities and policies, extract them and from the final input for each run.
- For each run we send a HTTP request to the 'ast-transformation' server, that calls an OCaml tool for building a summary dependency graph.

After observing the Leaks-When report and finding the interesting disclosure occurrences, user can move on to the quantitative analysis by clicking 'Guessing Advantage' in the 'Analysis' dropdown

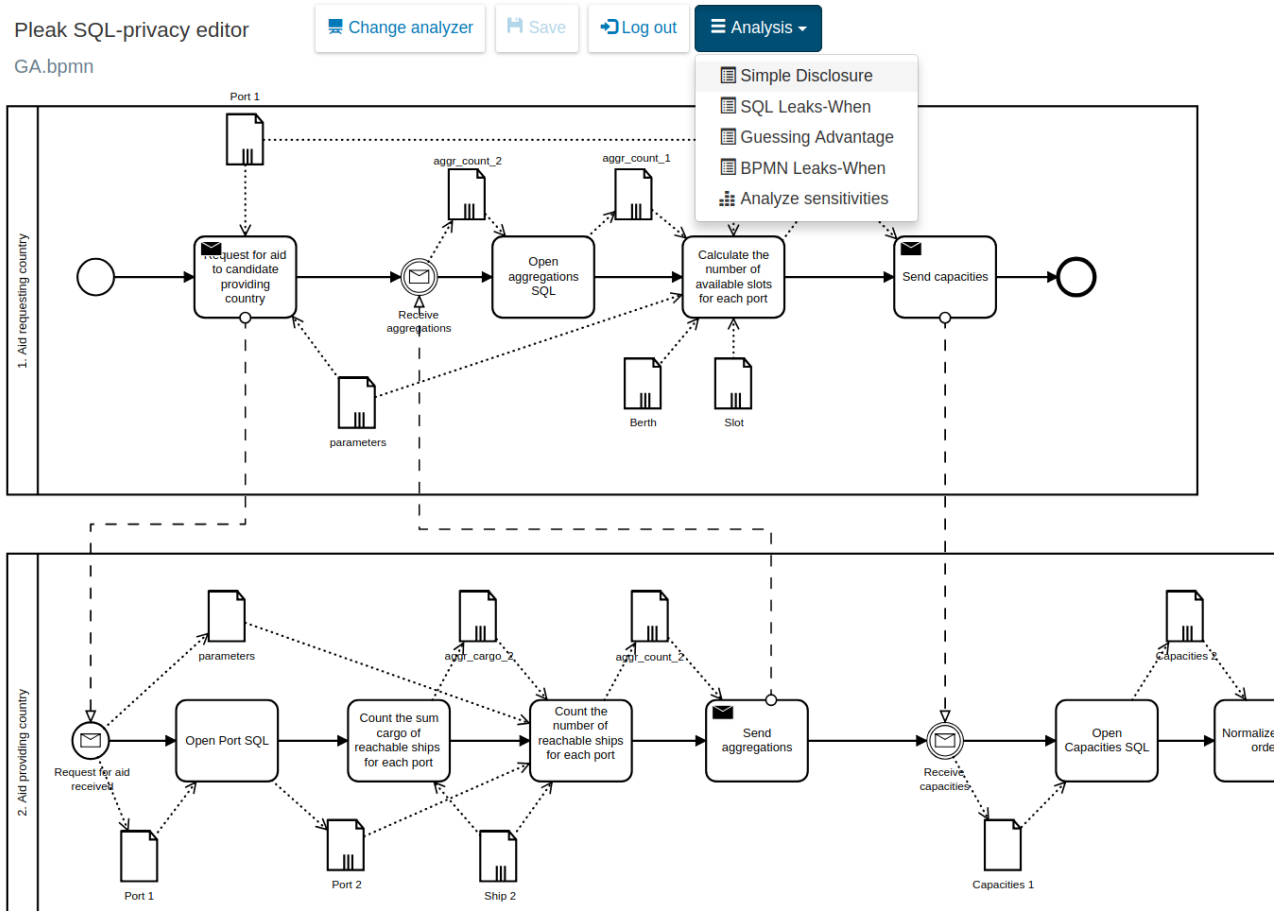


Figure 25: SQL-privacy editor

button. Here user selects an adversary party, the target party, the target data object and the bound of adversary's advantage. The target data object is highlighted with blue color, so that it is more visible what linked tasks will be included in the analysis. We conosider the integration of the Guessing Advantage analysis into the multi-level privacy analysis to happen through the common privacy policy of the collaborative process model. The detailed explanation is provided in Section 4.3.

