Business Process Model and Notation for Forensic-Ready Software Systems

Lukas Daubner[1][a], Raimundas Matulevičius[2][b], Barbora Buhnova[1][c] and Tomas Pitner[1][d]
[1]*Faculty of Informatics, Masaryk University, Brno, Czech Republic*
[2]*Institute of Computer Science, University of Tartu, Tartu, Estonia*

Keywords: Forensic Readiness, Forensic-Ready Software Systems, Modelling, BPMN, Software Design.

Abstract: The design and development of secure systems is an important and challenging task. However, such systems should also be prepared for eventual disputes or occurrences of a security incident. To solve this, forensic-ready software systems are, by-design, prepared to assist in the forensic investigation and to provide on-point data with high evidentiary value. However, software engineering support for the systematic development of such software systems is rather sparse. This paper tackles the problem by introducing novel modelling notation, called BPMN for Forensic-Ready Software Systems (BPMN4FRSS), including its syntax and semantics. The notation aims to capture the forensic-ready controls and enable reasoning over them, primarily focusing on potential digital evidence. Importantly, it is made to support forensic readiness oriented risk management decisions. The approach is then demonstrated in a scenario where the controls, which mitigate security and business risks, are properly represented.

1 INTRODUCTION

The importance of security in software systems is widely acknowledged. Consequently, the development of systems secure by design (Geismann and Bodden, 2020) is well established. However, the possibility of a cybersecurity incident cannot be ruled out entirely, even in highly secure systems (Casey and Nikkel, 2020). Once the incident occurs, it needs to be thoroughly investigated to uncover the root cause and possible culprit in a very reliable manner.

To meet the goals, digital forensics (Casey, 2011) techniques are employed. However, such investigation is a costly, laborious, time-consuming, and delicate endeavour, with uncertain success. The reason is strict requirements on digital evidence, on which the conclusions are built, to ensure a high level of reliability. For example, the digital evidence must be found, identified, collected, analysed, understood, and presented in a manner that does not compromise its integrity or meaning so it can be used in a potential legal proceeding (McKemmish, 2008).

[a] https://orcid.org/0000-0003-0853-2776
[b] https://orcid.org/0000-0002-1829-4794
[c] https://orcid.org/0000-0003-4205-101X
[d] https://orcid.org/0000-0002-2933-2290

Forensic readiness (Tan, 2001) was formulated to reduce the investigation cost and increase the value of digital evidence. It is a set of proactive measures, primarily oriented on organisational preparedness, including their systems for the possibility of forensic investigation (Rowlingson, 2004; Grobler and Louwrens, 2007). Recently, forensic readiness was approached from a software engineering perspective (Pasquale et al., 2018), making systems forensic by design, producing and handling the evidence in a forensically sound way. That includes, among others, ensuring the existence of useable digital evidence, protecting its integrity, and making it easily cross-referenceable (Sachowski, 2016).

While software systems generally produce a number of digital artefacts (e.g., logs (Studiawan et al., 2019)), their quality and completeness are not generally assured (Daubner et al., 2020a). Furthermore, if such ad-hoc artefacts are to be used, investigators must spend a great amount of time to find and understand them (Kävrestad, 2018). Also, digital evidence might be ruled inadmissible or has its evidentiary value reduced if it is based on unreliable data or without integrity guarantees (Casey, 2011). Furthermore, misleading data admitted as digital evidence can lead to miscarriage of justice (Henley, 2019).

This paper introduces a new approach to sup-

Daubner, L., Matulevičius, R., Buhnova, B. and Pitner, T.
Business Process Model and Notation for Forensic-Ready Software Systems.
DOI: 10.5220/0011041000003176
In *Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2022)*, pages 95-106
ISBN: 978-989-758-568-5; ISSN: 2184-4895
Copyright © 2022 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

port the design of forensic-ready software systems, enabling software engineers to build systems producing on-point data with high quality. To this end, we propose a novel modelling notation, an extension to Business Process Model and Notation (BPMN), called BPMN for Forensic-Ready Software Systems (BMPN4FRSS). It aims to capture the specific forensic-ready controls (i.e., implementation of requirements) and allow reasoning over them, including their validation and analysis. The primary concern is the representation of data sources potentially usable as digital evidence (i.e., potential digital evidence) within the software system. Furthermore, the BPMN4FRSS model captures mutual corroboration of potential digital evidence to analyse its sufficiency for an anticipated incident investigation. Lastly, the model is aimed to serve as documentation when a third-party investigator is required.

This paper is structured as follows: After the introduction, Section 2 explores related work. Section 3 describes the forensic-ready requirements and the aim of our effort. Then, Section 4 explains the syntax and semantics of the BPMN4FRSS, followed by Section 5 demonstrating its application. Finally, Section 6 concludes the paper and outlines future research steps.

2 RELATED WORK

Existing research on forensic readiness from a software engineering perspective has focused on proactive preservation of relevant and minimal evidence (Alrajeh et al., 2017), automated logging instrumentation (Rivera-Ortiz and Pasquale, 2020), and outlined their verification (Daubner et al., 2020b). There also exist frameworks focusing on broader organization-wide perspective (Elyas et al., 2015) and domain specifics (Grispos et al., 2017b). As for the forensic-ready requirements, a study mapping their understanding and approach in software engineering practice was conducted (Grispos et al., 2017a).

Aiming to identify the specific requirements, risk management is a viable strategy for forensic readiness (Bajramovic et al., 2016). A parallel can be found in the secure systems domain (Altuhhova et al., 2013) where risk management is principal in formulating security requirements (Matulevičius, 2017). In fact, the security risk-based approach was extended to cover the forensic-ready requirements as well (Daubner and Matulevičius, 2021).

So far, modelling approaches of forensic-ready software systems are scarce, describing code-level scenarios based on UML Sequence Diagram (Rivera-Ortiz and Pasquale, 2020) and incident investigation in cyber-physical systems, including their topology (Alrimawi et al., 2017). On the other hand, the situation in cybersecurity-focused modelling is different as there exist a wide range of approaches (Van den Berghe et al., 2017; Geismann and Bodden, 2020).

Notably, this paper extends an existing ad-hoc BPMN extension for representing potential evidence sources in forensic-ready software systems (Daubner and Matulevičius, 2021). Although the publication presents how such representation could assist in risk management, it has gaps in both syntax and semantics. Therefore, the main contribution of this paper is a proper specification of the extension's syntax and clarification of its concepts and scope to enable reasoning over forensic-ready software systems and allow further evolution of the approach.

