Noname manuscript No.
(will be inserted by the editor)

Modelling the Interplay of Security, Privacy and Trust in Sociotechnical Systems: A Computer-Aided Design Approach

Mattia Salnitri · Konstantinos Angelopoulos · Michalis Pavlidis · Vasiliki Diamantopoulou · Haralambos Mouratidis · Paolo Giorgini

Received: date / Accepted: date

Abstract Personal data has become a central asset for multiple enterprise applications and online services offered by private companies, public organisations or a combination of both. The sensitivity of such data and the continuously growing legislation that accompanies their management dictate the development of methods that allow the development of more secure, trustworthy software systems with focus on privacy protection. The contribution of this paper is the definition of a novel requirements engineering method that supports both early and late requirements specification, giving emphasis on security, privacy and trust. The novelty of our work is that it provides the means for software designers and security experts to analyse the system-to-be from multiple aspects, starting from identifying high level goals to the definition of business process composition, and elicitation of mechanisms to fortify the system from external threats. The method is supported by two CASE tools. To demonstrate the applicability and usefulness of our work, the paper shows its applications to a real-world case study.

Keywords Security · Privacy · Trust · sociotechnical systems · CASE tools

Mattia Salnitri
Department of Electronic, Computer Science and Bioengineering
Politecnico di Milano,
E-mail: mattia.salnitri@polimi.it

Konstantinos Angelopoulos
Collibra
E-mail: konstantinos.angelopoulos@collibra.com

Michalis Pavlidis, Haralambos Mouratidis
Centre for Secure, Intelligent and Usable Systems
University of Brighton,
E-mail: m.pavlidis@brighton.ac.uk, h.mouratidis@brighton.ac.uk

Vasiliki Diamantopoulou
Department of Information and Communication Systems Engineering
University of the Aegean
E-mail: vdiamant@aegean.gr

Paolo Giorgini
Department of Information Engineering and Computer Science
University of Trento
E-mail: paolo.giorgini@unitn.it

1 Introduction

Software engineers have always been challenged while capturing the intentions of stakeholders and analysing them in order to understand the problem for which they should provide a solution. The efforts to address this challenge led, as a result, to the foundation of Requirements Engineering (RE). An accurate definition for RE, which has been provided by Zave [97], is the following: *'Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families'.*

Research in the area of RE has provided well established approaches for capturing requirements such as the Unified Modelling Language (UML) [70] that follows an object oriented approach to model the problem described by the stakeholders. Goal-oriented requirements analysis [55] is a different approach of modelling the stakeholders' intentions for the system. Numerous general purpose goal-oriented modelling languages [4, 11, 16, 34, 57, 93] have been proposed over the years, focusing on different aspects of the RE process, e.g., cap-

turing social interactions, handling conflicting requirements, reasoning about alternative solutions.

The RE process is composed of two fundamental phases, the early requirements [95], where the organisational objectives of the system are specified, and the late requirements [45], where the functional aspects of the system that will meet the early requirements, are specified. In other words, the first phase focuses on the description of the problem the system-to-be will solve, whereas the latter captures the details of the solution. Frequently, the description of the problem includes organisational aspects, while the details of the solution includes concepts related to the software that has to be developed. For example, organisational aspects can be a requirement for anonymisation due to the movement of a dataset among companies, while the details of the solution can be a specific anonymisation technique. There is lack of methods to generate the specification of the expected functionality of the software system in terms of privacy, security, and trust, from the high level description of the security, privacy, and trust requirements respectively, that considers also the benefits of specific functionalities, and their implications to the rest of the system.

The aforementioned approaches usually cover only one of the two phases and, often, partially. This problem forces software designers to use multiple tools for providing a complete description of their design, that might not be related to each other, increasing the effort of learning how to use them and decreasing the final output's cohesion.

In addition to this issue in RE, during the last years, security, privacy, and trust gained a central position in software design, since consequences of unsecured systems are no more affordable by organisations and companies, and the earlier the security and privacy problems are discovered the lesser the consequences are. Moreover, new regulations, such as the General Data Protection Regulation (GDPR) [82], impose stricter constraints and higher sanctions. In particular, GDPR enforced by the European Union (EU) in May 2018 and all companies and organisations that manage personal data of citizens or customers are obliged to follow it. Privacy and security are aspects to be considered during the early requirements engineering phases and extensions of the aforementioned goal-oriented approaches [62, 53, 46] have been proposed to address security issues in software systems. These approaches enable the identification of external threats, vulnerabilities of the system-to-be and reasoning about counter-measures to mitigate them. There are also other approaches that allow analysing trust between interdependent components of a system [13, 63], which has become a common phenomenon as the scale of modern software systems keeps increasing. However, there are no approaches that tackle all the dimensions, security, privacy and trust, raising the same issues we mentioned earlier.

In this work we propose a novel method, named SePTA (Security, Privacy and Trust Approach), that supports a unified specification of security, privacy and trust requirements under one framework and enables software designers and security experts to enforce such requirements. SePTA is designed for sociotechnical systems, i.e. complex information systems such as the ones of public administrations and large companies, where there is an interplay between people and autonomous technical components that collaborate in order to achieve common objectives.

In this paper we focus on how security, privacy and trust requirements can be specified in early requirement phase using a goal-based modelling language, and how such requirements can be correctly enforced in the late requirement phase, using goal-based modelling languages and a modelling language for business processes. In particular, we adopted a business process modelling language for the definition of the late requirements since it can be used as a specification of how goals can be achieved.

More specifically, our work integrates and extends work from security, privacy and trust modelling languages to provide a method that introduces the following original contributions: i) provides a holistic requirements modelling and analysis approach that also includes security, privacy and trust requirements, supporting both early and late requirements elicitation; ii) provides a software tool that offers automated functionalities that reduce the effort of the designers, by not having to repeat modelling tasks in order to analyse a different aspect of the system; iii) facilitates the solution discovery in terms of security and privacy, by providing patterns that address common issues; iv) provides a method to enforce privacy and security requirements. Finally, we illustrate the application of the proposed approach and its benefits through a real-world case study from the domain of e-government.

The rest of this paper is organised as it follows: Section 2 describes the research framework used for the creation of SePTA, Section 3 presents the baseline of our work. Then, Section 4 describes in detail the SePTA method. Next, Section 5 shows how we apply SePTA on a real-world case study, while Section 6 presents the related work. Finally, Section 7 concludes the paper.

2 Method design

This section describes how we designed SePTA, using the design science approach defined by Hevner et al. in [32], with the following sections: (i) *problem relevance*, which highlights the importance of the problem that our approach addresses; (ii) *research rigour*, which describes the validity of the baseline of SePTA; (iii) *design and research process*, where the process used to create the method proposed in this paper, is defined; (iv) *artefact*, which highlights the outcomes described in this paper; (v) *design evaluation*, where we explain how SePTA has been evaluated; (vi) *research contribution*, which defines the contribution and the novelty of the proposal of this paper; while (vii) *research communication* describes how we communicate the content of our research work.

2.1 Problem relevance

As described in the introduction, it is extremely important to consider security and privacy during the requirements phase of sociotechical systems. However, as far as our knowledge goes, no method covers the entire requirement engineering process, from the definition of early requirements to late requirements. Consequently, security and privacy experts are forced to use multiple approaches [58] and this leads to conceptual and methodological gaps that result into consequences that span from requirements misaligned, to early requirements not enforced that, once (not) implemented, which leads to breaches that affect systems' security and/or users' privacy. Moreover, the satisfaction of security and privacy requirements is often assigned to human actors or to external systems, where there is no assurance that these entities will behave as expected. Therefore, there are a lot of trust assumptions about the satisfaction of security and privacy requirements that are made during the design of a system that remain overlooked [64]. These assumptions may be wrong and during the operation of the system will lead to security and privacy breaches.

2.2 Research rigour

The approach defined in this paper is based on four modelling languages: STS-ml[15], SecBPMN2 [75], Tropos [53] and JTrust [63]. Each language covers strategic aspects of early or late requirements. All modelling languages have been thoroughly validated using real case studies [20]. Moreover, the syntax of the STS-ml and SecBPMN2 languages has been formally defined [15] [75], while SecTro and JTrust have been semi-formally defined [53] [63]. Meta-models and detailed guidelines exist that can support the software designer in the development of the relevant models. The chosen baseline of SePTA offers a solid research foundation.

2.3 Design and research process

We defined as target users of SePTA security and privacy experts with a deep knowledge on the domain where the system will be deployed. We decided to involve the target users of the method directly in its definition. This decision allowed immediate feedback which eased the creation of an effective framework, based on experience of the users. The users involved were selected from heterogeneous domains such as health care, public administration and private IT companies, in order to avoid to be influenced by domain specific issues and requirements.

The creation of SePTA followed an iterative process where, in each iteration, the involved stakeholders evaluated the design of the method and the fulfilment of its requirements.

During the initial stage we interviewed a group of target users from different domains in order to define the initial set of requirements for the method [27]. We identified five main requirements, relevant to SePTA: (i) the method shall support security experts in identifying security and privacy requirements and in proposing security mechanisms to enforce them; (ii) the method shall facilitate the detection of weak trust relationships among the stakeholders of the system-to-be and refine the design in order to strengthen them; (iii) the method shall help users to move between different perspectives (i.e. security, privacy and trust perspectives) and between different level of abstractions (i.e. between goal diagrams and business process diagrams); (iv) the method shall be supported by a software that helps the users to analyse the models and check security, privacy and trust properties; (v) the method shall help users dealing with large sociotechnical systems, such as the ones of public administrations and large companies.

2.4 Artefact

The outcome of the design process, and what is described in this paper, is the SePTA method, which is composed of a process used to guide users in the definition of security, privacy and trust requirements, and a software tool that supports this process with graphical editors for the modelling language and automated verification engines for the analysis of security, privacy and

trust properties. The process commences after the activity of requirements elicitation, for which a range of available techniques already exist, such as interviews, document analysis, and questionnaires.

SePTA is a middle-range theory, as defined by Wieringa and Daneva in [89]. Therefore, it has no universal scope. In fact, it targets, and it is limited to sociotechnical systems, where human and autonomous components interact to achieve common objectives. In particular, it can be used during the early and late requirements phase of the design of such systems, where security and privacy experts need to perform both security, privacy and trust analyses.

2.5 Design evaluation

The involvement of targeted users in the creation of SePTA allowed us to evaluate the proposed method during its design and implementation. Moreover, once the method was released, we asked target users to apply SePTA in real world case studies [20]. In particular, the method was evaluated using three scenarios that represent three very different domains: (i) public administration; (ii) healthcare; and (iii) a private IT company. More information on the case study from the public administration domain is presented in Section 5.

2.6 Research contribution

The research contribution consists the framework as a set of guidelines for security and privacy experts and a set of model transformations that allow to automatically generate one model from another and, therefore, to facilitate experts in aligning early and late security and privacy requirements.

2.7 Research communication

This is a scientific publication, therefore, the communication style is technical and is aimed for an academic audience. However, especially during introduction, the discussion is less technical and we motivate the paper with real case examples in order to address a wider set of readers, such as IT experts and security and privacy engineers.

3 Baseline

Three of the four approaches we chose for modelling and analysing security, privacy and trust requirements, i.e. STS-ml, SecTro and JTrust, are based on a combination of modelling languages that are built on the foundations of Tropos [11] and i^* [93]. These two modelling languages combine Goal Oriented Requirements Engineering (GORE) with organisational and social aspects for designing a system.

We chose to use goal-based modelling languages since they are focused on the social relations among actors. This is an aspect that is central in early and late requirements in complex systems, such as the ones targeted by SePTA.

The fourth modelling language, i.e. SecBPMN2, is used for specifying security constraints of procedural aspects of systems. In particular, it is based on BPMN 2.0 [59], a well known standard notation for modelling business processes. We selected a procedural-based modelling language, since it permits to specify how the goals, defined in the goal-based modelling languages, can be achieved.

One of our objectives consisted in minimising the effort of SePTA users. We, therefore, avoided to create yet another set of new modelling languages and, instead, we adopted well known ones. Before selecting goal-based (STS-ml, SecTro and JTrust) and procedural (SecBPMN2) modelling languages, we examined other classes of languages, such as UML class diagram, Fault Tree Analysis (FTA), use cases and misuse cases. In particular, UML [60] class diagram is a widely used modelling language, however its scope (modelling of classes) targets the implementation of software and not the definition of social interactions between actors of sociotechnical systems. FTA [42] and attack trees [47] approaches and modelling languages allow to define security attacks on the system and possible mitigations. However, such modelling languages cannot be used for the analysis of the interaction between actors in sociotechnical systems. Use case [9] and misuse case [3] modelling languages are actor-centred and define the (mis)usage of functionalities of the systems-to-be. Unfortunately, they do not define the interactions between actors of a system, but only the interactions between users and the system-to-be.

For a detailed comparison of goal-based modelling languages and BPMN based modelling languages, please refer to Section 6.

In this section we present the fundamental concepts that our work inherits from Tropos, $i^*$and BPMN 2.0, while we explain their role in the requirements engineering process.

3.1 Goal Modelling

Goals state the intentions of stakeholders of a system-to-be and their satisfaction depends on actors or software components that compose it. More specifically, in the terminology of Tropos and i^*, goals represent functional requirements. A goal can be refined to more detailed goals (aka subgoals) using AND/OR refinements that follow Boolean Logic, forming acyclic directed graphs. Requirement engineers refine goals until the produced subgoals represent definite operations that can be executed either by a software component or a human that is part of the system-to-be. Similarly, non-functional requirements are represented by softgoals [14]. Such requirements capture qualities of the system-to-be that lack clear-cut criteria for their fulfilment, satisfied or not, whereas goals are satisfied or not.

The achievement of goals may require or may produce various resources in the system. For instance, in the case study that we will examine in the next sections, the Municipality of Athens (MoA) has as a goal to publish birth certificates. The achievement of this goal creates the resource, birth certificate, which, depending on the modelling language, can be referred also as a document. These resources are transmitted within the system-to-be raising security concerns, as we will see in the following sections.

