Deliverable D2.10

Certification Graph – v1

Editor(s):	Verena Geist, Stefan Schöberl
Responsible Partner:	Software Competence Center Hagenberg GmbH
Status-Version:	Final
Date:	31.01.2025
Type:	OTHER
Distribution level (SEN, PU):	PU

Project Number:	101120688
Project Title:	EMERALD

Title of Deliverable:	Certification Graph – v1
Due Date of Delivery to the EC:	31.01.2025

Workpackage responsible for the Deliverable:	WP2 – Methodology for Knowledge Extraction
Editor(s):	Verena Geist, Stefan Schöberl (SCCH)
Contributor(s):	Angelika Schneider, Florian Wendland, Christian Banse (FHG)
Reviewer(s):	Angela Fessl (KNOW) Cristina Martínez, Juncal Alonso (TECNALIA)
Approved by:	All Partners
Recommended/mandatory readers:	WP2, WP3

Abstract:	EMERALD aims to integrate evidence collected at different levels of the cloud service into a single graph-based structure, the *Certification Graph*. This document describes the interim version of the graph, i.e. its schema (or ontology), with semantically linked and combined evidence. The development mainly involves work of T2.1 and T2.6, but also inputs of T2.2, T2.3, T2.4, T2.5, and T3.1 are considered.
Keyword List:	Knowledge graph schema, ontology extensions, security features, knowledge integration, combined evidence analysis.
Licensing information:	This work is licensed under Creative Commons Attribution-ShareAlike 4.0 International (**CC BY-SA 4.0 DEED** https://creativecommons.org/licenses/by-sa/4.0/)
Disclaimer	Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. The European Union cannot be held responsible for them.

Document Description

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	03.12.2024	First draft version, key information, and TOC.	Verena Geist (SCCH)
v0.2	05.12.2024	Executive summary, introduction, functional description, requirements fulfilment, technical specifications, licensing information, and download.	Verena Geist (SCCH)
v0.3	09.12.2024	Recap and changes, architecture, and references.	Verena Geist (SCCH)
v0.4	08.01.2025	Details on ontology extensions, delivery, and usage	Stefan Schöberl (SCCH)
v0.5	10.01.2025	Review of content	Angelika Schneider (FHG)
v0.6	13.01.2025	Finalization before internal review	Verena Geist (SCCH) Stefan Schöberl (SCCH)
v0.7	15.01.2025	Internal Review	Angela Fessl (KNOW)
v0.8	23.01.2025	Addressing internal review and improving subcomponents description and illustrative example	Verena Geist (SCCH) Stefan Schöberl (SCCH)
v0.9	27.01.2025	Final reviewed version	Cristina Martínez, Juncal Alonso (TECNALIA)
v1.0	31.01.2025	Submitted to the European Commission	Cristina Martínez, Juncal Alonso (TECNALIA)

Table of contents

List of tables

List of figures

Terms and abbreviations

AI	Artificial Intelligence
AI-SEC	AI Security Evidence Collector
AMOE	Assessment and Management of Organizational Evidence
BSI	Bundesamt für Sicherheit in der Informationstechnik
BSI C5	BSI Cloud Computing Compliance Criteria Catalogue
CertGraph	Certification Graph
Codyze	Static Code Analyzer
EC	European Commission
eknows	Platform for software analysis
eknows-e3	Extractor component developed in the context of EMERALD
GA	Grant Agreement to the project
HTTP	Hypertext Transfer Protocol
ID	Identity
KR	Key Result
MEDINA	Predecessor project of EMERALD
ML	Machine Learning
NLP	Natural Language Processing
OTP	One-time password
OWL	Web Ontology Language
PDF	Portable Document Format
Protobuf	Protocol Buffers
RDF	Resource Description Framework
RCM	Repository of Controls and Metrics
SSO	Single Sign-On
SPARQL	SPARQL Protocol And RDF Query Language
SWRL	Semantic Web Rule Language
TLS	Transport Layer Security
UI	User Interface
URI	Uniform Resource Identifier
WP	Work Package
XML	Extensible Markup Language

Executive Summary

This deliverable describes the interim version of the central *Certification Graph* schema (i.e., the *CertGraph* ontology) for storing evidence in a graph-based format and is the refinement of the initial work on designing the *CertGraph* ontology in D2.1 [1]. This ontology serves as a common structure for semantically linked and combined evidence that is filled by all evidence extraction components of WP2.

