Attribute-Based Security Verification of Business Process Models

Nikolaos Argyropoulos, Haralambos Mouratidis, Andrew Fish
School of Computing, Engineering and Mathematics
University of Brighton, Brighton, UK
Email: {n.argyropoulos, h.mouratidis, andrew.fish}@brighton.ac.uk

***Abstract*—Business processes, as the instruments used by organisations to produce value, need to comply with a number of internally and externally imposed standards and restrictions. Since the majority of such processes involve the exchange of sensitive third party information, their compliance to security constraints needs to be verified before they can be implemented. Current attempts for the verification of security compliance of design-time business process models involve the transformation of both the model and the desired security properties into formal specifications, which can be then used as input for automated model checkers. Such an approach is usually costly both in terms of time and specialised knowledge, while also its coverage can be limited to specific types of security requirements. In this work we introduce an approach for the verification of security in business process models based on structural properties of the workflow of the process. To that end, we introduce a series of attributes to existing BPMN 2.0 concepts and algorithms for checking the compliance of a process model against the most common security requirements. Finally, a real-world business process is used to demonstrate and evaluate the applicability of our proposal.**

***Keywords*—Business Process Security, Security Verification, Business Process Modelling, BPMN**

I. Introduction

Since business processes are designed by humans using a number of different available modelling languages, they can include a certain degree of subjectivity and arbitrariness which may lead to varying interpretations of the same model by different stakeholders. It is, therefore, important that, before a business process model becomes an executable business process, its properties and functionalities are formally verified [1]. One of the most important properties of business process models, in need of verification, is security. Since most business processes revolve around the exchange of information between different participants, the security of such information assets becomes a critical factor for the success of the overall business process. Moreover, the compliance of an organisation's business processes to security standards can be an internally (e.g., organisational policies) and externally (e.g., laws and regulations) imposed requirement [2].

Business Process Compliance Management (BPCM) is the field of study involved with the compliance checking of business processes. Proactive, or forward compliance checking techniques can prevent the actual execution of non-compliant behaviour [3] as they are applied during the design of a business process. Such approaches often use formal specification languages and automated model checking to verify certain properties of a business process model. Due to the importance of security properties of business process models, a number of approaches specialising in security compliance verification have been developed. Nevertheless, as extensively discussed in Section V, since the formal specification of both the process models and their security requirements introduces considerable overhead in specialised knowledge and time, the adoption of such approaches in real life remains rather limited [2], [4].

In this work, we introduce an attribute-based security verification approach for business process models, which aims to provide increased usability and broad coverage for the traditional types of security requirements (authentication, authorisation, confidentiality, integrity, availability). To achieve that, a series of attributes are introduced to existing BPMN 2.0 concepts [5] in order to capture information relevant to the analysis of the security properties of the process model. Using such attributes, conditions that need to apply in a process model for the satisfaction of each type of security requirement are defined. Finally, for each type of security requirement, an algorithm is introduced, for verifying the compliance to such conditions. Our approach is applied to a real-life process from the public administration domain, in order to demonstrate and evaluate its functionality.

The rest of this work is structured as follows, Section II introduces the attributes necessary for the security verification process. Section III presents the process of instantiating the introduced attributes and the algorithms that make use of them to verify different types of security requirements. Section IV presents the application of our approach in a real-life scenario. Finally, related work and its limitations are discussed in Section V, while final remarks and future work directions are discussed in Section VI.

II. Security Related Attributes

The modelling of security related aspects is not natively supported by contemporary graphical process modelling languages such as BPMN [6]. Nevertheless, the ability to reason and verify the security properties of a business process model requires concepts able to capture security related aspects of its elements. To that end, we propose new attributes to be added to concepts of BPMN collaboration diagrams, which will then be used for security verification purposes. A partial metamodel containing the BPMN concepts relevant to our work, along with their newly introduced attributes is presented in Fig. 1.

TABLE I
OVERVIEW OF SECURITY-RELATED ATTRIBUTES

Attribute	**of BPMN concept**	**Type**	**Description**
id	Lane, Activity, Data Object	String	A unique identification text that describes each element of the process model.
authenticated	Lane	Boolean	A flag that if set to TRUE indicates that the lane has been successfully authenticated.
authorisation_level	Lane	Integer	The authorisation level of the lane.
owner	Activity, Data Object	Lane	The lane in which the activity or data object belongs.
source	Activity, Data Object	Activity	The activity or activities the execution of which directly precedes the activity at hand or produces the data object as output.
target	Activity, Data Object	Activity	The activity or activities the execution of which directly succeeds the activity at hand or uses the data object as input.
authentication_required	Activity, Data Object	Boolean	A flag that if set to TRUE indicates that authentication is required for to the execution of the activity or the modification of the data object.
authorisation_required	Activity, Data Object	Integer	The level of authorisation required for the execution of the activity or the modification of the data object.
security_objective	Security Implementing Activity	Enumeration	The type of security objective implemented by the activity.
integrity_required	Data Object	Boolean	A flag that if set to TRUE indicates that the integrity of the data object needs to be checked.
integrity_checked	Data Object	Boolean	A flag that if set to TRUE indicates that the integrity of the data object has been verified.
confidentiality_required	Data Object	Boolean	A flag that if set to TRUE indicates that the confidentiality of the data object needs to be ensured.
secure_channel[Lane]	Data Object	Boolean	A flag that if set to TRUE indicates that a secure channel exists for communicating the data object to Lane.
availability_required	Data Object	Boolean	A flag that if set to TRUE indicates that the availability of the data object needs to be checked.

