UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Aivo Toots

Tool Support for Privacy-Enhanced Business Process Model and Notation

Bachelor's Thesis (9 ECTS)

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

Tartu 2018

Tool Support for Privacy-Enhanced Business Process Model and Notation

Abstract:

This paper presents an implementation tool for a Privacy-Enhanced Business Process Model And Notation language (PE-BPMN) that extends Business Process Modal And Notation (BPMN) by adding constructs to specify privacy enhancing technologies to be used in process models. PE-BPMN language allows to visualize the movement and disclosure of private information between participants of business processes. The language is used as a basis for detecting privacy leakages in business processes.

The result of this work, the PE-BPMN editor provides a modelling tool for PE-BPMN. In addition, the tool supports the user by providing analyzers to check the syntactical correctness of these extended models. Syntactical correctness is a prerequisite of further analysis on PE-BPMN models. Currently, there are two analysis implemented. Combined results of these analysis give an overview of whether some information used in the business process is at risk of being leaked. Also, these results give an insight how to improve already existing processes or how to plan more secure new processes.

Keywords:
BPM, BPMN, PETs, PE-BPMN, PLEAK

CERCS: P175 Informatics, systems theory

Tugitööriist äriprotsessimudeli ja -notatsiooni privaatsuslaiendusele

Lühikokkuvõte:

Käesolev töö käsitleb tugitööriista äriprotsessimudeli ja -notatsiooni privaatsuslaiendusele, mis täiendab äriprotsesside modelleerimiskeelt võimalustega lisada äriprotsessi mudelitele privaatsustehnoloogiate kirjeldusi. Äriprotsessimudeli ja -notatsiooni privaatsuslaiendus võimaldab visualiseerida privaatse informatsiooni liikumist ja avalikustamist äriprotsessides erinevate osapoolte vahel.

Töö tulemusena valminud tööriist nimega *PE-BPMN editor* võimaldab luua privaatsuslaiendusega äriprotsessimudeli ja -notatsiooni mudeleid. Sealjuures pakub tööriist võimalust kontrollida nende mudelite süntaktilist korrektsust, mis on aluseks nende mudelite edasisteks analüüsideks. Praeguseks on võimalik kasutada kahte analüüsimeetodit, mille kombineeritud tulemus annab ülevaate äriprotsessis kasutatavast privaatsest informatsioonist, millel on oht lekkida. Saadud tulemus võimaldab täiustada olemasolevaid ja planeerida uusi turvalisemaid äriprotsesse.

Võtmesõnad:
BPM, BPMN, PETs, PE-BPMN, PLEAK

CERCS: P175 Informaatika, süsteemiteooria

Contents

1 Introduction

Every day, as the number of people grows, the number of parties wishing to obtain private information about people, companies, countries etc. also grows. For a long time, to obtain someone's private information, it was required to have a physical contact with the information – by means of stealing papers containing classified information, secretly taking photos of somebody or something, using devices to listen to secret conversations etc. Currently, globalization has allowed people to decrease the distance between people and has allowed to create channels to connect to others all around the world. One of the channels with increasingly growing importance and usage is the Internet. As the number of parties using it and the amount of (private) information moved through it is growing, the risk of privacy leakages is also growing. This means that the need to protect privacy is becoming more important every day. As all privacy related problems cannot be addressed in one thesis, a more narrow approach has been taken. This thesis seeks to provide means to prevent and detect privacy leakages in terms of business processes that have been modelled in Business Process Model And Notation (BPMN) language.

In practice, preventing privacy leakage of sensitive data along information flows is done by using privacy enhancing technologies (PETs). PETs also contribute to the analysis of movements of private information. Currently, as privacy was not a focus point in the development process of BPMN, there are no widely used means to visualize and analyze movements of private information as it is disclosed to participants of these processes. In order to provide these means to BPMN as well, Privacy-Enhanced Business Process Model And Notation (PE-BPMN) language has been proposed [PMB17]. Providing a tool support for Privacy-Enhanced Business Process Model and Notation by developing a tool entitled PE-BPMN editor is the main objective of this thesis.

The PE-BPMN editor is developed by the author of this thesis as an extension tool for Privacy LEAKage Analysis Tools (PLEAK)[1]. PLEAK is the NAPLES (Novel tools for Analyzing Privacy LEakageS)[2] project tool for modelling and analyzing business processes with regard to privacy. PLEAK is currently under development by Cybernetica AS and University of Tartu under the DARPA[3] Brandeis privacy technology development program. The author of this thesis works as a programmer at Cybernetica AS on the NAPLES project. Additionally, as a part of the work on this thesis, wiki[4] to document

[1] `https://pleak.io/`

[2] `https://cyber.ee/en/research/research-projects/naples/`

[3] 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.

[4] `https://pleak.io/wiki/`

PLEAK and its components, including PE-BPMN editor, is set up. Full development of the PE-BPMN editor and setup of the wiki has been done by the author of this thesis.

1.1 Problem statement and objectives

Currently, business process modelling provides standardized ways to express stakeholder collaboration and process support by technical solutions. But in order to visualize and analyze movements of private information, as it is disclosed to participants of these processes, extra means are required.

In this context, this thesis takes PE-BPMN language as a starting point. PE-BPMN language is an extension to BPMN notation with constructs to specify privacy enhancing technologies to be used in process models. Consequently, the main objective of this work is to implement prototypes that include a modelling tool for PE-BPMN and also analyzers to check the syntactical correctness of PE-BPMN models.

In more detail, the main objective of this thesis, creating the PE-BPMN editor, covers two sub-objectives. The first objective of this thesis is to provide a tool that would allow to extend BPMN models with PETs by using the PE-BPMN language. Using PE-BPMN allows to express the usage of privacy enhancing technologies and through that, to visualize the movement and disclosure of private information between participants of business processes.

The second objective is to provide support for the user of the tool while creating PE-BPMN models by equipping the tool with a validation process. The validation process helps to guarantee that related technologies are used correctly – syntactically and semantically – in order to give them a sensible meaning. The process covers each technology with specific correctness checks in two ways. The information concerning the requirements and restrictions of each technology is presented to the user firstly when the stereotype is added to the model, secondly, when the user has created a PE-BPMN model with stereotypes and runs the validation process on it. In terms of PE-BPMN and PE-BPMN editor, stereotype is a representation of privacy enhancing technology on BPMN model elements including the necessary technology-specific information required to use the technology. Additionally, a wiki to document PLEAK and all of its components, including PE-BPMN, has been set up in order to extend the support. Chapter of PE-BPMN editor covers a quick guide on how to set up and use the editor, also, it provides information about all the implemented stereotypes.

The result of combining these two objectives, the PE-BPMN editor, a user-facing frontend application that implements support for BPMN extension – PE-BPMN, is created by the author of this thesis. The editor allows to extend BPMN models by adding various PE-BPMN stereotypes to their elements and run analysis based on these extended models. Currently, these two objectives have been fulfilled and two different analysis can be run on PE-BPMN models.

1.2 Structure of the document

This section describes the structure of the rest of the thesis. Section 2 covers the background of the thesis. As the PE-BPMN editor is based on PE-BPMN and BPMN and is related to PETs and PLEAK, these are covered in the Background section. Next, Section 3 is about the PE-BPMN editor and PET stereotypes from the perspective of providing an implementation to PE-BPMN – providing a tool to create PE-BPMN models. It covers both principles and technological solutions used to create the implementation of PE-BPMN. Furthermore, Section 4 is about the validation process of PE-BPMN models. This section covers the principles and methodologies used to check the syntactical correctness of PE-BPMN models, which is the basis for a further privacy-leakages related analysis on these models. Finally, Section 5 covers the conclusion of this thesis, which also includes known issues of PE-BPMN editor and plans for future work.

2 Background

The goal of this section is to provide a brief introduction into the concepts, standards and technologies, using which the main goal of this thesis, creating the PE-BPMN editor, is achieved. Following section covers BPMN and PE-BPMN languages and PET technologies – these form the basis of the PE-BPMN editor. Additionally, as the PE-BPMN editor is one of the multiple extension tools for PLEAK, the concept and components of the PLEAK are briefly introduced.

2.1 Business Process Model and Notation

Business Process Management (BPM) is a discipline that seeks to deliver improvements in organizational performance, regulatory compliance and service quality [RLP16]. One popular approach to address this goal is to model organization's processes using Business Process Model And Notation (BPMN), a standard developed by the Object Management Group (OMG). The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes [OMG10, OMG11]. Modelling helps to understand processes and to share the understanding of the process with the people who are involved with the process on a daily basis. It also helps to identify and prevent issues. The prerequisite to conduct process analysis, redesign or automation is a profound understanding of the process [RLP16], BPMN modelling is a way to achieve it.

BPMN notation is rich in different elements and symbols, but in terms of this thesis, only the main elements that are mentioned in this work are described. The main elements of BPMN notation are visualized in Figure 1 and explained as follows.

Flow Objects are used to describe parts of a process where decisions are made. Flow objects can be divided into three groups – events, activities and gateways.

Events indicate occurrences taking place in a process. In this thesis, three types of event elements are considered – start event, message catch event and end event.

Start event (see Figure 2a) represents the beginning of the process.

Message catch event (see Figure 2b) expresses a step or a breakpoint where the process needs to wait for a message (a trigger) to continue.

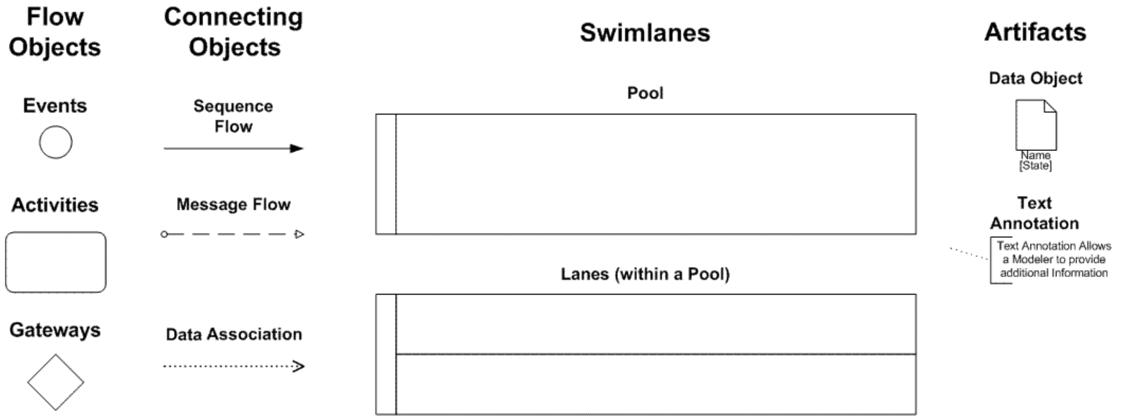
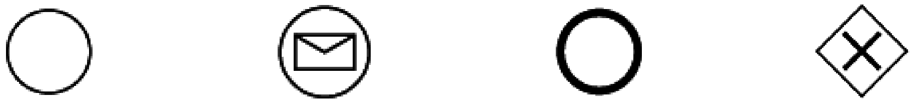


Figure 1. Core set of BPMN elements.
[Modified version of the image from
`http://www.omg.org/bpmn/Samples/Elements/Core_BPMN_Elements.htm`]



(a) Start Event (b) Message Catch Event (c) End Event (d) Exclusive Gateway

Figure 2. Some concrete BPMN elements.

End event (see Figure 2c) represents the ending of the process.

Activities describe the work being done in a process.

Tasks represent granular activities that cannot be divided into smaller units.

Gateways combine and separate flows in a process. There are many more gateway elements, but in this thesis only exclusive gateway is considered.

Exclusive gateways (see Figure 2d) evaluate the state of the process (based on a condition) and divide the process flow into one or more paths – these paths are mutually exclusive, which means that only one path can be chosen.

Swimlanes represent different actors (participants) and their roles in a process.

Pools represent participants of the process.

Lanes within pools help to express that certain roles are responsible for performing specific parts of a process.

Connecting Objects combine flow elements into a sequence.

Sequence flows connect events, tasks, gateways, etc. to each other. Direction of the sequence flows determines the order of the activities performed in the process.

Message flows express message sharing from one participant to another.

Data associations express the flow of information and their direction between activities.

Artifacts represent information relevant to an overall model.

Data objects represent the information used in the process by activities.

2.2 Privacy Enhancing Technologies

The Enterprise Privacy Group (EPG) [EPG08] states that:

> There is no commonly accepted definition of Privacy Enhancing Technologies (PETs), although all of the definitions seem to embody certain common characteristics. Specifically, a PET is something that:
>
> - reduces or eliminates the risk of contravening privacy principles and legislation;
> - minimises the amount of data held about individuals;
> - empowers individuals to retain control of information about themselves at all times.

The definition given by the European Commission [PET07] also includes the concept of using PETs at the design stage of new systems:

> The use of PETs can help to design information and communication systems and services in a way that minimises the collection and use of personal data and facilitates compliance with data protection rules. The use of PETs should result in making breaches of certain data protection rules more difficult and / or helping to detect them.

Table 1. Classification of Privacy Enhancing Technologies [PMB17, PTMT18].

Goal	Target	Examples of technology
Communication Protection	Secure	Client-Server encryption, TLS, IPSec, End-to-End encryption, PGP, OTR
	Anonymous	Proxies and VPN, onion routing, mix-networks, broadcast
Data Protection	Integrity	Message authentication codes, signatures
	Confidentiality	Encryption, secret sharing
Entity Authentication	Identity based	Username and password
	Attribute based	Credential used only once
Privacy Aware Computation	Confidential inputs	Homomorphic encryption, secure multiparty computation
	Privacy adding	Differential privacy, k-anonymity, anonymization
Human-Data Interaction	Transparency of data usage	Information flow detection, logging
	Intervenability	Information granularity adjustment, access control

These two definitions of PETs combined cover the principles of the PE-BPMN editor. The idea is that system builders should be able to use the tool to make decisions on the privacy solutions at the early stages of the development of a system and that the tool should enable auditors to analyse already existing systems from the perspective of private information movement and leakages in processes.

PETs considered in terms of this work are based on the PET classification from a report [DDFMH$^+$15], which is also the starting point for selecting privacy enhancing tools to develop Privacy-Enhanced Business Process Model And Notation (PE-BPMN) [PMB17, PTMT18] language. For privacy analysis, viewpoint based on the examples of PETs described in Table 1 is considered more useful than the one described in the survey. Privacy enhancing technologies important for this thesis are described in Section 2.3.

2.3 Privacy-Enhanced Business Process Model And Notation

Currently, business process modelling provides standardized ways to express stakeholder collaboration and process support by technical solutions, but in order to visualize and analyze movements of private information as it is disclosed to participants of these processes, extra means are required [PTMT18]. Privacy-Enhanced Business Process Model And Notation (PE-BPMN) is a proposed extension language to BPMN notation to visualize and analyze the movements of private information. PE-BPMN provides constructs to specify privacy enhancing technologies to be used on process models.

2.3.1 Categories of PETs in PE-BPMN

As described in [PMB17, PTMT18], there are five categories of PET technologies (see Table 1 for technology examples) chosen for PE-BPMN according to their application goals:

- *Communication protection* covers protection of the contents of the communication and also of the identities of the participants in the communication. Secure communication means that confidentiality and integrity are ensured in data transmissions. Anonymous communication protects the identity of participants;
- *Data protection* ensures integrity and confidentiality of the data. Confidentiality is guaranteed by protection mechanisms that allow only authorized parties to see the protected data. Integrity is achieved by using technologies that allow to verify if the data has been stored or transmitted in a way the creator of the data has expected;
- *Entity Authentication* proves that the user has the identity and / or the attributes that it claims to have. Identity authentication is a form of authentication that is used to prevent the user from pretending to be someone else. However, attribute based methods do not identify the user, but verify if the user has some required properties, such as the right age group;
- *Privacy-Aware Computations* focus on the usage of private data. Computations on confidential input data allow to securely process various operations (run functions on the private data) without removing protection mechanisms. Privacy adding computations usually add a layer of privacy to their outputs, for example by refusing to answer queries that reveal too much about the inputs or by adding nose to numeric outputs;
- *Human-Data Interaction* combines technical means and user actions. It means that technical means might require some user interaction to work. Basically, user has a chance to regulate how these means process some information.

Current implementation of PE-BPMN covers four of these categories, the category of Human-Data Interaction has not been addressed yet.

2.3.2 Examples of supported PETs

Extension of the BPMN concrete syntax to add PETs is done using stereotypes, which are the representatives of privacy enhancing technology groups described in Table 1 and Section 2.3.1. The stereotype characterizes the changed type of the BPMN construct and it includes the representative name of PET and all technology-specific information (such as parameters). See Section 2.3.3 about the main technology-related details of stereotypes.

In the current implementation of PE-BPMN, the categories described in Section 2.3.1 are entitled slightly differently. Here are described some technologies from the current implementation.

Communication protection

One example of communication protection is a SecureChannel, which is a message flow stereotype in Communication protection category. SecureChannel represents the secure communication where information is sent to another participant in the process without any interference. The information is sent by an activity (a task) to an actor (a task of another participant) that receives it through a message catch event. SecureChannel can also transfer multiple data objects. It is assumed that the data objects that were received are always the same that were sent [PTMT18]. Note that SecureChannel is more of a generic name for the principle rather than being an exact technology.

Public key encryption

One common data protection technology is a public key encryption [Men97]. Public key encryption is represented by two task stereotypes in the Confidentiality Protection category (a sub-category of Data Protection): PKEncrypt and PKDecrypt. PKEncrypt takes two inputs – a data object as data to be encrypted and a public key – and produces a ciphertext. PKDecrypt reverses encryption by taking a secret key and a ciphertext as inputs and producing the initial data object as a result. Encryption schemes may also be homomorphic and allow computations on encrypted data. It means that computations can be run on encrypted ciphertexts, generating an encrypted result. If this result is decrypted, the result is the same as it would be when those computations were run on plain (not encrypted) text. In current implementation, computations that operate on public and encrypted data and produce ciphertexts as outputs, are represented by a PKComputation stereotype. PKComputation is a task stereotype in Privacy Preserving (stereotypes) category (a sub-category of Data Processing).

Intel Software Guard Extensions

Intel Software Guard Extensions[5] (SGX) technology [AGJS13] offers a secure hardware platform to protect data and code during computations. In current implementation, secure communication with SGX is denoted SGXComputation stereotype and also define a SGXProtect stereotype as a representative of confidentiality protection mechanisms to provide inputs for SGXComputation stereotype.

A key component of SGX is its remote attestation process [JSR$^+$16] that enables the processor to prove that it runs the right code in a secure SGX environment. This is

[5] https://software.intel.com/en-us/sgx

considered as an entity authentication type of technology, which is represented by two task stereotypes: SGXAttestationEnclave and SGXAttestationChallenge. To define which computations belong to the same enclave, stereotype groups are used. This allows the group to use protected inputs and outputs of each other.

Secure multiparty computation

Privacy-Aware Computations category is entitled in the current implementation as Data Processing and it is divided into two categories – Privacy preserving and Privacy adding.