The languages that constitute the baseline of this work support the representation of social relationships. The fundamental concept of such relationship is, in Tropos terminology, the Actor. This concept represents a physical agent (e.g., a human) or a software component. Actors have goals and often depend on each other to fulfil theirs. Such a relationship between two actors is named dependency, the actor that relies on an other is referred as depender and the latter as dependee. Given that not every actor is reliable, dependencies in a system might lead to unsatisfied goals because a dependee didn't fulfil what was expected, causing the depender not to fulfil their own goals, creating a reaction of failures. Consequently, dependencies must concern the system designers with respect to the reliability and trustworthiness of the dependees. In this work we address such issues with trust analysis.

3.2 Business Process Modelling

In our approach we consider procedural aspects that can be used to specify organisational details of the system-to-be, such as the sequence of activities executed and the privacy and security constraints when such activities are executed. In particular, we focused on business processes, i.e. sequences of activities executed to achieve a business objective.

One of the most known and used modelling language for documenting business process is the Business Process Modelling and Notation (BPMN) standard. BPMN 2.0 [59] is the last version of the standard and specifies a rich and expressive modelling language. However, the basic concepts are simple yet effective. There are four main types of elements: activities, gateways, events and data objects, that are described in the rest of the section.

Activities are atomic set of instructions. The order of activity is specified with a control flow, that is represented with an arrow that connects two activities whose target activity is executed after the source activity.

Gateways specify forks in the sequence of activities. The Exclusive Gateway represents an evaluation of a statement, based on which one set of activities is executed instead of another one. The Parallel Gateway specifies that a set of activities is executed at the same time.

Events specify something that happens. The start of a business process and reception of a message are examples of two events.

Data Objects refer to containers of information that are used and/or produced by activities when they are executed.

4 The SePTA method

In this section we illustrate the motivation of our work and we present the Security, Privacy and Trust Approach (SePTA) to support the RE process in both early and late requirements specification.

SePTA involves four modelling languages, STS-ml [15], SecBPMN2 [75], Secure Tropos (SecTro) [53] and JTrust [63], each of them focusing on different aspects of the system and collaboratively enable designers analyse security and privacy requirements and examine the validity of trust relationships within the system-to-be. STS-ml and SecBPMN2 are implemented by the STS-tool[1], whereas Secure Tropos and JTrust by the SecTro tool[2] and can be used for facilitating the modelling of organisational aspects of sociotechnical systems, performing security analysis and capturing trust relationships, respectively. In our work we combine these tools by implementing transformations that allow these tools to exchange models and contribute by satisfying a subset of the requirements presented in the previous section. The purpose of combining these tools

[1] `http://www.sts-tool.eu/`

[2] `www.sense-brighton.eu/research/sectro-tool`

is to cover all the aspects of software designing every phase of the RE process.

4.1 Early Requirements

SocioTechnical Security (STS) [15] is a method focused on specifying early security requirements of sociotechnical systems. STS uses STS-ml, a goal-based modelling language that allows to specify components of a system, called Actors, their interactions and their privacy and security requirements.

STS-ml is a multi-view modelling language: it permits to specify a system from three perspectives. Each one highlights different properties of a sociotechnical system and it allows to easily define large systems. The three views supported by STS-ml are: (i) the social view, which is used to represent the objective of actors of a system; (ii) the information view, which is used to specify the content of documents exchanged between actors; (iii) the authorisation view, which is used to represent the flows of authorisations between actors. All views permit to specify two types of Actors: (i) Agent, which characterises a specific autonomous component; (ii) Role, which specifies an unknown (set of) components. For example, George is an agent and citizen is a role, since the former identifies a specific individual while the latter specifies a generic set of individuals, i.e. all the citizens.

STS-ml is focused on the social relationships among actors of a system. The assets it considers using this perspective are the interactions between actors. From our experience, the focus on such interactions helps security and privacy experts during the early requirements phase of a sociotechnical system. Consequently, the security requirements that can be defined using STS-ml target such relations. However, for the level of abstraction required in this phase, our stakeholders confirmed that the set of requirements that can be specified with STS-ml covers their security needs.

In the following, the three views of STS-ml are described.

Social View. The social view supports the specification of Goals of each Actor and delegations of goals. Furthermore, this view allows the specification how Documents, i.e. tangible entities, are used to store and transfer knowledge. Figure 2 shows an example of STS-ml social view; in the image the ID card is a document that can be used by the citizen or transmitted from the citizen to the MoA.

Information View. The information view is centred on the concept Information. An information, in STS-ml, is an intangible element, which represents a piece of knowledge, and it is stored in a document. Figure 3 shows an example of STS-ml information view; in the example, name and surname are two Information elements that are stored in an ID card Document. In the information view is possible to specify the set of information stored in each document.

Authorisation View. The authorisation view is used to specify the permissions granted between actors, for the usage of information. Each authorisation is composed of three parts: (i) the set of information, target of the authorisation, (ii) the operations (among read, modify, produce and transmit) that are allowed to be executed on the set of information, and (iii) the set of goals for which the authorisation is granted. Figure 4 shows an example of STS-ml information view; in the image a citizen authorises the MoA to read and modify the information family name only for achieving the goal of providing their birth certificate.

4.2 Late Requirements

In the second phase of the RE process, the designer must refine the initial requirements to more detailed ones. At this point, both stakeholders and designers have a more clear view of how the system-to-be will be structured and express additional requirements that are more system-specific. Modern agile software development approaches allow also early [85] or parallel [5] composition of the systems architecture which can be derived from the models of the previous phase. With this in mind, SePTA allows the specification of STS-ml model to a more detailed one, using Secure Tropos and JTrust that provide three additional views: a) Security Requirements view, b)Attacks view, c) Trust view. The Procedural view is supported by SecBPMN2. The transformations of the STS-ml concepts to Secure Tropos and JTrust concepts are depicted in Table 1, while the transformation from STS-ml to SecBPMN2 is described in [75].

The transformations of the models is supported by the software tool, provided with the method, in a semi-automated fashion. STS-ml diagrams are the first to be designed and, therefore, they cannot be generated, the other diagrams are then derived from the STS-ml models. After that, users will have to enrich the models, by using the concepts peculiar of the modelling language they want to use. For example, the software can automatically generate partial business processes from STS-ml diagrams, after that the user will have to enrich the business processes diagrams to create complete models.

Security Requirements View. The Security Requirements view displays agents and roles which are

STS	**SecTro** & **JTrust**
Agent\Role	Actor
Goal	Goal
Document	Resource
Delegation	Security Dependency
AND\OR	AND\OR

Table 1 Concept Transformations

referred as actors in the SecTro Notation, and their goals. In STS-ml there is distinction between agents and roles, as this language examines the system at an organisational level. However, Secure Tropos and JTrust analyse the system at an operational level and therefore, the nature of the entities involved in the system is not important. This has led to the decision to maintain the term of actor in Secure Tropos which represents both agents and roles.

SecTro enables software designers to add new goals and refine existing ones to plan, clear-cut operations that cannot be refined any further and are achieved by an actor of the system-to-be. Next, the security and privacy requirements specified over the delegations in STS-ml are transformed to Constraints, a central concept of this view. A Constraint is a restriction on an actor's function and can be either security or privacy related. For instance, if an actor has a goal of sending a confidential information to another actor, this goal shall be restricted by a security related constraint sender-confidentiality. Additionally, a constraint is related to an objective that needs to be fulfilled, such as confidentiality, integrity, authentication, etc. In this case, this goal is confidentiality. Constraints are satisfied by introducing mechanisms, system components that address security and/or privacy concerns in a system.

From a system design perspective, the purpose of a constraint is to indicate that the components that will fulfil the restricted goal should include counter-measures to external threats. In particular, a threat is a circumstance that could potentially cause damage to the system, such as information disclosure, tampering, spoofing to name some. A common approach to identify threats is STRIDE [78], which proposes six basic classes of threats and a systematic approach to identify them. However, the number of available threats in real world is continuously growing and new threats are continuously being documented.

When a threat is identified, mechanisms are proposed as counter-measures to mitigate it. A mechanism represents a software component or policy that must be implemented or adopted respectively, in order to fulfil a constraint. The identification of such mechanisms requires expertise, both in the domain of the system-to-be and security. As a result, proposing mechanisms can be a cumbersome process. In our approach, we have created a pattern library, where each pattern is a composition of alternative mechanisms that address various privacy-related Objectives that reduce the designer's effort. At the moment, the library includes only mechanisms, and not architectures, that mitigate privacy threats.

In particular, our library offers an implementation of previous work [18] that covers five basic privacy properties [67] that every data-centric sociotechnical system should satisfy. These properties and potential mechanisms that fulfil them are listed below.

- Anonymity. A system that satisfies this property should not permit the identification, direct or indirect, of a user by those who have access to this data. Anonymity can be achieved with the use of anonymisation services, such as virtual e-mail addresses and onion-routing, or with the use of track and evident erasers, such as spyware detection and removal software.
- Pseudonimity. A system that satisfies this property should guarantee that all entities associated with the data stored by or transferred through the system are not identifiable and alias is used instead of their real identity. This property can be achieved with the use of administrative tools such as smart cards, biometrics, or pseudonimiser tools, such as Mixmaster.
- Unlinkability. A system that satisfies this property should guarantee that when a user makes multiple uses of multiple resources of the system, relating these uses cannot reveal the user's identity. This property can be achieved with the use of anonymiser products (e.g., Tor, Hordes, GAP, etc), pseudonimiser tools or track and evident erasers.
- Undetectability. A system that satisfies this property should guarantee that a third party cannot distinguish whether a specific entity is a user of the system or not. This property can be achieved with the use of encryption, anononymiser tools, administrative tools, etc.
- Unobservability. A system that satisfies this property should guarantee that a user cannot be distinguished from the other users who belong in the same set while using the system. This property can be achieved with the use of administrative tools, anonymiser products, etc.

As it concerns security related mechanisms, the range of mechanisms is significantly higher than the privacy related ones. For this type of mechanisms there is guidance available for the designers in selecting mechanisms. This guidance is in the form of secu-

rity patterns [77] or online databases, such as the Common Attack Pattern Enumeration and Classification[3] (CAPEC) and the Top 10 and Proactive Controls projects from the Open Web Application Security Project (OWASP)[4].

Attacks View. Each Threat identified in the Security Requirements view can be further analysed in a separate view, namely Attacks view. The role of this view is to further analyse how a threat can be manifested and what are the specific vulnerabilities of the system-to-be. The main two concepts of this view is the attack method, which represents a method that a threat can be manifested and the vulnerability, a weakness of a system component or organisation. An attack method is performed by a malicious actor, named 'Attacker' and can be refined in the same manner goals do, using AND/OR. The purpose of attacks is to exploit vulnerabilities that are associated with goals or resources of the system. This view belongs to the late requirements phase because, in order to identify vulnerabilities, the architecture of the system must has already been drafted. The vulnerability detection requires domain knowledge and investigation by security engineers who usually rely on continuously update resources, such as the Common Vulnerability and Exposures (CVE) knowledge base[5].

Trust View. In sociotechnical systems, where the fulfilment of certain goals depend on human actors, often doubts are created on the fact if such actors will act according to the system's design or not. Hence, the trust, defined as the positive expectation of one actor from another that a specific goal assigned to the latter, will be fulfilled [56], must be analysed before proceeding to the system's implementation. This means that, in case of dependencies, the trust of the depender on the dependee (also referred as trustor and trustee, respectively) for the fulfilment of a specific goal should be examined. Therefore, trust on one actor is not the same for all the goals that the actor is assigned to fulfill, but it varies depending on the specific goal [65]. The same actor can be trusted for one goal to be fulfilled, but they may not be trusted for another goal. Such a goal can be to satisfy a security or a privacy requirement. The role, therefore, of this view is to support the software designer in analysing methodically and in detail on what type of trust the decision, of whether to trust an actor to satisfy a security or a privacy requirement, was made. In previous work [63], four types of trust have been identified within the context of sociotechincal systems. The first type is named experiential trust and originates from previous experience of the depender with the dependee, i.e. the dependee has repeatedly fulfilled the same or similar goal in the past. The second type of trust is the reported trust which originates from a third party who reports that the dependee is trustworthee. Next, the normative trust is a type of trust that originates from the system environment norm, e.g., a set of law or regulations that enforce the dependee to fulfil its goal. The last type of trust is the external trust, which originates from sources outside the environment of the system, e.g., governments or other authorities. These four types of trust represent direct trust relationships and are able to resolve dependencies between a depender and a dependee for the fulfilment of a security or a privacy requirement. In cases where there is no trust, the software designer has to choose a solution where the system should be implemented in a way to force the fulfilment of the security or privacy requirements.

Procedural View. Ultimately, goals are operationalised by defining sets of instructions to be executed in order to achieve such goals. In particular, in requirement engineering, goals are frequently operationalised using business processes. In the literature there are many methods for defining business processes with security aspects [75], however, as far as our knowledge goes, the most expressive, in terms of security aspects that can be expressed, and the most complete, in terms of security constraints, is SecBPMN2 [75]. It consists of: (i) SecBPMN2-ml, a modelling language that allows to specify business processes, security and privacy properties on such processes; (ii) SecBPMN2-Q, a modelling language that permits to specify security and privacy patterns (referred as security policies in SecBPMN2); (iii) a software that automates the analysis of security policies against business processes. Using SecBPMN2, designers and security experts can specify eleven different security concepts such as confidentiality, availability and integrity, and more specific ones such as non-repudiation. We selected this method and its modelling languages for the definition of business process in SePTA.