By developing the *CertGraph* ontology, this deliverable contributes to the key result CERTGRAPH (KR2) of the EMERALD project to provide a unified graph-based model of the cloud service under certification at different layers of the service. Following a knowledge graph-based approach in EMERALD, the ontology for storing and linking heterogenous evidence information is developed in WP2, and the model is then implemented as a knowledge graph in WP3.

First, this document starts with a recap of the *CertGraph* ontology from D2.1 [1] and indicates current changes. Second, the main part provides the functional and technical descriptions of the ontology, including its sub-ontologies and extensions to support the holistic approach to evidence collection. Some instructions for delivery and usage as well as current limitations are also presented. Third, a refined example of modelling and combining evidence information for TLS encryption from different sources illustrates the purpose and innovation of the ontology. Finally, the document concludes with a short summary and discussion of future work.

The main result of this deliverable is a uniform graph-based model to

(i) consolidate all extracted evidence information,
(ii) enable the retrieval of combined evidence by aggregating individual pieces of information to a higher-level viewpoint,
(iii) maintain traceability back to different information sources and extraction tools, and
(iv) provide all required concepts for resource types and security features to assess certification-relevant security metrics.

Based on this model, the uniform schema of evidence information will be further refined and analysed. The final version of the *CertGraph* ontology will then be reported in D2.11 [2], due in month 27.

1 Introduction

The *CertGraph* ontology, previously drafted in D2.1 [1], is a central graph-based model to support certification by bridging different layers and sources of extracted information, called evidence, from a cloud service. For this purpose, the ontology provides a highly structured, formal representation of a set of concepts (or classes) and their relationships and properties within the cloud service certification domain. It is created using a formal language, i.e. the Web Ontology Language (OWL), which supports complex expressions and logical inferences, including constraints, class hierarchies, and more. The main purpose of an ontology is to support knowledge sharing and reuse through structured domain knowledge as well as reasoning about the entities within the domain. In EMERALD, the *CertGraph* ontology enables harmonization of evidence gathering and assessment. Security controls defined in different schemas or catalogues are assigned to ontological concepts and those ontological types will be further used in metric definitions.

For automated compliance tools to work, suitable evidence needs to be extracted and linked. The evidence extractors developed in the EMERALD project and described in D2.2 [3], D2.4 [4], D2.6 [5], and D2.8 [6] extract and provide suitable evidence from

(i) the source code of services, often written in different programming languages, such as Java, Go, or Python (*Codyze* and *eknows-e3*[1]),
(ii) relevant parts of legal and policy documents, such as requirement or architecture documents (*AMOE*),
(iii) applied machine learning (ML) models with respect to various criteria, such as robustness, fairness, and explainability (*AI-SEC*), and
(iv) the virtual infrastructure, such as virtual machines, containers, or storage as well as runtime information, such as configuration or log files (*Clouditor-Discovery*).

The *CertGraph Ontology* with its respective extensions described in this document is a central tool to bridge those different layers and sources of evidence. Therefore, the ontology defines a vocabulary for mapping between the properties that shall be measured and the respective gathering of adequate evidence. It allows to aggregate individual aspects and fragments of information to a higher-level viewpoint of combined evidence, not previously detectable by a single tool.