An example of Privacy Preserving category would be a secure multiparty computation (MPC) [Yao82, GMW87], where multiple parties compute functions on their inputs while keeping these inputs private and not revealing them to other parties. In the current implementation of PE-BPMN, there is a general multiparty computation stereotype (MPC), but also stereotypes representing more specific technologies. For example secure computation can be achieved with homomorphic secret sharing [Sha79, Bla79]. Secret sharing technologies split secret data into multiple shares and define distributed protocols to compute on these shares while preserving the privacy of the inputs. The main tasks of secret sharing are producing the shares (SSSharing) and restoring the encoded data from the shares (SSReconstruction). In current implementation, the distributed computations, performed on the shares, that produce shared outputs are represented by SSComputation stereotype. MPC and SSComputation are task stereotypes in the Privacy Preserving category (a sub-category of Data Processing), while task stereotypes SSSharing and SSReconstruction are actually in Confidentiality Protection category (a sub-category of Data Protection).

All implemented stereotypes

In addition to previously described examples, there are many more stereotypes implemented – 32 task, 2 message flow and 2 data object stereotypes altogether. Full list of implemented stereotypes and their positioning in terms of the categories mentioned in this section, can be found in Section 3.2.1.

2.3.3 Stereotype details

Stereotypes are representatives of privacy enhancing technologies, which of each have technology-specific restrictions and requirements. These requirements are mainly related to numbers and types of inputs and outputs, values of parameters, and members of groups. There are also some restrictions concerning combining different stereotypes.

Numbers of inputs and outputs

Each technology has its specific requirement for the number of its inputs and outputs. For example, encryption process requires a key and data to be encrypted as inputs and provides encrypted data as an output. In this case, it is practical to expect that there are only two inputs – key and data to be processed. There is no need for extra inputs. If there were less inputs, the encryption process could not be initiated. PKEncrypt requires two inputs – a key and an input data and one output as a ciphertext.

Types of inputs and outputs

There are technologies that require their input to be of specific type. Almost every technology has certain expectations for the type of their inputs. Common examples would be decryption and computation technologies. For example, concerning decryption, the input data to be decrypted should be first encrypted, otherwise the process would have no meaningful output and also, to decrypt something, correct key is required. In terms of particular decryption technology, the key should be of appropriate type. PKDecrypt requires that the key is of PKPrivate type and from the same pair with the key (of type PKPublic) that the information to be decrypted was encrypted with.

Parameter values

Some technologies require additional parameters in addition to input and output data, for example computation script. To use technology, it might require that its parameters must have a value or / and this value must be in a fixed range to adjust the behaviour of the technology and to have a meaning. For example, secret sharing splits private values among participants so that some predefined groups of parties can collaboratively restore the secret. In case a (t, n)-threshold secret sharing scheme is used, the data is shared among n participants (e.g. number of shares) and any subset of $t \leq n$ or more participants is able to restore it. SSSharing uses (t, n)-threshold secret sharing scheme and has requirements concerning parameters – threshold (t) and number of computation parties (c) - $1 < t \leq n$ and $t \leq c \leq n$, (n is the number of outputs from the task). These requirements secure that one share or any set of shares, less than specified by the assumption, do not leak information about the shared secret and that only the threshold or qualified sets of parties can restore the secret.

Groups

There is an important part of privacy-enhancing technologies that require to run computations in parallel or in multiple parts. Groups denote computations that in some sense belong together. For example, in terms of multiparty computation, some technologies require that computations are run in parallel. On a model, this is expressed by dividing one computation into multiple computation tasks (sub-tasks of a joint task). Running computations in parallel means that computation tasks that multiple parties have, must be executed jointly (in parallel). These sub-tasks represent members of the stereotype group. MPC requires that there are at least two members in the group and they must run in parallel.

Specifying computations

There are computation technologies that can use the output of another stereotype (of different type of technology) as an input to run operations on. The operations of the computation stereotypes can be specified by computation scripts. In addition to computations, which take in the script as a simple parameter value, we also need to consider a layered approach where some computation protocol may actually compute another privacy enhancing technology. To this end, it is possible to also specify stereotype details with using another stereotype. For example, SGXComputation with PKEncrypt as a computation specification means that the computations are carried out using Intel SGX technology, but the operation that is computed is an encryption and the output of the computation is a ciphertext.

There are also some restrictions that are specific to only one stereotype. All stereotype-specific restrictions are covered under paragraphs of each related stereotype (see Appendix II).

2.3.4 Example of PE-BPMN model

Figure 3 describes a process with public-key encryption. There are also computations on the encrypted information and eventually, decryption of the encrypted information. For the visualizations of the main BPMN elements, see Figure 1.

The process begins with a start event and ends with an end event.

Participant 1 and *Participant 2* (Pool elements) represent different stakeholders. Message exchange between pools correspond to information sharing and, hence, potential leakages. However, the use of privacy enhancing technology (for example, public-key encryption – PKEncrypt) allows to reduce the risk of leakages.

Encrypt input data (with PKEncrypt stereotype) and *Send data* are examples of different activities (tasks) related to processes. Stereotypes (indicated by names in blue) mark that a certain task represents the activity using the privacy enhancing technology.

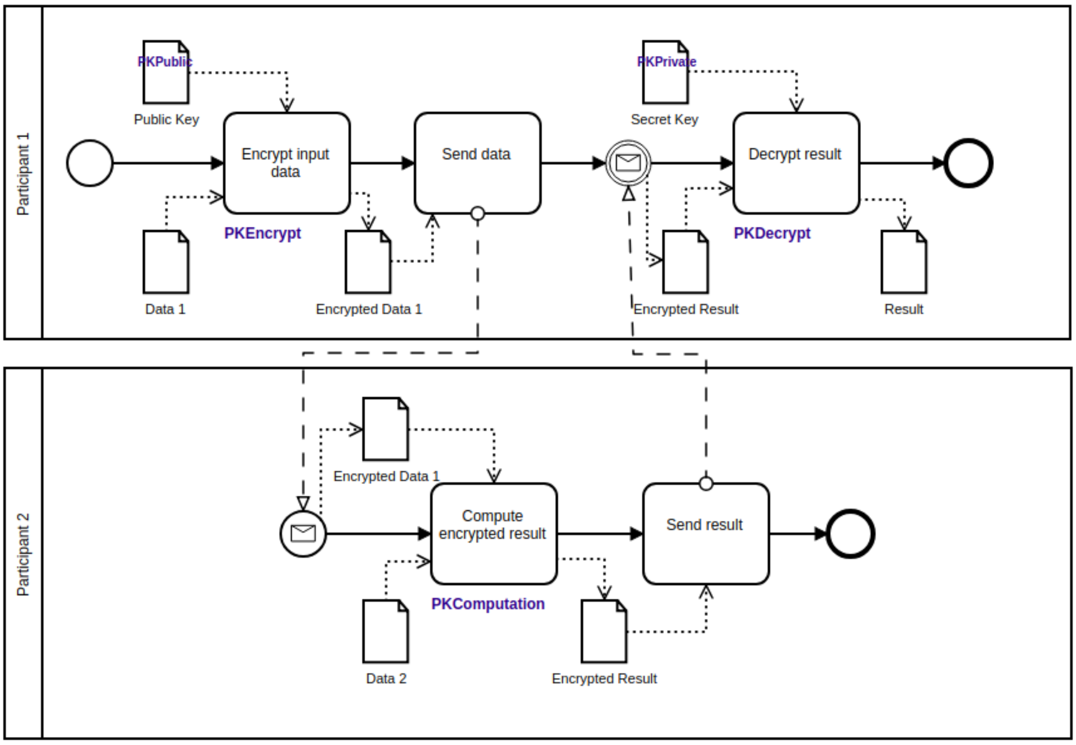


Figure 3. Representation of a process that uses encryption. Stereotypes attached to elements are expressed with names in blue text.

Elements entitled *Public key* and *Data 1* are representatives of information carriers or stores (for example, a database table). In context of this model, PKPrivate and PKPublic stereotypes mark that the elements they are attached to are keys of certain type used by PKEncrypt and PKDecrypt stereotypes. The encryption key used by PKEncrypt is of PKPublic type and the decryption key used by PKDecrypt is of PKPrivate type. The decryption key is also from the same pair as the encryption key used to encrypt the information being decrypted.

2.4 Privacy LEAKage Analysis Tools

The Privacy Leak Analyzer (PLEAK) is the NAPLES (Novel tools for Analyzing Privacy LEakageS) project tool for modelling and analyzing business processes with regard to privacy and experimenting with how different privacy technologies reduce the leakage of private data. The PLEAK tool is designed to support the analysis of private data flows in enterprise business processes (BP) and programs. PLEAK can be used by BP analysts, developers and maintainers in order to understand the privacy implications of the business processes used or planned by their organizations or customers. PLEAK provides them with a tool that they can use to describe their system by its private data elements, stakeholders, business processes and, optionally, data analysis algorithms. Once this is done, PLEAK will show how private data flows through a system, who does it leak to and to what extent does it leak.

Currently, PLEAK consists of backend, frontend and analyzers. Backend component provides file and user management platform and database. Frontend component provides a user interface to access, create, edit, share, publish, export BPMN models and separate them into folders. Frontend includes also a modeller component to create BPMN models.

In addition to main PLEAK platform, there are multiple analysis tools, such as SQL global sensitivity analyzer, SQL derivative sensitivity analyzer and leaks-when analyzer. All these analyzers are command-line applications that have an user-interface (an editor) built on top of them. BPMN models that are created in or imported into the modeller, can be used in all of these editors, where it is possible to attach analysis-related information to model elements, for example, SQL-scripts and -schemas in context of SQL global sensitivity analyzer.

The result of this thesis, the PE-BPMN editor, is another extension tool for PLEAK analysis tools. There are currently two analysis methods implemented into the editor – simple disclosure and data dependencies analysis. These analysis are beyond the scope of this thesis, but are described in [PTMT18].

2.5 Discussion

BPMN standard has been one of the most used notation to express business processes and information flows. It was chosen for PLEAK analysis tools for modelling and analyzing business processes with regard to privacy and privacy leakages. As currently the BPMN notation does not offer simple solutions to visualize and analyze movements of private information as it is disclosed to participants of business processes, PE-BPMN language was proposed to address the situation. PE-BPMN adds a new layer to the notation by extending it with ways to express the usage of privacy enhancing technologies. As there has been no implementation to use the PE-BPMN language yet and PE-BPMN based models are necessary for PLEAK analysis tools to express information flows of private data and detect privacy leakages in analysis based on these information flows, the result of this thesis is a tool that provides that implementation for PLEAK tools. However, the BPMN notation is rich of elements and symbols and the current implementation of PE-BPMN does not yet support full BPMN notation, so only the main elements of BPMN are described in this work.

3 PE-BPMN editor

PE-BPMN editor is a user-facing frontend application that implements support for BPMN extension (PE-BPMN). It allows to extend BPMN models by adding various PE-BPMN stereotypes to their elements and run analysis based on these extended models. Important part before running analysis is to validate whether the model to be analyzed is syntactically and semantically correct. The general idea of this section is to describe the implementation of PE-BPMN editor and its stereotypes that build up the basis for the validation of PE-BPMN models and further analysis on these models.

3.1 Implementation

PLEAK web-based tool with PE-BPMN editor is accessible at `https://pleak.io`. Full source code for the PLEAK tool is available at `https://github.com/pleak-tools` and for PE-BPMN editor at `https://github.com/pleak-tools/pleak-pe-bpmn-editor`. Installation instructions can be found in README files of these repositories.

3.1.1 Technologies used

PE-BPMN editor is an Angular[6] web-application built on a standard @angular/cli (version 1.6.7)[7] project. Graphical user-interface is based on HTML[8] and JavaScript[9], using Bootstrap (version 3.3.7)[10] and JQuery (version 3.3.1)[11] in addition. Bpmn-js (version 1.3.0)[12], an open-source BPMN 2.0 rendering toolkit and web modeller provided by Camunda Services GmbH[13], provides a toolkit to create and modify BPMN models.

Bpmn-js provides a visualization of XML[14]-format BPMN models, so that the editor:

- displays BPMN models and lets users to attach stereotypes to models (to add PETs using BPMN standard elements);
- saves the models using standard serialization format, which uses XML as a foundation.

[6] `https://angular.io/`

[7] `https://cli.angular.io/`

[8] `https://www.w3.org/html/`

[9] `https://www.javascript.com/`

[10] `https://getbootstrap.com/docs/3.3/`

[11] `http://jquery.com/`

[12] `http://bpmn.io/toolkit/bpmn-js/`

[13] `https://camunda.com/`

[14] `https://www.w3.org/XML/`

PE-BPMN editor extends BPMN models by attaching specific (named after stereotypes) labels into the XML code of the model file. This is explained more thoroughly in Section 3.1.2. Model contents are preserved as files, but meta-data of these files is stored in MySQL[15] database that is a part of PLEAK structure.

PE-BPMN editor is an extension tool to PLEAK, so its backend is provided by PLEAK. The backend provides REST[16]ful API service for user authentication, accessing model's meta-data and saving model's information.

3.1.2 XML-representation of stereotypes

BPMN models can be created and edited in PLEAK modeller. Created models are XML-format files with .bpmn extension. PLEAK uses Bpmn-js modeller tool to convert model elements as an input from graphical user interface into XML-format files. Thanks to Bpmn-js tool, there is no need for an extra code to create and parse XML code and to check whether model files are correctly formatted.

Each stereotype has a unique name and a specific set of details concerning the technology. For example, to use a stereotype that represents an encryption technology (such as PKEncrypt stereotype), information about its input data objects – key and data to be processed – must be somehow stored. Serving that purpose, stereotype details are packaged into a JavaScript object. This object is stringified (using JavaScript's JSON.stringify() method). As Bpmn-js parses XML-format BPMN model into a JavaScript object where information attached to a XML-tag can be accessed as a value of a property. The stereotype's information string can be set as a value to the property with the name of that stereotype, which means that model object's properties can be easily manipulated when required.

PE-BPMN editor uses Bpmn-js to translate .bpmn model files into graphical models. As Bpmn-js removes unknown XML-tags from the model during parsing process, specific XML-tag is defined for each stereotype and the list of these stereotypes is provided to Bpmn-js as a whitelist to be accepted. This approach allows to attach necessary information to all model elements.

In Figure 4 is an example of the PKEncrypt stereotype in a visual form. In the the Figure 5 is the same stereotype presented in XML format. Stereotype's information is saved into JSON string (of a JavaScript object) that is between <pleak:PKEncrypt>

[15] `https://www.mysql.com/`

[16] `https://www.service-architecture.com/articles/web-services/representational_state_transfer_rest.html`

and </pleak:PKEncrypt> tags. The JSON string contains IDs of a key and an input data objects (accordingly, Public Key and Data 1 in Figure 4). Properties *key* and *inputData* fix the roles of input data objects. The string is in a CDATA block, which means that the contents of that block could be interpreted as XML markup, but should not be. Otherwise, it would be possible to inject something into XML code as not just a string, but as a XML tag.

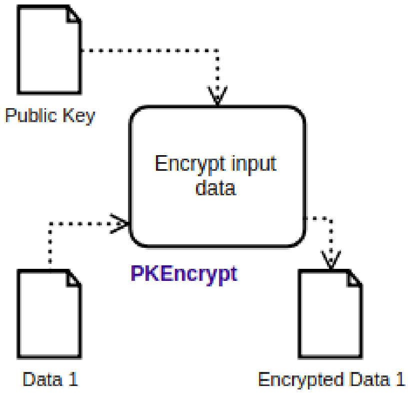


Figure 4. PKEncrypt stereotype attached to a task with two inputs and one output.

```
<pleak:PKEncrypt>
  <![CDATA[
    {
      "key":"DataObjectReference_0s3yw9a",
      "inputData":"DataObjectReference_10n52fp"
    }
  ]]>
</pleak:PKEncrypt>
```

Figure 5. XML representation of the PKEncrypt stereotype in Figure 4.

3.2 User interface

3.2.1 Stereotypes in menus

Stereotypes can be added to model elements by selecting them from the menu that appears next to the clicked element after clicking on a task, a data object or a message flow on a model.

There are two message flow stereotypes: CommunicationProtection and SecureChannel. For data objects, there are also two stereotypes: PKPublic and PKPrivate.

The rest of the stereotypes are task stereotypes and they are grouped based on the taxonomy. The goals that the task can have are data protection, data processing and entity authentication. Data protection has two possible targets: integrity and confidentiality protection. In data processing the targets are either privacy preserving or privacy adding computations. At the moment there are no stereotypes of the human-data interaction category implemented yet. After choosing the goal it is possible to choose the concrete stereotype. Each stereotype opens a stereotype settings menu (see Section 3.2.2) to specify the necessary parameters and roles of the inputs and outputs.

There are three categories of task stereotypes (see Figure 6) and thirty-two task stereotypes divided into these categories. These stereotypes are positioned in the menu based on the categorization of task stereotypes in Table 2. Note that some stereotypes are in multiple categories.

Table 2. Categorization of task stereotypes.

Category	Sub-category	Stereotypes
Data Protection	Integrity Protection	SGXQuoting, SGXQuoteVerification
	Confidentiality Protection	PKEncrypt, SKEncrypt, SSSharing, AddSSSharing, FunSSSharing, SGXProtect, ProtectConfidentiality, PKDecrypt, SKDecrypt, SSReconstruction, AddSSReconstruction, FunSSReconstruction, OpenConfidentiality
Data Processing	Privacy Preserving	PETComputation, MPC, SSComputation, AddSSComputation, FunSSComputation, GCEvaluate, GCGarble, GCComputation, SGXComputation, PKComputation, SKComputation, OTSend, OTReceive
	Privacy Adding	DimensionalityReduction, DifferentialPrivacy, PETComputation
Entity Authentication		SGXAttestationChallenge, SGXAttestationEnclave

See Figure 7 for an example of a sub-category in the main stereotypes menu. It shows that the Integrity Protection category (sub-category of Data Protection) is also divided into two more smaller sub-categories. This multi-level classification applies to other categories as well.

Some of the related technologies are described in Section 2.3.2, restrictions and requirements of each stereotype are covered in Appendix II.

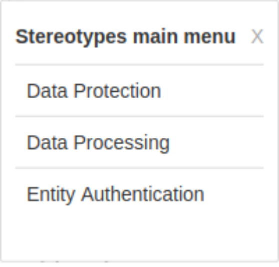


Figure 6. The main menu of task stereotypes.

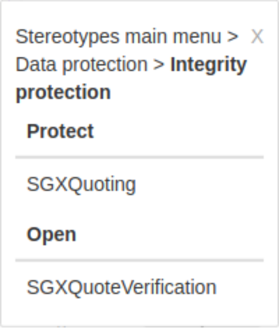


Figure 7. A menu of task stereotypes – category Data Protection, sub-category Integrity Protection.

3.2.2 Stereotype settings

Stereotype settings panel can be accessed in two ways. Firstly, if there are no stereotypes added to an element yet, the panel can be opened by clicking on an element and selecting the stereotype from stereotypes menu. Secondly, if there is already at least one stereotype attached to an element, the settings panel can be accessed by clicking on the element on the model. In both cases, the stereotype settings panel opens in a sidebar. Note that to really add the stereotype to an element, the *Save* button in the settings panel must be clicked. Otherwise, it will be not added as the adding processed is cancelled and the panel is closed after clicking outside of the sidebar or the element that the stereotype is supposed to be attached to. Stereotypes can be removed from an element by clicking the *Remove* button in the settings panel.