5.3 Features and Known Limitations

We extended the support of input BPMN process models and SQL language for different analyzers and created models for multi-level analysis. However, the coverage of the acceptable SQL is not full for all of the analysis. The analyzer accepts a large subset of SQL, with a syntax specific to PostgreSQL. This implies that the tool supports a subset of syntax of SQL3, with partial support to some extensions to SQL3 for geographical data, more specifically, to support the computation of distance between two geographical points based in their coordinates. However, the current version imposes some constraints to the syntax used thereof, because of the integration of the analyzers that has different support of the SQL.

In scope of the Master Thesis we do not seek to make changes in the existing analyzers, that is why some of the limitations stem from the core PLEAK tools. In terms of the integration, we try to match as close as possible the subset of BPMN input models and the acceptable input of the analyzers. Hence, the most important constraints include:

- The data types used for the SQL table schemas must be in compliance with the types supported by the Guessing Advantage analyzer. Since the support is more restricted for Guessing Advantage,

in the integrated scenarios the allowed data types are 'INT8', 'FLOAT8', 'TEXT' and 'BOOL'.

- The Guessing Advantage analyzer works only with SQL functions, that only have the aggregations in the output. There is no support for any of 'join' statements, so such operations should be moved outside the functions and can be still used for the Leaks-When analysis.
- The Guessing Advantage analyzer requires the SQL table schema for each of the input data objects of each of the tasks, whereas the Leaks-When analyzer assumes that the SQL table for the intermediate data objects must be created in the previous task using 'select-into' statement. So we must use both approaches, but the Leaks-When will ignore the user-created SQL schemas in the intermediate data objects.
- The 'select' clause does support the "*" parameter, but it works in a way that it always takes the primary key as a parameter attribute. As an alternative, user can explicitly enumerate the list of attributes that are projected.
- All the tables in the from clause must be associated with aliases. All the attributes in the list of projected attributes must also have an associated alias. Moreover, any reference to attributes must be qualified with the alias of the table, except in the case of the order by clause where attributes must be referred by the alias given in the select clause.
- In order to run the Extended Simple Disclosure analysis on collaborative BPMN models, the input data object of the sendinig activity must be received by the event of another party and not the task. Also the output data object of the receiving event should possess the same name, and only then we can filter out necessary attributes of that data object.

The final version of SQL-privacy editor, containing the multi-level privacy analysis along with privacy policies, has been tested mainly on the 'Aid Distribution' scenario, its extension and several abstract process models containing loops, BPMN parallel gateways and different combinations of BPMN primitives. However, non-adapted scenarios can be also used with single analyzers, suitable for the selected process model.

6 CONCLUSION

6.1 Summary

In this thesis, we incorporated a three-level policy-aware privacy analysis for the business processes with SQL Collaboration Workflows.

We added the privacy policies into the existing analyzers in scope of the PLEAK tool. They exist in form of the extension to the SQL 'grant' statements and allow to endow selected parties with access rights to the SQL tables and specific attributes. These rights can be granted for the entire process lifetime ('global' policy) or added dynamically ('local' policy). We have extended the Leaks-When analysis algorithm to perform the process-to-policy compliance check. This extension allows to display the policy violations - direct and indirect disclosures - in the Leaks-When report.

In addition, we integrated the Extended Simple Disclosure and Guessing-Advantage analyzers into the SQL-privacy editor, joining the Leaks-When and enabling the multi-level analysis. So now it is possible to run the analyzers of different levels of granularity in the same editor and for the same process model.

In the Extended Simple Disclosure analysis user can observe very high-level information about potential disclosure, judging only from the process model. We refined the analyzer with new output types and integrated it with the Leaks-When analyzer to simplify the report for the complicated models.

In the Guessing Advantage analysis user can measure the attacker's probability of guessing the value of the potentially disclosed attribute according to the Leaks-When report. Guessing Advantage analysis requires selecting the adversary and target parties and target data object (SQL table). Sensitivity attributes are attached to the policy, so the privacy policy is universal for both Leaks-When and Guessing Advantage analyzers. We developed an extension for the 'Aid Distribution' scenario that uses the SQL aggregation functions and works both for the Leaks-When analysis and the Guessing Advantage analysis.

6.1.1 Limitations

The detailed limitations of the developed tools are listed in Section 5.3. In general, we should mention that the support of SQL language is limited in the analyzers in different ways, as well as the input BPMN process model usually has to be adjusted to be able to work with different analyzers. Also PE-BPMN stereotypes remain detached from the SQL-privacy analyzer and Guessing Advantage analyzer.