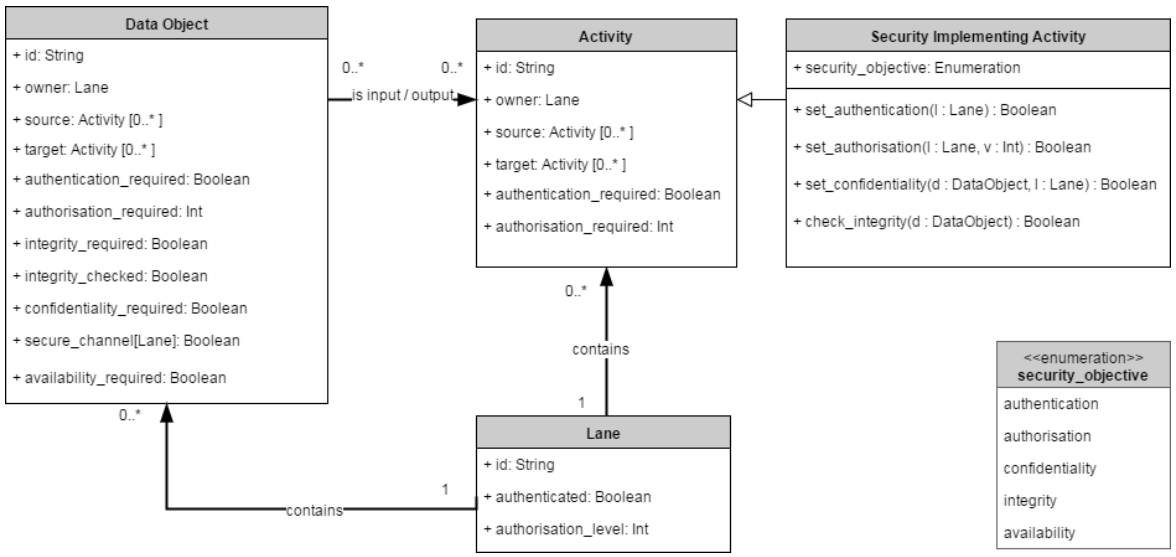

Fig. 1. Partial BPMN metamodel with security-related attributes

The newly introduced attributes, an overview of which is provided in Table I, capture information regarding properties of the business process elements which are essential for the verification of their security. The type of information they capture can be categorised in two groups, workflow related and security related information.

The workflow-related information is captured by the *owner*, *source* and *target* attributes, attached to the concepts of Activity and Data Object. These attributes aim to capture information regarding the position of each instance of activities and data objects within the workflow of a business process model. More specifically, for the concept of Activity, the *owner* attribute indicates the lane of which this activity is part of, thus relating information regarding the entity in charge of the activity's execution. For instance in the example process fragment of Fig. 2, the attribute instantiation *A1.owner* should return the value *L1*, since the activity with id *A1* belongs to the lane *L1*. The *source* and *target* attributes capture the activity executed immediately before and after the execution of the activity at hand, as dictated by the workflow of the business process. An example of the use of such attributes can be shown based on the process fragment of Fig. 2, where for the activity with id *A2* the attribute declaration *A2.source* returns *A1* and *A2.target* returns *A3*. As indicated by the multiplicity of the *source* and *target* attributes of the Activity concept in Fig. 1, there can be no source or target for an activity, in case an event triggers or is triggered by its execution (e.g., start or end events). It can also be the case that multiple sources or targets exist in case of workflow splits or joins due to gateways.

By checking the *owner* attribute of a *source* or *target* of an activity, we can deduce whether the workflow of the process is transferred from one lane to another, which is information of high relevance for the analysis and verification of security properties. Such deductions can be made using complex queries, which combine more than one type of attribute. For instance, in the example of Fig. 2, if the lane where the workflow leads after the execution of activity A2 needs to be identified, the attribute combination *(A2.target).owner* can be utilised. The first part of this declaration (i.e., *A2.target*) returns activity *A3*, and the next part of the declaration (i.e., *A3.owner*) returns *L2* as the lane that contains the activity that follows the execution of A2.

The same applies for the *owner*, *source* and *target* attributes of the Data Object concept, with the only difference being that the source and target represent the activities that create/modify or consume the data object at hand. For instance, *D1.source* in Fig. 2 should return *A3* while *D1.target* should return *A4*.