In case user has attached a stereotype to an element and clicks on the element, in addition to the settings panel appearing in the sidebar, input and output elements of the clicked element are highlighted. Input objects of a task are highlighted in green colour, output objects in pink. If a task with a stereotype belongs to a group, all group members

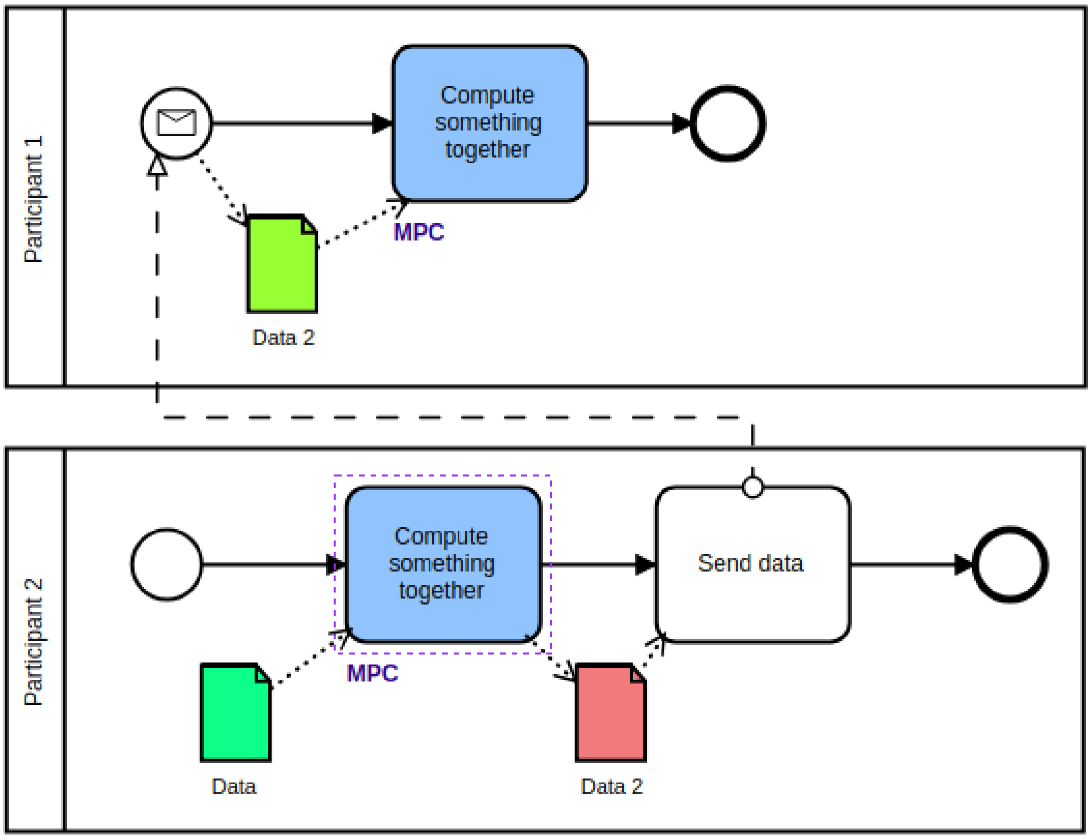
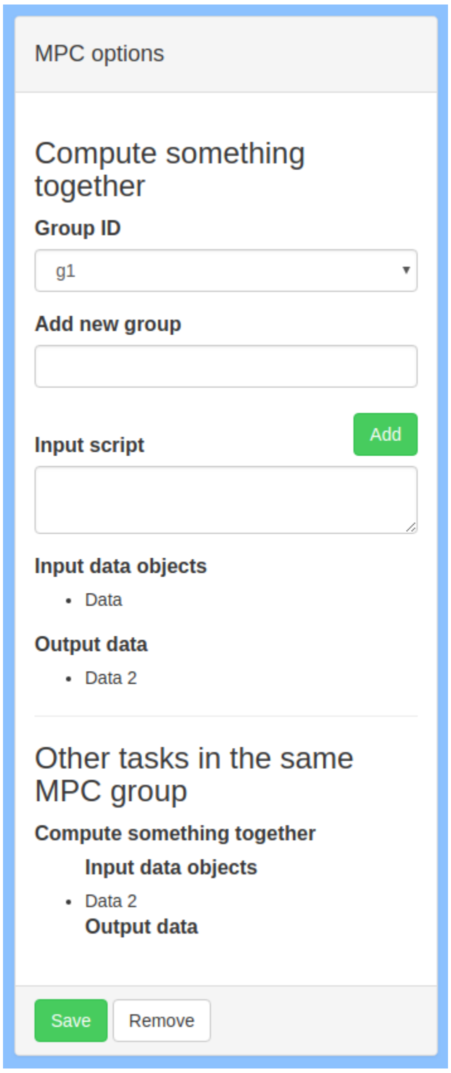


Figure 8. MPC stereotype group tasks and their inputs and outputs are highlighted after clicking on the *Compute something* together task of *Participant 2*.

are highlighted in blue and all group members' inputs and outputs are highlighted following the same logic as inputs and outputs of the clicked element. However, inputs and outputs of the clicked task are highlighted in slightly different tones of colours than the other tasks of the group. See Figure 8 for an example. Also, in case data objects have stereotypes attached to them, the objects are highlighted in blue.

Setting panel allows the user to choose different stereotype-specific options and set values to various parameters. One of the main parameters is the group of the computations. Some tasks form natural groups and are not meaningful on their own. For example, MPC stereotype has a meaning, if there are at least two members (tasks) in the stereotype group and they are all executed by different stakeholders. To create a new group in stereotype settings, the user should insert the group name into *Add new group* text input and click the *Add* button under it. After adding a new group, the name of it appears into *Group*

ID menu. In case there are already some existing groups, the user can choose the group from *Group ID* dropdown menu by clicking on the group name that is visible under the label *Group ID*. Changing the group of the task also changes the highlighting of the group members of the task accordingly – members of the selected group are highlighted together with the selected task. See Figure 9 for an example of stereotype settings panel of MPC stereotype that has already one group, entitled "g1", added and selected.



MPC options

Compute something together

Group ID: g1

Add new group

Input script

Input data objects

- Data

Output data

- Data 2

Other tasks in the same MPC group

Compute something together

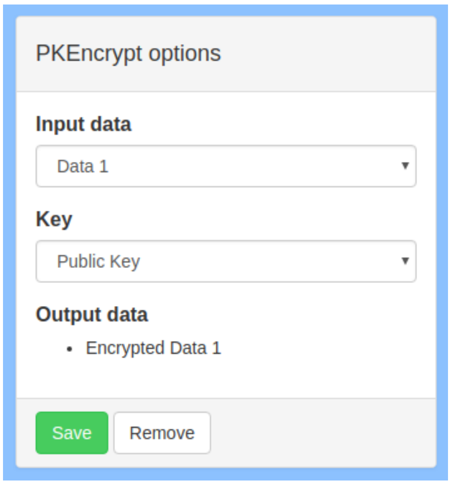
Input data objects

- Data 2

Output data

Figure 9. Stereotype settings of the task with MPC stereotype that belongs to a group g1. This is the settings panel that opens after clicking the task highlighted in Figure 8.

Another important parameter is a role for input and output data objects of a task. For example, for PKEncrypt stereotype it is necessary to specify which input acts as the key and which as the plaintext. See Figure 10 for an example.



PKEncrypt options

Input data

Data 1

Key

Public Key

Output data

- Encrypted Data 1

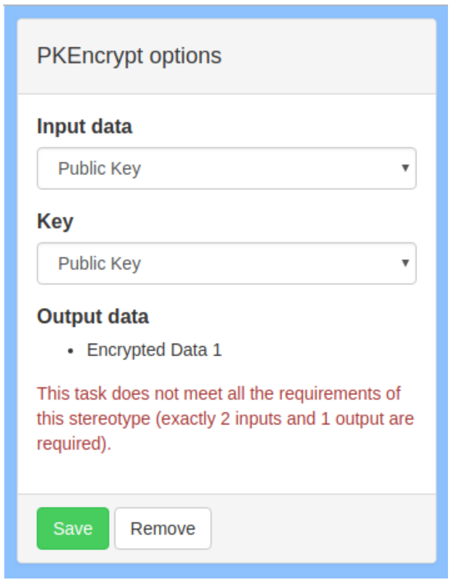
Figure 10. Stereotype settings of the PKEncrypt stereotype where the key and input data roles must be selected for both input objects.

One more common parameter is an input script. Some task stereotypes have it as a task specific script (for example PKComputation), some have it as a script shared and accessed between a group of tasks in a stereotype group (for example MPC). In both cases, input script can be inserted into *Input script* text input of stereotype settings panel (see Figure 9). Currently there are no restrictions for input script implemented in PE-BPMN editor yet.

There are also few parameters that are specific for only one stereotype. These parameters have mostly values of plaintext or numbers.

Note that not all stereotypes can be added to all elements, for example for tasks there has to be suitable number of inputs and outputs. For some stereotypes, it can be that there are special roles that the inputs or outputs have. For example, an encryption operation has two distinct inputs – the key and the plain text – that can be identified on the model. If an element conflicts one or more of stereotype restrictions, an error message is shown in stereotype settings panel and the stereotype cannot be attached to the element. Error message lists the number of expected inputs and outputs and also information about the requirements concerning parameters (see Figure 11 for an example).

Also, note that the user interface only allows to add a stereotype to an element that meets the requirements at the time of adding the stereotype. Later changes to the element or the model in general can make these requirements unmet. In case an element has a stereotype attached to it, but requirements of the stereotype are not anymore met, the validation report (see Section 4.2.2) is created (as a result of the validation process). The report is a list of errors and warnings that cover discrepancies between a model and these requirements.



PKEncrypt options

Input data

Public Key

Key

Public Key

Output data

- Encrypted Data 1

This task does not meet all the requirements of this stereotype (exactly 2 inputs and 1 output are required).

Figure 11. PKEncrypt stereotype cannot be attached to a task because the number of the inputs does not meet the requirements of the stereotype. An error message is shown in the settings panel of the stereotype.

3.3 Discussion

Previous sections have explained the choices made to implement a tool, entitled PE-BPMN editor, for using PE-BPMN language in practice. The implementation part of the PE-BPMN consists of two layers. Firstly, the technical solutions to implement the concept of the PE-BPMN, and secondly, the user interface. The main question behind these two layers has been how to represent the privacy enhancing technologies on BPMN models. As BPMN models are stored as XML files, extending XML files with extra labels was the approach that was chosen. The tool provides the user an interface to visually extend BPMN models with privacy enhancing technology representations (called stereotypes), as a result, PE-BPMN models are created. Under this user interface is a process of manipulating XML files.

Described approach was chosen, because PLEAK uses a tool (Bpmn-js) that allows easily to manipulate BPMN models through changing models' XML files, without the need to develop algorithms for the PE-BPMN editor to do it itself. With this approach, more focus can be put on expressing different privacy enhancing technologies, rather than finding ways how to store information related to these technologies. Described solutions are useful for the user of the tool because more technologies are more easily supported and also, creating PE-BPMN models in described way does not break syntax rules of BPMN, so these files can be also used in other BPMN tools, without needing to remove stereotypes from the model.

4 Validation process of PE-BPMN models

PE-BPMN stereotypes define various rules that need to be obeyed when adding stereotypes to the model in order for the model to have a reasonable semantics. Some rules are stereotype specific, e.g. the number of inputs and outputs a stereotyped task needs. This number is defined under each stereotype and it can be easily verified when adding the stereotype to the model. Additionally, there are some stereotype parameters that can be assigned through stereotype settings that must be consistent with the model. The interface does not let the user to insert incorrect values, but it is still necessary to verify these requirements in case the model has been tampered with by assigning these parameters through editing the model's file directly. There are also requirements that apply to several stereotypes at once and therefore require the general context of the process model to verify important properties. The validation process of PE-BPMN editor covers all of these and many other restriction and requirements checks related to stereotypes. The general idea of this section is to describe the implementation of PE-BPMN editor's validation process and the meaning of the validation of PE-BPMN models.

4.1 Implementation

4.1.1 Layers of validation

Validation process covers restriction and correctness checks of all tasks, data objects and message flows that have stereotypes attached to them. In general, validation process assumes that in terms of BPMN a model is syntactically correct, but it might have a structure that does not meet the requirements of specific technologies described on a model. These requirements and restrictions of privacy enhancing technology stereotypes are described in general in Section 4.1.2. Technology-specific requirements and restrictions are described under paragraphs of each each related stereotype (see Appendix II).

Before the real validation could start, preliminary information gathering is required. It is done in parallel with loading a model into the PE-BPMN editor. The process of information gathering, required to run validation process (correctness checks), covers:

- gathering information about all tasks, data objects and message flows and creating JavaScript objects to represent these elements – gathered information includes details about the element in terms of a BPMN model (element ID, name, etc.) and details about stereotypes attached to it in terms of PE-BPMN (stereotype name, parameters, etc.);

- gathering information about lanes and pools – this is necessary for correctness checks concerning parallelism (see Section 4.1.2.8).

It is necessary that this information gathering has finished before showing any results of the validation, otherwise, due to the limitations of the PE-BPMN editor, it might happen that user gets inconsistent validation results. As models could be large, containing hundreds of elements, this information gathering may take some more time than user would expect and to prevent further frustration because of long delay before seeing validation results, it is hidden into the loading process. It means visually that user cannot press Validate button before preliminary information gathering and model loading have finished. After these processes have finished, Validation button is turned active and user can start the main validation process.

The main validation process of a PE-BPMN model is divided into three sub-processes: validation of tasks, data objects and message flows. These sub-processes cover all task, data object and message flow stereotypes with correctness checks – first, all task stereotypes, after that, all data object stereotypes, and at last, all message flow stereotypes are checked. The validation process is not finished until all three parts have finished. To avoid running continuous checks whether all sub-processes have finished, they are run sequentially. When tasks with stereotypes are covered or there are no task stereotypes on the model, validation of data objects begins. Validation process of tasks, data objects and message flows follows the same logic. When the complete validation process has finished, the validation report (Section 4.2.2) is created.

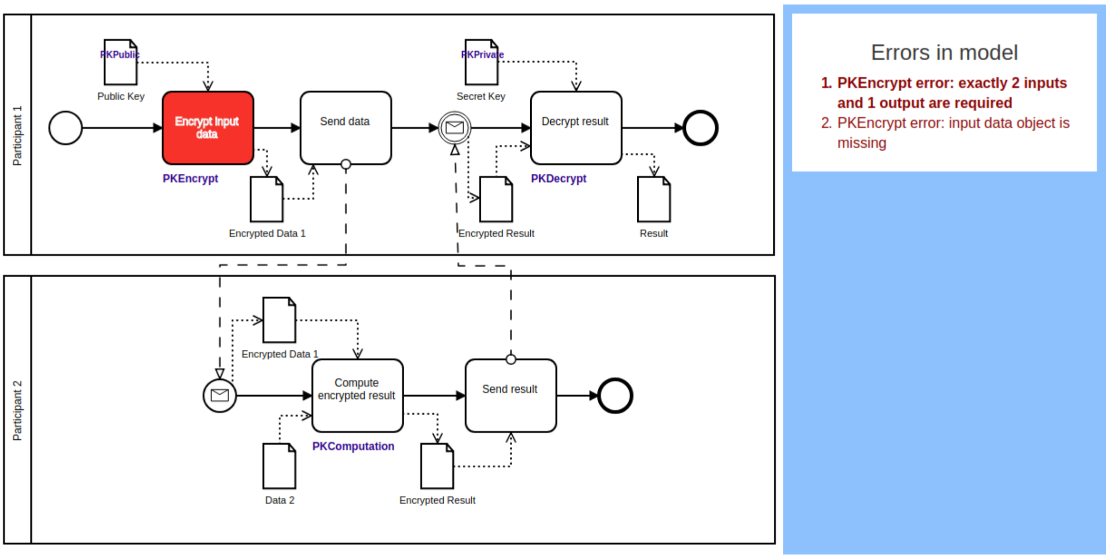
4.1.2 Types of correctness checks in validation process

In general, most stereotype-specific restrictions and requirements cover two cases. Firstly, the case when the user is going to attach a stereotype to an element and it needs to be checked whether the element meets all the requirements. Secondly, the case when an element has already a stereotype attached to it, but the model has been changed or the model's file has been manipulated. In more detail, these two types of restrictions and requirements can be divided into eight main types of correctness checks.

4.1.2.1 Type I: Number of inputs and outputs

Semantics:

Intuitively, each technology has its specific requirement for the number of its inputs and outputs. For example, encryption process requires a key and data to be encrypted as inputs and provides encrypted data as an output. In this case, it is practical to expect that there are only two inputs – key and data to be processed (see Figure 12). If there were less inputs, the editor would need a different approach to find the key or data to be processed and if there were more input objects, there would be just more information to extract the necessary part from. Same logic follows output objects. Almost all stereotypes require a



Errors in model

1. PKEncrypt error: exactly 2 inputs and 1 output are required
2. PKEncrypt error: input data object is missing

Figure 12. PKEncrypt encryption with a missing input data.

specific number of inputs and outputs.

Technical solutions:

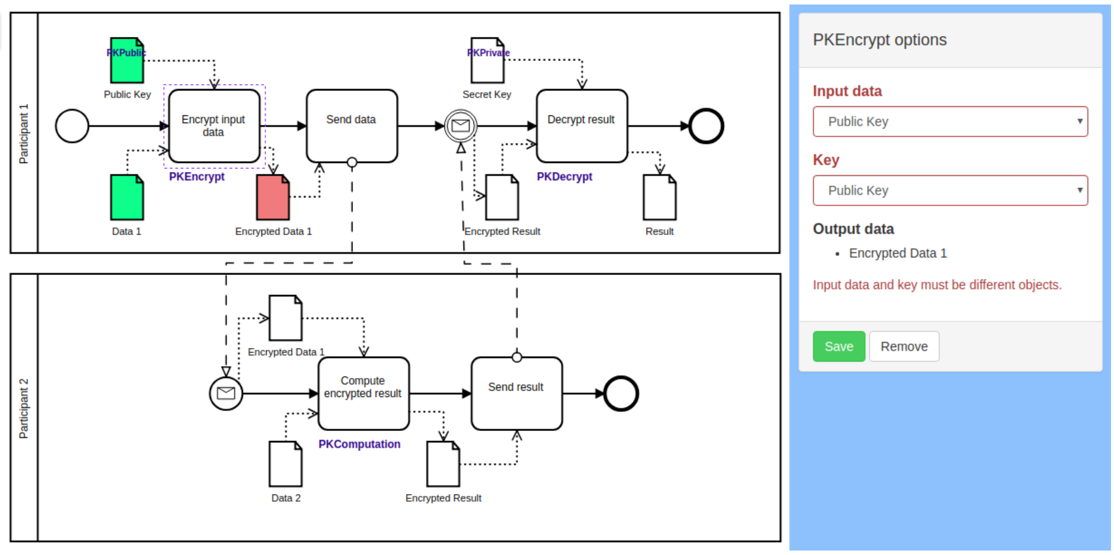
Between each task and data object connected to it, there is a specific connecting element (Data Association) that exists only in case both sides of the connection exist. These associations have a direction – from the perspective of task, incoming and/or outgoing. To get the number of input and output elements (data objects) these connections are counted and separated by their direction. These connections can be also bidirectional, which means that a data object can be for a task as an input and an output at the same time. That kind of elements are considered in both numbers of inputs and outputs.

Possible error messages and reasons behind them:

- **error and reason:** task must have *exactly* / *at least* / *at most* x inputs and *exactly* / *at least* / *at most* y outputs.

Restrictions related to numbers of inputs and outputs and parameters are covered in Table 3 and under paragraphs of each related stereotype (see Appendix II).

4.1.2.2 Type II: Existence and roles of inputs and outputs



PKEncrypt options

Input data

Public Key ▾

Key

Public Key ▾

Output data

- Encrypted Data 1

Input data and key must be different objects.