4.3 Process Overview

The modelling capabilities of the modelling languages we mentioned above allow designers to elicit the requirements of a sociotechnical system and build gradually its architecture with focus on security, privacy and trust. Each view of the aforementioned tools examine the system-to-be from a different angle and a different level of detail. From a technical point of view, all diagrams are stored in a database from where each tool can retrieve and process them. The external storage of the

[3] https://capec.mitre.org/

[4] https://www.owasp.org/index.php/Main_Page

[5] https://cve.mitre.org/

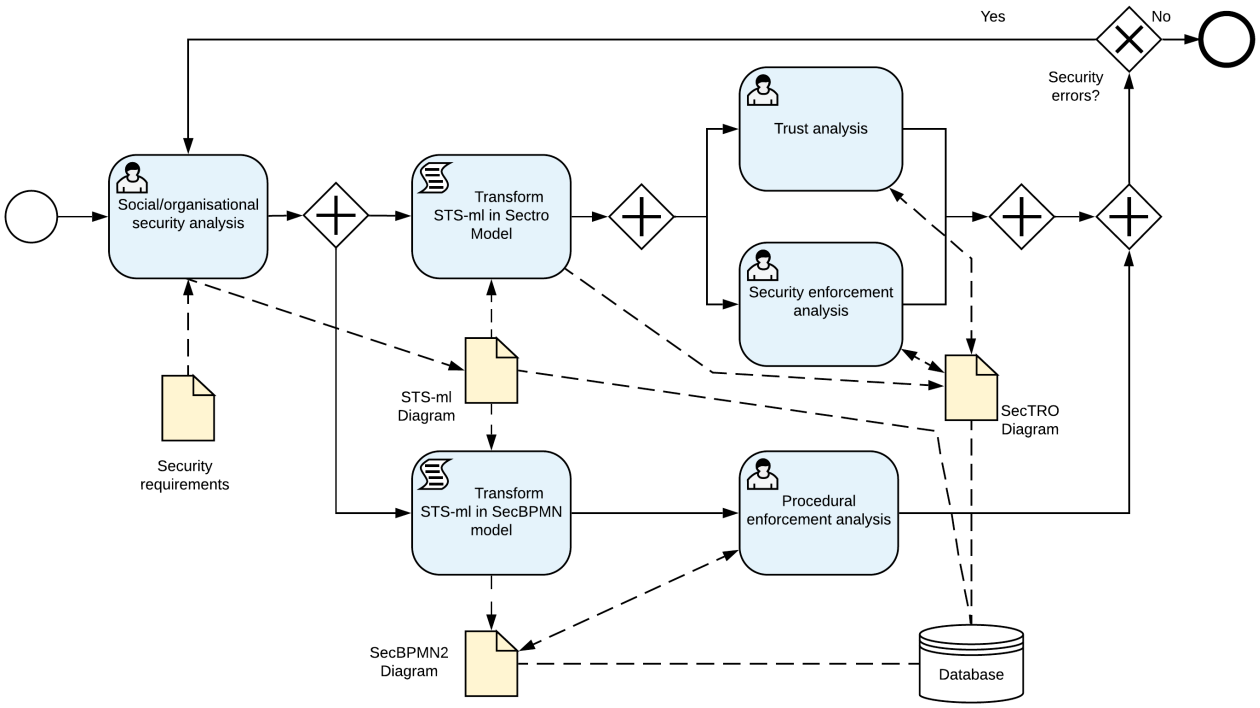

Fig. 1 Systematic approach proposed in this paper

diagrams facilitates the exchange of information among the tools.

Figure 1 shows the process that SePTA user will execute to perform the analyses. The first step is the definition of the analysis of early requirements. The first modelling language to be used is STS-ml, for the "Social/organisational analysis". Using this modelling language, relevant autonomous components of the analysed system are specified as actors and their objectives as goals. The interdependencies between actors are defined with delegations of goals and transmissions of documents. STS-ml is focused on privacy and security needs of each actor, therefore, it allows to specify security and privacy requirements in the interactions between actors. STS-Tool can be used to check automatically the well-formedness of the diagram and if there are inconsistencies in the security and privacy specification. Once software designers and security experts are satisfied with the level of privacy and security specified, they can export the diagram in the database, in order to make it available to other tools.

Once the model from STS is complete and stored to the database, the early security requirements phase is considered finished and the late security requirements phase starts. In this phase, the validation of the trustworthiness and security of the system-to-be is performed, along with the analysis of procedural aspects.

For what concerns the trustworthiness and security analyses, a SecTro diagram is generated by the software that supports SePTA. More specifically, the diagram is imported, from the shared database, in XML format and SecTro parses it to detect all the elements that exist in the diagram. From the STS diagram are imported the actors, their goals and dependencies as well as the resources (documents and information) that are produced and exchanged within the system-to-be. Such elements constitute the initial SecTRO diagram that is generated by the SePTA software.

After the generation of the SecTro model, the user performs the "Trust analysis" and the "Security analysis". The purpose of the former analyses consists in examining the trustworthiness of the system-to-be. In particular, the part of the SecTRO model dedicated to trust captures what preconditions must hold (referred as entailments) for an actor to be trusted and what counter-measures can be introduced in order to force actors to deliver their mandate. The latter process might introduce new actors in the system-to-be with goals related to the supervision and control of other actors. All in all, the analysis conducted with the use of the Trust View detects the weaknesses of the design of the system-to-be and proposes solutions to them.

The design process continues with the "Security analysis". Note that Trust View and Security Require-

ments View are synchronised. This means that every time a new actor, goal or resource is added in one view, so is to the other, decreasing the time required for the design process. This allows the trust analysis and the security analysis to be executed in parallel. In addition to the concepts we mentioned above, every security and privacy requirement specified in STS is transformed into a constraint. The role of the designers in this step of the process is twofold. First, to identify threats that could harm the system-to-be or prevent it from fulfilling its mandate and second, to propose defence mechanisms to protect the system. This is achieved with the use of the Attack View.

After the "Social/organisational analysis", the user performs also a "Procedural enforcement analysis". Such analysis is executed in parallel with the trust and security analysis, since the diagrams used by the two analyses are independent. STS-Tool integrates STS-ml and SecBPMN2 modelling languages, supporting a smooth transition between the STS-ml and SecBPMN2 diagrams. Software designers and security experts, as a first step, use STS-Tool to automatically generate business process diagrams from STS-ml goals and delegations ("Transform STS-ml in SecBPMN2 diagram" activity in Figure 1). The generated SecBPMN2 diagrams contain a process for each goal of the STS-ml diagram and one data object for each document of the STS-ml model. Delegation and transmission specified in STS-ml model are transformed in SecBPMN2 processes, as specified in [73].

The generated business processes will be enriched by the users of SePTA with SecBPMN2 elements ("Transform STS-ml in SecBPMN2 diagram" activity in Figure 1) and perform the procedural enforcement analysis ("Procedural enforcement analysis" activity in Figure 1). The two transformation activities in Figure 1 are partially automated, while the rest of the process is executed by a team composed by security and privacy experts.

The process defined in this section is iterative, meaning that it permits to revise the diagrams multiple times. The creation of a diagram may let the security experts to discover new security issues or new aspects of the system-to-be, that should be included in other models too.

Overall, STS-ml is useful for understanding and capturing the composition of the system-to-be and identifying its security and privacy requirements in terms of its organisational structure. Then, Jtrust and SecTro are used to validate the trustworthiness of the system-to-be, examine in depth the security and privacy vulnerabilities and propose mechanisms to enhance the system's security and privacy. Last, the objectives of the system-to-be and their security aspects are enacted and enforced in business processes using SecBPMN2.

Within each modelling language of the SEPTA method, the use of the various relationships between the entities asserts that one entity has some bearing on the other entity. This enables forward and backward traceabilty. In terms of the ability to follow the life of a specific entity across the whole method in both a forward and backwards direction, there is currently no automated support, but it can be achieved with manual intervention by cross referencing the related entities.

4.4 Software support

SePTA is provided with a software framework[6] which helps the users in performing security, privacy and trust analyses. The software integrates graphical editors for all the modelling languages provided, and performs automated analysis to check the well formdness of the models and security, privacy and trust properties. For more information on the analyses please refer to the specific documentation of STS [15], SecBPMN2 [75], Tropos [53] and JTrust [63].

The integration of STS-tool and SecTro was a challenging process, both from a conceptual and from a technical perspective. One of the motivations of this work was to allow software designers to analyse a system-to-be from multiple angles with the minimum effort possible. Therefore, having to redesign all common concepts for the modelling language would have a drawback. The advantage of combining these two tools is that both are built using the Eclipse Modelling Framework [80] (EMF). This means that both tools are implemented on the top of a metamodel that describes the concepts and the relationships among them, included in the modelling language that the tool implements. Hence, using the Java API provided by EMF and the concept mapping we presented in this section, we implemented a transformation mechanism inside SecTro that receives in XML form the STS-ml diagram and creates a Secure Tropos one. Finally, to support collaborative work, the exchange and storage of diagrams is conducted using a MongoDB database. This allows multiple stakeholders to have access, modify and provide feedback to the models they produced.

The software tool is not limited to an integration of the software, but it permits the usage of one diagram, for example an STS-ml diagram, as basis for other diagrams, for example SecBPMN2, Secure Tropos and JTrust. This reduces the effort that the users of SePTA

[6] http://salnitri.faculty.polimi.it/?page_id=327

will have to dedicate to the analysis and, therefore, will reduce the probability of errors in the models.

5 Case Study

We evaluated SePTA using a case study approach since our objective consists of examining the performance of the method and, therefore, the fulfilment of its requirements in a real world context of use [96]. We believe it is central, for such type of method, to be evaluated using complex scenarios that are common in real world sociotechnical systems.

In particular, we opted for a holistic case study with multiple scenarios [96]. The case study is considered *holistic* since we evaluated the overall method; with *multiple scenarios* since we performed it using three scenarios of different domains. Each scenarios target very different organisations: (i) a private company, DAEM S.A.[7], which is an IT company that collaborates with the municipality of Athens in Greece; (ii) a public administration, the Italian ministry for Economy Development (MiSE)[8]; (ii) two hospitals, the Hospital Infantil Niño Jesús[9] and Ospedale Bambino Gesú[10]. All organisations used SePTA to analyse parts of their sociotechnical systems. The decision of selecting scenarios of different domains allowed us to generalise the conclusions of the case study, as it is described in the analysis about the threats to external validity, later in this section.

The case study consisted in an application of the SePTA method to the three scenarios. For each one, we selected subjects (among the available employees of the companies involved) in order to match a typical team composition of experts that will use the framework, i.e. a domain expert, a security expert, a privacy expert, and a trust expert. Due to the low number of subjects involved in the use case, 2/3 subjects per scenario, we opted for a qualitative evaluation of SePTA. We conducted interviews with the objective of testing if the requirements of the method, defined in Section 2.3, were considered fulfilled. The execution was similar for all scenarios: first the subjects followed a training session about the method, after that the subjects worked in team (one for each scenario) and applied SePTA. The authors of the method helped the subjects correcting errors in the models created. There was no time limit on the scenarios execution, neither the scenarios were executed in the same time frame because of difficulties of finding common free slots for all the participants.

[7] http://www.daem.gr/en/

[8] https://www.sviluppoeconomico.gov.it

[9] http://www.madrid.org/hospitalninojesus/

[10] http://www.ospedalebambinogesu.it/home

The results of the empirical experiment hold when SePTA is applied on systems that correspond on the one defined in this paper as target of the method, i.e. large systems with social interactions between autonomous components (sociotechnical systems). Consequently, we cannot grant the same performance of the method if it is applied on a system with different characteristics. Similarly, the main objective of SePTA consists in analysing security, privacy and trust, therefore, such objectives were considered in the empirical experiment. Hence, if the method is used for other purposes, the results of the empirical experiment might not hold. In other words, if the SePTA will be used respecting the limitations described in the paper, we have a strong confidence that SePTA will perform as well as the experiment demonstrated. Furthermore, we believe that the domain of the analysed system will not influence significantly the performance of SePTA, if the limitations described above are satisfied, since the empirical experiment was performed in three, extremely different, domains and organisations.

In this paper we report only the DAEM scenario, for space issues. For more information on the other scenarios, please refer to the technical report [69].

5.1 The DAEM Scenario

Hereby, we present a real world case study provided by DAEM S.A. The company is responsible for the development and maintenance of the Municipality of Athens Computer Services (MACS), an information system of MoA that stores and manages personal data of the Athenian citizens. The main purpose of MACS is to facilitate the exchange of personal documents issued by public offices and other organisations that provide services exclusively to Athenian citizens. MACS is installed in the premises of the MoA.

The subject of this case study is George, an Athenian citizen, who requests a membership to a sports facility by using his MACS account, where he can also apply for a discount. A sports facility requires a birth certificate from MoA as a proof of residency and a medical certificate from the clinic where the citizen is registered. Copies of these certificates are sent in a digital form to the administrator of the sports facility who is responsible for approving the membership and the discount for George. When George visits the sports facility for the first time, he receives his badge from the receptionist, which he must present every time he needs to access the swimming pool. George is also able to monitor his visiting times at each of the facilities he is a member of. This information can be used by the administration of each sport facility for improving their

quality of service and, as a reward, provide personalised offers to the citizens for sharing their data. This information is recorded every time George enters the facility and uses his badge. Another service offered by MACS allows citizens to provide their bank details in order to set up direct debits for paying their council tax or even their membership to the swimming pool.

The challenge for PAs to design their system is to capture the type of information that needs to be transmitted in order for the goals of the involved entities to be achieved. The PAs also needs to examine how this information is managed by the different entities. For example, they must identify what authorisation each employee has over a citizen's certificate. In particular, what employees are allowed to read the certificates, make copies and to what other entities are legally authorised to redistribute them. This type of processes, apart from compliance issues, raises concerns about trust and security. The PAs must also identify the weaknesses of their system in terms of trust. For instance, what is the trust level to the sports facility for not exploiting the visiting time records or the personal information of the user for other commercial purposes. Furthermore, PAs must identify potential threats to the system's security and design mechanisms to protect its assets. For example, guarantee that the bank details of the citizen are safely stored and protected from cyber attacks. Finally, PAs must verify if defined early requirements are realised, considering the execution of their systems, i.e. they must verify if the executed business processes, enforce the requirements. Traditional requirements elicitation approaches render the design of systems with numerous dependencies. These interactions are considered as a cumbersome process for the PAs. Moreover, to the best of our knowledge, there is no such holistic approach that allows the verification of the enforcement of such requirements in the business processes executed in the system. Therefore, a tool supported method that allows PAs to analyse the aforementioned aspects is a necessity for designing e-government services and sociotechnical systems in general, that are compliant with laws, are secure and trustworthy.