6.2 Future Work

In the next iteration of the NAPLES project (leading to a software deliverable), we plan to continue with the implementation of the Leaks-When analysis and test it on more scenarios.

Having the Privacy Policies implemented and integrated in the Leaks-When analysis as a process-to-policy compliance checking, we can proceed with the implementation of the automated privacy optimization of the business process models specified in the PE-BPMN notation. This feature will enable us to figure out what PETs could complement a given business process in order to fulfill a given privacy policy.

In terms of the further integration, we can consider the PETs attached to the process model in the Leaks-When analysis. Also we will integrate other privacy analyzers of PLEAK with privacy policies.

Also we can consider a support of larger subset of PETs for Extended Simple Disclosure analysis. For example, we can take into account DifferentialPrivacy stereotype and add 'N' (noise) prefix to different type of disclosures, such as 'NV' or 'ND'. We can also start integration of the Leaks-When analysis with the PE-BPMN. For instance, the leakage of the SQL attribute can be considered differently if the attribute is encrypted.

REFERENCES

[AFG16] Myrto Arapinis, Diego Figueira, and Marco Gaboardi. Sensitivity of counting queries. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 120:1–120:13, 2016.

[ALL15] Rafael Accorsi, Andreas Lehmann, and Niels Lohmann. Information leak detection in business process models: Theory, application, and tool support. *Inf. Syst.*, 47:244–257, 2015.

[DAKG17] Vasiliki Diamantopoulou, Nikolaos Argyropoulos, Christos Kalloniatis, and Stefanos Gritzalis. Supporting the design of privacy-aware business processes via privacy process patterns. In *11th International Conference on Research Challenges in Information Science, RCIS 2017, Brighton, United Kingdom, May 10-12, 2017*, pages 187–198, 2017.

[DGL16] Marlon Dumas, Luciano García-Bañuelos, and Peeter Laud. Differential privacy analysis of data processing workflows. In *Graphical Models for Security - Third International Workshop, GraMSec 2016, Lisbon, Portugal, June 27, 2016, Revised Selected Papers*, pages 62–79, 2016.

[DGL18] Marlon Dumas, Luciano García-Bañuelos, and Peeter Laud. Disclosure analysis of SQL workflows. In *5th International Workshop on Graphical Models for Security, held in conjunction with the Federated Logic Conference (FLoC) 2018, GraMSec@FLoC 2018, Oxford, UK, July 8, 2018, Revised Selected Papers*, pages 51–70, 2018.

[dMB08] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Dwo06] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 1–12, 2006.

[FGF08] Simone Frau, Roberto Gorrieri, and Carlo Ferigato. Petri net security checker: Structural non-interference at work. In *Formal Aspects in Security and Trust, 5th International Workshop, FAST 2008, Malaga, Spain, October 9-10, 2008, Revised Selected Papers*, pages 210–225, 2008.

[GGL15] Roberto Guanciale, Dilian Gurov, and Peeter Laud. Business process engineering and secure multiparty computation. *Cryptology and Information Security Series*, 13:129–149, 01 2015.

[LPP18] Peeter Laud, Alisa Pankova, and Martin Pettai. Achieving differential privacy using methods from calculus. *CoRR*, abs/1811.06343, 2018.

[LPR18] Peeter Laud, Martin Pettai, and Jaak Randmets. Sensitivity analysis of SQL queries. In *Proceedings of the 13th Workshop on Programming Languages and Analysis for Security, PLAS@CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 2–12, 2018.

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 75–84, 2007.

[OMG11] OMG. Business Process Model and Notation (BPMN) Version 2.0. URL: `https://www.omg.org/spec/BPMN/2.0/`, January 2011.

[PMB17] Pille Pullonen, Raimundas Matulevicius, and Dan Bogdanov. PE-BPMN: privacy-enhanced business process model and notation. In *Business Process Management - 15th International Conference, BPM 2017, Barcelona, Spain, September 10-15, 2017, Proceedings*, pages 40–56, 2017.

[PTMT19] Pille Pullonen, Jake Tom, Raimundas Matulevičius, and Aivo Toots. Privacy-enhanced bpmn: enabling data privacy analysis in business processes models. *Software & Systems Modeling*, Jan 2019.

[RLP16] Marcello La Rosa, Peter Loos, and Oscar Pastor, editors. *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, volume 9850 of *Lecture Notes in Computer Science*. Springer, 2016.