Figure 13. PKEncrypt encryption, where the user is trying to use the *Public key* object as a key and input data (to be encrypted) at the same time.

Semantics:

In the description of the semantics of stereotype restriction Type I (Section 4.1.2.1) it is stated that encryption requires two input objects – a key and data to be processed. Logically, in order to use the information about these objects, these objects must exist and there needs to be a way to locate and distinguish them from other elements. In context of encryption, one of the inputs objects is supposed to contain information about the data to be processed, so the object must definitely exist (see error 2 in Figure 12). As the key and data to be processed are supposed to be different, they must be two different objects (see Figure 13, where user is stopped from using the Public Key object in two roles at the same time).

To be more precise about difference between objects, in terms of BPMN, different objects means that they have different names. There are cases when multiple elements on the model have the same name. Regarding the semantics of the model, these elements are understood to be the same. For example, the same encryption key (multiple objects with the same name) could be used multiple times in same processes. Key and data to be processed in the other hand can possibly have the same name but must have different IDs. In the context of currently described restrictions, difference between objects is considered in two possible meanings:

1. the requirement of having two elements that could have the same name, but must

have different IDs to be considered different;

2. the requirement of having two elements that have different IDs, but must have also different names to be considered different.

These restrictions extend to all stereotypes that need some specific information attached to them – for example, encryption and computation stereotypes, also stereotypes that form groups.

Technical solutions:

Firstly, in case the stereotype requires saving specific information (for example, ID of an input data object selected as a key for an encryption stereotype) into its JavaScript object / JSON string, there is a check to make sure that this stereotype object has all required properties. The check examines if all properties are defined and their values are set – in some cases, properties can have values as empty strings, but they cannot be undefined.

Secondly, in case there is information saved into stereotype object about model elements (their IDs or names), there is a check to make sure that elements with these IDs or names that are marked as inputs and outputs of a task in this object are also present on the model and have Data Association connections with the task. Bpmn-js provides a method to access all information related to an element by its ID. This method of Bpmn-js returns a value based on the input ID – if something is returned, the element with this ID exists. The check finds all input elements (that have Data Association connection with the task) and checks whether the supposed input element by its ID (the ID saved into stereotype object) is included in this returned list of input elements.

Also, in case the stereotype requires that input and output objects have specific fixed roles, there is a check to make sure that each object has its own role and the same role is not selected for multiple objects – whether the IDs or names of elements related to roles are different.

Possible error messages and reasons behind them:

- **error:** x is *undefined* / *undefined (it can be empty, but cannot be undefined)*;
  reason: x property must exist under stereotype details object and must not have been deleted from model's file;

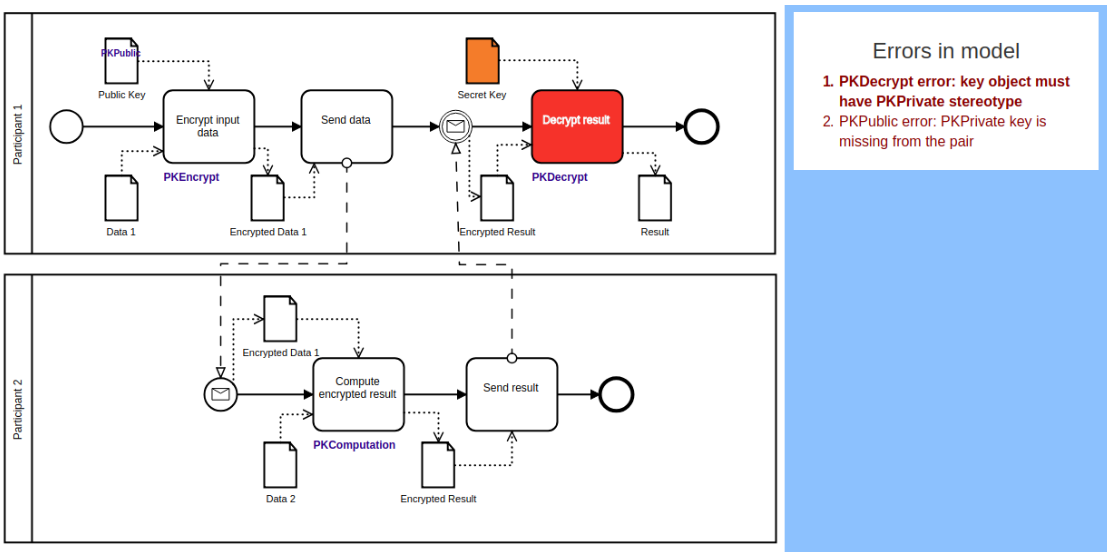
- **error:** x object is missing;
  reason: x object fixed (by its ID) in x property must exist also on model;

- **error:** x and y must be different objects;
  reason: x and y objects must be different data objects – they must have different IDs / names;

- **error:** x must be the same (with the same name) for all inputs;
  reason: all x input objects of input data objects that origin from y or z tasks must be with the same name.

Restrictions related to the existence and roles of inputs and outputs and parameters are covered in Table 3 and under paragraphs of each related stereotype (see Appendix II).

4.1.2.3 Type III: Types of inputs and outputs



Errors in model

1. **PKDecrypt error: key object must have PKPrivate stereotype**
2. PKPublic error: PKPrivate key is missing from the pair

Figure 14. PKDecrypt decryption with wrong type of key (missing PKPrivate stereotype).

Semantics:

There are technologies that require their input to be of specific type. Naturally, it is logical to assume that almost every technology has certain expectations for the type of their inputs. Common examples would be decryption and computation technologies. For example, concerning decryption, the input data to be decrypted should be first encrypted, otherwise the process would have no meaningful output. In addition, to decrypt something, correct key is required. In terms of particular decryption technology, the key should meet specific requirements – it should be of appropriate type. PKDecrypt requires that the input information to be decrypted has been encrypted using PKEncrypt technology (see Figure 14). The key should be of PKPrivate type and from the same pair with the key (of type PKPublic) that the information to be decrypted was encrypted with. These restrictions extend (in various sets) to all decryption and computation stereotypes.

Technical solutions:

Firstly, in case the stereotype requires that an input must have a certain type of stereotype attached to it, there is a check that uses a method (provided by Bpmn-js) to access all information related to an element by its ID. It checks whether the JavaScript object returned by this method has property named after the required stereotype. For example, PKDecrypt requires that the input key is of PKPrivate type – it means that they key object must have a PKPrivate stereotype attached to it.

Secondly, in case the stereotype requires that an input must be encrypted or encrypted with a certain encryption method, there is a check that checks if there exists a correct key (of specific type) for that input – it means that it searches through the incoming path to find out if there exists a task that uses acceptable technology (has specific stereotype) and has an input element with the same name that the input that is required to be encrypted. For example, PKDecrypt considers tasks in the incoming path to be acceptable if they have PKEncrypt or PKComputation stereotype attached.

Thirdly, in case the stereotype requires that at least one input must be selected as encrypted / private / SGXPrivate, there is a check that checks if user has selected in stereotype settings at least one input element to be encrypted / private / SGXPrivate.

Also, in case the stereotype requires that all inputs marked as encrypted / private / SGXPrivate are really encrypted / private / SGXPrivate, there is a check that checks if there exists a correct key for each of these inputs or if the input element is an output of a task that uses accepted technology (has specific stereotype). The task with accepted stereotype must have an output (with the same name as the input required to be encrypted) and it must be marked as encrypted under the stereotype settings of the task that this element is output of. More technically, it means that the check searches through the incoming path to find out if there exists a task that uses acceptable technology and has an input element with the same name as the input that is required to be encrypted or if there exists a task that uses acceptable technology and has the output with the same name as the input required to be encrypted marked in stereotype's JSON string as encrypted. In context of restrictions, incoming path is considered as a list of all elements that are sequentially connected to each other with directed Data Associations or other similar connections and from the perspective of an element the direction of all these connections is incoming. Same logic applies to outgoing path, but with opposite direction of connections.

Possible error messages and reasons behind them:

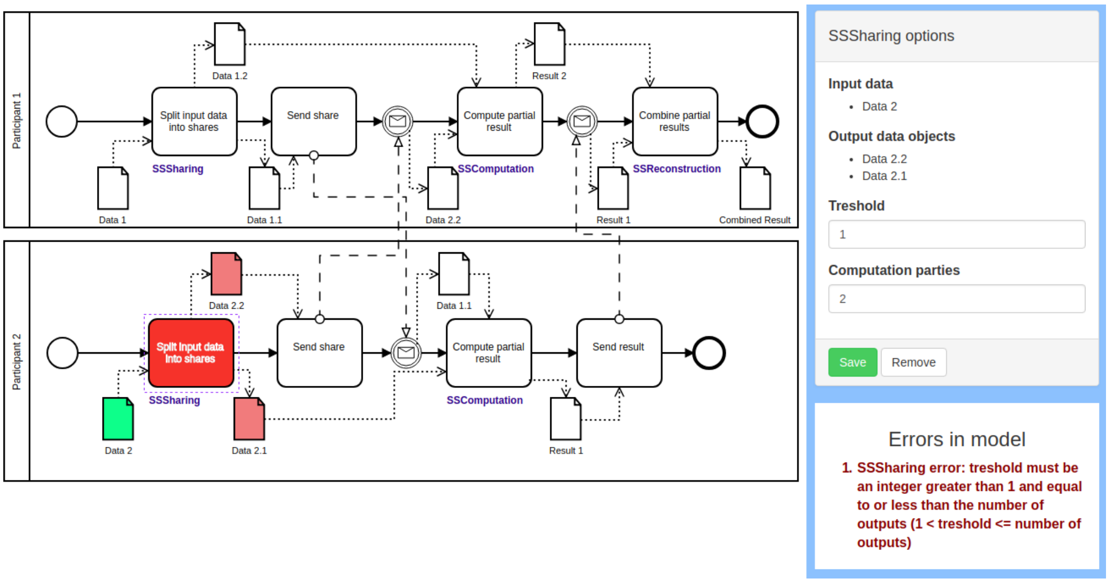
- **error and reason:** x data object must have y stereotype;
- **error:** x is encrypted with wrong encryption method or is not encrypted;
  reason: there must exist a correct key corresponding to the x – input data object selected as x must be an input object x from z task or one of the input objects from w task selected as *encrypted* / *private* / *SGXPrivate* – it must have the same name as input object x from z task or as one of the input objects of w task selected as *encrypted* / *private* / *SGXPrivate*;
- **error:** at least 1 input must be selected as encrypted;
  reason: at least one input data object must have type selected as *encrypted* / *private* / *SGXPrivate*;
- **error:** all inputs marked as encrypted are not encrypted;
  reason: all input data objects that have type selected as *encrypted* / *private* must be encrypted data objects from x or y tasks – there must exist a key (with the same name) for all inputs that are selected as *encrypted* / *private*.

Restrictions related to types of inputs and outputs are covered in Table 4 and under paragraphs of each related stereotype (see Appendix II).

4.1.2.4 Type IV: Requirements of parameters

Semantics:

Intuitively, to use a technology, it might require that its parameters must have a value or / and this value must be in a fixed range to adjust the behaviour of the technology and to have a meaning. For example, if a technology requires an input parameter to be an integer that matches the number of inputs, then it is logical that the value of this parameter should be not negative. For more detailed example, SSSharing stereotype has restrictions that include requirements for parameters – threshold (t) and number of computation parties (c) – $1 < t \leq n$ and $t \leq c \leq n$, (n is the number of outputs). See Figure 15 for an example (the correct value for the threshold would be two). Requirements for parameters are stereotype–specific and not generalizable for too many stereotypes.



SSSharing options

Input data

- Data 2

Output data objects

- Data 2.2
- Data 2.1

Treshold

1

Computation parties

2

Save Remove

Errors in model

1. **SSSharing error: treshold must be an integer greater than 1 and equal to or less than the number of outputs (1 < treshold <= number of outputs)**

Figure 15. SSSharing with wrong threshold parameter value.

Technical solutions:

There is a check to make sure that a parameter value matches with the real model – it means that if the parameter value represents a number of inputs and / or outputs then the check verifies that the number of input and / or output object elements on the model matches this parameter value.

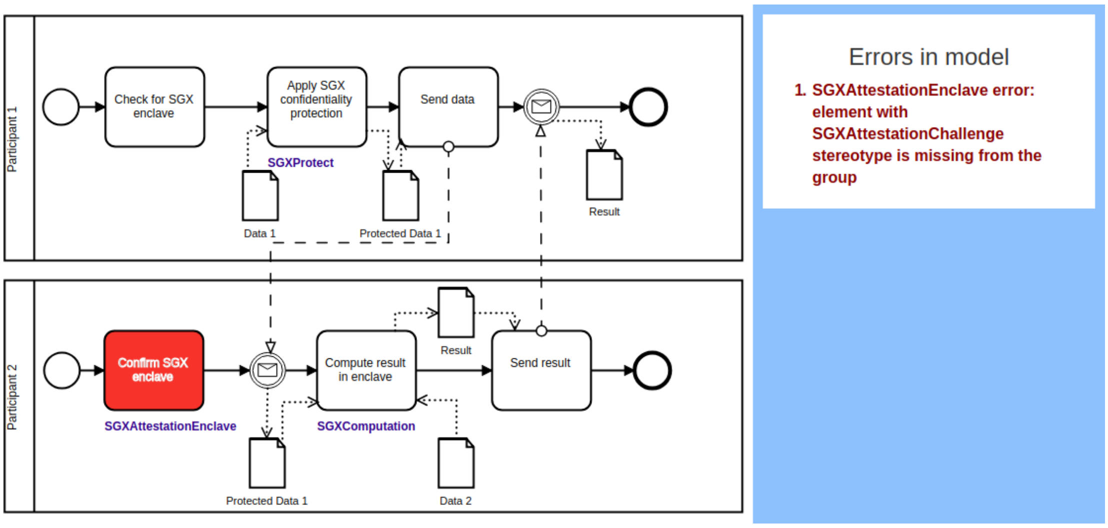
In general, these checks examine if user has inserted a (numeric) value to a parameter input and / or if the (numeric) value that user has inserted is in the required range or fits a certain criteria.

Possible error messages and reasons behind them:

- **error:** number of input and output pairs does not match with the number of x saved (some inputs or outputs might have been removed after adding stereotype); **reason:** number of inputs or outputs defined (by ID) in the parameter must match with the number of inputs and outputs that are actually present on a model;
- **error and reason:** x must be an integer *greater than / equal to* y and *less than / equal to* z ($y \leq x \leq z$);
- **error and reason:** x values cannot be empty.

Restrictions related to parameters are covered under paragraphs of each related stereotype (see Appendix II).

4.1.2.5 Type V: Group members



Errors in model

1. SGXAttestationEnclave error: element with SGXAttestationChallenge stereotype is missing from the group

Figure 16. SGXAttestationEnclave task missing the other required group member with SGXAttestationChallenge stereotype.

Semantics:

There are privacy-enhancing technologies that require to run computations in parallel or in multiple parts. These (parallel) computation parts are represented by a group of tasks. For example, MPC tasks are grouped together. Another example would be an Intel SGX technology, that has different grouping semantics where group tasks are all carried out in a single enclave. In particular, tasks with SGXAttestationChallenge stereotype must come in pairs with a task with SGXAttestationEnclave stereotype (see example in Figure 16), while this SGXAttestationEnclave stereotype task can also be in a group with multiple SGXComputation and SGXProtect tasks. These requirements for the stereotypes of group members extend to all tasks that form groups.

Technical solutions:

Firstly, in case the stereotype requires that there is a certain number of members in the same stereotype group, there is check that looks through JavaScript objects of all task / data object elements on the model (these JavaScript objects are created when model is loaded into the editor) and takes into account all tasks / data objects that have property with the name of the stereotype. All these tasks / data objects are divided into groups based on the value of group ID property of element's JavaScript object and counted. The number of members in the same group of the stereotype is compared to the requirements.

Additionally, in case the stereotype requires that there are members with certain stereotypes represented, there is a check that looks through JavaScript objects of all task elements on the model and takes into account all tasks that have property with the name of the required group stereotype. All these tasks are divided into groups based on the value of group ID property of task's JavaScripts object and if there exists a task with matching group ID. For example, in case of SGXAttestationChallenge stereotype task, there is a check to look for a task with SGXAttestationEnclave stereotype – these two stereotypes must be always used in pairs.

Possible error messages and reasons behind them:

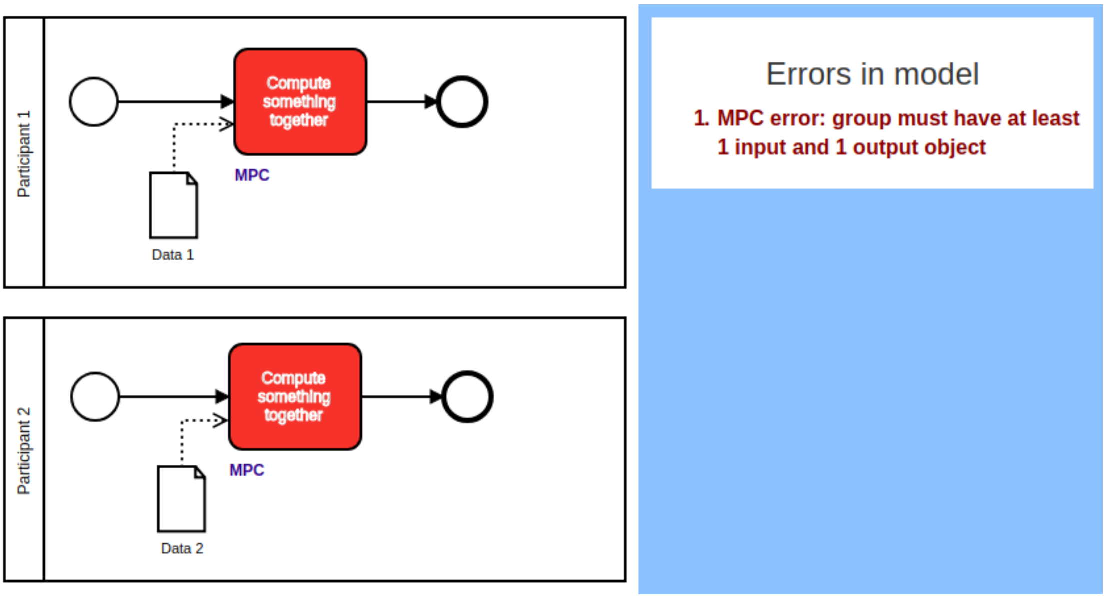
- **error:** group must have *at least* / *exactly* x members;
  reason: there must be *at least* / *exactly* x tasks in the same stereotype group;
- **error:** element with x stereotype is missing from the group;
  reason: there must be *at least* / *exactly* one x and one y task in a stereotype group.

Restrictions related to numbers and types of group members are covered in Table 5 and under paragraphs of each related stereotype (see Appendix II).

4.1.2.6 Type VI: Number of input and output objects per group

Semantics:
In terms of multi-party computation technology stereotypes that form groups, it is relevant to expect that numbers of inputs and outputs are fixed. Firstly, it is obvious that to compute something, there must be an input for at least one task from the stereotype group. Consequently, if there is an input that is the basis of computations, it should produce a result that is an output of the stereotype group. Also, if there are for example three computation parties, it should be possible to fix how many shares of these parties can get as inputs and in which form (in how many parts) is the result produced. In current implementation, groups of MPC stereotype tasks are expected to have at least one input and one output object per group (see Figure 17). In general, these restrictions are typical for multi-party computation stereotypes (that form groups).