5.2 Applying SePTA

Here we demonstrate how SePTA is applied in the case study we mentioned above. At each step of SePTA we present the corresponding diagram and explain the notation used by the modelling language.

5.2.1 STS-ml

STS-ml fits the purpose of this paper since it allows to specify privacy and security requirements in the interaction of actors of sociotechnical systems, i.e., when goals are delegated and documents are transmitted.

Figures 2, 3 and 4 show examples of the social, information and authorisation views for the DAEM case study. Figure 2 shows an example of the social view. In this example there are seven actors: there are agents, for example George, and roles, for example Clinic. Goals are represented with green squares with rounded corners. For example, Certificates verified is a goal of SP Information System, which specifies that the actor has the objective of verifying certificates. In Figure 2 only the goals of SP Information System are shown, because of space limitations. A goal is assigned to an actor if it is placed within its scope, which is represented with a circle that intersects the actor itself. A goal can be linked to one or more documents, in order to specify that the achievement of such goal will imply an operation of the documents. For example, Certificates verified is linked with a read relation to Birth certificate document, which specifies that SP Information System will read the document in order to achieve that goal. There are three types of operation links: (i) read, which specifies that the document is viewed only; (ii) modify, which specifies that the document is also changed; (iii) produce, which specifies that the document is created when the goal is achieved.

Social View. The social view allows to specify AND and OR decompositions of goals, delegations of goals between actors and transmissions of documents. A delegation is represented with an arrow from the depender actor to the delegated goal and from the goal to the dependee actor. For example, Athenian Citizen delegates the goal Medical certificate issued to Clinic. The transmission of a document is specified with a similar representation but with the document in the place of the goal. When a goal is delegated, appears in the target actor scope in dark green. Similarly, when documents are transmitted, they appear in dark grey in the scope of the actor who received them.

In this view, it is possible to specify a set of privacy and security requirements in the delegations and transmissions. Such requirements are represented with small boxes under the relations. For example, in the transmission of bank details from Athenian Citizen to SP Information System, there are two security and privacy requirements: confidentiality, represented with a brown box with the string Con, and integrity, represented with a pink box with the string Int. The former specifies that Athenian Citizen requires that the communication channel where the document is sent denies unauthorised users

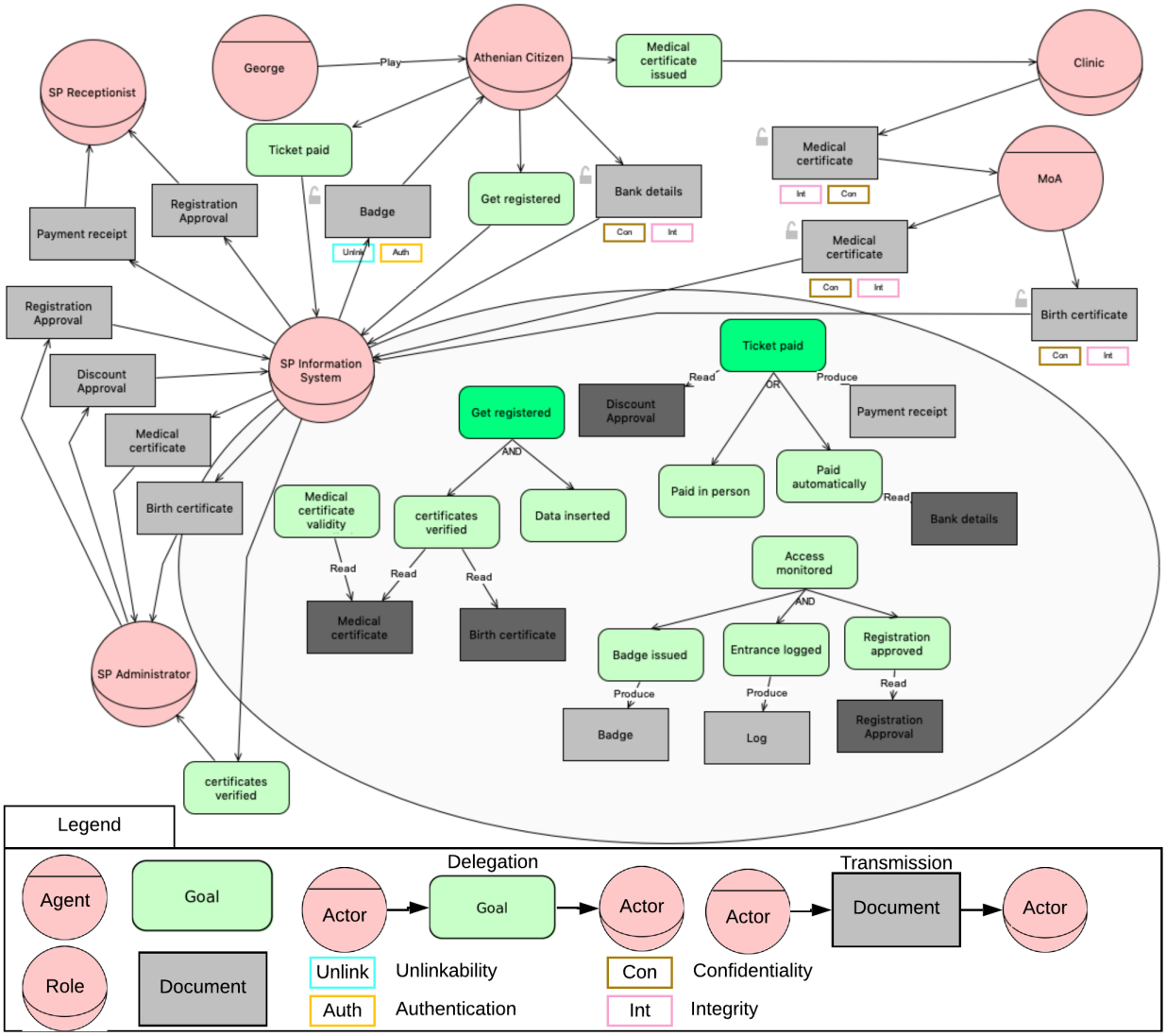

Fig. 2 Example of STS-ml social view

to read/receive the document. The latter specifies that the document transmitted should be received without modifications of its content.

Information View. Figure 3 shows the information view. This view is centred on two elements: information and document. It allows to specify what information is made tangible by a document, i.e. is stored in a document, and what is the owner of such information. For example, Name and surname are two Information elements that are Made Tangible By the documents Medical certificate and Badge. name and surname are owned by the Athenian Citizens.

Authorisation View. Figure 4 shows the authorisation view, where are specified what operations are granted to be executed on information. An authorisation is a relation between two actors and it is composed of three parts: the operations granted (which are Read, Modify, Produce and Transmit), the set of information targeted and for which goals the authorisation is valid. For example, in Figure 4 Athenian Citizen authorises MoA to Read and Transmit (the green tick on the boxes R and T) but not to Modify or Produce (the red cross on the boxes M and P) the information IBAN, Surname, Place of birth, etc., only to reach goal the Certificates provided.

5.2.2 SecTro

Trust View. The SecTro tool imports the social view from the STS-ml model. The next step of the SePTA process prescribes the creation of the Trust view, shown

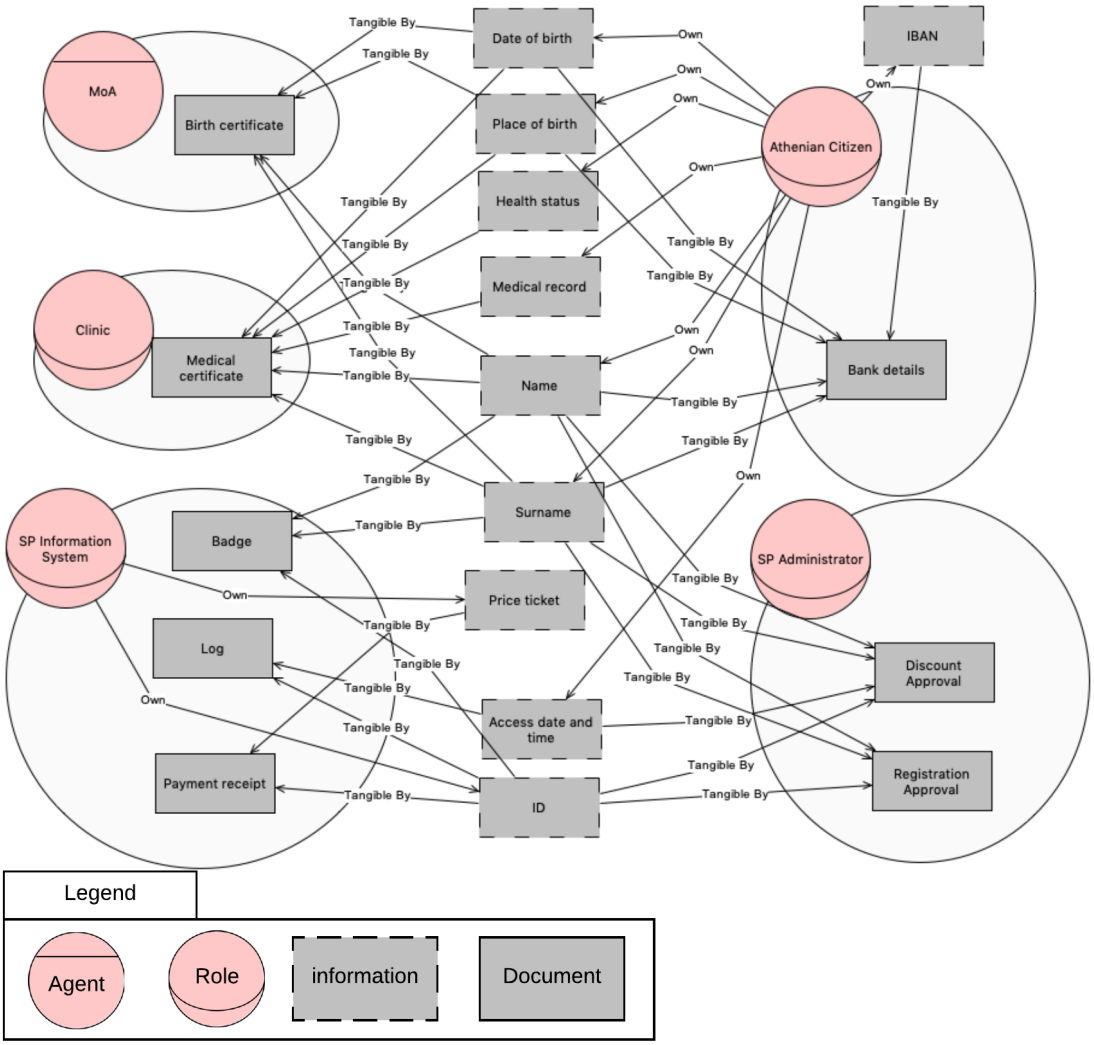

Fig. 3 Example of STS-ml information view

in Figure 5. The actors and their goals, as defined in the STS-ml diagram, are automatically recreated during the import. The goal delegations are converted to Dependencies, which have similar notation in SecTro. For more detailed analysis in SecTro notation, please refer to [19].

For our example, there is a dependency of the Athenian citizen on the Clinic to issue the medical certificate for him. This dependency introduces a potential vulnerability as the Clinic may delay in producing the certificate or, even worse, not produce the certificate at all. Therefore, this dependency has to be resolved either through trust (orange parallelogram) or control (red parallelogram). Because there is no trust on the Clinic to do so, the dependency is resolved through control where the Greek Ministry of Health is the controller and there, there is an assumption that the Ministry can be trusted. As this resolution introduces a new dependency now on the Ministry to control the Clinic, normative trust is identified as a resolution of this dependency, which assumes that there is trust on the social norm. From evidence collected it is shown that it is indeed the social norm the Ministry to control the Clinic and, therefore, the entailment that the ministry can be trusted to control the Clinic is true. So, the Clinic will be controlled to issue the medical certificate once the system is put in operation. A second dependency of the Athenian Citizen is on the Administrator of the Swimming Pool to verify the validity of the medical certificate. Similarly, this dependency introduces another potential vulnerability to the system in the case where the Administrator does not verify the validity or maliciously verifies the validity of the certificate.