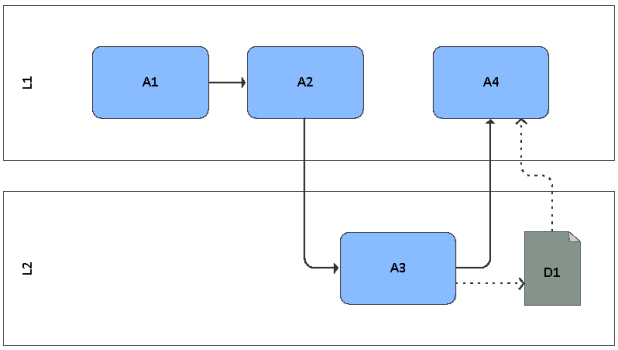

Fig. 2. Example process fragment

The second group of attributes captures security needs and properties of the *Lane*, *Activity* and *Data Object* elements. More specifically, the attributes introduced at the Lane concept indicate whether or not the entity represented by such a lane has been authenticated and what is its level of authorisation. Such properties of a lane are vital for the verification of security properties, as they indicate whether the entity modelled by the lane can access certain activities or data objects. The Data Object concept includes a number

of attributes in order to capture different types of security needs (e.g., *authentication_required, authorisation_required, confidentiality_required* etc.). The attributes relating to the need of authentication and authorisation are also included in the Activity concept. Such attributes are used for identifying which types of security needs must be checked during the security verification. Other than attributes used to capture needs, the Data Object concept also includes attributes for capturing certain security-related properties, such as the existence of secure channels between the data object and a lane. Such properties are an important component of the security verification process, which will be presented in the next section.

Finally, other than the introduction of attributes to existing concepts, we have also introduced a new type of BPMN activity called *Security Implementing Activity*. Such a type of activity is concerned with the operationalisation of security at the process level by the implementation of security mechanisms and countermeasures. The type of security objective fulfilled by each security implementing activity is captured by its *security_objective* attribute, while a set of methods are available for allowing such activities to interact with the attributes of other process elements. The selection of appropriate security mechanisms is considered to be outside the scope of this work and so security implementing activities are considered as "black boxes". The security verification process proposed in this work is, therefore, implementation agnostic and mainly concerned with the effect that the structural properties of a business process model have on the satisfaction of the security requirements of the process.

III. Attribute Instantiation and Security Verification

The attributes presented in the previous section are utilised for the verification of security objectives. The process for the instantiation of such attributes and the algorithm used for the verification of each security objective will be presented in the rest of this section.

1) Authentication: Authentication is defined as the provision of assurance that a claimed characteristic of an entity is correct [7]. In the context of business processes, authentication entails the verification of a credential of a subject using security mechanisms [8]. The subjects of a business process are its participating entities, which can be, among others, individuals or groups of human participants, software systems or organisations. (Swim)lanes are used in BPMN 2.0 as a graphical representation of a participant in a business process model [5]. Therefore, authentication is a security objective associated with the lanes of a business process model.

To capture the authentication property of a process participant, the attribute *authenticated* has been introduced at the Lane concept, as illustrated in Fig. 1. Security implementing activities which operationalise the authentication security objective, as indicated by the value of their *security_objective* attribute, can access the *authenticated* attribute of a lane l and set it to $TRUE$ using their *set_authentication(l)* method. The attribute *authentication_required* has been introduced to the Activity and Data Object concepts to capture whether they require participants to be authenticated before accessing them.

Algorithm 1 defines the steps for the verification of the authentication property of activities and data objects. The procedure AUTHENTICATION_CHECK_A takes an activity as input (line 1) and identifies each of its preceding activities(line 2). If such activities belongs to a different lane that the activity at hand and if that lane is authenticated (line 3), then the authentication constraint of the activity is considered satisfied. Similarly, the procedure AUTHENTICATION_CHECK_DO takes a data object as input (line 9) and, for each of the activities having the data object as input (line 10), checks whether they belong to a different lane than the data object and whether that lane is authenticated (line 11).

2) Authorisation: Authorisation requires the restriction of access to assets based on certain business or security requirements of an entity [7]. In the context of a business process model, authorisation involves a lane, representing the entity that wants to access an asset, the authorisation level of that entity, and the asset itself, which can be either an activity or a data object [8].

A number of attributes have been introduced, as shown in Fig. 1, for the instantiation and checking of the authorisation objective. More specifically, the attribute *authorisation_level* is used for capturing the level of authorisation of each process lane. The attribute *authorisation_required* is used to capture the minimum level of authorisation required by an entity for accessing an activity or data object. Finally, security implementing activities with the *security_objective* attribute set to *authorisation*, perform the *set_authorisation(l, v)* method to set the *authorisation_level* of a lane l to a value v.

In the context of a business process model, authorisation checking, performed using Algorithm 2, involves following the workflow of the process to identify all the entities that interact with the authorisation-constraint process elements. In case of an authorisation-constraint activity, procedure AUTHORISATION_CHECK_A identifies each activity that precedes it in the workflow (line 2). If that activity belongs to a different lane than the constraint activity (line 3) and the authorisation level of that lane is greater or equal to the minimum authorisation level required by the constraint activity (line 4), the authorisation constraint is satisfied. In the case of a data object, a similar authorisation checking process is followed using the procedure AUTHORISATION_CHECK_DO but, in this case, each activity using the data object as input is identified (line 12). If such activity belongs to a lane different than the data object (line 13), then the authorisation level of such lane is compared to the authorisation level required by the constraint data object (line 14) and if it is greater or equal the authorisation constraint is considered satisfied (line 15).