1.1 About this deliverable

This document aims to describe the *CertGraph* ontology for modelling evidence information in the cloud service certification domain as a common structure for semantically linked and combined evidence. It consists of a core ontology and several sub-ontologies for capturing security features as well as domain *concepts* and *relationships* of different extensions. In this deliverable, the structure and the main concepts of the extensions to consolidate all extracted evidence information in terms of taxonomies is presented. In addition, *properties* are discussed which maintain traceability back to different information sources and extraction tools. For better illustration, we base the explanations on an example which uses one selected security criteria "encryption of data for transmission", which is specified in the BSI C5:2020[2] (CRY-02).

The *CertGraph* ontology represents the basis for integrating and instantiating the knowledge graph in the *Evidence Store* component in Task 3.1. It is also the foundation for analysing the semantic information and context of the heterogeneous evidence information in Task 2.6 to enable the retrieval of combined evidence. Another important aspect is to semantically describe

[1] Note that the component was renamed from *eknows* to *eknows evidence extractor* (*eknows-e3*)
[2] https://www.bsi.bund.de/dok/13368652

how specific *Resource Types* are related to *Security Features*, which are essential concepts to assess certification-relevant security metrics in Task 3.4.

1.2 Document structure

The document is structured as follows.

In Section 2, we give a short recap of the *CertGraph* ontology as introduced in D2.1 [1]. We also indicate any changes from the initial draft and discuss current refinements.

Section 3 provides functional and technical descriptions of the *CertGraph* ontology at the current development stage, as well as information on delivery and usage. Details on the sub-ontologies and extensions for the different cloud service layers are presented, i.e., for extracted evidence from source code, from policy documents, from ML models, and from cloud runtime environments. We further discuss refinements and limitations of concepts for combining evidence and supporting traceability, as well as for security features to assess new security metrics.

In Section 4, the illustrative example originally outlined in D2.1 [1] for modelling and combining extracted evidence information from different sources is refined.

Section 5 ends up with the conclusions, including a short summary of the content presented, open challenges, and future work.

2 Recap of the initial draft of the *CertGraph* ontology and changes

The *CertGraph* ontology introduced in D2.1 [1] is based on the *Cloud Property Graph* [7] ontology from the MEDINA H2020 project[3], which proposes a vendor-independent ontology of cloud resources and related security features. It has the major advantage that metrics (or rules) can be defined for abstract resource types and/or security features, while the extractor tools can agnostically gather evidence for these abstract concepts as well.

Figure 1 shows the different sub-ontologies and extensions of the overall *CertGraph* ontology, which together represent a unified source of types in the cloud service certification domain. The basic architecture remains unchanged. The *Core* ontology, together with the *Security Feature* ontology, builds the foundation of the ontology and contains base classes and properties. Specifically, *Security Feature* models different security related concepts. The extensions are built on top of this foundation, and each extension models the evidence gathered by a different type of extractor (i.e., *eknows-e3* / *Codyze*, *AMOE*, *AI-SEC*, and *Clouditor-Discovery*). The collected evidence from the extractors is represented as instances within a separate part that, in turn, is built upon the ontology and implemented in the *Evidence Store* (see D2.1 [1]).