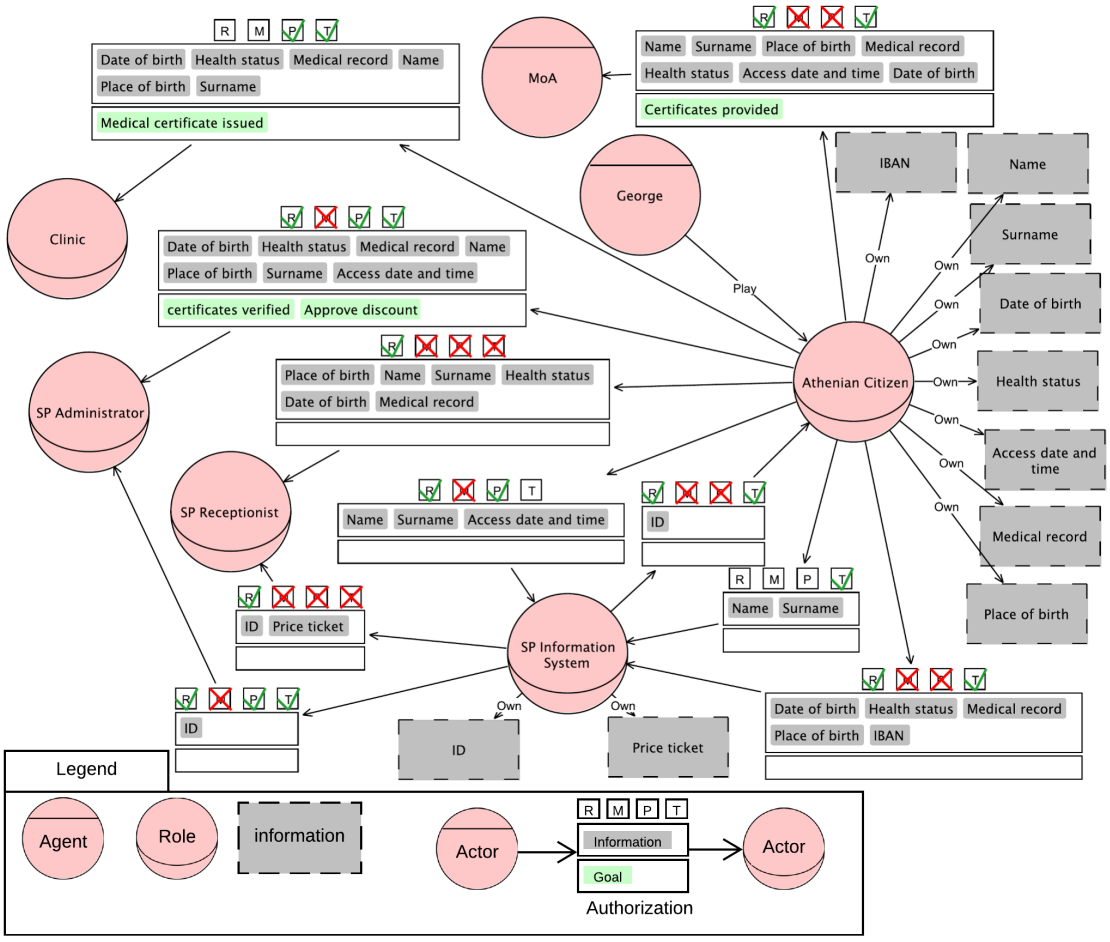

Fig. 4 Example of STS-ml authorisation view

Since there is direct experience with the administrator, experiential trust has been identified as a potential resolution, which assumes the entailment that we can trust our judgement. This entailment that we can trust our judgement for the decision on whether the administrator can be trusted is true, as we have had long direct experience with him in the past.

Security Requirements View. The next step of the SePTA process is to complete the Security Requirements view. From the previous steps, the actors, their goals and their dependencies are maintained in an automated fashion. Moreover, the new actor introduced in the Trust view is also presented along with its goals. In addition to that, the security and privacy requirements defined in STS are converted to Constraints (red hexagons), as shown in Figure 6. Such constraints are the Sender Confidentiality and Sender Integrity, associated with the medical certificate which is created within the Clinic. The designers and the security experts must now identify threats to the system. In this case, three threats are identified. The first threat found is the Man-in-the-middle threat that impacts the medical certificate that is exchanged between MoA and the Clinic and the bank details sent from the Athenian Citizen to the SP Information System. The next threat identified is tampering that impacts the birth certificate created and stored by MoA and the bank details provided by the Athenian Citizen. The last threat is identifiability and impacts the medical certificate that is stored by the SP Information System. In principle, the administrators who check the medical certificates of the applicants shall not be able to identify who is the applicant.

Attacks View. Now that the threats are identified, mechanisms (green hexagons) shall be proposed to protect the system-to-be from any potential harm. To examine threats in further detail, the designers can create the Attacks view for each of them. Due to space limitation, hereby, we present the analysis for the Man-in-the-middle threat, which is shown in Figure 7. In this case, there are two types of vulnerabilities (red el-

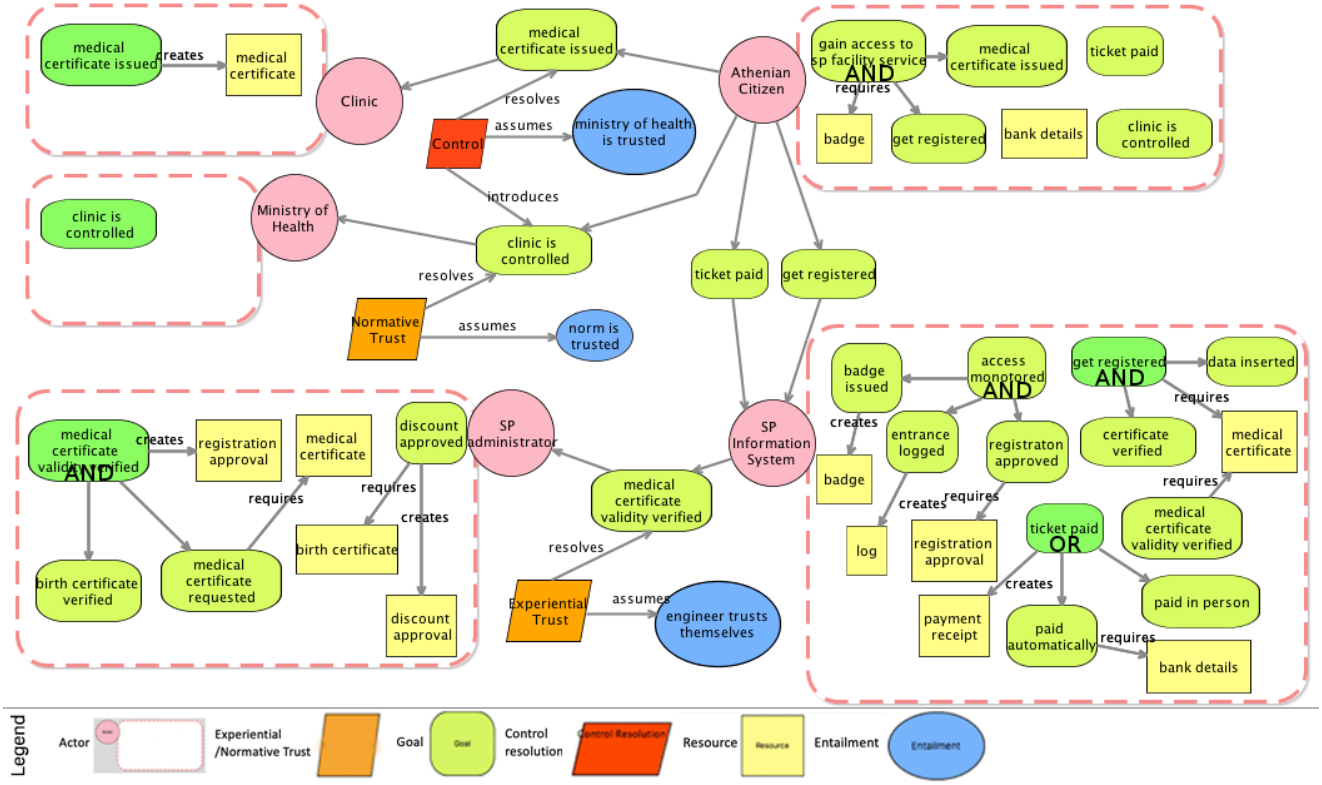

Fig. 5 Example of Trust view

lipse) detected. First, is the insecure communication channel through which the medical certificate is transferred from the Clinic to MoA. Next, the bank details of the Athenian Citizen as well as the medical and birth certificates are stored in a database, which makes them vulnerable to SQL injection attacks (orange hexagon). For each of the vulnerabilities, designers and security experts select appropriate mechanisms to protect the system-to-be for the potential attacks. For example, in the case of the Clinic, Mutual authentication and encryption have been selected as countermeasures to a man-in-the-middle attack.

The mechanisms defined in the Attacks view also appear within the corresponding actor in the Security Requirements view. Then, each mechanism should be associated with the Constraints that satisfies. It is important to mention that the creation of an Attacks view for every threat is not necessary. In particular, for some well known threats, the designers could propose mechanisms in the Security Requirements view directly. More specifically, SecTro offers a pattern library for privacy-related threats such as Identifiability. The designers can look for specific threats and directly import and customise a set of mechanisms that has been proven to work effectively as a countermeasure.

5.2.3 SecBPMN2

SecBPMN2 is built on top of BPMN 2.0 [59]. Figure 8 shows an example of a business process for the transmission of medical certificates, specified using SecBPMN2-ml. In the example, there are two participants, i.e. executors of processes, that are P.MOA and P.SP Information System. Each participant contains a process, which starts with one start event, represented with a green solid circle, and ends with an end event, which is represented with a red solid circle. The basic element of a process is a task, which is an atomic activity. For example, Verify completeness of medical certificate is an activity executed by P.MOA. To specify the sequence of activities executed, the control flow relation is used. It is represented with a black arrow which starts from the precedent element and targets the element executed immediately after. For example, the control flow that connects StartEvent 5 with Verify completeness of medical certificate specifies that, immediately after the start of the business process, the activity is executed. The control flow of a business process can be split with two types of gateways: (i) parallel gateways, represented with a yellow diamond with an "+", that specifies that all outgoing control flows are executed at the same time; (ii) Exclusive gateways, represented with a yellow diamond with

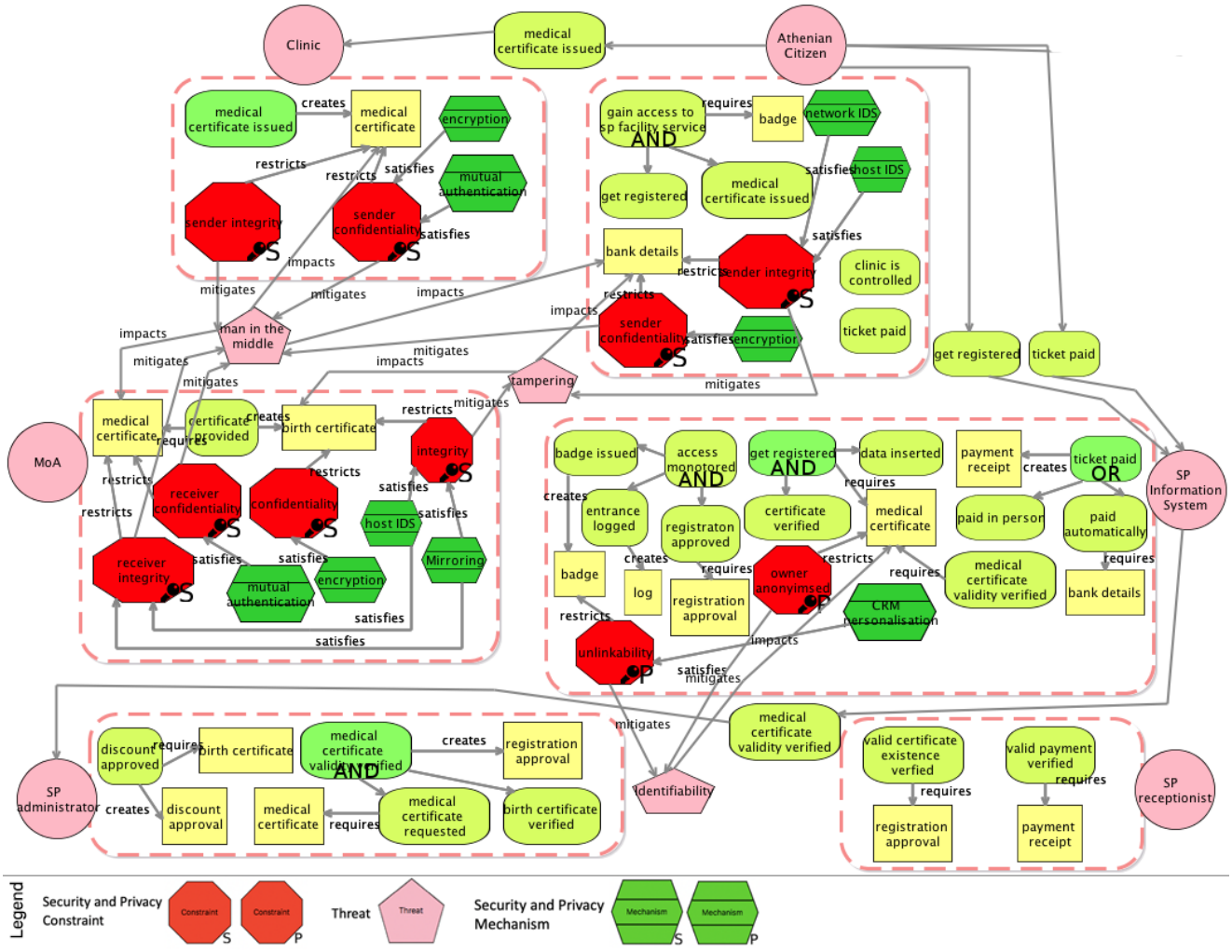

Fig. 6 Example of Security Requirements view

an "X", that specifies that only one outgoing control flow is executed. For example, only one activity between Identify missing information and Send medical certificate is executed, based on the outcome of the Complete? question: if it is Yes then the lower control flow is executed, otherwise the upper control flow is executed. Data objects represent documents that are read or written by activities or events. For example, Medical certificate is written (represented by an arrow from the activity/event to the data object) by the event StartEvent5 and read (represented by an arrow from the data object to activity/event) by Verify completeness of medical certificate. Document can be sent using message flows and messages. For example, the message M.Medical certificate is sent from P.MOA to P.SP Information System.

SecBPMN2-ml extends BPMN2.0 with security annotations. Such annotations are represented with an orange solid circle, with different icons that specify security and privacy concepts. In Figure 8, four security annotations are used. Linked to the activity Produce missing information there are the annotations No Delegation, which means that the entire activity must be executed by the specified participant, and No repudiation, which imposes to keep a legal proof of execution of the activity. Linked to the data object there is the integrity security annotation, which specifies that the document should be protected from intentional corruption. There is a confidentiality annotation linked to the message flow which specifies that only authorised people can receive and send messages on that channel. For more information on SecBPMN2 security annotations please refer to [75].