[SDSM17] Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez, and David Megías. Individual differential privacy: A utility-preserving formulation of differential privacy guarantees. *IEEE Trans. Information Forensics and Security*, 12(6):1418–1429, 2017.

[Too18] Aivo Toots. Tool Support for Privacy-Enhanced Business Process Model and Notation, 2018. Bachelor's thesis.

[TTY+19] Aivo Toots, Reedik Tuuling, Maksym Yerokhin, Marlon Dumas, Luciano García-Bañuelos, Peeter Laud, Raimundas Matulevicius, Alisa Pankova, Martin Pettai, Pille Pullonen, and Jake Tom. Business process privacy analysis in pleak. In *Fundamental Approaches to Software Engineering - 22nd International Conference, FASE 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, pages 306–312, 2019.

APPENDIX

1 SQL STATEMENTS FOR THE 'REACHABLE PORTS' STEP OF THE 'AID DISTRIBUTION' SCENARIO

```
CREATE TABLE port_1 (
  port_id bigserial PRIMARY KEY,
  name text NOT NULL,
  latitude DOUBLE PRECISION,
  longitude DOUBLE PRECISION,
  offloadcapacity bigint,
  offloadtime bigint,
  harbordepth bigint,
  available boolean
);

CREATE TABLE parameters (
  param_id UNIT PRIMARY KEY,
  deadline INT,
  shipname TEXT
);

SELECT p1.port_id as port_id,
  p1.latitude as latitude,
  p1.longitude as longitude
INTO port_2
FROM port_1 as p1;

CREATE TABLE ship_2 (
  ship_id bigserial PRIMARY KEY,
  name text NOT NULL,
  cargo bigint,
  latitude DOUBLE PRECISION,
  longitude DOUBLE PRECISION,
  length bigint,
  draft bigint,
  maxspeed bigint
);

create or replace function earliest_arrival(
  ship_lat DOUBLE PRECISION,
  ship_long DOUBLE PRECISION,
  port_lat DOUBLE PRECISION,
  port_long DOUBLE PRECISION,
  max_speed BIGINT)
  returns BIGINT as
$$
  select ceil((POINT(ship_lat, ship_long) <@> POINT(port_lat, port_long)) / max_speed)::BIGINT
$$
language SQL IMMUTABLE returns NULL on NULL INPUT;

create or replace function compute_reachable_ports(deadline BIGINT, shipname TEXT)
  returns TABLE (port_id BIGINT, arrival BIGINT) as
  $$
  select port_2.port_id as port_id, ship_2.cargo as cargo, ship_2.draft as draft,
      earliest_arrival(ship_2.longitude, ship_2.latitude,
                       port_2.longitude, port_2.latitude, ship_2.maxspeed) as arrival
  from port_2, ship_2
  where earliest_arrival(ship_2.longitude, ship_2.latitude,
      port_2.longitude, port_2.latitude, ship_2.maxspeed) <= deadline;
  $$
  language SQL;
```

```
SELECT rports.port_id as port_id,
  rports.cargo as cargo,
  rports.draft as draft
  INTO ship_requirements_2
  FROM parameters as p cross join lateral compute_reachable_ports(p.deadline, p.shipname) as rports;

SELECT rports.port_id as port_id,
  rports.arrival as arrival
  INTO reachable_ports_2
  FROM parameters as p cross join lateral compute_reachable_ports(p.deadline, p.shipname) as rports;

SELECT sr1.port_id as port_id,
  sr1.cargo as cargo,
  sr1.draft as draft
  INTO ship_requirements_1
  FROM ship_requirements_2 as sr1;

SELECT rports.port_id as port_id,
  rports.arrival as arrival
  INTO reachable_ports_1
  FROM reachable_ports_2 as rports;
```

2 SQL STATEMENTS FOR THE 'FEASIBLE PORTS' STEP OF THE 'AID DISTRIBUTION' SCENARIO

```
create or replace function compute_feasible_ports()
  returns TABLE (port_id BIGINT) as
$$
  select rp1.port_id as port_id,
    p1.harbordepth as harbordepth,
    p1.offloadcapacity as offloadcapacity
  from reachable_ports_1 as rp1, ship_requirements_1 as sr1, port_1 as p1
  where p1.port_id = sr1.port_id
    and p1.port_id = rp1.port_id
    and rp1.port_id = sr1.port_id
    and p1.available
    and p1.harbordepth >= sr1.draft
    and p1.offloadcapacity >= sr1.cargo;
$$
language SQL;

select fports.port_id as port_id,
  fports.harbordepth as harbordepth,
  fports.offloadcapacity as offloadcapacity
into feasible_ports_1
from compute_feasible_ports() as fports;

SELECT fp1.port_id as port_id,
fp1.harbordepth as harbordepth,
fp1.offloadcapacity as offloadcapacity
INTO feasible_ports_2
FROM feasible_ports_1 as fp1;
```