Concerning the other BPMN approaches, several extensions that partially overlap with forensic readiness are focused primarily on cybersecurity. An example is the modelling of security requirements, including non-repudiation and attack detection (Rodríguez et al., 2007; Chergui and Benslimane, 2018). Another approach focuses on defining security policies by annotating the BPMN model with new graphical elements and predicates (Salnitri et al., 2014). Similarly, a language for structured textual annotations of a BPMN model aims for the model transformation into security policies and composition with pre-existing fragments (Mülle et al., 2011). Security risk management is also addressed by a BPMN extension to describe risks and support decisions (Altuhhova et al., 2013; Matulevičius, 2017). Lastly, an example of a privacy-focused BPMN extension includes privacy-enhancing technologies into the model (Pullonen et al., 2017) to allow an analysis of data leakage. It is supported by a tool enabling detailed analysis of those models (Pullonen et al., 2019).

3 FORENSIC-READY SOFTWARE SYSTEM REQUIREMENTS

With the coinage of the term forensic-ready software systems, Pasquale et al. (Pasquale et al., 2018) formulated a set of high-level requirements for such systems. The requirements are listed in Table 1, containing names and short descriptions. Furthermore, the work also elicits the open software engineering challenges for forensic-ready systems. This paper explicitly focuses on one particular challenge: *Representing and reasoning about forensic-ready systems*.

Tacking the challenge is a pivotal step in supporting the development of forensic-ready software systems. Proper representation aids in the mapping

Table 1: Requirements on forensic-ready software.

Availability	All useful potential evidence is preserved, prepared and retrievable if needed.
Relevance	Preserved potential evidence is relevant to considered incidents and scenarios.
Minimality	Potential evidence unnecessary for the expected investigation is not preserved.
Linkability	Preserved pieces of potential evidence can be linked with other pieces.
Completeness	Preserved potential evidence is sufficient to satisfy or refute the considered investigation hypothesis.
Non-Repudiation (Admissibility)	Preserved potential evidence should conform as admissible evidence. Its integrity and authenticity are ensured.
Data Provenance	The handling process of potential evidence records all operations made.
Legal Compliance	The potential evidence handling process is compliant with laws and regulations.

of the high-level requirements to their instantiation as forensic-ready control on a model of a software system. Additionally, it is crucial for the verification (Daubner et al., 2020b). The importance and usefulness of modelling have been shown in a risk-based approach for forensic-ready software systems (Daubner and Matulevičius, 2021), where BPMN-based models support the risk management process. We aim to expand this idea further to enable a richer and more detailed expression of forensic-ready controls.

For this paper, we formulate the following research question to drive the effort: **How to model the forensic-ready software systems to support the risk management decisions?**

We further decompose the research question into three goals to reflect the requirements in Table 1. Notably, this work does not explicitly consider the *Legal Compliance* requirement, but the general good practice of digital forensics is reflected in the design.

G1: Model Incident Scenario and the Relevant Potential Evidence. The aim is to model the *Availability* and *Relevance* of the potential evidence by mapping its sources and types directly into a scenario in question. Secondarily, with potential evidence explicitly elicited, *Completeness* and *Minimality* can be addressed by analyzing the model.

G2: Model Relationships between the Potential Evidence. Represent relationships between the potential evidence. Allow specification of potential evidence types to explore timing, mutual dependence, and common data fields, addressing *Linkability*.

G3: Model the Lifecycle and Properties of Potential Evidence. By modeling the entire lifecycle of potential evidence from the creation to the preservation, as well as any operations performed, *Data Provenance* can be established as needed. Additionally, proofs on integrity or authenticity can be established at a certain point within the lifecycle, which adds to its *Non-Repudiation*.

4 BPMN EXTENSION FOR FORENSIC-READY SOFTWARE SYSTEMS

This section describes the syntax and semantics of BPMN4FRSS based on BPMN 2.0 (OMG, 2010). While designing the extension, we followed the existing definition of BPMN abstract syntax and semantics and methodology utilised in related BPMN extensions (Altuhhova et al., 2013; Pullonen et al., 2017). However, the semantics of BPMN4FRSS is described in textual form. While the formal definition of semantics is essential to avoid ambiguity and allow for precise analytic methods (Harel and Rumpe, 2004), we plan it as a part of future work, together with model analysis discussed in Section 6.2.

The BPMN-based approach was chosen for three main reasons. (1) The process models capture a dynamic behaviour that allows inspecting the interplay between pieces of potential evidence and their lifecycle. (2) A high level of abstraction for the process model contributes to platform independence, in contrast to a similar platform-dependent approach (Rivera-Ortiz and Pasquale, 2020). (3) Interconnection with risk management, as traditionally forensic readiness is established by formulating and evaluating risk scenarios (Rowlingson, 2004). In this regard, the BPMN was motivated by the existing extension for security risks (Altuhhova et al., 2013), which BPMN4FRSS complements. The downside of this approach is the difficult mapping of potential evidence on the modelled system's broader context, namely high-level architecture.

4.1 Business Process Model and Notation

BPMN is a language for constructing a business process model, a standard controlled by Object Management Group (Matulevičius, 2017). While syntactically similar to the flow charts, it has a rich semantic model (Dijkman et al., 2008; OMG, 2010), supporting even the model execution (Silver, 2011). Figure 1 contains an example BPMN diagram, which includes the basic BPMN constructs relevant to this paper.

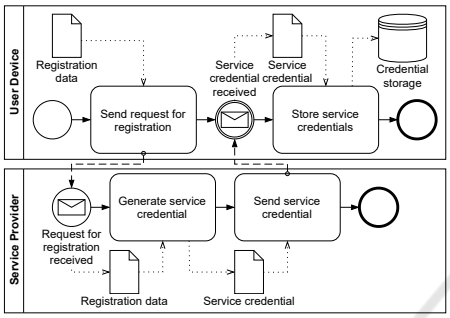


Figure 1: Example of BPMN diagram.

The diagram contains two *Pools* (User Device and Service Provider), each containing a number of *Flow Objects* connected by *Sequence Flows* within a single pool and by *Message Flows* between the pools. The most prominent *Flow Object* is *Task* (e.g., Send request for registration) representing an atomic activity.

An *Event* is a flow object with multiple specialised types, depending on its location and trigger condition. In the example, *Start Event* denotes the start of a process and *Message Start Event*, requiring a message as a trigger condition. On the other hand, the end is marked with an *End Event*. Inside the process, a *Message Intermediate Event* (Service credential received) requires "catching" a message for process continuation. The two types of flow objects not present in the example are *Gateway* representing a control point (e.g., branching) and *Sub-Process* representing a composite activity.