3) Confidentiality: Confidentiality refers to the protection of information from disclosure to unauthorised entities [9]. Therefore, in terms of business process models, confidentiality is a property of a data object, which is the concept BPMN 2.0 utilises to capture information assets. Defining confidentiality

Algorithm 1 Algorithm for authentication checking

```
1: procedure AUTHENTICATION_CHECK_A(Activity)
2:     for all Activity.source do
3:         if Activity.owner ≠ (Activity.source).owner and ((Activity.source).owner).authenticated == TRUE then
4:             return TRUE
5:         end if
6:     end for
7: end procedure
8:
9: procedure AUTHENTICATION_CHECK_DO(DataObject)
10:    for all DataObject.target do
11:        if DataObject.owner ≠ (DataObject.target).owner and ((DataObject.target).owner).authenticated == TRUE then
12:            return TRUE
13:        end if
14:    end for
15: end procedure
```

Algorithm 2 Algorithms for authorisation checking

```
1: procedure AUTHORISATION_CHECK_A(Activity)
2:     for all Activity.source do
3:         if Activity.owner ≠ (Activity.source).owner then
4:             if ((Activity.source).owner).authorisation_level ≥ Activity.authorisation_required then
5:                 return TRUE
6:             end if
7:         end if
8:     end for
9: end procedure
10:
11: procedure AUTHORISATION_CHECK_DO(DataObject)
12:     for all DataObject.target do
13:         if DataObject.owner ≠ (DataObject.target).owner then
14:             if ((DataObject.target).owner).authorisation_level ≥ DataObject.authorisation_required then
15:                 return TRUE
16:             end if
17:         end if
18:     end for
19: end procedure
```

also requires the identification of authorised entities that can access the information [10]. Thus, the concept of a swimlane is, once again, required for the definition of confidentiality in the context of business processes.

A number of attributes have been introduced for reasoning about confidentiality in business process models, as shown in Fig. 1. The attribute *confidentiality_required* introduced in the Data Object concept indicates whether the confidentiality objective has to be met for accessing a data object. The attribute *secure_channel[Lane]*, also introduced in the data object concept, indicates whether a communication channel capable of confidential data transmission exists between the data object and a specific entity, modelled as a lane in the business process. In order to establish confidentiality, appropriate security implementing activities need to be introduced in the business process. To that end, security implementing activities operationalising the confidentiality security objective (i.e., *security_objective* attribute is set to *confidentiality*) have the method *set_confidentiality()*. That method takes as input a confidentiality-constraint data object and a lane and, if a secure connection exists between them, assigns the value $TRUE$ to the *secure_channel[Lane]* attribute of the data object.

Algorithm 3 verifies whether the confidentiality objective of a data object is met by a business process model. The algorithm takes a data object as input and checks all the outgoing workflows sourcing from that data object (line 2). For each outgoing workflow leading to a lane that is different than the one currently owning the data object (line 3), the *authorisation_level* of that lane is compared to the minimum authorisation level required by the data object (*authorisa-*

Algorithm 3 Algorithm for confidentiality checking

```
1: procedure CONFIDENTIALITY_CHECK(DataObject)
2:     for all (DataObject.target).owner do
3:         if DataObject.owner ≠ (DataObject.target).owner then
4:             if ((DataObject.target).owner).authorisation_level ≥ DataObject.authorisation_required then
5:                 if DataObject.secure_channel[(DataObject.target).owner] == TRUE then
6:                     return TRUE
7:                 end if
8:             end if
9:         end if
10:    end for
11: end procedure
```

Algorithm 4 Algorithm for integrity checking

```
1: procedure INTEGRITY_CHECK(DataObject)
2:     for all (DataObject.target).owner do
3:         if DataObject.owner ≠ (DataObject.target).owner and DataObject.integrity_checked == TRUE then
4:             return TRUE
5:         end if
6:     end for
7: end procedure
```

Algorithm 5 Algorithm for availability checking

```
1: procedure AVAILABILITY_CHECK(DataObject)
2:     for all DataObject.target do
3:         if DataObject.owner ≠ (DataObject.target).owner then
4:             if ((DataObject.target).owner).authorisation_level ≥ DataObject.authorisation_required then
5:                 if DataObject.source ≠ IS_UNIQUE then
6:                     return TRUE
7:                 end if
8:             end if
9:         end if
10:    end for
11: end procedure
```

tion_required attribute of data object) (line 4). Finally, the existence of a secure communication channel between any authorised target lane and the data object is checked via the *secure_channel[Lane]* attribute of the data object (line 5). If the attribute has a value of $TRUE$ for each target lane then the confidentiality objective is satisfied.