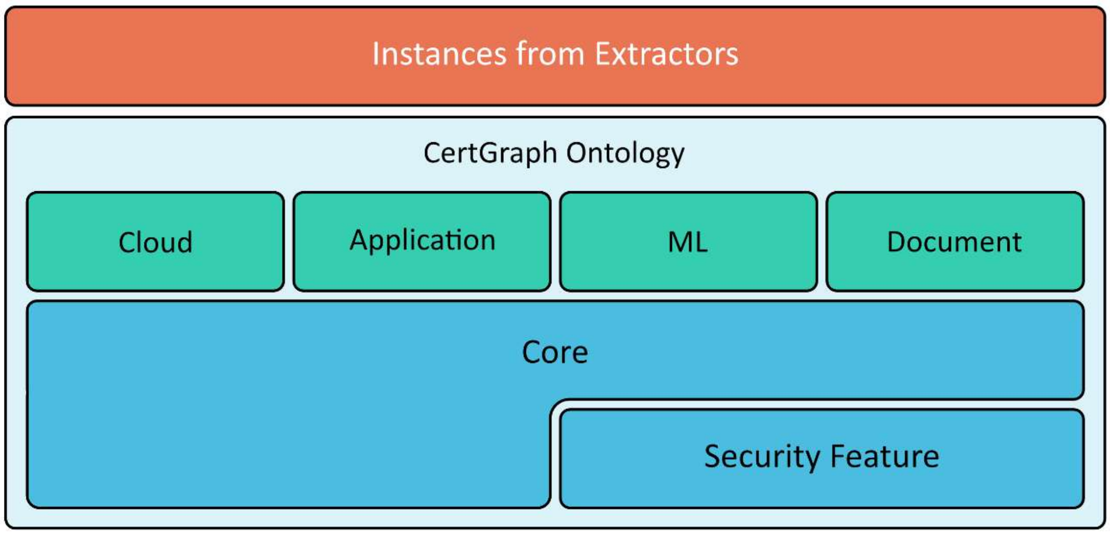

Figure 1. Initial design of the CertGraph ontology with sub-ontologies and extensions from D2.1 [1]

Changes at the current development stage concern:

- Reuse concepts from the *Cloud Property Graph* [7] to build *Core* (including reuse of security features) and *Cloud.*
- Model evidence and resources in *Core*, including further refinement.
- Refinement of *Core* into smaller sub-ontologies to improve the ontology structure.
- Refinement of extensions, in particular:
  - *Application*: Model applications including source code.
  - *Document*: Model documents, add corresponding properties.
  - *ML*: Create first draft, define relevant aspects.
  - *Cloud*: Extension and refinement of the cloud concept and its properties.

The refinement of all ontologies including extensions can be found in Section 3.2.2.

[3] https://medina-project.eu/

3 The *CertGraph* ontology (interim version)

In this section, we describe the interim version of the *CertGraph* ontology, the schema of the envisaged *Certification Graph* designed to streamline security certification which integrates evidence from multiple sources [8].

3.1 Functional description

Motivation and scope. The foundation of our knowledge graph is an ontology to store and link evidence and the fusion of extracted knowledge from different sources. We consider the complete stack from software to policies and enable the fusion of evidence from different views and sources. Its extensible ontology is designed to accommodate multiple domains, including cloud security, ML models, and source code. By providing an automated and systematic approach to extract evidence from different sources and build an ontology, the resulting *Certification Graph* aims to facilitate more effective security certification and compliance verification [8].

Requirements. In D2.1 [1], a list of requirements for developing the ontology was introduced. In the following, their respective implementation state (partially / fully /not implemented) and a brief description of how they are / will be implemented are given in tables from Table 1 to Table 7.

Table 1. REQ.01 - Formal language

Field	Description
Requirement ID	REQ.01
Short title	Formal language
Description	The ontology should be defined using a formal language that allows for the expression of concepts, relationships, instances, and axioms.
Progress	Fully Implemented – 100 %

The ontology is defined using the Web Ontology Language (OWL)[4].

Table 2. REQ.02 - Clear conceptualization

Field	Description
Requirement ID	REQ.02
Short title	Clear conceptualization
Description	The ontology should provide a clear and comprehensive conceptualization of the domain it represents.
Progress	Partially implemented – 30%

The ontology extensions need to be further refined, in particular for application, ML, and document.

Table 3. REQ.03 - Hierarchical structure of concepts

Field	Description
Requirement ID	REQ.03
Short title	Hierarchical structure of concepts

[4] https://www.w3.org/OWL/

Description	The ontology should support the creation of a hierarchical structure of concepts, allowing for subclass relationships and the organization of concepts into a taxonomy.
Progress	Fully Implemented – 100 %

The ontology extensions are modelled as taxonomies.