Data exchanged between events and tasks are represented with a *Data Object* (e.g., Registration data), which has its lifecycle tied to the process instance. In contrast, persistent storage is denoted with a *Data Store* (Credential storage). Associations of those constructs are called *Data Associations*.

4.2 Semantics

Planning the implementation of forensic readiness includes identifying scenarios where a digital forensics approach could be required (Rowlingson, 2004). These scenarios could originate from risk management describing security risks (Daubner and Matulevičius, 2021) or incidents from past experience (Rivera-Ortiz and Pasquale, 2020) that forensic readiness should cover. Such a scenario can be described as a business process model using BPMN. However, without the principal component – potential digital evidence[1]. Hence, BPMN4FRSS allows the expression of potential digital evidence, including its origin, handling, storage, and corroboration with others within a business process model (i.e., a scenario).

Potential evidence is naturally the core concept of BPMN4FRSS. Because the potential evidence is essentially a piece of data, it is represented as a BPMN *Data Object*, although with a specific lifecycle. Each potential evidence is considered an instance of a particular type, called *Potential Evidence Type*, that describes its content. Additionally, the notion of evidence context regarding accessibility to potential evidence during an investigation is captured. The motivation behind this is the difference between a cooperative device that will surrender the evidence (e.g., server under organization control) and a non-cooperative device whose evidence will not be available (e.g., user's phone). In the business process terms, different devices correspond to different participants, thus BPMN *Pools*.

The first step of the potential evidence lifecycle is to mark a point of origin, called the *Potential Evidence Source*, which denotes a point in time (process) where the potential evidence is created. Furthermore, one *Potential Evidence Source* can denote the origin of multiple *Potential Evidence Types* (e.g., sending a message generates log records on the application and web server). Such simplification allows the process to be high-level view or focus on a particular detail, effectively splitting the source as needed. Finally, the evidence needs to be persistently stored to be used later during an investigation, which is a trait of the BPMN *Data Store*.

No digital evidence is self-sufficient to make conclusions. It must be corroborated with other pieces, and there should be strong assurance about its integrity and authenticity. To that end, the BPMN4FRSS model captures relations between the *Potential Evidence Types* to establish links between them and their nature. The number of relations and their chaining has a direct impact on expected evidentiary value. A very specific relationship is formed with *Proof*, which goal is to provide guarantees regarding other *Potential Evidence Types* (e.g., a hash provides proof

[1]Note the difference: potential digital evidence – potentially useable for future investigation, and digital evidence – used to satisfy or refute the investigation hypothesis.

of integrity for a log record). Naturally, the guarantees depend on specific technologies, so BPMN4FRSS is open for extensions, allowing to specify specialised proofs while keeping the core semantical relationships valid.

During its lifecycle, multiple factors can impact the potential evidence in both a positive and negative manner. A simple example is a computation over the potential evidence, which is represented as BPMN *Task*. However, the nature of this computation must be noted as it can influence the meaning of said potential evidence. In BPMN4FRSS, we define three core computation groups: integrity, authenticity, and data transformation computation. Currently, they are meant for documentation purposes but remain open for extensions to represent concrete methods and protocols while fitting to the overall lifecycle scheme.

In addition to the guarantees from internal sources (w.r.t. modelled scenario), external sources could also influence the potential evidence in the form of service. A good example is Trusted Timestamping (Ćosić and Bača, 2010), a 3rd party service testifying on integrity, timeliness, and possibly authenticity. We included support for such sources to BPMN4FRSS, as it allows to express a wide range of mechanisms that establishes guarantees without the need to model them in detail. As such, they are expressed by the combination of BPMN *Pool* and *Potential Evidence Type*. Generally, however, the services are not required to be 3rd party but can be an integral part of the target system, only abstracted for the scope of the scenario. Similarly to the proofs, BPMN4FRSS is open to extensions representing concrete services. Explicitly in this paper, extensions for PKI-based (Ćosić and Bača, 2010) and blockchain-based (Weilbach and Motara, 2019) services are demonstrated.

4.3 Abstract Syntax

Figure 2 depicts how BPMN is extended by BPMN4FRSS extension on BPMN abstract syntax (OMG, 2010). It contains all the essential concepts, as well as possible extension points. The model is intentionally left incomplete to accommodate more specific technologies relevant for forensic ready software like integrity or non-repudiation controls.

The core syntactical constructs of BMPN4FRSS are *Potential Evidence Source* and *Potential Evidence Type*. The former can be applied on either BPMN *Task*, *Event*, or *Data Store*, but only on one of those simultaneously as the same point of origin cannot be shared. Originating from a particular *Potential Evidence Source*, multiple *Potential Evidence Types* can be defined. Furthermore, *Data Relation* relationships can exist between the *Potential Evidence Types*, either whole or its fields, with an expression specifying it. A core construct denoting the locations (multiple are permitted) where *Potential Evidence Types* are stored is the *Evidence Storage*, a specialization of BPMN *Data Store*.