4) Integrity: Integrity is concerned with ensuring that information is protected from improper modifications so as to avoid intentional or accidental unauthorised changes to system data [9]. Similar to confidentiality, the entities relating to integrity, in terms of business process models, are the data object, which models the data handled during the process execution, and the lane which models the entities exchanging said data.

As shown in Fig. 1, to capture aspects relating to integrity, the *integrity_required* and *integrity_checked* attributes have been introduced in the data object concept. When the *integrity_required* attribute has a $TRUE$ value, an integrity constraint exists on the data object at hand, while if *integrity_checked* attribute is set to $TRUE$ the integrity of the data object has been confirmed by appropriate security mechanisms. The activities modelling the operationalisation of such integrity implementing mechanisms are modelled as security implementing activities with their *security_objective* attribute set to *integrity*. To signify that the integrity checking has been performed, such activities include the method *check_integrity()*, which takes a data object as input and changes the value of its *integrity_checked* attribute to $TRUE$.

For the verification of the integrity objective of data objects in a business process model, Algorithm 4 has been developed. The algorithm takes as input a data object and identifies the lane of each activity that consumes the data object (line 2). If the data object's source lane is different than its target lane (line 3), which indicates that a data transfer has taken place, the *integrity_checked* value of the data object is checked (line 3). If the value is $TRUE$ a successful integrity checking is assumed to have been executed, thus signifying the satisfaction of the integrity objective.

5) Availability: Availability describes the property of system resources being accessible and usable upon demand by an authorised entity [7]. Therefore, in terms of a business process model, a system resource, modelled as a data object, needs to be available to an authorised entity, modelled as a lane. To capture aspects relating to availability, the extended metamodel of Fig. 1 introduces the *availability_required* attribute in the concept of Data Object, which indicates that such an element has an availability constraint placed upon it, if its value equals $TRUE$.

The satisfaction of the availability constraint relates to the structure of the workflow of a process model. Since a data object need to be available upon demand, there is a need for redundancy built into the workflow in order to ensure that there is always more than one ways to reach the availability-constraint process element. This means that an availability-constraint data object, for instance, should be able to be produced as the output of more than one activities. Therefore, to check the satisfaction of an availability-constrain data object we introduce Algorithm 5. This algorithm first checks if each activity requiring the data object (line 2) belongs to a lane different that the owner of the data object (line 3) and whether that lane has the appropriate authorisation for accessing it (line 4). Finally, it checks whether the constraint data object sources from more than one activities (line 5). If a value of $TRUE$ is returned, the availability objective for said data object is assumed satisfied.

IV. Application of Attribute-based Security Verification

To demonstrate how the attribute-based security verification can be applied, a business process from the public administration domain has been selected. The process involves citizens residing in the municipality of Athens, Greece, who want to get registered for using public swimming pool facilities. In order to complete their registration, a number of documents (i.e., medical certificate, birth certificate) issued by different organisations (i.e., local clinic, municipality of Athens) need to be collected and forwarded to the administration of the swimming pool facility. Once the collected documents are verified and the citizen is registered to the swimming pool facility's information system, a badge is provided so the citizen can access the facilities.

A. Attribute instantiation

The process model describing the interaction between the different roles, as illustrated in Fig. 3, was created with the cooperation of DAEM S.A., the organisation in charge of developing the information systems for the municipality of Athens. The same stakeholders also assisted us in the elicitation of the security requirements and the introduction of the appropriate security implementing activities in the process model, denoted with a padlock symbol at their top left corner. Due to space limitations, in this section we will explore only a subset of the elicited requirements in order to demonstrate and evaluate the functionality of our approach. More specifically, the security requirements that will be examined concern the *Confidentiality* and *Integrity* of data objects *D1: Medical Certificate, D2: Birth Certificate* and *D3: Medical certificate [Municipality of Athens Copy]*, while also the *Authentication* required for the execution of activity *A16: "Provide access to pool facilities"*.

The instantiated attributes of the security-constraint data objects D1, D2 and D3 are included in Fig 3. First, the attributes *owner*, *source* and *target* were automatically instantiated according to the structure of the business process model. Next, the attributes *integrity_required* and *confidentiality_required* were manually set to $TRUE$ to reflect the elicited security requirements. Finally, the attributes *integrity_checked* and *secure_channel[Lane]* were automatically instantiated by the methods *check_integrity()* and *set_confidentiality()*, as soon as the confidentiality implementing activities A3 and A7 and the integrity implementing activities A5 and A9 were introduced in the process model.

A similar series of actions was followed for the instantiation of the attributes of the authentication-constraint activity A16. Its owner lane (SP Information System) and its source (A15) and target (A17) activities were automatically instantiated according to the existing connections of the process model. The *authentication_required* attribute was set to $TRUE$ during the security requirement elicitation. Finally, activity A15 authenticates the citizen to the swimming pool facility using the *set_authentication(Athenian Citizen)* method to set the *authenticated* attribute of the Athenian Citizen lane to $TRUE$.