Technical solutions:



Errors in model

1. MPC error: group must have at least 1 input and 1 output object

Figure 17. MPC with missing group output.

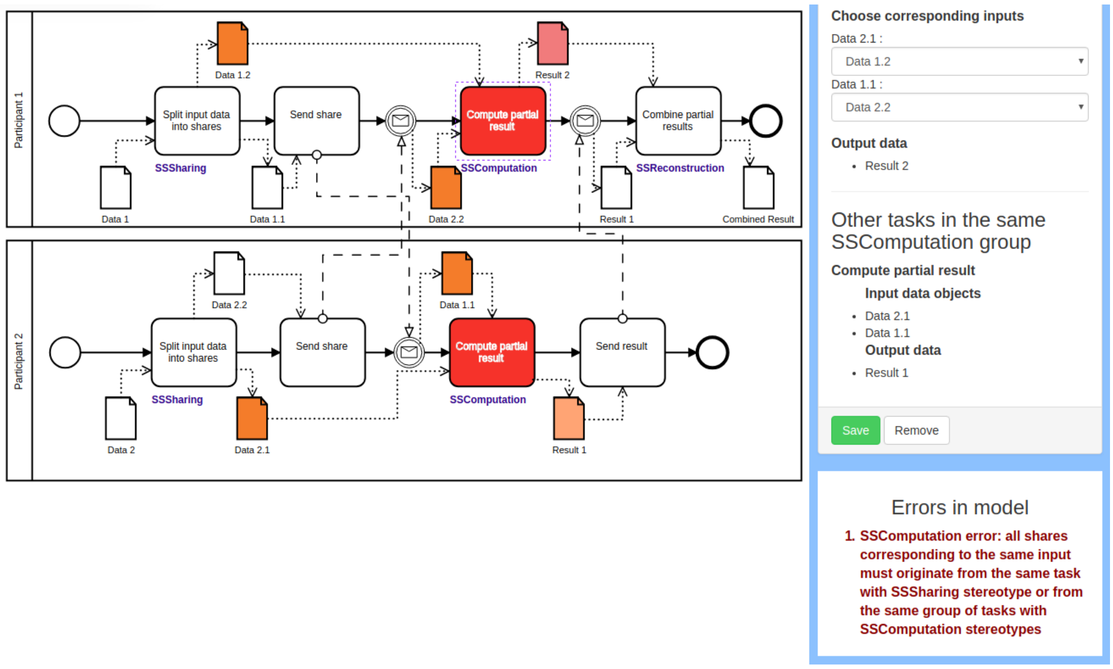
Stereotypes that require that there is a certain number of inputs and outputs per stereotype group are covered by checks that use the same approach as checks described in Section 4.1.2.5 to gather information about other members of the same stereotype group. To get the number of inputs and outputs of stereotype group members, logic described in Section 4.1.2.1 is used. Numbers of inputs and outputs for each group member are summed and compared to the requirements.

Possible error messages and reasons behind them:

- **error and reason:** group must have *at least* / *exactly* x input and *at least* / *exactly* y output objects;
- **error and reason:** each group task must have the same number of inputs and outputs.

Restrictions related to numbers of inputs and outputs per group are covered in Table 5 and under paragraphs of each related stereotype (see Appendix II).

4.1.2.7 Type VII: Relations between inputs and outputs of group members



Choose corresponding inputs

Data 2.1 : Data 1.2

Data 1.1 : Data 2.2

Output data

- Result 2

Other tasks in the same SSComputation group

Compute partial result

Input data objects

- Data 2.1
- Data 1.1

Output data

- Result 1

Errors in model

1. SSComputation error: all shares corresponding to the same input must originate from the same task with SSSharing stereotype or from the same group of tasks with SSComputation stereotypes

Figure 18. SSComputation with function shares from different SSSharing tasks.

Semantics:

Considering stereotypes that form groups, for example computation stereotypes, in addition to the requirements for numbers of inputs and outputs, it is also important that the input data is produced or composed by acceptable technology. Supposing there is a specific technology used to produce shares from a secret. As shares produced by a specific technology have explicit characteristics, it is clear that a random reconstruction technology cannot be used to restore the share. For example, in case SSSharing is used to split the secret into shares, FunSSReconstruction or AddSSReconstruction technologies cannot be used, but SSReconstruction is required to restore the secret. Also, to run computations on these shares, applicable computation technology is required (in this example, SSComputation). In addition to the requirement of acceptable technology, it is also essential that all these input shares are produced by the same stereotype task or group of tasks, as it is logical that shares produced by one task or group of tasks should be reconnectable. On the contrary, for mixed shares that are produced by multiple tasks or task groups, it should not be possible to reconstruct any secrets from them (see Figure 18, where SSComputation shares corresponding to one input originate from two different SSSharing tasks). These restrictions are typical for reconstruction and computations stereotypes that are used in groups.

Technical solutions:

Firstly, in case the stereotype requires that all input objects are of the same origin – outputs of the same task or tasks group – there is a check, that looks through JavaScript objects of tasks from incoming path. Only tasks with JavaScript objects that have a property with applicable stereotype names are considered. In case an acceptable task is found, the list of its output elements is compared to the list of inputs of the task that has a stereotype with currently considered restriction. In case an acceptable task is found, but the stereotype attached to it forms groups, instead of task's outputs, all outputs of the stereotype group are considered. To find output elements of a stereotype group, an approach described in Section 4.1.2.6 is used. For both cases, all inputs must be included in the list of these outputs.

Secondly, in case the stereotype requires that all input objects corresponding to the same input are of the same origin – outputs of the same task or tasks group – there is a check, that uses a similar approach that is used in previously described check. In previous case, all group input elements were considered as a group of elements, but in this case, there is an extra layer to it – all input elements of group members are combined into sub-groups. For example, if there are six inputs per group, these inputs could be separated into two or three sub-groups, instead of using one group. Previously described check is used on each of these sub-groups, not on all inputs together.

Finally, cases where stereotype requires that all shares or inputs corresponding to the same input must be different or same, follow the same logic as requirements concerning the difference and similarity of objects described in Section 4.1.2.2.

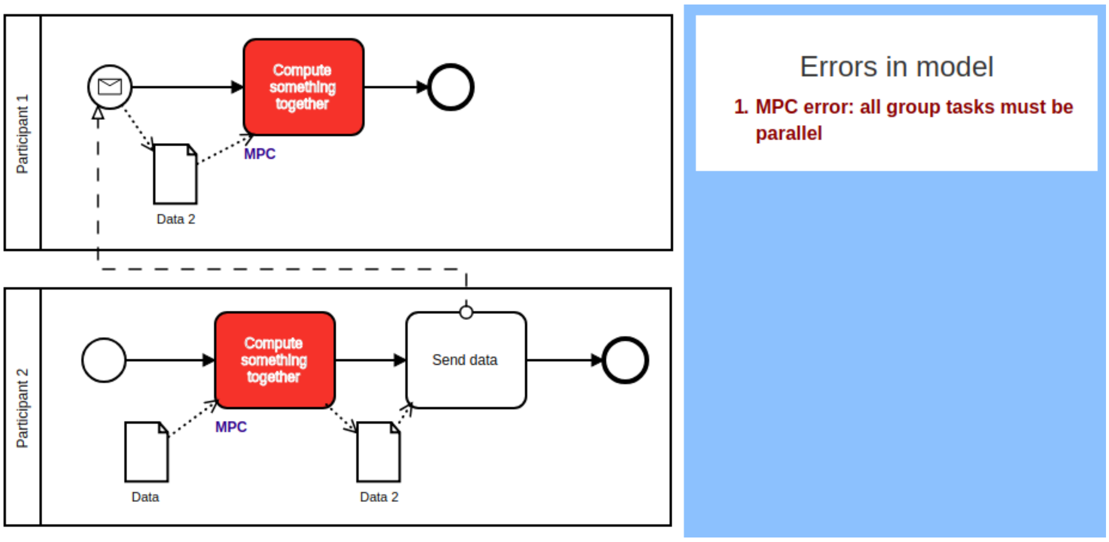
Possible error messages and reasons behind them:

- **error:** *both* / *all* input function shares must originate from the same task with x stereotype *or from the same group of tasks with y stereotypes*;
  reason: all input data objects must be outputs of x task or y task group and they must be all from the same stereotype group;
- **error:** all shares corresponding to the same input must originate from the same task with x stereotype or from the same group of tasks with y stereotypes;
  reason: all input data objects corresponding to the same input must be outputs of x tasks or y task groups and they must be all from the same stereotype group;
- **error:** all shares corresponding to the same input must be different;
  reason: all names of input data objects corresponding to the same input must be different;

- **error:** x objects must be same for both group members;
  reason: both stereotype group tasks must have the same x data object – x data object must be with the same name for both tasks.

Restrictions concerning relations between inputs and outputs of group members are covered in Table 4 and under paragraphs of each related stereotype (see Appendix II).

4.1.2.8 Type VIII: Parallelism of group members



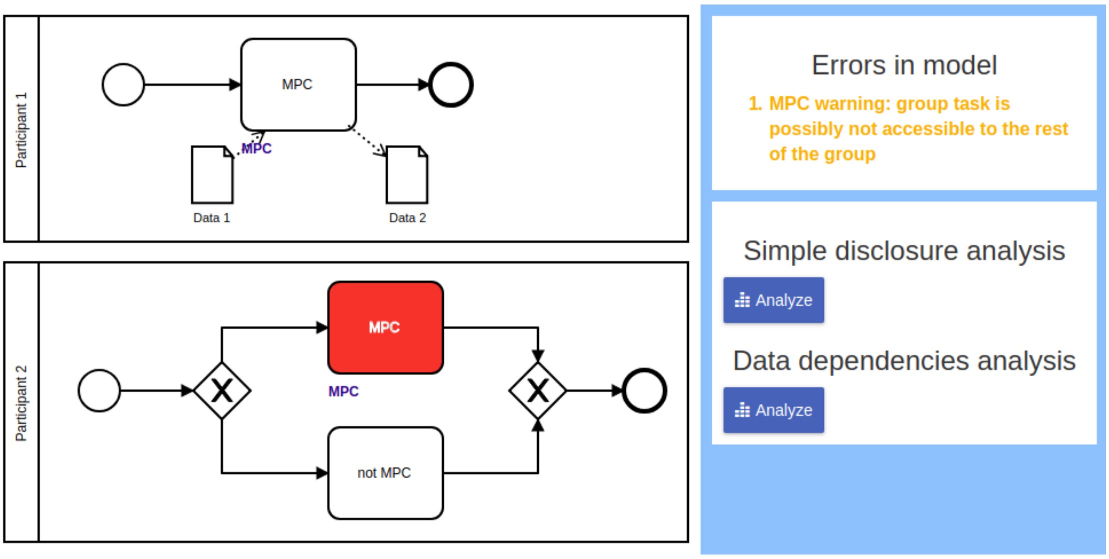
Errors in model

1. MPC error: all group tasks must be parallel

Figure 19. MPC with group members in separate pools, but directly connected.

Semantics:

There are computation technologies, for example secure multi-party computation, that require that their computation is run in parallel. In practice, some of these technologies are designed to be run on CPUs with many cores or on GPUs to reduce the calculation time. In terms of multi-party computation, running computations in parallel means that computation tasks that multiple parties have, must be executed jointly (in parallel). On a model, this is expressed by dividing one computation into multiple computation tasks (sub-tasks of a joint task) onto separate parallel lanes or pools. These sub-tasks represent members of the stereotype group. There is also a requirement that members of the group have no connection between each other. By connection, it is meant that one task from the stereotype group is not in the incoming or outgoing path of another task from the same stereotype group. For example, MPC stereotype requires that all group members are located onto separate lanes or pools and that there is no connection between them (see Figure 19).



Errors in model

1. MPC warning: group task is possibly not accessible to the rest of the group

Figure 20. MPC with group members possibly not reachable to each other.

Also, if there are Exclusive Gateway elements used on the model, a process flow is split into multiple paths and exactly one path in the flow must be taken. When one of the stereotype group members is located in one of these paths, it might happen that based on the context of the model, the path is never chosen and the group member is not reachable for other members of the same stereotype group. In PE-BPMN editor, there is no implementation to set conditions for these gateways yet. It is currently not possible to tell which path from the gateway is actually chosen so all outgoing paths are covered. In these situations, the PE-BPMN editor does not give an error, but instead warns the user that there is a possibility that one of the members of the stereotype group could be not accessible to other members of the same stereotype group (see an example in Figure 20).

Additionally, if there are multiple computations used on a model and multiple stereotype groups are positioned on same lanes or pools, ordering of these groups' representatives on each lane or pool is also important, otherwise some stereotype group tasks might be possibly not accessible to the rest of the group. These restrictions are typical for most of the computation stereotypes that form groups.

In contrast, some technologies require that there is no parallelism between group members and all group members are on the same lane, for example SGXComputation.

It is also possible, that group members must be located onto separate lanes, but there must be a connection between them, so the order of their location can be determined – element with GCGarble stereotype in a group must be before the element with GCEvaluate stereotype.

Technical solutions:

Firstly, in case the stereotype requires that all members (tasks) of the same stereotype group are on the same or separate lane, there is a check, that finds an ID of each task's lane and compares them. As described in Section 4.1.2.2, Bpmn-js provides a method to get information about model elements by their IDs. This information includes also the information about element's parent elements (task is on a lane, so the lane is a parent element of the task), in this case, it allows to find lane or pool ID for each task just by running the method for each group member.

Secondly, in case the stereotype requires that one of the stereotype group members must be before another group member (in terms of sequence flow) or there is no connection between group members at all (they are parallel), there are checks that check for each group member (task) of the stereotype group if it is in incoming or outgoing path of another group member. In case one group task is in the incoming path of another group task, but not in the outgoing path of it, the one task is located before the other task. In case one group task is not in the incoming or outgoing path of another group task and that for all group members, all group members are parallel.

Also, in case members of the stereotype group are considered to be parallel based on previous checks, there are checks to ensure that these group tasks are also accessible to each other.

In case the stereotype requires that all group members (tasks) are parallel and Exclusive Gateway elements are used on the model, there is a check that checks if all tasks of the stereotype group are accessible to the rest of the group. The check finds a Start Event element from the incoming path of a task and finds all possible outgoing paths starting from that start event. All these paths are compared to each other. If there is the same Exclusive Gateway element (by ID) in both compared paths, the path is split from the exclusive gateway. The task must be in both of these paths starting from the exclusive gateway. This check is run on all group members and if it does not fail, it can be assumed that no matter which paths from exclusive gateways are chosen, all group tasks are accessible to all other group members. In case the check fails, a warning is returned.

In order to run the previously described check, there is a check to see if there is a Start Event element in the incoming path of the stereotype task. This check returns also a warning.

In addition, there is a check to make sure that if there are multiple stereotypes used that require their computations to be run in parallel, all tasks from these different stereotype groups are in the same order on each lane. The check gathers group IDs of these stereotype groups and creates lists of these IDs for each lane or pool, later, all lists are compared. As there could be possibly situations where orders of group IDs in these lists differ, but stereotype group could still run in parallel, the check does not return an error, but warning instead.

Possible error messages and reasons behind them:

- **error:** *both* / *all* group tasks must be on *the same* / *separate* lane;
  reason: *both* / *all* group tasks from the same stereotype group must be on *the same* / *different* lanes;
- **error:** element with x stereotype in this group must be before element with y stereotype;
  reason: task from the stereotype group must be in the incoming path, but not in the outgoing path of y task from the same stereotype group;
- **error:** *both* / *all* group tasks must be parallel;
  reason: any task from the stereotype group must not be in the incoming or outgoing path of another task from the same stereotype group;
- **warning:** Start Event element is missing;
  reason: there must be at least one Start Event element (in the incoming path of the task with related stereotype) on the model;
- **warning:** group task is possibly not accessible to the rest of the group;
  reason: all tasks from the same stereotype group that have common Exclusive Gateway elements in their paths (all tasks that have the same exclusive gateway in their paths) must exist in all paths that start from these common exclusive gateways;
- **warning:** all group tasks are possibly not parallel;
  reason: all tasks from different stereotype groups must be in the same order on each lane – ordered lists of stereotype groups must match for each lane.

Restrictions related to parallelism of group members are covered in Table 5 and under paragraphs of each related stereotype (see Appendix II).

There are also some restrictions that are specific to only one stereotype. All stereotype-specific restrictions are covered under paragraphs of each related stereotype (see Appendix II).

4.2 User interface

4.2.1 Validating models

In addition to one of the PE-BPMN editor's main functionalities of adding stereotypes to model elements, there is an important operation of validating syntactical correctness of these models with stereotypes. In order to start the validation of a model, the *Validate* button (see Figure 21) on top of the PE-BPMN editor's page should be pressed.

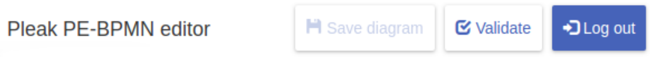


Figure 21. *Validate* button is located next to the *Save* button in PE-BPMN editor. *Save* button is inactive, because there are no new unsaved changes on the model.

Pressing the *Validate* button starts the validation process described in Section 4.1.1 and if the process has finished, the validation report (Section 4.2.2) is created.

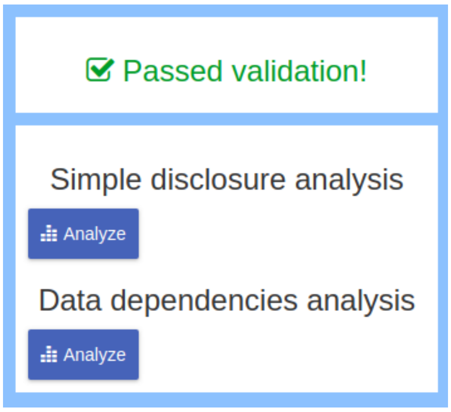
In general, validation can be run at any time as wished – whether there are any new changes on the model or not.

4.2.2 Validation report

Validation report consists of the list of errors and warnings, accordingly coloured red and yellow. In case there are any errors or warnings, list (entitled Errors in model) appears. Clicking on one of the errors in the list highlights concerned model elements – tasks are coloured red and data objects orange. See Figures 12, 13, 14, 15, 16, 17, 18, 19 and 20 for examples. All possible error and warning messages are covered under paragraphs of each stereotype. If there are no errors, message *Passed validation!* appears (see Figure 22). To run analysis on PE-BPMN model, no stereotype errors is a requirement. Warnings (see Figure 20) do not stop running analysis, but could mean that analysis results are not fully correct or are influenced by the reason behind these warnings.

4.2.3 Analysis on a validated model

Currently, there are two analysis implemented in PE-BPMN editor – simple disclosure analysis and data dependencies analysis. These analysis can be run on validated models



☑ Passed validation!

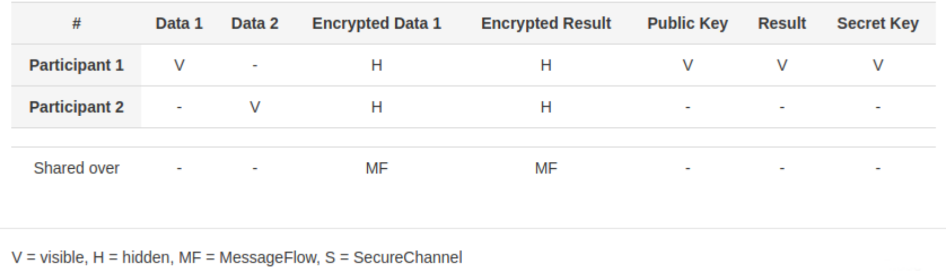
Simple disclosure analysis

Data dependencies analysis

Figure 22. There are no errors found on the model so the *Validation passed!* message is shown and analysis can be run.

that have no errors (warnings are allowed). Buttons to run the analysis appear under the *Passed validation!* message (see Figure 22).