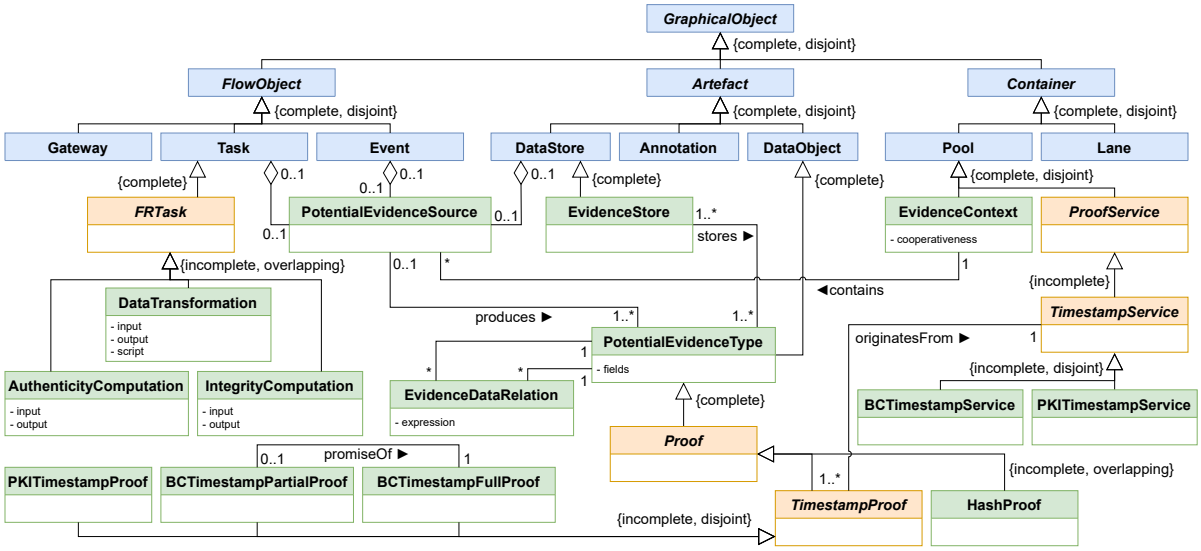
We defined two types of extensions to BPMN *Pool*. First, *Evidence Context*, as the name suggests, specifies the *Pool* where *Potential Evidence Sources*, therefore also *Potential Evidence Types* can be defined with information regarding the cooperativeness level of the context. The cooperativeness can be: (1) Non-Cooperative: potential evidence is inaccessible (e.g., personal device), (2) Cooperative: potential evidence is accessible during every investigation, and (3) Semi-Cooperative: potential evidence accessibility is context-sensitive and is further specified in Cooperativeness Context. The second extension, *Proof Service*, is an abstract construct allowing the definition of specific services, external from the modelled process point of view, which provides proofs for *Potential Evidence Types*.

The proofs themselves are organised under abstract *Proof*, a special kind of *Potential Evidence Type*. They can originate from within the system, meaning a result of a *Task* in the modelled process, but also form *Proof Service* by a *Message Flow*. Regardless of origin, they define further assurance of other *Potential Evidence Types* (e.g., corroborated data, integrity).

Both the *Proof Service* and *Proof* are meant as explicit extension points for more concrete and technology-dependent constructs. The formulation of extension points in BPMN4FRSS is motivated by streamlining the extensibility. For example, core BPMN-4FRSS defines *Hash Proof* to represent the output of a cryptographic hash algorithm commonly used as a proof of integrity. Other possible extensions could be constructs for digital signatures or hash structures.

Specialised *Proof Service* can also define a specialised and related *Proof*. An example is tuple *BC Timestamp Partial Proof* and *BC Timestamp Full Proof*, related to *BC Timestamp Service*, corresponding to timestamping services based on blockchain. It is particularly illustrating as it demonstrates the support for different real services. For example, ChainPoint (Liang et al., 2017) and OpenTimestamps (Weilbach and Motara, 2019) require both partial and full proof as the full proof will be committed with a time delay, so partial proof is generated as a temporal receipt. On the other hand, OriginStamp (Hepp et al., 2018) can generate full proof immediately, if required.

The last group is three specializations of the



Note: Classes highlighted in green represents the BMPN4FRSS constructs, and classes highlighted in orange are the BMPN4FRSS extension points.

Figure 2: Abstract Syntax of BPMN4FRSS.

BPMN *FRTask*, which represents a *Task* relevant for forensic readiness scope. These represent specific computations related to establishing data integrity, authenticity, and handling. Each of them specifies input and output data. Additionally, the *Data Transformation* defines a script for the transformation, allowing for its documentation and validation to assert the impact on the meaning of potential evidence. Again, it is possible to extend either of them to address specific schemes. Notably, the three specializations are not disjoint to support schemes that address both authenticity and integrity simultaneously.

4.4 Concrete Syntax

Realization of the BMPN4FRSS modelling notation is done by introducing new visual elements and stereotypes for existing BPMN constructs. Moreover, the stereotypes are further specified by parameters, which provide details regarding the application of a particular stereotype. The stereotype-parameter concept is rooted in the concept of profiles, stereotypes, and tagged values used in UML (Arlow and Neustadt, 2005). For the sake of clarity, we opt to use green colour to highlight the constructs.

Table 2 contains a summary of the concrete syntax and its mapping on the abstract syntax. Each abstract syntax construct is mapped on either visual element, stereotype, or parameter in the concrete syntax. Therefore, some of the relationships are expressed as parameters rather than visually. This choice was motivated by finding a balance between visual expressiveness and readability.

Core Concepts of the BPMN4FRSS deal with establishing the potential evidence, including its origin, context, and storage. Figure 3 describes a simple scenario, where both Business Data and Audit log are marked as potential evidence. The difference between those two is that Business Data is an integral part of the scenario, while the Audit log is its byproduct but relevant for the possible investigation. Both are organised under *Evidence Context*, which parameter *Cooperativeness* informing that the potential evidence is always available if needed (e.g., under full control of the responsible organization). Lastly, the scenario deals with the persistent storage of Business Data, so the *Data Store* is marked as such.

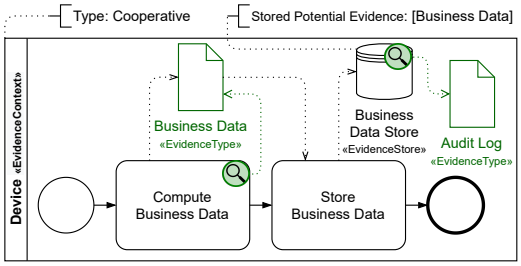


Figure 3: Concrete syntax of BPMN4FRSS core concepts.

Proof specifies an explicit corroboration of some other potential evidence captured by the Evidence Data Relation, which is visible in the Evidence View (see Section 4.5). Figure 4 represents a scenario where a digest was created for Business Data to prove its integrity. The principal part is the *Integrity Com-*

Table 2: Concrete syntax of BPMN4FRSS.

Abstract Syntax	Concrete Syntax	Parameters
Potential Evidence Source		
Potential Evidence Type	«EvidenceType»	Data Fields, Lifecycle Process
Produces		
Evidence Data Relation		Expression
Evidence Context	«EvidenceContext»	Cooperativeness
Evidence Store	«EvidenceStore»	Stored Potential Evidence
Hash Proof	«HashProof»	
PKI Timestamp Proof	«PKITimestampProof»	Originates From
BC Timestamp Partial Proof	«BCTimestampPartialProof»	Originates From
BC Timestamp Full Proof	«BCTimestampFullProof»	Originates From
Promise Of		
PKI Timestamp Service	«PKITimestampService»	
BC Timestamp Service	«BCTimestampService»	
Authenticity Computation	«AuthenticityComp»	Input, Output
Integrity Computation	«IntegrityComp»	Input, Output
Data Transformation	«DataTransformation»	Input, Output, Script

putation Task which takes one *Data Object* as input and produces a *Hash Proof* as an output. While not included in the example, the storage of both the potential evidence and proof greatly impacts its qualities (e.g., storing them on separate devices). The same input-output logic applies to other *FRTasks* as well, and possible extensions might include additional input or output (e.g., key).