SecBPMN2-Q is a modelling language, based on SecBPMN2-ml, that permits to specify security and privacy policies, i.e. constraints on SecBPMN2-ml business processes. For further information on SecBPMN2-Q please refer to [75].

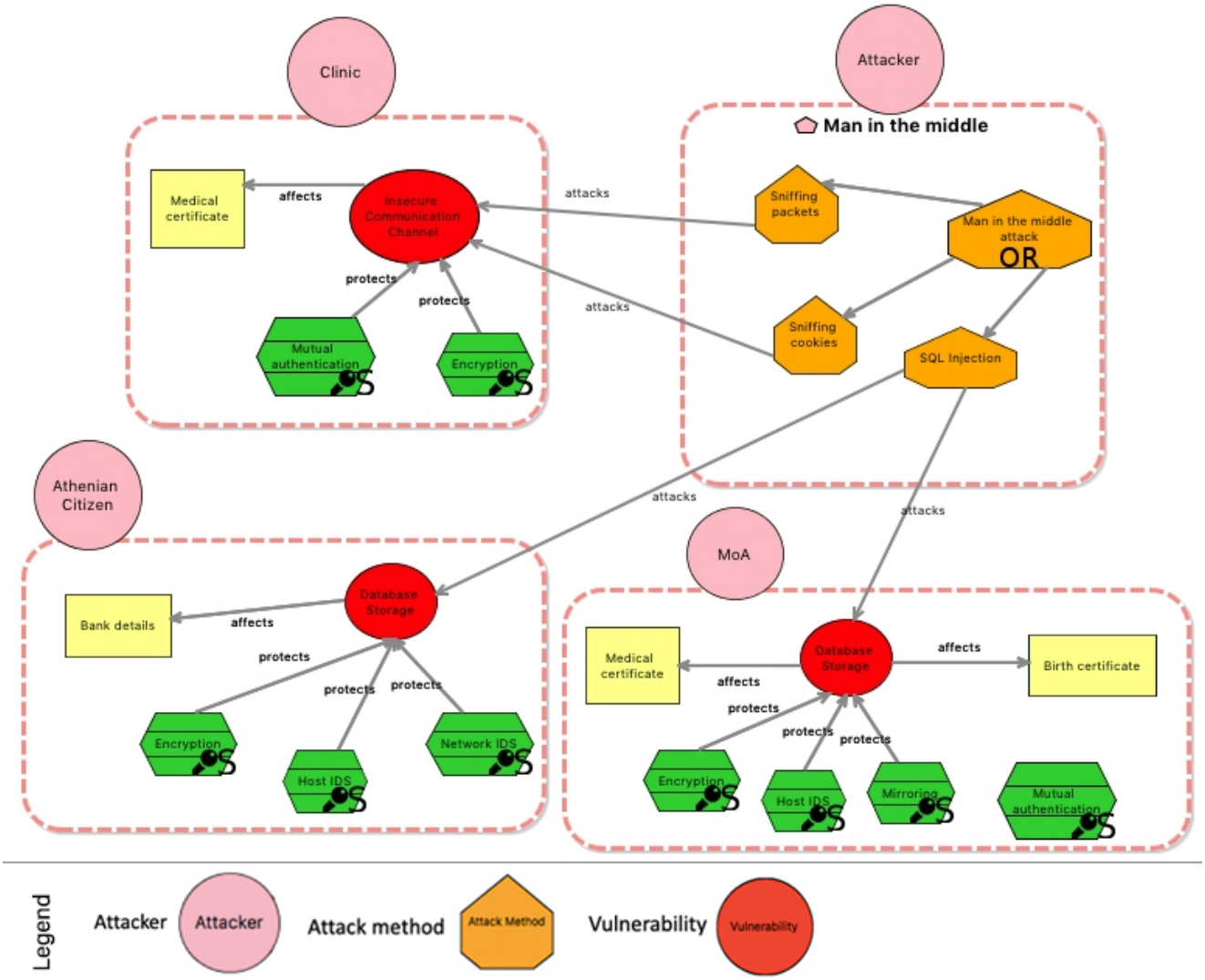

Fig. 7 Example of Attacks view

SecBPMN2 is integrated in SePTA since skeletons of business processes are generated from STS-ml models in an automated fashion, while security policies are automatically generated from security requirements of STS-ml [74].

5.3 Threats to validity

We analyse the validity of the case study following the approach described by Yin [91], where four types of tests are specified. Here, we describe each of these tests and how we mitigated them.

5.3.1 Construct validity

Construct validity consists in "establishing correct operational measures for the concept being studied" [91]. Yin in [91] suggests to cover two steps:

1. select specific types of changes;
2. demonstrate that the selected measures of the changes do reflect the specific changes.

For what concerns the first step, we clearly defined the requirements of the method, that are the criteria to be evaluated.

The second step is a critical point: since we interviewed the subjects who performed the case study, we did not perform any quantitative analysis, because of the low statistical power that the analysis would have had (2/3 subjects per case study). We indeed preferred to retrieve as much information as possible with interviews, due to the high effort put in the integration of SePTA in real word case study: the application of SePTA on parts of real sociotechnical systems lasted a few months. However, since we had specific requirements, we structured the interviews in order to check if the requirements of the method were satisfied.

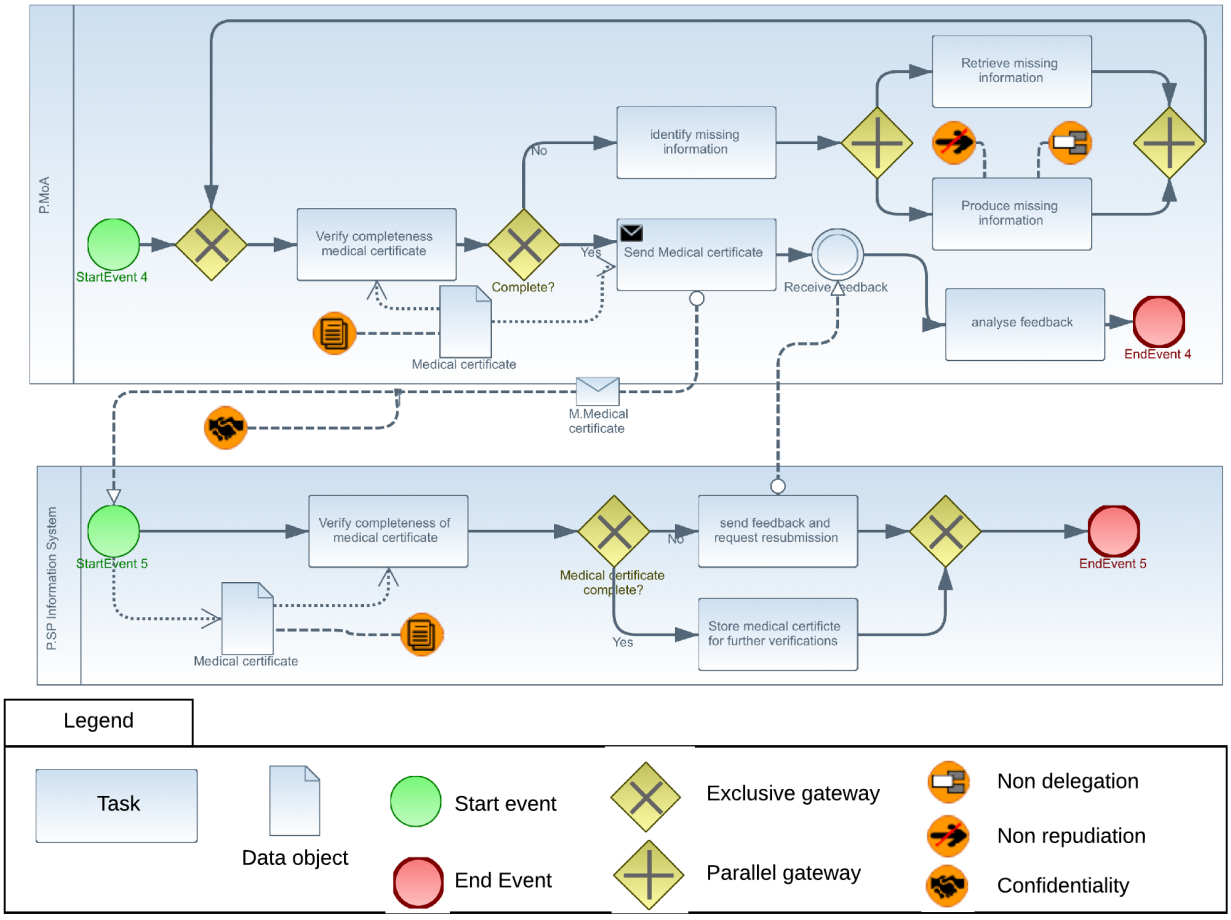

Fig. 8 Example of SecBPMN2

5.3.2 Internal validity

Internal validity is about the correct deduction of the results starting from the original data. In other words, being sure that the treatment causes the changes that are observed [90]. In order to mitigate the threats to internal validity we selected scenarios where security, privacy and trust were central issues, and new functionalities had to be introduced in already present (and stable from the point of view of the development) sociotechnical systems. This permitted to avoid interference that may has be generated from new systems. This allowed the subjects to be focused only in the parts of the sociotechnical systems to be analysed with SePTA. Moreover, we have been careful to give the subjects a software as stable as possible, in order to avoid them being (negatively) influenced by software issues. Furthermore, subjects were highly motivated and we had no mortality, i.e. no subject left, during the execution of the case study.

5.3.3 External validity

External validity deals with the problem of "knowing whether a study's findings are generalisable beyond the immediate case study" [91]. This is an inherent weakness of using a case study research method since, inevitably, case studies are focused on their specific domain. In order to mitigate such threat we chose to perform three scenarios in three very different domains: one in an IT company (DAEM), one in a Ministry and one in two hospitals. The conclusions were conducted considering the results of the case studies and, therefore, eliminating issues and interference specific of single domains.

Moreover, we believe that in complex scenarios as the one targeted by SePTA, in-vitro empirical experiments are not a valid option, since they cannot reproduce the complexity of real sociotechnical systems.

5.3.4 Reliability

A case study is reliable if "a later investigator followed exactly the same procedures as described by an earlier investigator and conducted the same case study all over again, the later investigator should arrive at the same findings and conclusions" [91]. The DAEM case study has been detailed in this paper, moreover the three scenarios are described, in detail, in a technical report [69]. The reproducibility of the case study is granted by the diagrams that were defined by the subject and reported in this paper and in the technical report [69]. Another research can examine the diagrams and reach the same conclusions. Unfortunately, the nature of sociotechnical systems is so complex and dynamic that it will not be possible to recreate the same system and perform again the case study. However, we are confident that results will be similar if SePTA will be applied to other sociotechnical systems, since we applied the method to three, very different, scenarios.

5.4 Discussion

SePTA enabled software designers and security and privacy experts to analyse the requirements of sociotechnical systems from multiple viewpoints. Thanks to the inclusion of stakeholders in the process for the design of SePTA, we had several workshops that led to accurate feedback from the users. This allowed us to, possibly, correct the method during its creation and, therefore, to create a method that satisfies the requirements of its target users, by design. The users included in the design process were not included in the evaluation of SePTA, in order to avoid biased answers.

In particular, for what concerns the *requirements (i)* defined in Section 2.3, the method captures the organisational setting of the system-to-be by identifying the stakeholders involved in it and what are their goals. Software designers captured the data exchanged within the system-to-be and modelled in detail the information included in this data. This part of the design process revealed the security and privacy weaknesses of the system's design that required further attention. At the late requirements phase, software designers along with the security and privacy experts analysed in detail the security and privacy vulnerabilities of the system-to-be and proposed mechanisms to guarantee that such threats will not put the system's operation at risk. The aforementioned two steps are fulfilled with the use of both STS-tool and SecTro, where the first enables the elicitation of security and privacy goals and specifies their operationalisation using secure business processes, whereas the latter supports an in depth analysis of the system's security and privacy vulnerabilities, providing the means to security experts to propose appropriate defence mechanisms.

The *requirement (ii)* defined in Section 2.3, is satisfied using the Trust View of SecTro, which supports the analysis of the trustworthiness of the dependencies among actors, satisfying the third requirement of the PAs.

We consider *requirement (iii)* satisfied since the users of the approach can easily switch models to analyse different perspectives of the same system, using the transformation rules defined in this paper. Indeed, following an iterative process, the user can easily analyse the system from a social/organisational, security, privacy and trust perspective, from the abstraction level of goal-model and business process.

Requirement (iv) is fulfilled by the software we created. We maintained the functionalities of the original software we integrated, in order to provide the verification mechanisms already implemented.

Requirement (v) is fulfilled since subjects, during the interviews, highlighted both the importance of the holistic perspective offered by the approach proposed in this paper and the usefulness for the sociotechnical systems they analysed.

Putting in practice SePTA was a cumbersome process. The main difficulty was to train the users of the tools to use each modelling language correctly. Due to the rich notation and, in some cases, the lack of expertise in conceptual modelling, the users found initially hard to design their systems. More particularly, initial efforts resulted in not fully correct models, since users, although they possessed domain knowledge, however they didn't know how to represent it in the models. However, after performing a training session for each modelling language, the users, with the consultancy of the tool-owners, managed to design their systems using SePTA correctly. In terms of the understandability of the models, the tools have the functionality to provide exploration mechanisms to navigate through the models, instead of navigating one model. This is achieved through the existence of various views of the models that allow users to search and focus on information of interest and deal with the complexity of the requirements models. This feature also supports scalability of the models [19], as large size models with many nodes are decomposed to smaller more manageable views. Obviously, the provision of training and consultancy by the tool owners is not a cost-effective way to develop models, but once users were trained, they were able to build meaningful models that could substantially inform the design of their systems. The users evaluated SePTA in the scope of the case study, admitting that the pro-

vided training was effective and the tools assisted them to improve their design [87].