B. Security verification results

In order to verify the satisfaction of the identified security requirements by the security implementing activities introduced to the process by the system designers, the algorithms introduced earlier in this work were provided to them (i.e., Algorithms 3 and 4 for data objects D1, D2 and D3 and Algorithm 1 for activity A16). The authentication checking, performed using procedure AUTHENTICATION_CHECK_A(A15) of Algorithm 1, indicated the satisfaction of the authentication constraint for the activity A16 by the process model of Fig. 3. The same was true regarding the integrity checking for data objects D1, D2 and D3.

Nevertheless, the confidentiality checking for each of the three data objects identified a violation. More specifically, since the confidentiality requirement has authorisation as a prerequisite, the authorisation checking performed in line 4 of Algorithm 3 returned a $FALSE$ value. This happens due to the lack of any authorisation implementing activities in the process model constructed by the system designers. Therefore, even if a secure communication channel is established between the clinic and the municipality for the transfer of the medical certificate document (D1) and between the municipality and the swimming pool facility's information system for the transfer of documents D2 and D3, the same entities also need the appropriate level of authorisation for handling such documents, which they do not currently possess. Therefore, authorisation implementing activities need to be introduced to

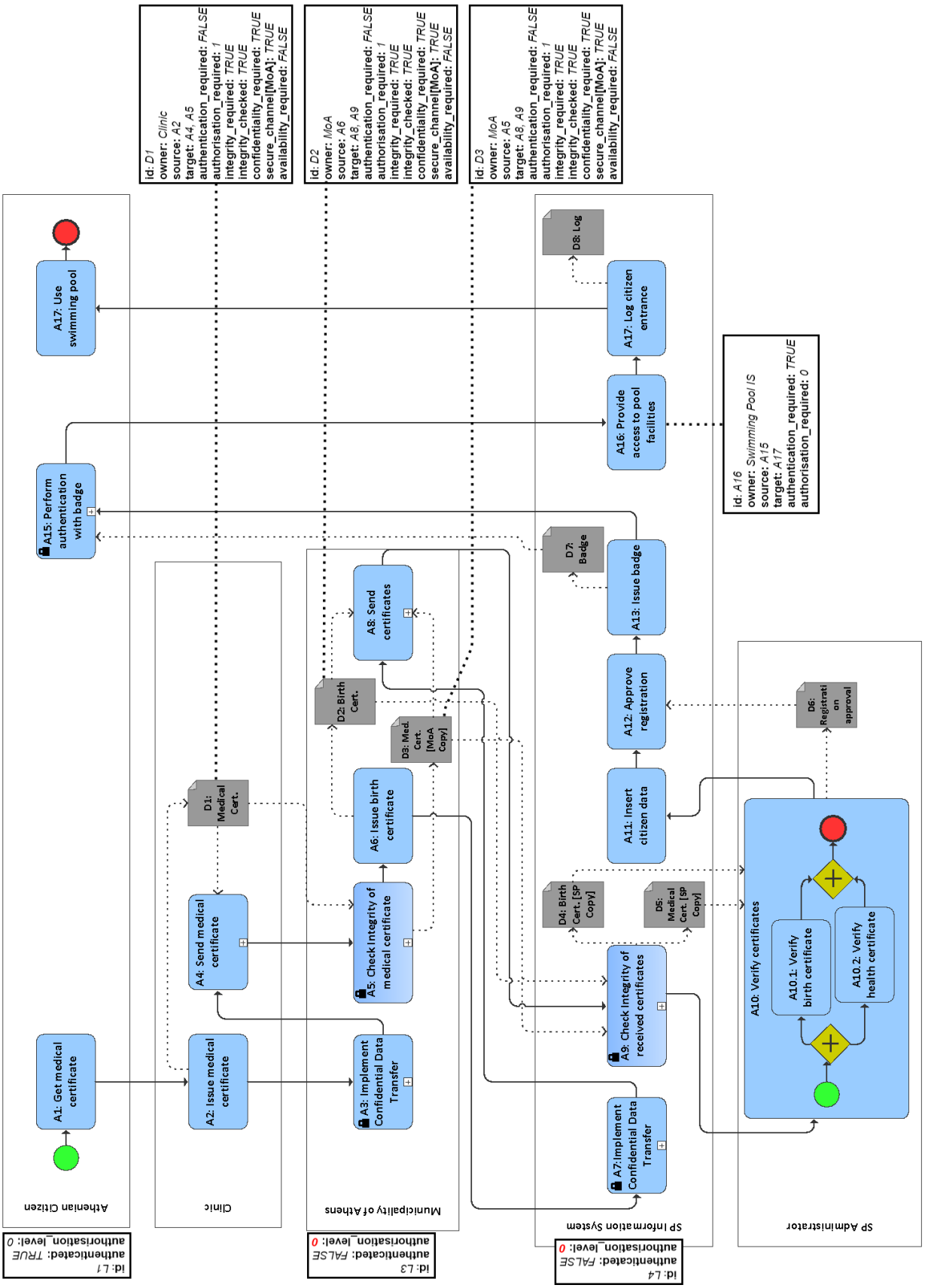

Fig. 3. Security-extended BPMN process model for citizen registration to swimming pool facility

the existing process model for the lanes of the municipality of Athens and the swimming pool's information system.