As the development of these analysis is not part of this thesis, they are described only briefly. The analysis have been described in the [PTMT18] paper as follows in this section.

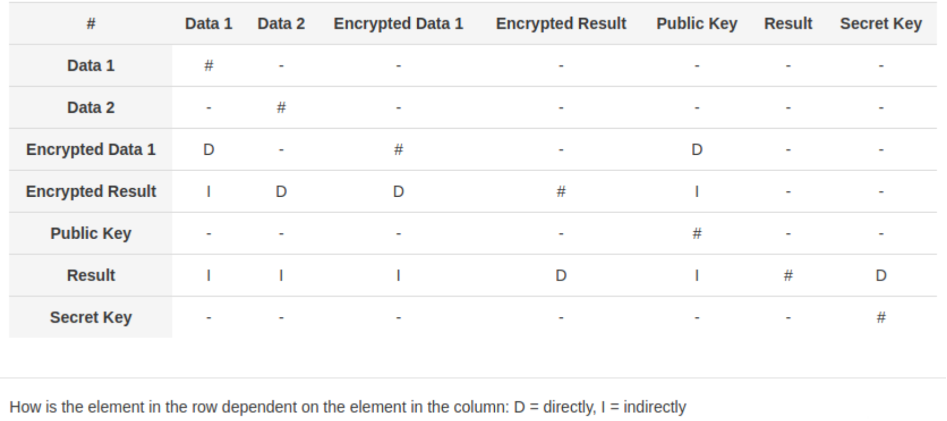


#	Data 1	Data 2	Encrypted Data 1	Encrypted Result	Public Key	Result	Secret Key
Participant 1	V	-	H	H	V	V	V
Participant 2	-	V	H	H	-	-	-
Shared over	-	-	MF	MF	-	-	-

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

Figure 23. The result of the simple disclosure analysis on the model in Figure 3.

Simple disclosure analysis gives an overview of which data objects are seen by which participants in the process. It describes for each data object if contents of the data object are available to the data holder (marked as visible (V)) or they are protected with some PET (marked as hidden (H)). In addition, it lists which data objects are sent over the network and whether they are sent over a public channel (marked as a message flow (MF)) or a secure network channel (marked as S). See Figure 23 for an example of the result of the analysis.



#	Data 1	Data 2	Encrypted Data 1	Encrypted Result	Public Key	Result	Secret Key
Data 1	#	-	-	-	-	-	-
Data 2	-	#	-	-	-	-	-
Encrypted Data 1	D	-	#	-	D	-	-
Encrypted Result	I	D	D	#	I	-	-
Public Key	-	-	-	-	#	-	-
Result	I	I	I	D	I	#	D
Secret Key	-	-	-	-	-	-	#

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

Figure 24. The result of the data dependencies analysis on the model in Figure 3.

Data dependencies analysis summarizes the data dependency graph in a matrix form (see Figure 24). Letter D in the cell means that the data object in the column depends on the data object in the row directly. Letter I means that data objects in the cell and row are connected indirectly. For example, in the Figure 24 Data 1 is directly connected to the Encrypted data 1, but to the Encrypted Result, Data 1 is connected indirectly. Also, Data 1 is not connected to the Secret Key – this is marked with symbol $-$.

The combination of these matrices gives us an overview of whether any data objects are at risk of being leaked.

4.3 Discussion

Previous section described the concept and solutions of the ways how this work achieves the goal of providing support to the user of the PE-BPMN editor tool. The concept is described through the semantics of different components and layers of validation process. Solutions are explained by describing the logic behind the user interface and by providing guidance how to use it. In general, the validation process is necessary to explain to the user the requirements of different privacy enhancing technologies, also, it helps to detect syntactical errors in the model.

Validation process covers two main cases of restrictions and requirements checks. Firstly, each technology has its own requirements to use it, so the tool lets the user know if any of the requirements are not met while adding the representation of the technology (stereotype) on the model. Secondly, if privacy enhancing technologies are already used on the model, but the model has been changed and all the requirements of these technologies are not anymore met, the tool lets the user know what needs to be changed to make the model correct again.

In addition to being a supportive measure for the user, validation process is also required to achieve the essential goal of running analysis on PE-BPMN models. To run analysis and detect privacy leakages in a PE-BPMN model, the model needs to be correct from the perspective of the used technologies. The correct usage of all implemented technologies (stereotypes) can be verified with correctness checks, so the prerequisites for further analysis are covered.

There are two analysis implemented in the PE-BPMN editor. Current analysis can already provide some information about the movement of private information between different participants, so it gives the users of the tool already some insight how to improve the processes that they have modelled. Analysis could help to improve already existing processes or help to plan more secure new processes. These analysis have been also briefly described in the previous section.

5 Conclusions and future work

The main goal of this thesis was to implement a prototype that includes a modelling tool for PE-BPMN and also validators to check the syntactical correctness of PE-BPMN models. As a result of this thesis, the author of this work has implemented the PE-BPMN editor tool that supports thirty-six different stereotypes. All these stereotypes are covered with validators to find and to help the user to fix syntactical mistakes in PE-BPMN models. Additionally, the wiki to describe PLEAK tools, including PE-BPMN has been set up. Finally, one of the goals was and is to run different analysis on PE-BPMN models. Currently, two analysis have been implemented, which means, at least in terms of these analysis and currently implemented stereotypes, that the goal of providing support to find and correct syntactical errors in PE-BPMN models, has been achieved. The PE-BPMN editor has been connected to PLEAK and is live and running in `https://pleak.io`. Concerning the usage and users of the PE-BPMN editor, there are already analysts who use it to model real-life processes in order to reduce the risk of privacy leakages in these processes. Registration to PLEAK is currently available for invited users only, but as the project is open-sourced, the code of PE-BPMN is available for download at `https://github.com/pleak-tools/pleak-pe-bpmn-editor`.

The development of PE-BPM editor is an on-going process, which means that more stereotypes will be added in order to support PE-BPMN notation more precisely. Also, better support of BPMN syntax is a future goal. Details concerning known limitations and future work are described in the next section.

5.1 Future work

In the current implementation of PE-BPMN editor, there are four main known limitations.

Firstly, in PE-BPMN, it is expected that the whole process is described in one diagram. The stereotypes only consider a process that starts from public data objects and are then protected or computed upon as necessary. Hence, the editor is currently unable to consider fragments of processes that already start from some protected data. It is future work to consider intermediate models where the inputs may already be protected. This requires introducing data stereotypes that correspond to the current protection type task stereotypes.

Secondly, syntactic verification supports only limited combinations of stereotypes. Mainly, it is expected that each protection stereotype takes unprotected data as an input, which means that it is not possible to correctly treat cases where, for example, a ciphertext is given as an input to secret sharing and then later restored and decrypted. Currently the secret sharing reconstruction would be verified, but verification of decryption would fail. Additionally, there are also not considered cases where only part of the data object is encrypted, for example an encrypted column in an otherwise unprotected database.

Thirdly, there is a limitation in support for BPMN syntax. One of the unsupported BPMN elements is a Subprocess, that is necessary to consider more practical models than the ones currently supported. In general, adding each new BPMN element requires an individual approach, in terms of subprocesses, a link to connect data objects inside and outside of the process is required.

Fourthly, the PE-BPMN editor has some challenges in checking parallelism of grouped stereotypes. There are two problematic situations. Firstly, in some cases there is an extra condition (exclusive gateway) in the process flow to determine whether a task from the stereotype group is going to be executed or not. There is currently no implementation for checking these semantic conditions, so it is not possible to be sure whether the task is accessible and can be run in parallel at all. Secondly, in some cases multiple parallel tasks are combined sequentially, but their running order may differ. For instance, in some cases the previous task has to be finished while in other cases the following task can already start while the first one is still running. As there are no real running processes under these stereotypes yet, it is currently not possible to be sure whether required computations have been completed and these tasks can run in parallel when necessary. The first situation could be addressed by implementing setting and checking conditions of gateway elements so it would be know for correctness checks of which paths would be chosen based on gateway conditions. The second issue could be solved by translating a model into a Petri net and then using the tools of Petri net analysis in order to identify problems, such as race conditions.

References

[AGJS13] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative Technology for CPU Based Attestation and Sealing. In *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy, volume 13*. ACM New York, NY, USA, 2013.

[Bla79] George Robert Blakley. Safeguarding Cryptographic Keys. In *Proceedings of the 1979 AFIPS National Computer Conference, pages 313-317*. AFIPS Press, 1979.

[DDFMH$^+$15] G. Danezis, J. Domingo-Ferrer, J.-H. Hoepman M. Hansen, D. L. Metayer, R. Tirtea, and S. Schiffner. Privacy and Data Protection by Design-from Policy to Engineering. Technical report, European Union Agency for Network and Information Security, 2015.

[EPG08] EPG. Privacy by Design - An Overview of Privacy Enhancing Technologies, prepared by Enterprise Privacy Group on behalf of the UK Information Commissionerś Office. URL: `http://www.dsp.utoronto.ca/projects/surveillance/docs/pbd_pets_paper.pdf`, November 2008.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.

[JSR$^+$16] Simon Johnson, Vinnie Scarlata, Carlos Rozas, Ernie Brickell, and Frank Mckeen. Intel® Software Guard Extensions: EPID Provisioning and Attestation Services. *White Paper*, 1:1–10, 2016.

[Men97] Alfred J Menezes. Handbook of Applied Cryptography / Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, 1997. chapter 8.

[OMG10] OMG. BPMN 2.0 by Example: non-normative OMG document with BPMN 2.0 examples. URL: `https://www.omg.org/cgi-bin/doc?dtc/10-06-02.pdf`, June 2010.

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

[PET07] European Union. Press release: Privacy Enhancing Technologies (PETs). URL: `http://europa.eu/rapid/press-release_MEMO-07-159_en.htm`, May 2007.

[PMB17] Pille Pullonen, Raimundas Matulevičius, and Dan Bogdanov. PE-BPMN: Privacy-Enhanced Business Process Model and Notation. In *In Proceedings of the 15th International Conference on Business Process Management (BPM)*, pages 40–56. Springer, 2017.

[PTMT18] Pille Pullonen, Jake Tom, Raimundas Matulevičius, and Aivo Toots. Privacy-Enhanced BPMN: A Multi-Level Approach to Information Disclosure Analysis, 2018. submitted for publication.

[RLP16] Marcello La Rosa, Peter Loos, and Oscar Pastor. *Business Process Management*. Springer, 2016.

[Sha79] Adi Shamir. How to Share a Secret. Communications of the ACM, 22(11):612-613, November 1979.

[Yao82] Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium*, pages 160–164. IEEE, 1982.

Appendix

I. Tables of stereotype restrictions and requirements

Table 3. Restrictions and parameters of the implemented Task stereotypes.

Stereotype	Inputs	Outputs	Other parameters
ProtectConfidentiality	1: data	1: protected data	
SGXProtect	1: data	1: enclave data	group with SGXComputation
PKEncrypt	2: public key, data	1: ciphertext	
SKEncrypt	2: secret key, data	1: ciphertext	
SSSharing	1: data	$2 - \ldots$: shares	number of shares, threshold; $(t \leq c \leq n)$; $(1 < t \leq n)$
AddSSSharing	1: data	$2 - \ldots$: additive shares	
FunSSSharing	1: function	2: function shares	
OpenConfidentiality	1: protected data	1: data	
PKDecrypt	2: private key, ciphertext	1: data	
SKDecrypt	2: secret key, ciphertext	1: data	
SSReconstruction	$2 - \ldots$: shares	1: secret data	
AddSSReconstruction	$2 - \ldots$: additive shares	1: secret data	
FunSSReconstruction	2: function shares	1: secret function	
PETComputation	$1 - \ldots$: protected data, data	1: protected data or data	script
SGXComputation	$1 - \ldots$: enclave data, data	$1 - \ldots$: enclave data or data	script, group with SGXComputation, group with SGXProtect, group with SGXAttestationEnclave
PKComputation	$1 - \ldots$: ciphertexts, data	1 : ciphertext	script

SKComputation	$1 - \ldots$: ciphertexts, data	1 : ciphertexts	script
MPC	$0 - \ldots$: data	$0 - 1$: data	script, group
SSComputation	$1 - \ldots$: shares, data	1: share	script, group
FunSSComputation	2: function share, evaluation point	1: additive share	group
AddSSComputation	$1 - \ldots$: additive shares, data	1: additive share	script, group
GCGarble	0:	2: garbled circuit, input encodings	script, grouped with GCEvaluate
GCEvaluate	2: garbled circuit, input encodings	1: computation output	grouped with GCGarble
OTSend	1: input data	0:	grouped with OTReceive
OTReceive	1: query	1: input	grouped with OTSend
DimensionalityReduction	2: projection matrix, data	1: feature vector	
DifferentialPrivacy	$1 - \ldots$: data	1 : data	script or (ε, δ) for each input-output pair; ε values cannot be empty
SGXAttestationEnclave	0-...: enclave measurement	0:	group with SGXAttestationChallenge, group with SGXComputation
SGXAttestationChallenge	0-...: challenge	1: attestation outcome	group with SGXAttestationEnclave
SGXQuoting	2: challenge, measurement	1: quote	
SGXQuoteVerification	3: quote, certificate, revocation list	1: verification outcome	

Table 4. Conditions for allowed opening and computation stereotypes

Stereotype	Success conditions
OpenConfidentiality	input comes from ProtectConfidentiality or PETComputation
PKDecrypt	ciphertext input comes from PKEncrypt or PKComputation, uses the PKPrivate corresponding to the PKPublic used by the input
SKDecrypt	ciphertext input comes from PKEncrypt or PKComputation, uses the same secret key as the input
SSReconstruction	all input shares correspond to the same secret shared value (outputs of one SSSharing or one group of SSComputation), at least *threshold* shares available
AddSSReconstruction	all input shares correspond to the same secret shared value (outputs of one AddSSSharing or one group of AddSSComputation or FunSSComputation), all shares available
FunSSReconstruction	all input shares correspond to the same secret shared value (outputs of one FunSSSharing), both shares available
SGXComputation	enclave data comes from SGXProtect or as enclave data from SGXComputation in the same group
PKComputation	ciphertext inputs come from PKEncrypt or PKComputation, all ciphertext inputs correspond to the same PKPublic public key, the output corresponds to the same key
SKComputation	ciphertext inputs come from SKEncrypt or SKComputation, all ciphertext inputs correspond to the same secret key, the output corresponds to the same key
SSComputation	share inputs come from SSSharing or SSComputation tasks, all shared inputs have the same threshold, the output has the same threshold as the shared inputs
FunSSComputation	function share input from FunSSSharing, output is two party additive secret shared value
AddSSComputation	additive share inputs from AddSSSharing or AddSSComputation tasks, all additive share inputs have the same number of shares, output is additive secret sharing for the same number of shares as the inputs

Table 5. Restrictions for stereotype groups

Stereotype	# tasks	Restrictions
MPC	$2 - \ldots$	Parallel tasks, at least one input per group, at least one output per group
SSComputation	$2 - \ldots$	Parallel tasks, at least one shared input, all tasks have the same number of inputs and outputs, the number of tasks is the number of shares in each shared input values, the same data inputs for all tasks, each task has one distinct share input for each shared input
FunSSComputation	2	Parallel tasks, one shared input, the same evaluation point input for both tasks
AddSSComputation	$2 - \ldots$	Parallel tasks, at least one shared input, all tasks have the same number of inputs and outputs, the number of tasks is the number of shares in each shared input values, the same data inputs for all tasks, each task has one distinct share input for each shared input
SGXComputation, SGXProtect, SGXAttestationEnclave	$1 - \ldots$	All group tasks are on the same lane (same CPU executes them)
SGXAttestationChallenge, SGXAttestationEnclave	2	Parallel tasks, both exist
OTSend, OTReceive	2	Parallel tasks, both exist, may have a message flow from OTSend to OTReceive
GCComputation	2	Parallel tasks, at least one input per group, at least one output per group
GCGarble, GCEvaluate	2	Both exist, on separate lanes, GCGarble garbled circuit output is the garbled circuit input to GCEvaluate, GCGarble input encodings output is used to derive encoded input for GCEvaluate

II. Restrictions and requirements of each stereotype

This section contains the documentation of stereotype restrictions and requirements at the time of submitting the thesis, the most recent documentation can be found from the wiki.