3 SQL STATEMENTS FOR THE 'SELECT SHIP' STEP OF THE 'AID DISTRIBUTION' SCENARIO

```
create or replace function select_ships()
returns TABLE (port_id BIGINT, ship_id bigserial, length BIGINT) as
$$
  select port_2.port_id as port_id, ship_2.length as length, ship_2.ship_id as ship_id,
      earliest_arrival(ship_2.longitude, ship_2.latitude,
                       port_2.longitude, port_2.latitude, ship_2.maxspeed) as arrival
  from port_2, ship_2, feasible_ports_2 as fp2
  where port_2.port_id = fp2.port_id
$$
language SQL;


SELECT ships.port_id as port_id,
ships.ship_id as ship_id,
ships.length as length
INTO ship_information_2
FROM parameters as p cross join lateral select_ships(p.deadline) as ships;

SELECT si2.ship_id as ship_id,
si2.length as length
INTO ship_information_1
FROM ship_information_2 as si2;
```

4 SQL STATEMENTS FOR THE 'SLOT ASSIGNMENT' STEP OF THE 'AID DISTRIBUTION' SCENARIO

```
CREATE TABLE berth (
  port_id bigint,
  berth_id bigint,
  berthlength bigint,
  PRIMARY KEY (port_id, berth_id)
);

CREATE TABLE slot (
  port_id bigint,
  berth_id bigint,
  slot_id bigint,
  ship_id bigint REFERENCES ship NOT NULL,
  slotstart bigint,
  slotend bigint,
  PRIMARY KEY (port_id, berth_id, slot_id),
  FOREIGN KEY (port_id, berth_id) REFERENCES berth (port_id, berth_id)
);

SELECT slot.port_id as port_id,
      slot.berth_id as berth_id,
      row_number() OVER (PARTITION BY port_id, berth_id) AS row_id,
      slot.slotstart as slotstart,
      slot.slotend as slotend
INTO slot1
FROM slot
ORDER BY port_id, berth_id, slotstart;

SELECT COALESCE(slot1.port_id, slot2.port_id) AS port_id,
  COALESCE(slot1.berth_id, slot2.berth_id) AS berth_id,
  row_number() OVER (PARTITION BY port_id, berth_id) AS slot_id,
  COALESCE(slot1.slotend, slot2.slotstart) AS gap,
  COALESCE(slot1.slotend, 0) AS slotstart,
  COALESCE(slot2.slotstart, 30) AS slotend
into available_slots
FROM slot1
FULL JOIN slot1 AS slot2 ON
  slot1.port_id = slot2.port_id AND
  slot1.berth_id = slot2.berth_id AND
  slot1.row_id + 1 = slot2.row_id
WHERE COALESCE(slot1.slotend, 0) < COALESCE(slot2.slotstart, 30)
ORDER BY port_id, berth_id, gap;


SELECT port_1.port_id as port_id,
  availslot.berth_id as berth_id,
  availslot.slot_id as slot_id,
  GREATEST(rport.arrival, availslot.slotstart) AS offloadstart
INTO slot_assignment_1
FROM reachable_ports_1 AS rport, feasible_ports_1 AS fport, port_1,
available_slots AS availslot, berth, ship_information_1
WHERE port_1.port_id = fport.port_id
AND port_1.port_id = rport.port_id
AND port_1.port_Id = berth.port_id
AND availslot.port_id = berth.port_id
AND availslot.berth_id = berth.berth_id
AND berth.berthlength >= ship_information_1.length
AND rport.arrival <= deadline AND availslot.slotstart <= deadline
AND rport.arrival + port_1.offloadtime <= availslot.slotend
AND availslot.slotstart + port_1.offloadtime <= availslot.slotend
ORDER BY offloadstart, port_id, berth_id;
```