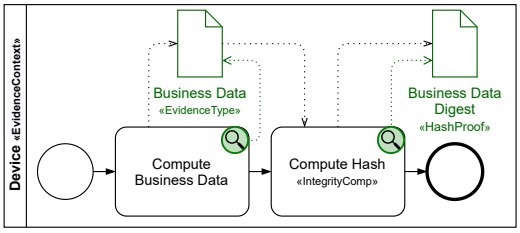


Figure 4: Concrete syntax of BPMN4FRSS proof.

Proof Service establishes an external entity to corroborate a *Data Object*. Figure 5 presents a scenario of using a blockchain-based timestamping service. To that end, a signed digest (output of *Authenticity and Integrity Computation*) is sent to the service to receive a partial proof. While some services offer an instantaneous commitment to a blockchain, in which the partial proof is skipped, we model a case when the service firstly presents an acknowledgement of receiving the data and the full proof at a later time, when it is actually committed. In all cases, the full proof is the desired and reliable guarantee. The PKI-based timestamping service is modelled analogously, as the *Proof Service* abstracts the inner mechanisms. However, there are different guarantees from different services and technologies in practice.

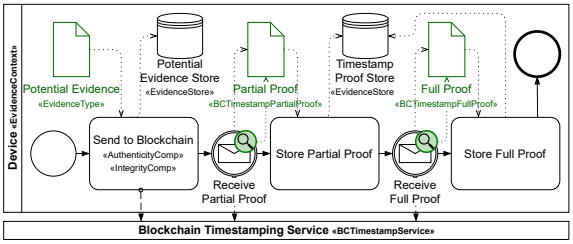


Figure 5: Concrete syntax of BPMN4FRSS proof service.

4.5 Model Views

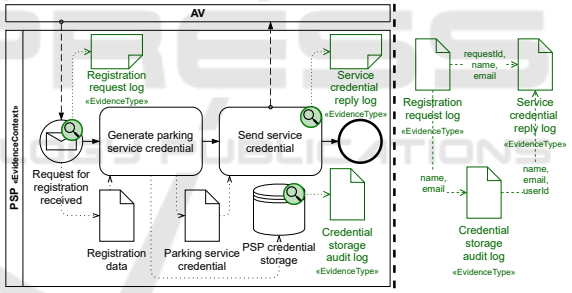


Figure 6: Scenario View (left) and Evidence View (right).

Visual presentation of every part of BPMN4FRSS, specifically *Evidence Data Relations*, can be problematic, resulting in an overly complicated model diagram. A possible solution could be not to visualise some relations and leave them in purely textual form. Instead, we opted to provide two concurrent views on a model, a well-known approach for software architecture modelling (Kruchten, 1995). Specifically, one view focuses on the scenario and the other on the evidence relationships.

Furthermore, we adopt the model view approach, allowing us to combine several models into a single goal-oriented view (Bruneliere et al., 2019). Specifically, a separately modelled and reusable process describing the lifecycle of potential evidence is formulated to supplement the main process model.

Scenario View is a default view displaying the

core BPMN diagram enhanced by the BPMN4FRSS visual elements and stereotypes. However, the *Evidence Data Relations* between *Potential Evidence Types* are omitted, as the clarity of the scenario is favoured. An example is depicted in Figure 6 (left).

Evidence View is a specialised view focusing on the relationships between Potential Evidence Types. This includes annotated *Evidence Data Relations* and timing information derived from the process represented with the direction arrow. An example is depicted in Figure 6 (right). The labels on each *Evidence Data Relation* depicts expression parameter. For brevity, the shared data fields are displayed, and more complex functions are enlosed in curly brackets.

Lifecycle Process is a separate process, an example depicted in Figure 5, capturing the lifecycle of potential evidence. It also covers its processing and strengthening (integrity and availability proofs). Although the same can be expressed in the main process, the separation promotes reusability and readability. Furthermore, the Lifecycle Processes can be shared among multiple *Potential Evidence Types* and cascaded (i.e., new *Potential Evidence Type* in a Lifecycle Process can have its own) but must be acyclic. The Lifecycle Process cannot be used if the target *Potential Evidence Type* is handled in the main process apart from its creation.

5 APPLICATION OF BPMN4FRSS

To demonstrate the BPMN4FRSS, we apply the notation to represent forensic-ready controls in the Automated Valet Parking (AVP) scenario (Nwaokolo, 2020). This particular scenario was previously utilised to demonstrate a risk-based approach for forensic-ready software systems (Daubner and Matulevičius, 2021), which discussed the risks and their mitigations by forensic-ready controls. While the work provides a good foundation for designing forensic-ready software systems, the used BPMN modelling notation aims for ad-hoc support of risk management decisions, with only a basic syntax definition. Its major shortcoming is the inability to support deeper reasoning and analysis of the scenario models due to a deficiency in both syntax and semantics. We expanded on the results and filled the gap by introducing a properly defined notation with validation and analysis in mind. Consequently, the scenario is ideal for validating our approach, allowing a comparison to the current state of the art in forensic-ready software design.

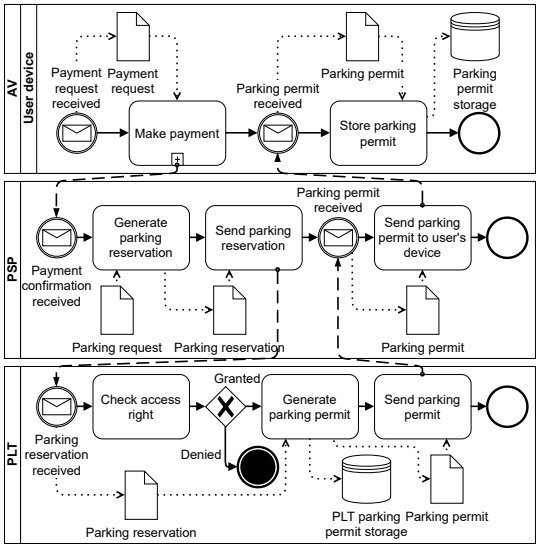


Figure 7: Excerpt from Automated Valet Parking scenario: Issuing a parking ticket.