AddSSComputation

Required input objects
1...n data objects (additive shares) from AddSSSharing tasks / AddSSComputation or FunSSComputation task groups

Required input parameters
group (with AddSSComputation tasks)

Optional input parameters
input script

Required output objects
1 data object (data)

Restrictions

1. **Restriction:** task must have at least one input and exactly one output data objects
   Error: at least 1 input and exactly 1 output are required

2. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined

3. **Restriction:** "inputScript" property must exist under stereotype details object – "inputScript" can be empty, but must not have been deleted from model's file
   Error: inputScript is undefined

4. **Restriction:** "inputs" property must exist under stereotype details object – "inputs" information must not have been deleted from model's file
   Error: inputs are undefined

5. **Restriction:** all names of input data objects must be different
   Error: all input function shares must be different

6. **Restriction:** all names of input data objects corresponding to the same input must be different
   Error: all shares corresponding to the same input must be different

Group specific restrictions

1. **Restriction:** all input data objects fixed (by their IDs) in "inputs" property (for all group tasks) must exist also on model
   Error: one or more shares (data objects) corresponding to the same input of the group are missing
2. **Restriction:** all input data objects corresponding to the same input must be outputs of AddSSSharing task / AddSSComputation or FunSSComputation task groups and they must be all from the same stereotype group
   Error: all shares corresponding to the same input must originate from the same task with AddSSSharing stereotype or from the same group of tasks with AddSSComputation or FunSSComputation stereotypes
3. **Restriction:** there must be at least two tasks in the same stereotype group
   Error: group must have at least 2 members
4. **Restriction:** all group tasks from the same stereotype group must be on different lanes
   Error: each group task must be on separate lane
5. **Restriction:** any task from the stereotype group must not be in the incoming or outgoing path of another task from the same stereotype group
   Error: all group tasks must be parallel
6. **Restriction:** all tasks from the stereotype group must have the same number of input and output data objects
   Error: each group task must have the same number of inputs and outputs
7. **Restriction:** all tasks from different stereotype groups must be in the same order on each lane – ordered lists of stereotype groups must match for each lane
   Warning: all group tasks are possibly not parallel
8. **Restriction:** all tasks from the same stereotype group that have common Exclusive Gateway elements in their paths (all tasks have the same exclusive gateway in their paths) must exist in all paths that start from these common exclusive gateways
   Warning: group task is possibly not accessible to the rest of the group
9. **Restriction:** there must be at least one Start Event element on the model (to run parallelism checks)
   Warning: Start Event element is missing

AddSSReconstruction

Required input objects
2...n data objects (additive shares) from AddSSSharing task / AddSSComputation or FunSSComputation task groups

Required output objects
1 data object (secret data)

Restrictions

1. **Restriction:** task must have at least two input and exactly one output data objects
   Error: at least 2 inputs and exactly 1 output are required

2. **Restriction:** all input data objects must be outputs of AddSSSharing task / AddSSComputation or FunSSComputation task groups and they must be all from the same stereotype group
   Error: all input function shares must originate from the same task with AddSSSharing stereotype or from the same group of tasks with AddSSComputation or FunSSComputation stereotypes

3. **Restriction:** all names of input data objects must be different
   Error: all input function shares must be different

AddSSSharing

Required input objects
1 data object (data)

Required output objects
2...n data objects (additive shares)

Restrictions

1. **Restriction:** task must have exactly one input and at least two output data objects
   Error: exactly 1 input and at least 2 outputs are required

2. **Restriction:** all names of output data objects must be different
   Error: output objects must have different names

CommunicationProtection

Required input objects
1...n data object

Required output objects
1...n data objects

Restrictions

1. **Restriction:** message flow must have at least one input-output data objects pair
   Error: at least 1 incoming-outgoing pair of data objects is required

2. **Restriction:** message flow must have the same number of input and output data objects
   Error: number of incoming-outgoing pair of data objects does not match

3. **Restriction:** the list of names of input data objects must match the list of names of output data objects
   Error: names of incoming and outgoing data objects must match

DifferentialPrivacy

Required input objects
1...n data objects (data)

Optional input parameters
computation script or (ε, δ) for each input-output pair

Required output objects
1 data object (data)

Restrictions

1. **Restriction:** task must have at least one input and exactly one output data objects
   Error: at least 1 input and exactly 1 output are required

2. **Restriction:** "inputScript" property must exist under stereotype details object – "inputScript" can be empty, but must not have been deleted from model's file
   Error: inputScript is undefined (it can be empty, but cannot be undefined)

3. **Restriction:** "delta" property must exist under stereotype details object – "delta" can be empty, but must not have been deleted from model's file
   Error: delta is undefined (it can be empty, but cannot be undefined)

4. **Restriction:** "epsilons" property must exist under stereotype details object – "epsilons" can be zeros, but must not have been deleted from model's file
   Error: epsilons are undefined

5. **Restriction:** number of inputs or outputs defined (by ID) in the parameter must match with the number of inputs and outputs that are actually present on a model
   Error: number of input and output pairs does not match with the number of epsilons saved (some inputs or outputs might have been removed after adding stereotype)

6. **Restriction:** epsilon values must not be empty
   Error: epsilon values cannot be empty

DimensionalityReduction

Required input objects
2 data objects (data, projection matrix)

Required output objects
1 data object (feature vector)

Restrictions

1. **Restriction:** task must have exactly two input and one output data objects
   Error: exactly 2 inputs and 1 output are required

2. **Restriction:** "data" property must exist under stereotype details object – "data" must have been selected and saved and "data" information must not have been deleted from model's file
   Error: data is undefined

3. **Restriction:** "projectionMatrix" property must exist under stereotype details object – "projectionMatrix" must have been selected and saved and "projectionMatrix" information must not have been deleted from model's file
   Error: projectionMatrix is undefined

4. **Restriction:** input data object fixed (by its ID) in "data" property must exist also on model
   Error: data object is missing

5. **Restriction:** input projectionMatrix object fixed (by its ID) in "projectionMatrix" property must exist also on model
   Error: projectionMatrix object is missing

6. **Restriction:** data and projectionMatrix objects must be different data objects – they must have different IDs
   Error: data and projectionMatrix must be different objects

FunSSComputation

Required input objects
2 data objects (evaluation point, function share), share from FunSSSharing task

Required input parameters
group (with FunSSComputation tasks)

Required output objects
1 data object (additive share)

Restrictions

1. **Restriction:** task must have exactly two input and one output data objects
   Error: exactly 2 inputs and 1 output are required

2. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined

3. **Restriction:** "evaluationPoint" property must exist under stereotype details object – "evaluationPoint" must have been selected and saved and "challenge" information must not have been deleted from model's file
   Error: evaluationPoint is undefined

4. **Restriction:** "shareOfFunction" property must exist under stereotype details object – "shareOfFunction" must have been selected and saved and "shareOfFunction" information must not have been deleted from model's file
   Error: shareOfFunction is undefined

5. **Restriction:** input data object fixed (by its ID) in "evaluationPoint" property must exist also on model
   Error: evaluationPoint object is missing

6. **Restriction:** input data object fixed (by its ID) in "shareOfFunction" property must exist also on model
Error: shareOfFunction object is missing

7. **Restriction:** evaluation point and function share objects must be different data objects – they must have different IDs
Error: evaluation point and function share must be different objects

Group specific restrictions

1. **Restriction:** there must be exactly two tasks in the same stereotype group
Error: group must have exactly 2 members

2. **Restriction:** both group tasks from the same stereotype group must be on different lanes
Error: both group tasks must be on separate lane

3. **Restriction:** any task from the stereotype group must not be in the incoming or outgoing path of another task from the same stereotype group
Error: group tasks must be parallel

4. **Restriction:** all tasks from different stereotype groups must be in the same order on each lane – ordered lists of stereotype groups must match for each lane
Warning: all group tasks are possibly not parallel

5. **Restriction:** all tasks from the same stereotype group that have common Exclusive Gateway elements in their paths (all tasks have the same exclusive gateway in their paths) must exist in all paths that start from these common exclusive gateways
Warning: group task is possibly not accessible to the rest of the group

6. **Restriction:** there must be at least one Start Event element on the model (to run parallelism checks)
Warning: Start Event element is missing

7. **Restriction:** both input data objects must be outputs of the same FunSSSharing task
Error: both group input function shares must originate from the same task with FunSSSharing stereotype

8. **Restriction:** both input function shares of the stereotype group must be different – they must have different names
Error: both group members must have different input function shares

9. **Restriction:** both stereotype group tasks must have the same input evaluation point data object – evaluation point data object must be with the same name for both tasks
   Error: evaluation point must be the same (with same name) for both group members

FunSSReconstruction

Required input objects
2 data objects (function shares) from FunSSSharing task

Required output objects
1 data object (secret function)

Restrictions

1. **Restriction:** task must have at exactly two input and one output data objects
   Error: exactly 2 inputs and 1 output are required
2. **Restriction:** both input data objects (function shares) must be outputs of the same FunSSSharing task
   Error: both input function shares must originate from the same task with FunSSSharing stereotype
3. **Restriction:** names of input data objects must be different
   Error: input function shares must be different

FunSSSharing

Required input objects
1 data object (function)

Required output objects
2 data objects (function shares)

Restrictions

1. **Restriction:** task must have at exactly one input and two output data objects
   Error: exactly 1 input and 2 outputs are required
2. **Restriction:** names of output data objects must be different
   Error: output objects must be different

GCComputation

Required input parameters
group (with GCComputation tasks)

Optional input parameters
input script

Restrictions

1. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined

2. **Restriction:** "inputScript" property must exist under stereotype details object – "inputScript" can be empty, but must not have been deleted from model's file
   Error: inputScript is undefined

Group specific restrictions

1. **Restriction:** stereotype group must have at least one input and one output data object
   Error: group must have at least 1 input and 1 output object

2. **Restriction:** there must be exactly two tasks in the same stereotype group
   Error: group must have exactly 2 members

3. **Restriction:** all group tasks from the same stereotype group must be on different lanes
   Error: each group task must be on separate lane

4. **Restriction:** any task from the stereotype group must not be in the incoming or outgoing path of another task from the same stereotype group
   Error: all group tasks must be parallel

5. **Restriction:** all tasks from different stereotype groups must be in the same order on each lane – ordered lists of stereotype groups must match for each lane
   Warning: all group tasks are possibly not parallel

6. **Restriction:** all tasks from the same stereotype group that have common Exclusive Gateway elements in their paths (all tasks have the same exclusive gateway in their paths) must exist in all paths that start from these common exclusive gateways
   Warning: group task is possibly not accessible to the rest of the group

7. **Restriction:** there must be at least one Start Event element on the model (to run parallelism checks)
   Warning: Start Event element is missing

GCEvaluate

Required input objects
2 data objects (garbled circuit, input encodings), circuit from GCGarble task

Required input parameters
group (with GCGarble task)

Required output objects
1 data object (computation output)

Restrictions

1. **Restriction:** task must have exactly two input and one output data objects
   Error: exactly 2 inputs and 1 output are required

2. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined

3. **Restriction:** "inputScript" property must exist under stereotype details object and must not have been deleted from model's file
   Error: inputScript is undefined

4. **Restriction:** "garbledCircuit" property must exist under stereotype details object – "garbledCircuit" must have been selected and saved and "garbledCircuit" information must not have been deleted from model's file
   Error: garbledCircuit is undefined

5. **Restriction:** "inputEncoding" property must exist under stereotype details object – "inputEncoding" must have been selected and saved and "inputEncoding" information must not have been deleted from model's file
   Error: inputEncoding is undefined

6. **Restriction:** input data object fixed (by its ID) in "garbledCircuit" property must exist also on model
   Error: garbledCircuit object is missing

7. **Restriction:** input data object fixed (by its ID) in "inputEncoding" property must exist also on model
   Error: inputEncoding object is missing

8. **Restriction:** garbled circuit and input encoding objects must be different data objects – they must have different IDs
   Error: garbledCircuit and inputEncoding must be different objects

Group specific restrictions

1. **Restriction:** there must be exactly one GCEvaluate and one GCGarble task in a stereotype group
   Error: element with GCGarble stereotype is missing from the group

2. **Restriction:** both stereotype group tasks must have the same garbled circuit data object – garbled circuit data object must be with the same name for both tasks
   Error: garbledCircuit objects must be same for both group members

3. **Restriction:** input data object fixed (by its ID) in "garbledCircuit" property must be an output element of a GCGarble task from the same stereotype group
   Error: garbledCircuit object must originate from the element with GCEvaluate stereotype of this group

4. **Restriction:** both group tasks from the same stereotype group must be on different lanes
   Error: both group tasks must be on separate lane

5. **Restriction:** GCGarble task from the stereotype group must be in incoming path, but not in outgoing path of GCEvaluate task from the same stereotype group
   Error: element with GCGarble stereotype in this group must be before element with GCEvaluate stereotype

GCGarble

Required input objects
2 data objects (garbled circuit, input encodings)

Required input parameters
group (with GCEvaluate task)

Optional input parameters
input script

Required output objects
0 data objects

Restrictions

1. **Restriction:** task must have exactly two input and zero output data objects
   Error: exactly 2 outputs and no inputs are required
2. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined
3. **Restriction:** "inputScript" property must exist under stereotype details object – "inputScript" can be empty, but must not have been deleted from model's file
   Error: inputScript is undefined
4. **Restriction:** "garbledCircuit" property must exist under stereotype details object – "garbledCircuit" must have been selected and saved and "garbledCircuit" information must not have been deleted from model's file
   Error: garbledCircuit is undefined
5. **Restriction:** "inputEncoding" property must exist under stereotype details object – "inputEncoding" must have been selected and saved and "inputEncoding" information must not have been deleted from model's file
   Error: inputEncoding is undefined
6. **Restriction:** input data object fixed (by its ID) in "garbledCircuit" property must exist also on model
   Error: garbledCircuit object is missing
7. **Restriction:** input data object fixed (by its ID) in "inputEncoding" property must exist also on model
   Error: inputEncoding object is missing
8. **Restriction:** garbled circuit and input encoding objects must be different data objects – they must have different IDs
   Error: garbledCircuit and inputEncoding must be different objects

Group specific restrictions

1. **Restriction:** there must be exactly one GCEvaluate and one GCGarble task in a stereotype group
   Error: element with GCEvaluate stereotype is missing from the group

2. **Restriction:** both stereotype group tasks must have the same garbled circuit data object – garbled circuit data object must be with the same name for both tasks
   Error: garbledCircuit objects must be same for both group members

3. **Restriction:** both group tasks from the same stereotype group must be on different lanes
   Error: both group tasks must be on separate lane

4. **Restriction:** GCGarble task from the stereotype group must be in incoming path, but not in outgoing path of GCEvaluate task from the same stereotype group
   Error: element with GCGarble stereotype in this group must be before element with GCEvaluate stereotype

MPC

Required input parameters
group (with MPC tasks)

Optional input parameters
input script

Restrictions

1. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined

2. **Restriction:** "inputScript" property must exist under stereotype details object – "inputScript" can be empty, but must not have been deleted from model's file
   Error: inputScript is undefined

Group specific restrictions

1. **Restriction:** stereotype group must have at least one input and one output data object
   Error: group must have at least 1 input and 1 output object

2. **Restriction:** there must be at least two tasks in the same stereotype group
   Error: group must have at least 2 members

3. **Restriction:** all group tasks from the same stereotype group must be on different lanes
   Error: each group task must be on separate lane

4. **Restriction:** any task from the stereotype group must not be in the incoming or outgoing path of another task from the same stereotype group
   Error: all group tasks must be parallel

5. **Restriction:** all tasks from different stereotype groups must be in the same order on each lane – ordered lists of stereotype groups must match for each lane
   Warning: all group tasks are possibly not parallel

6. **Restriction:** all tasks from the same stereotype group that have common Exclusive Gateway elements in their paths (all tasks have the same exclusive gateway in their paths) must exist in all paths that start from these common exclusive gateways
   Warning: group task is possibly not accessible to the rest of the group

7. **Restriction:** there must be at least one Start Event element on the model (to run parallelism checks)
   Warning: Start Event element is missing

OpenConfidentiality

Required input objects

1 data object (protected data) from ProtectConfidentiality or PETComputation task

Required output objects

1 data object (data)

Restrictions

1. **Restriction:** task must have exactly one input and one output data objects
   Error: exactly 1 input and 1 output are required

2. **Restriction:** input and output objects must be different data objects – they must have different IDs
   Error: input and output objects must be different data objects

3. **Restriction:** input data object must be output of ProtectConfidentiality or PETComputation task and if it is an output of PETComputation task then the output data object must be selected as "Private" under PETComputation stereotype settings of the output object's task
   Error: input object is not private

OTReceive

Required input objects
1 data object (query)

Required input parameters
group (with OTSend task)

Required output objects
1 data object (input)

Restrictions

1. **Restriction:** task must have exactly one input and one output data objects
   Error: exactly 1 input and 1 output are required

2. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined

Group specific restrictions

1. **Restriction:** there must be exactly one OTReceive and one OTSend task in a stereotype group
   Error: element with OTSend stereotype is missing from the group

2. **Restriction:** both group tasks from the same stereotype group must be on different lanes
   Error: both group tasks must be on separate lane

3. **Restriction:** tasks of stereotype group must be parallel or OTReceive task of the stereotype group must be directly connected (through message flow) to the OTSend task of the same group
   Error: element with OTReceive stereotype in this group must be directly connected (through message flow) to the element with OTSend stereotype and there must be no other connections between them

4. **Restriction:** OTSend task from the stereotype group must be in incoming path, but not in outgoing path of OTReceive task from the same stereotype group
   Error: element with OTSend stereotype in this group must be before element with OTReceive stereotype

5. **Restriction:** all tasks from different stereotype groups must be in the same order on each lane – ordered lists of stereotype groups must match for each lane
Warning: all group tasks are possibly not parallel

6. **Restriction:** all tasks from the same stereotype group that have common Exclusive Gateway elements in their paths (all tasks have the same exclusive gateway in their paths) must exist in all paths that start from these common exclusive gateways
Warning: group task is possibly not accessible to the rest of the group

7. **Restriction:** there must be at least one Start Event element on the model (to run parallelism checks)
Warning: Start Event element is missing

OTSend

Required input objects
1 data object (input data)

Required input parameters
group (with OTReceive task)

Required output objects
0 data objects

Restrictions

1. **Restriction:** task must have exactly one input and zero output data objects
Error: exactly 1 input and no outputs are required

2. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
Error: groupId is undefined

Group specific restrictions

1. **Restriction:** there must be exactly one OTReceive and one OTSend task in a stereotype group
Error: element with OTReceive stereotype is missing from the group

2. **Restriction:** both group tasks from the same stereotype group must be on different lanes
Error: both group tasks must be on separate lane

3. **Restriction:** tasks of stereotype group must be parallel or OTReceive task of the stereotype group must be directly connected (through message flow) to the OTSend task of the same group
   Error: element with OTReceive stereotype in this group must be directly connected (through message flow) to the element with OTSend stereotype and there must be no other connections between them

4. **Restriction:** OTSend task from the stereotype group must be in incoming path, but not in outgoing path of OTReceive task from the same stereotype group
   Error: element with OTSend stereotype in this group must be before element with OTReceive stereotype

5. **Restriction:** all tasks from different stereotype groups must be in the same order on each lane – ordered lists of stereotype groups must match for each lane
   Warning: all group tasks are possibly not parallel

6. **Restriction:** all tasks from the same stereotype group that have common Exclusive Gateway elements in their paths (all tasks have the same exclusive gateway in their paths) must exist in all paths that start from these common exclusive gateways
   Warning: group task is possibly not accessible to the rest of the group

7. **Restriction:** there must be at least one Start Event element on the model (to run parallelism checks)
   Warning: Start Event element is missing

PETComputation

Required input objects
1...n data objects (protected data and/or data) from ProtectConfidentiality or PETComputation task

Optional input parameters
input script

Required output objects
1 data object (protected data or data)

Restrictions

1. **Restriction:** task must have at least one input and exactly one output data objects
   Error: at least 1 input and exactly 1 output are required

2. **Restriction:** "inputScript" property must exist under stereotype details object – "inputScript" can be empty, but must not have been deleted from model's .bpmn file
   Error: inputScript is undefined (inputScript is undefined
3. **Restriction:** "inputTypes" property must exist under stereotype details object – input types must have been selected and saved and "inputTypes" information must not have been deleted from model's .bpmn file
   Error: input types are undefined
4. **Restriction:** "outputTypes" property must exist under stereotype details object – output type must have been selected and saved and "outputTypes" information must not have been deleted from model's .bpmn file
   Error: output type is undefined
5. **Restriction:** each input data object must be output of ProtectConfidentiality or PETComputation task and if it is output of another PETComputation task then the output data object must be selected as "Private" under PETComputation stereotype settings of the output object's task
   Error: all inputs marked as private are not private

PKComputation

Required input objects
1...n data objects (ciphertexts, data) from PKEncrypt or PKComputationtask

Optional input parameters
input script

Required output objects
1 data object (ciphertext)

Restrictions

1. **Restriction:** task must have at least one input and exactly one output data objects
   Error: at least 1 input and exactly 1 output are required
2. **Restriction:** "inputScript" property must exist under stereotype details object – "inputScript" can be empty, but must not have been deleted from model's file
   Error: inputScript is undefined

3. **Restriction:** "inputTypes" property must exist under stereotype details object – input types must have been selected and saved and "inputTypes" information must not have been deleted from model's file
   Error: input types are undefined

4. **Restriction:** at least one input data object must have type selected as "Ecrypted"
   Error: at least 1 input must be selected as encrypted

5. **Restriction:** all key input objects of input data objects that origin from PKEncrypt or PKComputation tasks must be with the same name
   Error: key must be the same (with the same name) for all inputs

6. **Restriction:** all input data objects that have type selected as "Encrypted" must be encrypted data objects from PKEncrypt or PKComputation tasks – there must exist a key (with the same name) for all inputs that are selected as "Encrypted"
   Error: all inputs marked as encrypted are not encrypted

PKDecrypt

Required input objects
2 data objects (private key, ciphertext) from PKEncrypt or PKComputationtask

Required output objects
1 data object (data)

Restrictions

1. **Restriction:** task must have at exactly two input and one output data objects
   Error: exactly 2 inputs and 1 output are required