In a more general approach for the evaluation of the tools in the context of the presented case study, the results of which have been presented in [20], the method for the evaluation followed a systematic approach, combining on one hand the objectives the users set, i.e. the improvement of users' security, privacy and trust when they use PAs services, and on the other hand, the requirements related to the design of SePTA method. Each requirement could be technical, and therefore its fulfilment could be verified by checking the existence of the functionality(ies) implemented, or could refer to human perspective, such as usability of the software. The users evaluated the whole process of the identification of security, privacy, and trust requirements as easy, even though they were not familiar with such tools and methodologies at the start of the project. They mentioned some difficulties during their first interaction with the tools, the effort they put in order to integrate SePTA in their already existing platforms, and in understanding the functionalities of each tool. However, they realised the importance of such a method that should support the services they deliver, since they can declare they take into consideration citizens' privacy, by securing their sociotechnical systems, respecting the privacy requirements, and by analysing the trust relationships they have developed with external entities. Finally, a remarkable comment was that the users considered SePTA as a method that could act as an additional trusted layer between the existing services that the MoA provides to the citizens of Athens, acting as a facilitator between a public body and a citizen, increasing trust to offered public services.

6 Related Work

There are various different lines of research that are focusing on trust modelling during the requirements analysis stage. In [28] the authors extend the Tropos methodology [10] with the concepts of ownership, trust, delegation, permission, and monitoring. The concept of trust is introduced in order to capture the existence or non-existence of trust in the cases of delegation, since sometimes actors delegate goals to actors that they do not trust as long as there are ways to hold such dependees accountable. Similarly to delegation, there is trust of permission and trust of execution. In the first case the depender trusts that the dependee will not go beyond the achievement of the goal, while in the second case the depender trusts that the dependee will at least achieve the goal for him/her. Furthermore, in cases where there is no trust, the concept of monitoring is introduced. The act of monitoring can be done by the delegator himself/herself or by another actor who plays the role of monitor in order to check the violation of trust. Through this way, the developer can capture trust relationships in a normal functional requirements model. However, compared to our proposed approach, this work does not support the software engineer in modelling and reasoning why there is trust and also how the dependency on the monitoring entity can be resolved.

The work in [8] extends the Secure Tropos [54] methodology with the concepts of request, action, trust relationship, trusting intention, reputative knowledge, recommendation and consequence in order to model trust. A trust relationship indicates that one actor expects another actor to behave in a certain way. Trusting intention is defined as the intent of the trustor on how far he/she actually trusts the trustee to carry out the action to her request. Reputative knowledge is the knowledge that the trustor has about the trustee. Recommendation is the representation in favour of another actor while consequence is the effect of a trust relationship. The developer is guided through a series of models, using the aforementioned concepts, in order to analyse and reason about trust relationships. Although this approach provides a mechanism for reasoning about trust relationships, it fails to capture indirect relationships as well. In addition, there are no constructive techniques to guide the design of the system in case of luck of trust in a dependency.

Another approach [21] proposes a method for discovering trade-offs that trust relationships bring between trust and control. The method contains seven steps: identification of actors and their dependencies; modelling and reasoning of actors' goals; modelling trust relationships; recording trust rationale; replacement of the trustee party with a malicious party; analysis of vulnerabilities; and analysis of the trade-offs. In this approach the trust modelling techniques from i* [92] and [28] are adopted and the trust rationale is captured as a belief from the viewpoint of the depender in order to reveal implicit trust assumptions. Then, the dependee entity is replaced with a potential malicious entity that has the same access and capabilities as the legitimate entity. As a result, the developer can then model and analyse the vulnerabilities and their impact that the potential malicious entity may bring. The analysis is a cost/benefit analysis where the cost is the risks that the malicious entity brings. The aim of the trade-off analysis is to evaluate if the potential vulnerabilities, because of lack of trust in the entities that have been assigned with goals, outweigh the benefits of the dependency relationship. If the potential vulnera-

bilities outweigh the benefits, then the developer can choose an alternative dependee that offers better ratio of benefits and vulnerabilities. The costs and benefits of each alternative dependee are evaluated in terms of satisfaction or denial of top goals of the depender. In this approach, the solution is considered to be the identification of alternative dependee, however, in our approach we believe that control of the dependee can be a solution, and thus we provide the required abstractions that will enable the software engineer to enforce the fulfilment of dependencies in cases where there is not trust.

Similarly to our approach, the aforementioned works focus on modelling trust in social relationships. However, there are other lines of research such as [94], [68], [29] that treat trust as a non-functional requirement of the system. We consider these approaches outside of the scope of our paper.

In addition to trust requirements, there is rich literature on security and privacy requirements. SQUARE (Security Quality Requirements Engineering) methodology [49] is a risk-driven methodology that supports the elicitation, categorisation, prioritisation and inspection of the security requirements through a number of specific steps. It also supports the performance of risk assessment to verify the tolerance of a system against possible threats. The method outputs all the necessary security requirements that are essential for the satisfaction of the security goals of a system. The methodology introduces the concepts of security goal, threat, and risk, but does not consider the assets and the vulnerabilities of a system. All the required security requirements should be identified by the requirements engineering team and the relevant stakeholders.

After the realisation of the importance of privacy during the development of software systems, the authors of SQUARE methodology adapted their approach accordingly to support the elicitation of privacy requirements at the early stages of software development process [7]. The extended framework follows the same steps as the first approach of SQUARE methodology and it also integrates a technique that supports the elicitation and prioritisation of privacy requirements, namely Privacy Requirements Elicitation Technique (PRET) [52]. However, similarly the approach does note consider the vulnerabilities of the system and in addition it majorly depends on a database of privacy requirements that has to be regularly updated.

In [71] the authors propose the Model Oriented Security Requirements Engineering (MOSRE) framework for Web Applications which considers security requirements at the early stages of the development process. It covers all phases of requirements engineering and suggests the specification of the security requirements in addition to the specification of systems requirements. The objectives, stakeholders, and assets of the Web application are identified during the inception phase. The final security requirements are elicited after a sequence of actions that include the identification - categorisation - prioritisation of threats and system vulnerabilities, the risk assessment process, the analysis and modelling, and finally, the categorisation - prioritisation - validation of the final security requirements. However, this approach employs UML diagrams to model threats and doesn't allow modelling of all security concepts together.

Another approach is the Security Requirements Engineering Framework (SREF) [31] which enables the elicitation and analysis of security requirements. This framework includes four stages. Firstly, it identifies functional requirements and afterwards, the security goals. Continuing, it identifies the security requirements of the functional requirements. Each security requirement satisfies one or more security goals. After these steps, the framework verifies if the under examination system satisfies the security requirements. The framework also includes the notion of trust assumptions that can impact the satisfaction of security requirements but does not provide guidelines to further analyse those trust assumptions.

The authors in [1] introduced an asset-based approach for the elicitation of security goals from business process models which are then translated into security requirements. This method follows a sequence of steps. During the first step, an early analysis is performed that identifies the business assets that are valuable and must be protected against security risks. The second step is dedicated to the elicitation of security requirements during the examination of the security risk of business assets. The final stage is the elicitation of security requirements which results to the generation of business rules that satisfy security goals of the system under examination.

A well-known goal-oriented requirements engineering approach, KAOS [83] was introduced for the elaboration of requirements from high level goals. Exceptional agent behaviours, namely obstacles, were responsible for the fulfilment of goals. These obstacles were identified and resolved through the elaboration of scenarios between software and agents, responsible for the production of a reliable system [86,61]. The KAOS methodology has been extended [84] in order to elaborate security requirements as well. The output of this extension is the development of two models. The first model corresponds to the system-to-be, aiming to describe the software and the relations between goals, agents, objects, operations, and requirements. The sec-

ond model, which is regarded as an 'anti-model', captures possible attackers, their goals, as well as system vulnerabilities, in order to elicit potential threats and security requirements for the prevention of these threats. The aforementioned security requirements that the anti-model derives are regarded as countermeasures and are integrated to the first model.

In [23,24] the authors propose the Problem-based Security Requirements Elicitation (PresSuRE) Methodology that facilitates the identification of security needs during requirements analysis of software systems. More specifically, it provides a computer security threat recognition and the development of security requirements. This methodology uses problem diagrams to support the modelling of functional requirements. Firstly, based on its contents, this methodology identifies system's assets accompanied with the rights of authorised entities. Then, it determines possible attackers and their abilities. Based on these steps, PresSuRE generates graphs which depict threats on system's assets. Every functional requirement of each asset is related with possible threats and security requirements.

In the area of privacy requirements, in [17] the authors present LINDDUN, a privacy threat analysis framework which, in its first release, aimed at the elicitation and fulfilment of privacy requirements in software-based systems. The process that LINDDUN follows is that a data flow diagram (DFD) of the system is designed and then the identified privacy threats are related to DFD elements. Privacy threat trees and misuse cases are used for the collection of threat scenarios that might affect the system. Moreover, this methodology supports the elicitation of the final privacy requirements and the selection of appropriate privacy enhancing technologies. The final stage of this methodology is the prioritisation and validation of privacy threats through risk assessment. LINDDUN also provides a map that connects privacy enhancing technologies with each privacy requirement, facilitating thus the system designers to select the most appropriate techniques that are able to satisfy privacy requirements.

Similarly, the Privacy Safeguard (PriS) [40], a privacy requirements engineering methodology, incorporates privacy requirements into the system design process. The authors of PriS aim to cover the gap between system design and implementation phase. PriS considers privacy requirements as organisational goals and through the use of privacy-process patterns in order to describe the impact of privacy goals to the affected organisational processes. The next step is the modelling of the privacy-related organisational processes. These processes aim to support the selection of the system architecture that best satisfies them.

Our approach provides a holistic requirements modelling and analysis approach that includes security, privacy and trust requirements, offering functionalities that minimise the effort of the systems' designers, since they avoid repetition on modelling tasks, while they can analyse the system from different perspectives.

7 Conclusions

In this work we presented a modelling approach, named SePTA, that enables software designers and security and privacy experts to explore multiple aspects of sociotechnical systems in both early and late requirements phases. More specifically, we interviewed stakeholders of large Public Administrations and IT companies. The output of the interviews prescribed that to design software supported services in such environments requires tools that can capture the organisational setting with specific focus on mitigating security and privacy threats as well as enhancing the trust relationships among the components of the system.

Toward this direction, we integrated two CASE tools, namely STS-tool and SecTro where the first implements the STS-ml and SecBPMN2 modelling languages, whereas the second implements Secure Tropos and JTrust. Both tools support multiple views in order to satisfy the requirements provided by the stakeholders of our project. STS-tool is used to identify the strategic goals of the entities involved in the system-to-be, their relationships with each other and what are the security and privacy requirements of the system. On the other hand, SecTro is used to analyse in further detail the security and privacy requirements that have been elicited with the STS-tool and propose mechanisms to mitigate external threats and strengthen relationships among codependent entities of the system.

As future work we plan to include in SePTA an explicit traceability of privacy and security requirements, defined in STS-ml and addressed/enforced in SecTro, JTrust and SecBPMN2. Moreover we will examine the possibilities of making the notation of the modelling languages in SePTA more concise to make it easier to learn for potential users without losing though any of their expressiveness. Additionally, as part of our research agenda, we plan to add more views to increase the level of detail of the final design. More specifically, we will explore the architecture of the system-to-be in a higher level of detail, combining modern agile approaches for its derivation such as the Three-Peaks approach [5] and UMLsec for analysing security and privacy aspects.

Acknowledgements This paper is supported by European Union Horizon 2020 research and innovation programme under grant agreement No 653642, project VisiOn (Visual Privacy Management in User Centric Open environments).