5.1 Default Scenario

Figure 7 is an excerpt from the BPMN diagram describing a scenario of issuing a parking ticket. Concretely, the presented part describes the process from the payment made on an Autonomous Vehicle (AV) to generating a parking permit by Parking Lot Terminal (PLT), all facilitated by Parking Service Provider (PSP), which sends the parking permit back to AV – the user device.

In the default setting, numerous vulnerabilities and a complete lack of sufficient forensic-ready properties have been shown (Daubner and Matulevičius, 2021), giving malicious parties not only the possibility to attack but also making them hard-to-detect. To enumerate the risks related to the scenario, a risk management process is employed (Matulevičius, 2017). This demonstration assumes that risk management was performed and the specific risks (including security and business) are known. Specifically, the following four risks are considered in the demonstration:
R1: A malicious insider injects a parking permit into the PLT parking permit storage out-of-the-process due to their capability to access it. Leading to a loss of parking permit integrity.
R2: An attacker fabricates a fake Parking reservation and sends it to PLT due to access control tampering. Leading to the loss of Parking permit integrity.
R3: A dishonest customer repudiates a Parking permit received from PSP, demanding a reimbursement, leading to a financial loss.
R4: An attacker uses zero-day vulnerability to fabricate a Parking reservation. Leading to the loss of

parking permit integrity.

Reliable digital evidence is needed to detect and prove the occurrence of the listed risks. To this end, forensic-ready controls must be established, typically in the form of new potential evidence sources and potential evidence.

5.2 Forensic-ready Scenario

Figure 8 describes the scenario enhanced with forensic-ready controls represented by BPNM4FRSS. It explicitly covers all the listed risks by ensuring that their occurrence can be detected and investigated. Additionally, the relations between the potential evidence, important for the risk treatment evaluation, are visualised in Figure 9 depicting the evidence view of the scenario.

The evaluation is performed manually by comparing the potential evidence types produced in the parts of the process describing the event of interest. Additionally, the evidentiary value of potential evidence is determined based on relations with others. The risks are addressed as follows:

R1: The actions of a malicious insider injecting a parking permit into the PLT parking permit storage are recorded as instances of the PLT permit storage audit. Moreover, if the injection is performed out-of-process, some evidence links will be missing (PLT reservation incoming log) and thus detectable. A possible issue involves tampering with potential evidence, mitigable by adequate storage, e.g., remote storage, representable in a lifecycle process.

R2: By itself, a fabricated parking reservation is indistinguishable for PLT, but it is detectable based on the potential evidence introduced by forensic-ready controls. Firstly, the instances of the PLT access audit should record any modifications. Secondly, the fabrication can be detected by reconstructing the evidence links (e.g., PSP reservation outgoing log will be missing) and comparing the contents of PLT and PSP parking permit stores.

R3: If a dishonest customer repudiates a parking permit, any potential evidence on the user device is unavailable due to the privacy and user's unwillingness (i.e., non-cooperating device). The overall goal is to reinforce the timeliness, authenticity, and integrity of parking permits. This was achieved by introducing the blockchain-based timestamping service OpenTimestamps, which generates proofs of the parking permit verifiable by 3rd party. BPMN4FRSS captures the proofs and their relationships within the scenario.

R4: The forensic-ready controls assist in tackling the zero-day vulnerability by proactively preserving potential evidence in key areas. It is characterised as "making the attacker noisy". Examples of such areas are interfaces and storage as denoted on the model.

It could be argued that traditional security controls could address the risks without the need for forensic readiness. However, they do not fully mitigate the risk in many cases. For example, the security controls might not adequately cover business risks or risks from insider threats, like R3 and R1. Forensic-ready controls are in a sense complementing the security controls to provide means to detect and investigate an occurrence of an incident.

6 CONCLUSION

Our work tackles the challenge of representing the forensic-ready software system controls, and with the combination of risk management, making reasoning over them accessible also for software engineers with only a basic knowledge of digital forensics. Additionally, the intended use goes beyond software development, promoting documentation of the systems useable during the investigation. Consequently, our result contributes to the evolving state-of-the-art in the development of forensic-ready systems and is an enabler for evolution in this area.

6.1 Answers to Research Goals

The introduced BPMN extension can support the development of forensic-ready software systems by enabling their modelling. We validated this claim by enhancing an existing scenario with defined risks to model their mitigation using forensic-ready controls systematically. Arguably, BPMN4FRSS met the set goals in the following way:

G1: Modeling the scenario itself is possible using plain BPMN, which is often the case in practice and demonstrated in Figure 7. The BPMN4FRSS enables enhancement of the scenario by marking the potential evidence and their points of origin within the scenario. As a result, the scenario can be evaluated if it is sufficiently covered with respect to the risks.

G2: BPMN4FRSS allows detailed documentation of potential evidence in a textual way. However, to provide deeper insight into the relationships, we proposed a specialised evidence view that visualises the relationships in a tangible way. Consequently, non-linked potential evidence and the clusters of related or inadequately linked potential evidence are visually detectable. Arguably, such presentation could be beneficial also during the investigation to communicate the available data with an investigator.

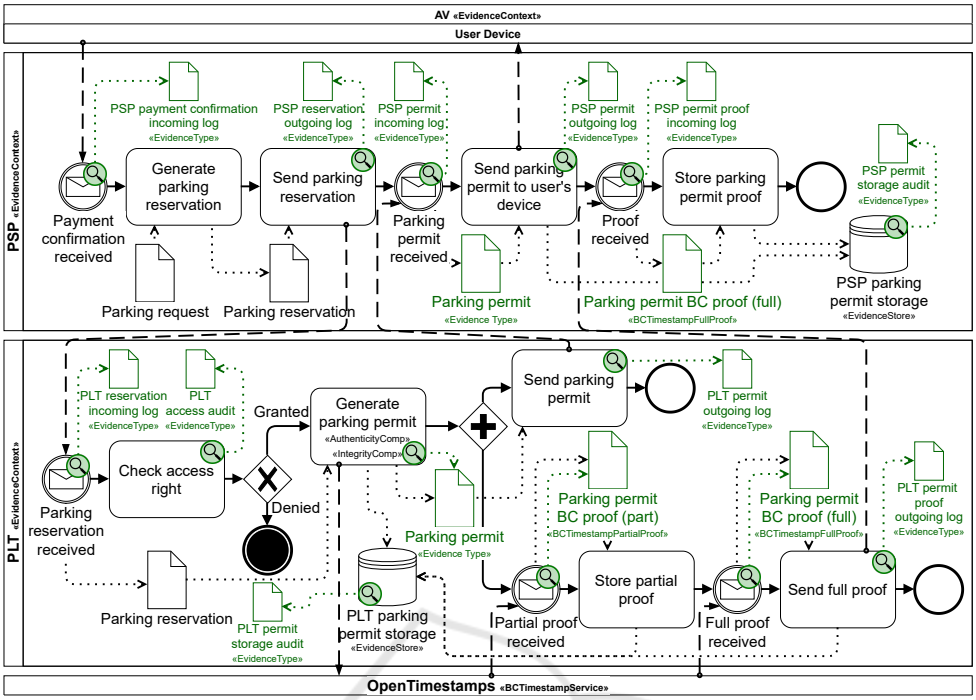


Figure 8: BPMN4FRSS model of enhanced AVP scenario (scenario view).