2. **Restriction:** "ciphertext" property must exist under stereotype details object – "ciphertext" must have been selected and saved and "ciphertext" information must not have been deleted from model's file
   Error: ciphertext is undefined

3. **Restriction:** "key" property must exist under stereotype details object – "key" must have been selected and saved and "key" information must not have been deleted from model's file
   Error: key is undefined

4. **Restriction:** input data object fixed (by its ID) in "ciphertext" property must exist also on model
   Error: ciphertext object is missing

5. **Restriction:** input data object fixed (by its ID) in "key" property must exist also on model
   Error: key object is missing

6. **Restriction:** ciphertext and key objects must be different data objects – they must have different IDs
   Error: ciphertext and key must be different objects

7. **Restriction:** key data object must have PKPrivate stereotype
   Error: key object must have PKPrivate stereotype

8. **Restriction:** there must exist a correct key corresponding to the ciphertext – input data object selected as key must be an input object "key" from PKEncrypt task or one of the input objects from PKComputation task selected as "Encrypted" – it must have the same name as input object "key" from PKEncrypt task or as one of the input objects of PKComputation task selected as "Encrypted"
   Error: ciphertext is encrypted with wrong encryption method or is not encrypted

9. **Restriction:** input data object selected as key must be from the same PKPublic-PKPrivate pair (group) as all other input "encryption key" objects from PKEncrypt tasks
   Error: all keys must be from the same PKPublic-PKPrivate key pair

PKEncrypt

Required input objects
2 data objects (public key, data)

Required output objects
1 data object (ciphertext)

Restrictions

1. **Restriction:** task must have at exactly two input and one output data objects
   Error: exactly 2 inputs and 1 output are required

2. **Restriction:** "inputData" property must exist under stereotype details object – "inputData" must have been selected and saved and "inputData" information must not have been deleted from model's file
   Error: inputData is undefined

3. **Restriction:** "key" property must exist under stereotype details object – "key" must have been selected and saved and "key" information must not have been

deleted from model's file
Error: key is undefined

4. **Restriction:** input data object fixed (by its ID) in "inputData" property must exist also on model
Error: input data object is missing

5. **Restriction:** input data object fixed (by its ID) in "key" property must exist also on model
Error: key object is missing

6. **Restriction:** input data and key objects must be different data objects – they must have different IDs
Error: input data and key must be different objects

7. **Restriction:** key data object must have PKPublic stereotype
Error: key object must have PKPublic stereotype

PKPrivate

Required input parameters
pair (with PKPublic and PKPrivate tasks)

Restrictions

1. **Restriction:** "groupId" property must exist under stereotype details object – pair ID ("groupId") must have been selected and saved and "groupId" information must not have been deleted from model's file
Error: groupId is undefined

Group specific restrictions

1. **Restriction:** there must be at least one PKPublic and one PKPrivate data object in a pair
Error: PKPublic key is missing from the pair

2. **Restriction:** all PKPrivate data objects in the same pair (with same groupId) must have the same name
Error: all PKPrivate keys of the pair must have the same name

PKPublic

Required input parameters
pair (with PKPublic and PKPrivate tasks)

Restrictions

1. **Restriction:** "groupId" property must exist under stereotype details object – pair ID ("groupId") must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined

Group specific restrictions

1. **Restriction:** there must be at least one PKPublic and one PKPrivate data object in a pair
   Error: PKPrivate key is missing from the pair

2. **Restriction:** all PKPublic data objects in the same pair (with same groupId) must have the same name
   Error: all PKPublic keys of the pair must have the same name

ProtectConfidentiality

Required input objects

1 data object (data)

Required output objects

1 data object (protected data)

Restrictions

1. **Restriction:** task must have exactly one input and one output data objects
   Error: exactly 1 input and 1 output are required

2. **Restriction:** input and output objects must be different data objects – they must have different IDs
   Error: input and output objects must be different data objects

SecureChannel

Required input objects

1...n data object

Required output objects

1...n data objects

Restrictions

1. **Restriction:** there must be at least one input-output data objects pair going through the message flow
   Error: at least 1 incoming-outgoing pair of data objects is required

2. **Restriction:** numbers of incoming and outgoing data objects of the message flow must match
   Error: number of incoming-outgoing pair of data objects does not match

3. **Restriction:** the list of names of incoming data objects must match with the list of names of outgoing data objects
   Error: names of incoming and outgoing data objects must match

SGXAttestationChallenge

Required input objects
0...n data objects (challenge)

Required input parameters
group (with SGXAttestationEnclave task)

Required output objects
1 data object (attestation outcome)

Restrictions

1. **Restriction:** task must have exactly one output data objects
   Error: exactly 1 output is required

2. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined

Group specific restrictions

1. **Restriction:** there must be exactly one SGXAttestationChallenge and one SGXAttestationEnclave task in a stereotype group
   Error: element with SGXAttestationEnclave stereotype is missing from the group

2. **Restriction:** both group tasks from the same stereotype group must be on different lanes
   Error: both group tasks must be on separate lane

3. **Restriction:** any task from the stereotype group must not be in the incoming or outgoing path of another task from the same stereotype group
   Error: both group tasks must be parallel

4. **Restriction:** all tasks from different stereotype groups must be in the same order on each lane – ordered lists of stereotype groups must match for each lane
   Warning: all group tasks are possibly not parallel

5. **Restriction:** all tasks from the same stereotype group that have common Exclusive Gateway elements in their paths (all tasks have the same exclusive gateway in their paths) must exist in all paths that start from these common exclusive gateways
   Warning: group task is possibly not accessible to the rest of the group

6. **Restriction:** there must be at least one Start Event element on the model (to run parallelism checks)
   Warning: Start Event element is missing

SGXAttestationEnclave

Required input objects
0...n data objects (enclave measurement)

Required input parameters
group (with SGXAttestationChallenge or SGXComputation tasks)

Required output objects
0 data objects

Restrictions

1. **Restriction:** task must have no output data objects
   Error: no outputs are required

2. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model’s file
   Error: groupId is undefined

.0.0.1 Group specific restrictions

1. **Restriction:** there must be exactly one SGXAttestationChallenge and one SGXAttestationEnclave task in a stereotype group
   Error: element with SGXAttestationChallenge stereotype is missing from the group

2. **Restriction:** both group tasks from the same stereotype group must be on different lanes
   Error: both group tasks must be on separate lane

3. **Restriction:** any task from the stereotype group must not be in the incoming or outgoing path of another task from the same stereotype group
   Error: both group tasks must be parallel

4. **Restriction:** all tasks from different stereotype groups must be in the same order on each lane – ordered lists of stereotype groups must match for each lane
   Warning: all group tasks are possibly not parallel

5. **Restriction:** all tasks from the same stereotype group that have common Exclusive Gateway elements in their paths (all tasks have the same exclusive gateway in their paths) must exist in all paths that start from these common exclusive gateways
   Warning: group task is possibly not accessible to the rest of the group

6. **Restriction:** there must be at least one Start Event element on the model (to run parallelism checks)
   Warning: Start Event element is missing

SGXComputation

Required input objects
1...n data objects (enclave data, data) from SGXComputation or SGXProtect tasks

Required input parameters
group (with SGXComputation, SGXProtect or SGXAttestationEnclave tasks)

Optional input parameters
input script (direct input or input as an output of another stereotype task)

Required output objects
1 data object (enclave data or data)

Restrictions

1. **Restriction:** task must have at least one input and exactly one output data objects
   Error: at least 1 input and exactly 1 output are required

2. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined

3. **Restriction:** "inputScript" property must exist under stereotype details object – "inputScript" can be empty, but must not have been deleted from model's file
   Error: inputScript is undefined

4. **Restriction:** in case input script is selected as an output of another stereotype task, this task must have selected stereotype attached to it on model and in model's file
   Error: inputScript stereotype is missing

5. **Restriction:** "inputTypes" property must exist under stereotype details object – input types must have been selected for each input and saved and "inputTypes" information must not have been deleted from model's file
   Error: input types are undefined

6. **Restriction:** "outputTypes" property must exist under stereotype details object – output type must have been selected and saved and "outputTypes" information must not have been deleted from model's file
   Error: output type is undefined

7. **Restriction:** at least one input data object must have type selected as "SGXPrivate"
   Error: at least 1 input must be selected as encrypted

8. **Restriction:** all input data objects that have type selected as "SGXPrivate" must be data objects from SGXProtect tasks or data objects selected as SGXPrivate from SGXComputation tasks
   Error: all inputs marked as SGXPrivate are not SGXPrivate

Group specific restrictions

1. **Restriction:** all group tasks from the same stereotype group must be on the same lane
   Error: all group tasks must be on the same lane

SGXProtect

Required input objects
1 data object (data)

Required input parameters
group (with SGXProtect, SGXComputation or SGXAttestationEnclavetasks)

Required output objects
1 data object (protected data)

Restrictions

1. **Restriction:** task must have exactly one input and one output data objects
   Error: exactly 1 input and 1 output are required

2. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined

SGXQuoteVerification

Required input objects
3 data objects (quote, certificate, revocation list)

Required output objects
1 data object (verification outcome)

Restrictions

1. **Restriction:** task must have exactly three input and one output data objects
   Error: exactly 2 inputs and 1 output are required

2. **Restriction:** "quote" property must exist under stereotype details object – "quote" must have been selected and saved and "quote" information must not have been deleted from model's file
   Error: quote is undefined

3. **Restriction:** "certificate" property must exist under stereotype details object – "certificate" must have been selected and saved and "certificate" information must not have been deleted from model's file
   Error: certificate is undefined

4. **Restriction:** input data object fixed (by its ID) in "quote" property must exist also on model
   Error: quote object is missing

5. **Restriction:** input certificate object fixed (by its ID) in "certificate" property must exist also on model
   Error: certificate object is missing

6. **Restriction:** "revocationList" property must exist under stereotype details object – "revocationList" must have been selected and saved and "revocationList" information must not have been deleted from model's file
   Error: revocationList is undefined

7. **Restriction:** input certificate object fixed (by its ID) in "revocationList" property must exist also on model
   Error: revocationList object is missing

8. **Restriction:** quote, certificate and revocation list objects must be different data objects – they must have different IDs
   Error: quote, certificate and revocation list must be different objects

SGXQuoting

Required input objects
2 data objects (challenge, measurement)

Required output objects
1 data object (quote)

Restrictions

1. **Restriction:** task must have exactly two input and one output data objects
   Error: exactly 2 inputs and 1 output are required

2. **Restriction:** "challenge" property must exist under stereotype details object – "challenge" must have been selected and saved and "challenge" information must not have been deleted from model's file
   Error: challenge is undefined

3. **Restriction:** "measurement" property must exist under stereotype details object – "measurement" must have been selected and saved and "measurement" information must not have been deleted from model's file
   Error: measurement is undefined

4. **Restriction:** input data object fixed (by its ID) in "challenge" property must exist also on model
   Error: challenge object is missing

5. **Restriction:** input measurement object fixed (by its ID) in "measurement" property must exist also on model
   Error: measurement object is missing

6. **Restriction:** challenge and measurement objects must be different data objects – they must have different IDs
   Error: challenge and measurement must be different objects

SKComputation

Required input objects
1...n data objects (ciphertexts, data) from SKEncrypt or SKComputationtask

Optional input parameters
input script

Required output objects
1 data object (ciphertext)

Restrictions

1. **Restriction:** task must have at least one input and exactly one output data objects
   Error: at least 1 input and exactly 1 output are required

2. **Restriction:** "inputScript" property must exist under stereotype details object – "inputScript" can be empty, but must not have been deleted from model's file
   Error: inputScript is undefined

3. **Restriction:** "inputTypes" property must exist under stereotype details object – input types must have been selected and saved and "inputTypes" information must not have been deleted from model's file
   Error: input types are undefined

4. **Restriction:** at least one input data object must have type selected as "Encrypted"
   Error: at least 1 input must be selected as encrypted

5. **Restriction:** all key input objects of input data objects that origin from SKEncrypt or SKComputation tasks must be with the same name
   Error: key must be the same (with the same name) for all inputs

6. **Restriction:** all input data objects that have type selected as "Encrypted" must be encrypted data objects from SKEncrypt or SKComputation tasks – there must exist a key (with the same name) for all inputs that are selected as "Encrypted"
   Error: all inputs marked as encrypted are not encrypted

SKDecrypt

Required input objects
2 data objects (secret key, ciphertext) from SKEncrypt or SKComputationtask

Required output objects
1 data object (data)

Restrictions

1. **Restriction:** task must have at exactly two input and one output data objects
   Error: exactly 2 inputs and 1 output are required

2. **Restriction:** "ciphertext" property must exist under stereotype details object – "ciphertext" must have been selected and saved and "ciphertext" information must not have been deleted from model's file
   Error: ciphertext is undefined

3. **Restriction:** "key" property must exist under stereotype details object – "key" must have been selected and saved and "key" information must not have been deleted from model's file
   Error: key is undefined

4. **Restriction:** input data object fixed (by its ID) in "ciphertext" property must exist also on model
   Error: ciphertext object is missing

5. **Restriction:** input data object fixed (by its ID) in "key" property must exist also on model
   Error: key object is missing

6. **Restriction:** ciphertext and key objects must be different data objects – they must have different IDs
   Error: ciphertext and key must be different objects

7. **Restriction:** input data object selected as key must be an input object "key" from SKEncrypt task or one of the input objects selected as "Encrypted" from SKComputation task – it must have the same name as input object "key" from

SKEncrypt task or as one of the input objects selected as "Encrypted" from SKComputation task
Error: ciphertext is encrypted with wrong encryption method or is not encrypted

8. **Restriction:** input data object selected as key must be with the same name as all input "encryption key" objects from SKEncrypt and SKComputation tasks – there must be only one unique key name concerning particular input ciphertext
Error: decryption key must be the same (with the same name) as the encryption key

SKEncrypt

Required input objects
2 data objects (secret key, data)

Required output objects
1 data object (ciphertext)

Restrictions

1. **Restriction:** task must have at exactly two input and one output data objects
Error: exactly 2 inputs and 1 output are required

2. **Restriction:** "inputData" property must exist under stereotype details object – "inputData" must have been selected and saved and "inputData" information must not have been deleted from model's file
Error: inputData is undefined

3. **Restriction:** "key" property must exist under stereotype details object – "key" must have been selected and saved and "key" information must not have been deleted from model's file
Error: key is undefined

4. **Restriction:** input data object fixed (by its ID) in "inputData" property must exist also on model
Error: input data object is missing

5. **Restriction:** input data object fixed (by its ID) in "key" property must exist also on model
Error: key object is missing

6. **Restriction:** input data and key objects must be different data objects – they must have different IDs
Error: input data and key must be different objects

SSComputation

Required input objects
1...n data objects (shares, data), shares from SSSharing tasks or SSComputation task groups

Required input parameters
group (with SSComputation tasks)

Optional input parameters
input script

Required output objects
1 data object (share)

Restrictions

1. **Restriction:** task must have at least one input and exactly one output data objects
   Error: at least 1 input and exactly 1 output are required
2. **Restriction:** "groupId" property must exist under stereotype details object – group must have been selected and saved and "groupId" information must not have been deleted from model's file
   Error: groupId is undefined
3. **Restriction:** "inputScript" property must exist under stereotype details object – "inputScript" can be empty, but must not have been deleted from model's file
   Error: inputScript is undefined
4. **Restriction:** "inputs" property must exist under stereotype details object – "inputs" information must not have been deleted from model's file
   Error: inputs are undefined
5. **Restriction:** all names of input data objects must be different
   Error: all input function shares must be different

Group specific restrictions

1. **Restriction:** all input data objects fixed (by their IDs) in "inputs" property (for all group tasks) must exist also on model
   Error: one or more shares (data objects) corresponding to the same input of the group are missing

2. **Restriction:** all input data objects corresponding to the same input must be outputs of SSSharing tasks or AddSSComputation task groups and they must be all from the same stereotype group
   Error: all shares corresponding to the same input must originate from the same task with SSSharing stereotype or from the same group of tasks with SSComputation stereotypes

3. **Restriction:** all names of input data objects corresponding to the same input must be different
   Error: all shares corresponding to the same input must be different

4. **Restriction:** there must be at least two tasks in the same stereotype group
   Error: group must have at least 2 members

5. **Restriction:** the number of members in stereotype group must be greater than or equal to the number on computation parties of each SSSharing tasks
   Error: the number of members in SSComputation group is not correct (it should be equal to or greater than the number of computation parties of each task with SSSharing stereotype)

6. **Restriction:** all group tasks from the same stereotype group must be on different lanes
   Error: each group task must be on separate lane

7. **Restriction:** any task from the stereotype group must not be in the incoming or outgoing path of another task from the same stereotype group
   Error: all group tasks must be parallel

8. **Restriction:** all tasks from the stereotype group must have the same number of input and output data objects
   Error: each group task must have the same number of inputs and outputs

9. **Restriction:** all tasks from different stereotype groups must be in the same order on each lane – ordered lists of stereotype groups must match for each lane
   Warning: all group tasks are possibly not parallel

10. **Restriction:** all tasks from the same stereotype group that have common Exclusive Gateway elements in their paths (all tasks have the same exclusive gateway in their paths) must exist in all paths that start from these common exclusive gateways
    Warning: group task is possibly not accessible to the rest of the group

11. **Restriction:** there must be at least one Start Event element on the model (to run parallelism checks)
    Warning: Start Event element is missing

SSReconstruction

Required input objects
2...n data objects (shares) from SSSharing task / SSComputation task group

Required output objects
1 data object (secret data)

Restrictions

1. **Restriction:** task must have at least two input and exactly one output data objects
   Error: at least 2 inputs and exactly 1 output are required

2. **Restriction:** all input data objects must be outputs of SSSharing task / SSComputation task group and they must be all from the same stereotype group
   Error: all input function shares must originate from the same task with SSSharing stereotype or from the same group of tasks with SSComputation stereotypes

3. **Restriction:** all names of input data objects must be different
   Error: all input function shares must be different

4. **Restriction:** the number of input data objects must be greater than or equal to the number of threshold
   Error: the number of input function shares from the task with SSSharing stereotype or from the group of tasks with SSComputation stereotypes must be greater than or equal to the number of threshold

SSSharing

Required input objects
1 data object (data)

Required input parameters
threshold, number of computation parties

Required output objects
2...n data objects (shares)

Restrictions

1. **Restriction:** task must have exactly one input and at least two output data objects
   Error: exactly 1 input and at least 2 outputs are required

2. **Restriction:** "threshold" property must exist under stereotype details object – "threshold" must have been inserted and saved and "threshold" information must not have been deleted from model's file
   Error: threshold is undefined

3. **Restriction:** "computationParties" property must exist under stereotype details object – "computationParties" must have been inserted and saved and "computationParties" information must not have been deleted from model's file
   Error: computationParties is undefined

4. **Restriction:** threshold must be an integer greater than one and less than or equal to the number of output objects
   Error: threshold must be an integer greater than 1 and equal to or less than the number of outputs (1 < threshold <= number of outputs)

5. **Restriction:** computation parties must be an integer greater than or equal to threshold and less than or equal to the number of output objects
   Error: computation parties must be an integer greater than or equal to threshold and less than or equal to the number of outputs (threshold <= computation parties <= number of outputs)

III. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Aivo Toots**,

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

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Tool Support for Privacy-Enhanced Business Process Model and Notation

   supervised by Pille Pullonen and Luciano García-Bañuelos

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 14.05.2018