References

1. Naved Ahmed and Raimundas Matulevicius. A method for eliciting security requirements from the business process models. In *CAiSE (Forum/Doctoral Consortium)*, pages 57–64, 2014.
2. Z Al-Adawi, S Yousafzai, and J Pallister. Conceptual model of citizen adoption of e-government. In *The Second International Conference on Innovations in Information Technology (IIT05)*, pages 1–10. Citeseer, 2005.
3. Ian Alexander. Misuse cases: Use cases with hostile intent. *IEEE software*, 20(1):58–66, 2003.
4. Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. A goal modeling framework for self-contextualizable software. *BMMDS/EMMSAD*, 9:326–338, 2009.
5. Konstantinos Angelopoulos, Vítor E Silva Souza, and John Mylopoulos. Capturing variability in adaptation spaces: A three-peaks approach. In *International Conference on Conceptual Modeling*, pages 384–398. Springer, 2015.
6. Annie I Anton and Julia B Earp. A requirements taxonomy for reducing web site privacy vulnerabilities. *Requirements engineering*, 9(3):169–185, 2004.
7. Ashwini Bijwe and Nancy R Mead. Adapting the square process for privacy requirements engineering. Technical report, Software Engineering Institute, 2010.
8. Kamaljit Kaur Bimrah. *A Framework for Modelling Trust During Information Systems Development.* PhD thesis, University of East London, 2009.
9. Kurt Bittner. *Use case modeling.* Addison-Wesley Longman Publishing Co., Inc., 2002.
10. Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
11. Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
12. Lemuria Carter and France Bélanger. The utilization of e-government services: citizen trust, innovation and acceptance factors. *Information systems journal*, 15(1):5–25, 2005.
13. Amit K Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos. Reasoning about agents and protocols via goals and commitments. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 457–464. International Foundation for Autonomous Agents and Multiagent Systems, 2010.
14. Lawrence Chung, Brian A Nixon, Eric Yu, and John Mylopoulos. Non-functional requirements in software engineering, 2012.
15. Fabiano Dalpiaz, Elda Paja, and Paolo Giorgini. *Security requirements engineering: designing secure socio-technical systems.* MIT Press, 2016.
16. Anne Dardenne, Axel Van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of computer programming*, 20(1-2):3–50, 1993.
17. Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering*, 16(1):3–32, 2011.
18. Vasiliki Diamantopoulou, Christos Kalloniatis, Stefanos Gritzalis, and Haralambos Mouratidis. *Supporting Privacy by Design Using Privacy Process Patterns*, pages 491–505. Springer International Publishing, Cham, 2017.
19. Vasiliki Diamantopoulou and Haralambos Mouratidis. Applying the physics of notation to the evaluation of a security and privacy requirements engineering methodology. *Information & Computer Security*, 26(4):382–400, 2018.
20. Vasiliki Diamantopoulou and Haralambos Mouratidis. Evaluating a reference architecture for privacy level agreements management. In *12th Mediterranean Conference on Information Systems (MCIS 2018)*. AIS, 2018.
21. Golnaz Elahi and Eric Yu. Trust trade-off analysis for security requirements engineering. In *Requirements Engineering Conference, 2009. RE'09. 17th IEEE International*, pages 243–248. IEEE, 2009.
22. Fred E Emery. Characteristics of socio-technical systems. Technical report, London: Tavistock Institute, 1959.
23. Stephan Faßbender, Maritta Heisel, and Rene Meis. Functional requirements under security pressure. In *Software Paradigm Trends (ICSOFT-PT), 2014 9th International Conference on*, pages 5–16. IEEE, 2014.
24. Stephan Faßbender, Maritta Heisel, and Rene Meis. Problem-based security requirements elicitation and refinement with pressure. In *International Conference on Software Technologies*, pages 311–330. Springer, 2014.
25. David Gefen, Elena Karahanna, and Detmar W Straub. Trust and tam in online shopping: an integrated model. *MIS quarterly*, 27(1):51–90, 2003.
26. Panagiotis Germanakos, Eleni Christodoulou, and George Samaras. A european perspective of e-government presence–where do we stand? the eu-10 case. In *International Conference on Electronic Government*, pages 436–447. Springer, 2007.
27. Mohamad Gharib, Mattia Salnitri, Elda Paja, Paolo Giorgini, Haralambos Mouratidis, Michalis Pavlidis, José F Ruiz, Sandra Fernandez, and Andrea Della Siria. Privacy requirements: Findings and lessons learned in developing a privacy platform. In *Requirements Engineering Conference (RE), 2016 IEEE 24th International*, pages 256–265. IEEE, 2016.
28. P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling security requirements through ownership, permission and delegation. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 167–176, Aug 2005.
29. Janusz Gorski, A Jarzkebowicz, Rafal Leszczyna, Jakub Miler, and Marcin Olszewski. Trust case: Justifying trust in an it solution. *Reliability Engineering & System Safety*, 89(1):33–47, 2005.
30. Sonja Grabner-Kräuter and Ewald A Kaluscha. Empirical research in on-line trust: a review and critical assessment. *International Journal of Human-Computer Studies*, 58(6):783–812, 2003.
31. Charles Haley, Robin Laney, Jonathan Moffett, and Bashar Nuseibeh. Security requirements engineering: A framework for representation and analysis. *IEEE Transactions on Software Engineering*, 34(1):133–153, 2008.

32. Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *Management Information Systems Quarterly*, 28(1):6, 2008.
33. Janine S Hiller and France Belanger. Privacy strategies for electronic government. *E-government*, 200:162–198, 2001.
34. Jennifer Horkoff, Yijun Yu, and SK Eric. Openome: An open-source goal and agent-oriented model drawing and analysis tool. *iStar*, 766:154–156, 2011.
35. Mark Horst, Margôt Kuttschreuter, and Jan M Gutteling. Perceived usefulness, personal experiences, risk perception and trust as determinants of adoption of e-government services in the netherlands. *Computers in Human Behavior*, 23(4):1838–1852, 2007.
36. Siv Hilde Houmb, Shareeful Islam, Eric Knauss, Jan Jürjens, and Kurt Schneider. Eliciting security requirements and tracing them to design: an integration of common criteria, heuristics, and umlsec. *Requirements Engineering*, 15(1):63–93, 2010.
37. Marijn Janssen, Yannis Charalabidis, and Anneke Zuiderwijk. Benefits, adoption barriers and myths of open data and open government. *Information Systems Management*, 29(4):258–268, 2012.
38. Ivan Jureta, John Mylopoulos, and Stephane Faulkner. Revisiting the core ontology and problem in requirements engineering. In *International Requirements Engineering, 2008. RE'08. 16th IEEE*, pages 71–80. IEEE, 2008.
39. Jan Jürjens. Model-based security testing using umlsec. *Electron. Notes Theor. Comput. Sci.*, 220(1), December 2008.
40. Christos Kalloniatis, Evangelia Kavakli, and Stefanos Gritzalis. Addressing privacy requirements in system design: the pris method. *Requirements Engineering*, 13(3):241–255, 2008.
41. Costas Lambrinoudakis, Stefanos Gritzalis, Fredj Dridi, and Günther Pernul. Security requirements for e-government services: a methodological approach for developing a common pki-based security policy. *Computer Communications*, 26(16):1873–1883, 2003.
42. Wen-Shing Lee, Doris L Grosh, Frank A Tillman, and Chang H Lie. Fault tree analysis, methods, and applications a review. *IEEE transactions on reliability*, 34(3):194–203, 1985.
43. Tong Li, Elda Paja, John Mylopoulos, Jennifer Horkoff, and Kristian Beckers. Security attack analysis using attack patterns. In *Research Challenges in Information Science (RCIS), 2016 IEEE Tenth International Conference on*, pages 1–13. IEEE, 2016.
44. Luncheng Lin, Bashar Nuseibeh, Daniel Ince, Michael Jackson, and Jonathan Moffett. Analysing security threats and vulnerabilities using abuse frames. *ETAPS-04*, 2003.
45. Alicia Martínez, Oscar Pastor López, and Hugo Estrada. A pattern language to join early and late requirements. *Journal of Computer Science & Technology*, 5, 2005.
46. Fabio Massacci, John Mylopoulos, and Nicola Zannone. Security requirements engineering: the si* modeling language and the secure tropos methodology. *Advances in Intelligent Information Systems*, pages 147–174, 2010.
47. Sjouke Mauw and Martijn Oostdijk. Foundations of attack trees. In *International Conference on Information Security and Cryptology*, pages 186–198. Springer, 2005.
48. D Harrison McKnight, Vivek Choudhury, and Charles Kacmar. Developing and validating trust measures for e-commerce: An integrative typology. *Information systems research*, 13(3):334–359, 2002.
49. Nancy R Mead and Ted Stehney. *Security quality requirements engineering (SQUARE) methodology*, volume 30. ACM, 2005.
50. Daniel Mellado, Eduardo Fernandez-Medina, and Mario Piattini. Applying a security domain requirements engineering process for software product lines. *IEEE Latin America Transactions*, 6(3):298–305, 2008.
51. Petar Milić, Kristijan Kuk, Turhan Civelek, Brankica Popović, and Stefan Kartunov. The importance of secure access to e-government services. *Communications*, 26(16):1873–18893, 2016.
52. Seiya Miyazaki, Nancy Mead, and Justin Zhan. Computer-aided privacy requirements elicitation technique. In *Asia-Pacific Services Computing Conference, 2008. APSCC'08. IEEE*, pages 367–372, 2008.
53. Haralambos Mouratidis, Nikolaos Argyropoulos, and Shaun Shei. Security requirements engineering for cloud computing: the secure tropos approach. In *Domain-Specific Conceptual Modeling*, pages 357–380. Springer, 2016.
54. Haralambos Mouratidis and Paolo Giorgini. Secure tropos: A security-oriented extension of the tropos methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(02):285–309, 2007.
55. John Mylopoulos, Lawrence Chung, and Eric Yu. From object-oriented to goal-oriented requirements analysis. *Commun. ACM*, 42(1), January 1999.
56. Guido Mllering. The trust/control duality. *International Sociology*, 20(3):283–305, 2005.
57. Chi Mai Nguyen, Roberto Sebastiani, Paolo Giorgini, and John Mylopoulos. Multi-objective reasoning with constrained goal models. *Requirements Engineering*, 2016.
58. Armstrong Nhlabatsi, Arosha Bandara, Shinpei Hayashi, Charles Haley, Jan Jurjens, Haruhiko Kaiya, Atsuto Kubo, Robin Laney, Haralambos Mouratidis, Bashar Nuseibeh, et al. Security patterns: Comparing modeling approaches. In *Software engineering for secure systems: Industrial and research perspectives*, pages 75–111. IGI Global, 2011.
59. OMG. Bpmn 2.0. Technical report, OMG, Jan 2011.
60. OMG. Uml 2.5.1. Technical report, OMG, Dec 2017.
61. Styliani Pachidi. Goal-oriented requirements engineering with kaos, 2009.
62. Elda Paja, Fabiano Dalpiaz, and Paolo Giorgini. Modelling and reasoning about security requirements in sociotechnical systems. *Data & Knowledge Engineering*, 98:123–143, 2015.
63. Michalis Pavlidis, Shareeful Islam, Haralambos Mouratidis, and Paul Kearney. Modeling trust relationships for developing trustworthy information systems. *International Journal of Information System Modeling and Design (IJISMD)*, 5(1):25–48, 2014.
64. Michalis Pavlidis, Haralambos Mouratidis, and Shareeful Islam. Modelling security using trust based concepts. *International Journal of Secure Software Engineering (IJSSE)*, 3(2):36–53, 2012.
65. Michalis Pavlidis, Haralambos Mouratidis, Shareeful Islam, and Paul Kearney. Dealing with trust and control: A meta-model for trustworthy information systems development. In *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*, pages 1–9, May 2012.
66. Paul A Pavlou. Consumer acceptance of electronic commerce: Integrating trust and risk with the technology acceptance model. *International journal of electronic commerce*, 7(3):101–134, 2003.

67. Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. 2010.
68. Stéphane Lo Presti, Michael Butler, Michael Leuschel, and Chris Booth. Holistic trust design of e-services. *Trust in E-Services: Technologies, Practices and Challenges*, pages 113–139, 2006.
69. VisiOn European project consortium. VisiOn Pilots Report - final version. Technical report, VisiOn, 2017. https://www.visioneuproject.eu/wp-content/uploads/2018/11/2017-VSN-RP-145-D5.2-VisiOn-Pilots-Report-final.pdf.
70. James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified modeling language reference manual, the*. Pearson Higher Education, 2004.
71. P Salini and S Kanmani. Model oriented security requirements engineering (mosre) framework for web applications. *Advances in Computing and Information Technology*, pages 341–353, 2013.
72. Mattia Salnitri, Achim D Brucker, and Paolo Giorgini. From secure business process models to secure artifact-centric specifications. In *International Conference on Enterprise, Business-Process and Information Systems Modeling*, pages 246–262. Springer, 2015.
73. Mattia Salnitri and Paolo Giorgini. Transforming socio-technical security requirements in secbpmn security policies. In *iStar*, 2014.
74. Mattia Salnitri, Elda Paja, and Paolo Giorgini. Preserving compliance with security requirements in socio-technical systems. In *Cyber Security and Privacy Forum*, pages 49–61. Springer, Cham, 2014.
75. Mattia Salnitri, Elda Paja, and Paolo Giorgini. Maintaining secure business processes in light of socio-technical systems' evolution. In *Requirements Engineering Conference Workshops (REW), IEEE International*, pages 155–164. IEEE, 2016.
76. Markus Schumacher. *Security Engineering with Patterns: Origins, Theoretical Models, and New Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
77. Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, and Peter Sommerlad. *Security Patterns: Integrating security and systems engineering*. John Wiley & Sons, 2013.
78. Adam Shostack. *Threat modeling: Designing for security*. John Wiley & Sons, 2014.
79. Shirish C Srivastava and Thompson Teo. Citizen trust development for e-government adoption: Case of singapore. *PACIS 2005 Proceedings*, page 59, 2005.
80. Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.
81. Maximilian Teltzrow and Alfred Kobsa. Impacts of user privacy preferences on personalized systems. In *Designing personalized user experiences in eCommerce*, pages 315–332. Springer, 2004.
82. European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, L119/59, May 2016.
83. Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249–262. IEEE, 2001.
84. Axel Van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. In *Proceedings of the 26th International Conference on Software Engineering*, pages 148–157, 2004.
85. Axel Van Lamsweerde. Requirements engineering: From system goals to uml models to software, 2009.
86. Axel Van Lamsweerde and Emmanuel Letier. Handling obstacles in goal-oriented requirements engineering. *IEEE Transactions on software engineering*, 26(10):978–1005, 2000.
87. VisiOn-Consortium. D6.3 training activities manual. Technical report, VisiOn, 2017.
88. Merrill Warkentin, David Gefen, Paul A Pavlou, and Gregory M Rose. Encouraging citizen adoption of e-government by building trust. *Electronic markets*, 12(3):157–162, 2002.
89. Roel Wieringa and Maya Daneva. Six strategies for generalizing software engineering theories. *Science of computer programming*, 101:136–152, 2015.
90. C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An Introduction*. Springer Science & Business Media, 2000.
91. Robert K Yin. *Case study research and applications: Design and methods*. Sage publications, 2017.
92. Eric Yu. *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, 1995.
93. Eric Yu. Modelling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering*, 11:2011, 2011.
94. Eric Yu and Lin Liu. Modelling trust for system design using the i * strategic actors framework. In Rino Falcone, Munindar Singh, and Yao-Hua Tan, editors, *Trust in Cyber-societies*, volume 2246 of *Lecture Notes in Computer Science*, pages 175–194. Springer Berlin Heidelberg, 2001.
95. Eric SK Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 226–235. IEEE, 1997.
96. Zaidah Zainal. Case study as a research method. *Jurnal Kemanusiaan*, 5(1):1–6, 2007.
97. Pamela Zave. Classification of research efforts in requirements engineering. *ACM Computing Surveys (CSUR)*, 29(4):315–321, 1997.