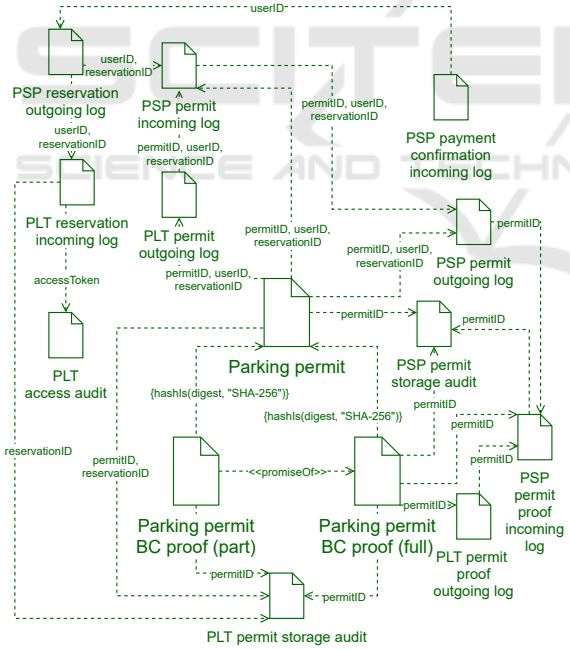


Figure 9: BPMN4FRSS model of enhanced AVP scenario (evidence view).

G3: Establishing the potential evidence origin is not sufficient for a holistic representation of its lifecycle. For that reason, BPMN4FRSS contains constructs to express storage and handling of the evidence using BPMN terms. A particular focus is given to strengthening the potential evidence by providing guarantees on its authenticity, integrity, and timeliness, which is a key task of forensic-ready software systems. We provided support for both the internal creation of guarantees and the involvement of an external service to give as much flexibility as possible. While it is unreasonable to elicit every possible method, we opted to create extension points within BPMN4FRSS to include them once needed.

6.2 Future Work

The modelling notation is an important contribution in approaching forensic readiness from a software engineering point of view. It is an enabler for analysing forensic-ready risk scenarios, and in extension, forensic-ready software systems. While currently, the analysis is performed manually and informally, we build the BPMN4FRSS to support automated analysis, which is the objective for future work. Furthermore, BPMN4FRSS models are a key component in a bigger picture towards assurance methods for forensic-ready software systems (Daubner et al., 2020b), representing the manifestation of forensic-ready requirements within the system for validation and verification.

Semantic Validity of the model must be established in the first place (Dijkman et al., 2008). While the abstract syntax of BPMN4FRSS poses static re-

Table 3: Examples of BPMN4FRSS model semantic restrictions.

Rule	Explanation
Data Objects and Data Stores with the same name in a single model namespace are considered the same object for all intents and purposes.	This rule enables seamless linkability across multiple models into a comprehensive view.
Non-Cooperative Evidence Context must not contain Potential Evidence Sources nor Potential Evidence Types.	This rule forbids invalid definitions of potential evidence in contexts where it would be inaccessible during an investigation.
Hash Proof must be an output of an Integrity Computation Task.	This rule enforces the correct semantic of a cryptographic hash function.
PKI Timestamp Proof must be an output of Event, which has Message Flow directed to it from a PKI Timestamp Service Pool.	This rule enforces the correct usage of PKI Timestamp Service.
Related BC Timestamp Partial Proof and Timestamp Full Proof must be produced from the same BC Timestamp Service Pool.	This rule enforces the consistency of BC Timestamp Proofs.

strictions, the semantic ones are no less important. Their examples are listed in Table 3. While the semantic restrictions can be formulated in a textual way, analogous to the catalogue of mistakes in UML diagrams (Chren et al., 2019), a formal definition of semantics is desirable for automation and ambiguity reduction. For example, first-order logic could be utilised, as shown on the inter-process communication in BPMN (Houhou et al., 2019).

Model-based Risk Analysis should automatically investigate the effect of risk occurrence within the modelled scenario. The goal is to determine whether the employed forensic-ready controls are sufficient to detect the risk in question. For example, the outcome would be sequences of potential evidence generated under the default setting and under the risk, which must be different. This approach would have to combine additional modelling support to represent a cybersecurity risk in BPMN, such as Security Risk-Oriented BPMN (Matulevičius, 2017), which can be employed side-to-side with BPMN4FRSS.

Toolset is needed for the effective creation, validation, and analysis of BPMN4FRSS-enhanced models. Besides being crucial for broader adoption of BPMN4FRSS, the need for proper tool support has been identified as a software challenge for forensic-ready software systems (Pasquale et al., 2018).

ACKNOWLEDGEMENTS

This research was supported by ERDF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

REFERENCES

Alrajeh, D., Pasquale, L., and Nuseibeh, B. (2017). On evidence preservation requirements for forensic-ready systems. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, page 559–569. ACM.

Alrimawi, F., Pasquale, L., and Nuseibeh, B. (2017). Software engineering challenges for investigating cyber-physical incidents. In *2017 IEEE/ACM 3rd International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS)*, pages 34–40.

Altuhhova, O., Matulevičius, R., and Ahmed, N. (2013). An extension of business process model and notation for security risk management. *International Journal of Information System Modeling and Design*, 4:93–113.

Arlow, J. and Neustadt, I. (2005). *UML 2 and the unified process: practical object-oriented analysis and design*. Pearson Education.

Bajramovic, E., Waedt, K., Ciriello, A., and Gupta, D. (2016). Forensic readiness of smart buildings: Preconditions for subsequent cybersecurity tests. In *2016 IEEE International Smart Cities Conference (ISC2)*, pages 1–6.