C. Discussion

The security verification of the above described real-life process provided some insights regarding the applicability and effectiveness of the proposed approach. The system designers of the participating organisation had already identified the security requirements that needed to be satisfied and were able to instantiate the provided attributes and apply the proposed verification algorithms without any significant issues. The whole process did not require any prior knowledge of either formal specification languages or technical security implementations. In fact, the prerequisites for using the proposed approach is rather limited, as it requires users to have elicited the security requirements of the process at hand and be familiar with workflow elements of BPMN, which is an advantage in terms of usability.

As discussed above, the application of the verification algorithms identified some security-related issues at the existing process model. The identification of such issues prompted the system designers to update the business process model in order to also implement authorisation where required. The ability of the algorithms proposed by this work to identify potential security violations and pinpoint their location within the workflow of the process, constitutes a positive indication of the overall effectiveness of the approach.

The application of the approach also revealed a number of aspects that need to be further refined. More specifically, the assignment of the security implementing activities, which are required for our approach to properly function, requires some intervention from experts. This is due to the fact that such activities may need to introduced at specific places within the workflow (e.g., directly before the execution of a constraint activity). It can also be the case that certain pre-existing process activities already perform security-implementing tasks but still need to be identified and marked as such. For instance, in the process illustrated in Fig. 3, activity A15 already existed as part of the process but needed to be marked as an authentication-implementing activity by experts, while confidentiality-implementing activities A3 and A7 where placed after the elicitation of the respective security requirements. Therefore, in the future, a more structured approach towards the integration of security-implementing activities needs to be established, as it is rather ad-hoc at its current stage.

The types of security requirements covered, as well as the development of tool support, are some aspects the further development of which would benefit the applicability and effectiveness of our approach. More specifically, while in its current form this approach is able to successfully verify a number of traditional security requirements, more security-related aspects have been identified by literature (e.g., [10] lists accountability, auditability and non-repudiation) which could be covered. Finally, the approach would benefit from the development of automated modelling tool support, which could facilitate the instantiation of the proposed attributes on-the-fly, during the creation of the model and the execution of the verification algorithms in the same environment. This issue was not as prevalent in the application of our approach in this specific case study but it could be an issue as the scale and complexity of the models to be verified increases.

V. Related Work

The variability introduced by the numerous available modelling languages, combined with the subjectivity and arbitrariness of manually created business process models, creates the need for formal approaches to verify the produced process designs [1]. Additionally, the verification of the compliance of an organisation's internal business processes to certain restrictions, internally (i.e., organisational standards and policies) and externally (i.e., laws and regulations) imposed, is often a legal requirement [2].

A common approach for checking the security properties of business process models involves the specification of the process model as a formal graph, the definition of the security properties using formal propositional languages and the use of an automated model checker, which takes as input the graph and the formal property definitions to perform the model checking. The work presented in [11] follows such approach by checking delegation and revocation functionalities, expressed using linear temporal logic (LTL), of business process models, captured as finite state machines (FSMs). In [8] LTL is used to express authorisation constraints of process models, translated in FSMs, in order to be verified by a model checker. In [12] the security properties which the process must fulfil are defined by ϕ formulas and the process model is formally defined as a planning system. Similarly, in [13] Petri-Nets are used to capture process models and automated tool support is provided for the identification of transitions between process elements that may lead to information leaks. The work presented in [14] validates security-related aspects of business processes by formalising both the created process model and its "security desiderata" before using them as input to a model checker. The model checker automatically analyses the formal model to check for violations of the security constraints and translates the results of this analysis to easily comprehensible graphical notation. A similar attempt is presented in [15] where Security Validation as a Service (SVaaS) is introduced as a cloud-based approach to business process security compliance checking where all relevant security related information along with the process model are captured in the form of a Business Process Compliance Problem (BPCP). The BPCP is translated into a formal specification language and used as input for a model checker, which returns the analysis results in a graphical manner. Finally, [16] presents an approach for the static checking of conformance of service implementations to security requirements specified at the process level via source code analysis and covers access control, separation of duty and need-to-know properties.

Most of the approaches introduced by the scientific literature of the area introduce overwhelming complexity for non-expert

users which severely hinders their widespread adoption by the industry today [2], [4]. One important drawback of such approaches is their limited support for modelling techniques, as most of them require process models to be transformed in a specific manner (e.g., Petri-nets, FSMs) before they can be used as input for a model checker. This contrasts with the variety of modelling languages used in practice and introduces a considerable overhead in terms of time and expert knowledge [17], as large numbers of processes need to be remodelled using a specific modelling technique. In contrast, the approach presented in this work uses BPMN 2.0, the "de-facto" standard for business process modelling [6], without the need to further translate neither the process model, nor the security requirements in formal specifications. Additionally, the range of compliance rules supported by works in the area of security verification is limited [2], as most approaches support a subset of security properties related to role assignment and user permissions (e.g., separation of duty, access control). Our work shifts the focus towards traditional security requirements (authentication, authorisation, confidentiality, integrity, availability), which can be verified by the structure of the workflow of the process.