5 SQL STATEMENTS FOR THE 'CALCULATE SHIP AGGREGATIONS' STEP OF THE EXTENSION OF THE 'AID DISTRIBUTION' SCENARIO

```
create table port_1 (
  port_id INT8 primary key,
  name TEXT,
  latitude INT8,
  longitude INT8,
  offloadcapacity INT8,
  offloadtime INT8,
  harbordepth INT8,
  available Bool
);

select p1.port_id as port_id,
  p1.name as name,
  p1.latitude as latitude,
  p1.longitude as longitude,
  p1.offloadcapacity as offloadcapacity,
  p1.offloadtime as offloadtime,
  p1.harbordepth as harbordepth,
  p1.available as available
into port_2
from port_1 as p1;

CREATE TABLE parameters
(
  param_id INT8 PRIMARY KEY,
  deadline INT8,
  portname TEXT
);

create or replace function aggr_count(portname TEXT)
  returns TABLE(cnt INT8) as
$$
  select count(ship_2.ship_id) as cnt
  from ship_2, port_2, parameters
  where port_2.name = parameters.portname
    AND (point(ship_2.latitude, ship_2.longitude) <@>
    point(port_2.latitude, port_2.longitude)) / ship_2.max_speed <= parameters.deadline
$$ language SQL IMMUTABLE returns NULL on NULL INPUT;

select p.name as name, res.cnt as cnt
into aggr_count_2
from port_2 as p cross join aggr_count(p.name) as res;
```

6 SQL STATEMENTS FOR THE 'CALCULATE SLOT AGGREGATIONS' STEP OF THE EXTENSION OF THE 'AID DISTRIBUTION' SCENARIO

```
create table berth (
  berth_id INT8 primary key,
  port_id INT8,
  berthlength INT8
);

CREATE TABLE slot (
  slot_id INT8 primary key,
  port_id INT8,
  berth_id INT8,
  slotstart INT8,
  slotend INT8
);

select ac2.name as name, ac2.cnt as cnt
into aggr_count_1
from aggr_count_2 as ac2;

create or replace function slots_count(portname TEXT)
  returns TABLE(slots_number INT8) as
$$
  select count(slot.slot_id) as slots_number from port_1, aggr_count_1, slot, berth, parameters
  where aggr_count_1.name = port_1.name
    AND port_1.name = parameters.portname
    AND port_1.port_id = berth.port_id
    AND slot.port_id = berth.port_id
    AND slot.berth_id = berth.berth_id
    AND slot.slotstart <= parameters.deadline AND slot.slotstart + port_1.offloadtime <= slot.slotend
  $$ language SQL IMMUTABLE returns NULL on NULL INPUT;

select p.name as portname, res.slots_number as slots_number
into capacities_1
from port_1 as p cross join slots_count(p.name) as res;
```

7 SQL STATEMENTS FOR THE 'MATCH SHIPS' STEP OF THE EXTENSION OF THE 'AID DISTRIBUTION' SCENARIO

```
select cap1.portname as portname, cap1.slots_number as slots_number
into capacities_2
from capacities_1 as cap1;

create or replace function earliest_arrival(
ship_lat INT8,
ship_long INT8,
port_lat INT8,
port_long INT8,
max_speed INT8)
returns INT8 as
$$
  select ceil((POINT(ship_lat, ship_long) <@> POINT(port_lat, port_long)) / max_speed)::INT8
$$
language SQL IMMUTABLE returns NULL on NULL INPUT;

SELECT ship_2.ship_id as ship_id,
port_2.name as portname, earliest_arrival(ship_2.longitude, ship_2.latitude,
      port_2.longitude, port_2.latitude, ship_2.max_speed) as arrival, ship_2.cargo as cargo,
      ship_2.draft as draft,
      row_number() OVER (PARTITION BY ship_2.ship_id) AS row_id
INTO ordered_ships
FROM ship_2, port_2, parameters
WHERE earliest_arrival(ship_2.longitude, ship_2.latitude,
      port_2.longitude, port_2.latitude, ship_2.max_speed) <= parameters.deadline
ORDER BY ship_id;

select os.portname as portname, os.arrival as arrival
into selected_ships
from ship_2, ordered_ships as os, capacities_2 as cap2
where os.portname = cap2.portname AND os.row_id <= cap2.slots_number;
```

8 LICENCE

Non-exclusive licence to reproduce thesis and make thesis public

I, _____ Maksym Yerokhin _____,
(*author's name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

_____ Multi-Level Policy-Aware Privacy Analysis _____,
(*title of thesis*)

supervised by _____ Pille Pullonen and Luciano García-Bañuelos_____
(*supervisor's name*)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Maksym Yerokhin
16/05/2019