Bruneliere, H., Burger, E., Cabot, J., and Wimmer, M. (2019). A feature-based survey of model view approaches. *Software & Systems Modeling*, 18(3):1931–1952.

Casey, E. (2011). *Digital evidence and computer crime*. Academic Press, 3rd ed edition.

Casey, E. and Nikkel, B. (2020). Forensic analysis as iterative learning. In *The Security of Critical Infrastructures*, pages 177–192. Springer.

Chergui, M. E. A. and Benslimane, S. M. (2018). A valid bpmn extension for supporting security requirements based on cyber security ontology. In *Model and Data Engineering*, pages 219–232. Springer.

Chren, S., Buhnova, B., Macak, M., Daubner, L., and Rossi, B. (2019). Mistakes in uml diagrams: Analysis of student projects in a software engineering course. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 100–109.

Daubner, L., Macak, M., Buhnova, B., and Pitner, T. (2020a). Towards verifiable evidence generation in forensic-ready systems. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 2264–2269.

Daubner, L., Macak, M., Buhnova, B., and Pitner, T. (2020b). Verification of forensic readiness in software development: A roadmap. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, page 1658–1661. ACM.

Daubner, L. and Matulevičius, R. (2021). Risk-oriented design approach for forensic-ready software systems. In

The 16th International Conference on Availability, Reliability and Security, ARES 2021. ACM.

Dijkman, R. M., Dumas, M., and Ouyang, C. (2008). Semantics and analysis of business process models in bpmn. *Information and Software technology*, 50(12):1281–1294.

Elyas, M., Ahmad, A., Maynard, S. B., and Lonie, A. (2015). Digital forensic readiness: Expert perspectives on a theoretical framework. *Computers & Security*, 52:70–89.

Geismann, J. and Bodden, E. (2020). A systematic literature review of model-driven security engineering for cyber–physical systems. *Journal of Systems and Software*, 169:110697.

Grispos, G., García-Galán, J., Pasquale, L., and Nuseibeh, B. (2017a). Are you ready? towards the engineering of forensic-ready systems. In *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, pages 328–333.

Grispos, G., Glisson, W. B., and Choo, K.-K. R. (2017b). Medical cyber-physical systems development: A forensics-driven approach. In *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pages 108–113.

Grobler, C. P. and Louwrens, C. P. (2007). Digital forensic readiness as a component of information security best practice. In *New Approaches for Security, Privacy and Trust in Complex Environments*, pages 13–24. Springer.

Harel, D. and Rumpe, B. (2004). Meaningful modeling: what's the semantics of "semantics"? *Computer*, 37(10):64–72.

Henley, J. (2019). Denmark frees 32 inmates over flaws in phone geolocation evidence. *The Guardian*.

Hepp, T., Schoenhals, A., Gondek, C., and Gipp, B. (2018). Originstamp: A blockchain-backed system for decentralized trusted timestamping. *it - Information Technology*, 60(5-6):273–281.

Houhou, S., Baarir, S., Poizat, P., and Quéinnec, P. (2019). A first-order logic semantics for communication-parametric bpmn collaborations. In *Business Process Management*, pages 52–68, Cham. Springer.

Kävrestad, J. (2018). *Fundamentals of Digital Forensics*. Springer.

Kruchten, P. (1995). The 4+1 view model of architecture. *IEEE Software*, 12(6):42–50.

Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., and Njilla, L. (2017). Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 468–477.

Matulevičius, R. (2017). *Fundamentals of secure system modelling*. Springer.

McKemmish, R. (2008). When is digital evidence forensically sound? In *Advances in Digital Forensics IV*, pages 3–15. Springer.

Mülle, J., Stackelberg, S. v., and Böhm, K. (2011). A security language for bpmn process models. Technical Report 9, Karlsruher Institut für Technologie.

Nwaokolo, A. O. (2020). A comparison of privacy enhancing technologies in internet of vehicle systems. Master's thesis, University of Tartu.

OMG (2010). Business process model and notation. https://www.omg.org/spec/BPMN/2.0/.

Ćosić, J. and Bača, M. (2010). (im)proving chain of custody and digital evidence integrity with time stamp. In *The 33rd International Convention MIPRO*, pages 1226–1230.

Pasquale, L., Alrajeh, D., Peersman, C., Tun, T., Nuseibeh, B., and Rashid, A. (2018). Towards forensic-ready software systems. In *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, ICSE-NIER '18, page 9–12. ACM.

Pullonen, P., Matulevičius, R., and Bogdanov, D. (2017). Pe-bpmn: Privacy-enhanced business process model and notation. In *Business Process Management*, pages 40–56. Springer.

Pullonen, P., Tom, J., Matulevičius, R., and Toots, A. (2019). Privacy-enhanced bpmn: Enabling data privacy analysis in business processes models. *Software and Systems Modeling*, 18(6):3235–3264.

Rivera-Ortiz, F. and Pasquale, L. (2020). Automated modelling of security incidents to represent logging requirements in software systems. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20. ACM.

Rodríguez, A., Fernández-Medina, E., and Piattini, M. (2007). A bpmn extension for the modeling of security requirements in business processes. *IEICE - Trans. Inf. Syst.*, E90-D(4):745–752.

Rowlingson, R. (2004). A ten step process for forensic readiness. *International Journal of Digital Evidence*, 2.

Sachowski, J. (2016). *Implementing Digital Forensic Readiness*. Syngress.

Salnitri, M., Dalpiaz, F., and Giorgini, P. (2014). Modeling and verifying security policies in business processes. In *Enterprise, Business-Process and Information Systems Modeling*, pages 200–214. Springer.

Silver, B. (2011). *BPMN Method and Style, with BPMN Implementer's Guide: A structured approach for business process modeling and implementation using BPMN 2.0*. Cody-Cassidy Press Aptos, CA, USA.

Studiawan, H., Sohel, F., and Payne, C. (2019). A survey on forensic investigation of operating system logs. *Digital Investigation*, 29:1–20.

Tan, J. (2001). Forensic readiness. Technical report, @stake, Inc.

Van den Berghe, A., Scandariato, R., Yskout, K., and Joosen, W. (2017). Design notations for secure software: a systematic literature review. *Software & Systems Modeling*, 16(3):809–831.

Weilbach, W. T. and Motara, Y. M. (2019). Applying distributed ledger technology to digital evidence integrity. *SAIEE Africa Research Journal*, 110(2):77–93.