VI. Conclusion

The verification of security properties of business processes is an important aspect of their development lifecycle, as it can validate that a process model is indeed secure before its execution begins. Nevertheless, the applicability of existing approaches for security verification suffers due to their complexity, lack of practicality and dependency on specialised knowledge. This work proposes an attribute-based approach for process security verification, as an attempt to overcome the above limitations. The proposed approach extends existing BPMN process elements with attributes that capture information about certain properties of the workflow. Based on such attributes, a number of conditions are defined, which need to be met by the structure of the models workflow for the satisfaction of each security requirement. Finally, algorithms that check such conditions are also developed, the execution of which performs the security verification of a process model.

The application of the proposed approach in a real life system provided some initial positive insights regarding its applicability and effectiveness, while also identified some aspects in need of further refinement. More specifically, the positive aspects included the ease-of-use of the approach, since it did not require any prior knowledge of formal languages by the process designers and analysts and its effectiveness, since it was able to identify and pinpoint parts of the designed process that violated specific security requirements.

Future directions of this work will focus on the extending the proposed approach to provide coverage to a wider range of security requirements. Its application will be further supported with the addition of rules to explicitly define the integration of security implementing activities during the design of business process models. Automated tool support will also be an important step towards increasing the ease-of-use of the proposed approach, as it will limit the need for manual intervention during the instantiation and verification process. Finally, a more extensive evaluation of the approach in real-life settings will allow us to extract further conclusions regarding its performance and applicability.

Acknowledgement

This research was partially supported by the Visual Privacy Management in User Centric Open Environments (VisiOn) project, supported by the EUs Horizon 2020 programme, Grant agreement No 653642.

References

[1] S. Morimoto, "A Survey of Formal Verification for Business Process Modeling," in *The 8th International Conference on Computational Science*. Springer, 2008, pp. 514–522.

[2] J. Becker, P. Delfmann, M. Eggert, and S. Schwittay, "Generalizability and Applicability of Model-Based Business Process Compliance-Checking Approaches A State-of-the-Art Analysis and Research Roadmap," *BuR-Business Research*, vol. 5, no. 2, pp. 221–247, 2012.

[3] M. El Kharbili, A. K. A. de Medeiros, S. Stein, and W. M. van der Aalst, "Business process compliance checking: Current state and future challenges." *MobIS*, vol. 141, pp. 107–113, 2008.

[4] G. Müller and R. Accorsi, "Why are business processes not secure?" *Lecture Notes in Computer Science*, vol. 8260, pp. 240–254, 2013.

[5] Object Management Group, "Business Process Model and Notation (BPMN) 2.0," Tech. Rep., 2011.

[6] M. Leitner, S. Schefer-Wenzl, S. Rinderle-Ma, and M. Strembeck, "An experimental study on the design and modeling of security concepts in business processes," in *The Practice of Enterprise Modeling*, 2013, pp. 236–250.

[7] ISO, "ISO/IEC 27000 Information technology Security techniques Information security management systems Overview and vocabulary," Tech. Rep., 2014.

[8] C. Wolter, M. Menzel, A. Schaad, P. Miseldine, and C. Meinel, "Model-driven business process security requirement specification," *Journal of Systems Architecture*, vol. 55, no. 4, pp. 211–223, 2009.

[9] G. Stonebumer, A. Goguen, and A. Fringa, "Risk management guide for information technology systems," *Recommendations of the National Institute of Standards and Technology*, 2002.

[10] Y. Cherdantseva and J. Hilton, "A reference model of information assurance & security," in *The 8th International Conference on Availability, reliability and security (ARES)*. IEEE, 2013, pp. 546–555.

[11] A. Schaad, V. Lotz, and K. Sohr, "A model-checking approach to analysing organisational controls in a loan origination process," in *The 11th Symposium on Access Control Models and Technologies*. ACM, 2006, pp. 139–149.

[12] A. Armando and S. E. Ponta, "Model checking of security-sensitive business processes," *Lecture Notes in Computer Science*, vol. 5983, pp. 66–80, 2010.

[13] A. Lehmann and D. Fahland, "Information flow security for business process models - Just one click away," in *The 10th International Conference on Business Process Management*. Springer, 2012, pp. 34–39.

[14] W. Arsac, L. Compagna, G. Pellegrino, and S. E. Ponta, "Security Validation of Business Processes via Model Checking," *Lecture Notes in Computer Science*, vol. 6542, pp. 29–42, 2011.

[15] L. Compagna, P. Guilleminot, and A. D. Brucker, "Business process compliance via security validation as a service," in *The 6th International Conference on Software Testing, Verification and Validation*. IEEE, 2013, pp. 455–462.

[16] A. D. Brucker and I. Hang, "Secure and compliant implementation of business process-driven systems," in *The 10th International Conference on Business Process Management*. Springer, 2012, pp. 662–674.

[17] H. Groefsema and D. Bucur, "A Survey of Formal Business Process Verification: From Soundness to Variability," in *Proceedings of the 3rd International Symposium on Business Modeling and Software Design*, 2013, pp. 198–203.