Table 4. REQ.04 - Reasoning and consistency checking

Field	**Description**
Requirement ID	REQ.04
Short title	Reasoning and consistency checking
Description	The ontology should be compatible with inference engines and allow for the definition of logical rules that enable automated reasoning about the concepts and their relationships.
Progress	Fully Implemented – 100 %

The selected modelling tool supports a reasoning component to derive new information based on rules and to detect inconsistencies in the ontology.

Table 5. REQ.05 - Interoperability and extensibility

Field	**Description**
Requirement ID	REQ1.05
Short title	Interoperability and extensibility
Description	The ontology should be developed in a way that ensures interoperability with other ontologies, facilitating data exchange and integration across different layers of a cloud service.
Progress	Fully Implemented – 100 %

The selected modelling tool supports the splitting of the ontology into multiple files for better structuring and linking of concepts using different namespaces, i.e., through different sub-ontologies and extension.

Table 6. REQ.06 - Documentation and annotation

Field	**Description**
Requirement ID	REQ.06
Short title	Documentation and annotation
Description	Comprehensive documentation and annotation of the ontology should be available.
Progress	Partially implemented – 20%

Documentation and annotation of the ontology needs to be improved. Only few concepts are documented yet.

Table 7. REQ.07 - Versioning

Field	**Description**
Requirement ID	REQ.07
Short title	Versioning

Description	There should be a clear strategy for handling the releases of the ontology (e.g., annually, quarterly, or on demand) and how changes and new versions are announced.
Progress	Partially implemented – 30%

Currently, the used version control tool supports versioning of OWL files and collaboration, but no strategy for releases and changes has yet been defined.

Innovation. Previous approaches [7] [9] [10] [11] that build on the notion of gathering evidence – from sources such as the cloud infrastructure – to demonstrate compliance to certain standards or regulations have several shortcomings. They perform a mapping to a structure described in an ontology to harmonize evidence gathered from various cloud providers and technologies, but they are not very comprehensive in terms of semantic modelling. For example, they mostly focus on cloud infrastructure resources. However, in a real-world certification scenario, many more resource types, such as source code, policy documents or other data assets need to be assessed. Second, previous approaches created different, independent kinds of evidence for each resource and stored them into information silos, even if they describe the same aspect (e.g., configuration of encryption), but from different viewpoints.

The knowledge graph-based approach in EMERALD will go beyond these shortcomings: First, the *Certification Graph* aims to be a systematic approach to building an ontology for security certifications spanning the complete stack from infrastructure layer, source code, data to policies and procedures [8]. By providing a schema for storing and linking the heterogeneous evidence information, EMERALD can provide a unified view of the cloud service under certification. Second, by linking generic models at different levels of abstraction the *Certification Graph* enables the development and assessment of complex mapping rules. It provides an initial approach for the fusion of evidence coming from different views/sources of the same resource [8]. This approach allows to aggregate individual aspects and fragments of information to a higher-level of combined evidence, while providing support for traceability to information sources and extraction processes.

This way, the *Certification Graph* serves as a common structure that is filled by all evidence extraction tools and can be leveraged by the assessment tools to measure security metrics relevant for certification, which makes the EMERALD knowledge graph outstanding and innovative. Assessing (partial) evidence from different sources also allows a qualitative statement about the accuracy of measured results for auditors and, furthermore, enables the comparison between specification (e.g., in policy documents) and implementation (e.g., in source code) of security features.

Fitting into overall EMERALD Architecture. How the *CertGraph* ontology fits into the overall EMERALD architecture and how it is related with the other components was already presented in D2.1 [1]. Figure 2 shows the EMERALD high-level architecture as a component diagram. There have been no changes to the component diagram so far: The extraction components for collecting evidence, i.e., *AMOE*, *Codyze*, *eknows-e3*, *AI-SEC*, and *Clouditor-Discovery* are represented at the bottom part of Figure 2. They map the extracted information to the EMERALD evidence format using the terms described in the *CertGraph* ontology. This raw evidence is then delivered to the *Evidence Store* following the defined schema and is used to assess the metrics defined in the *Repository of Controls and Metrics* (RCM).

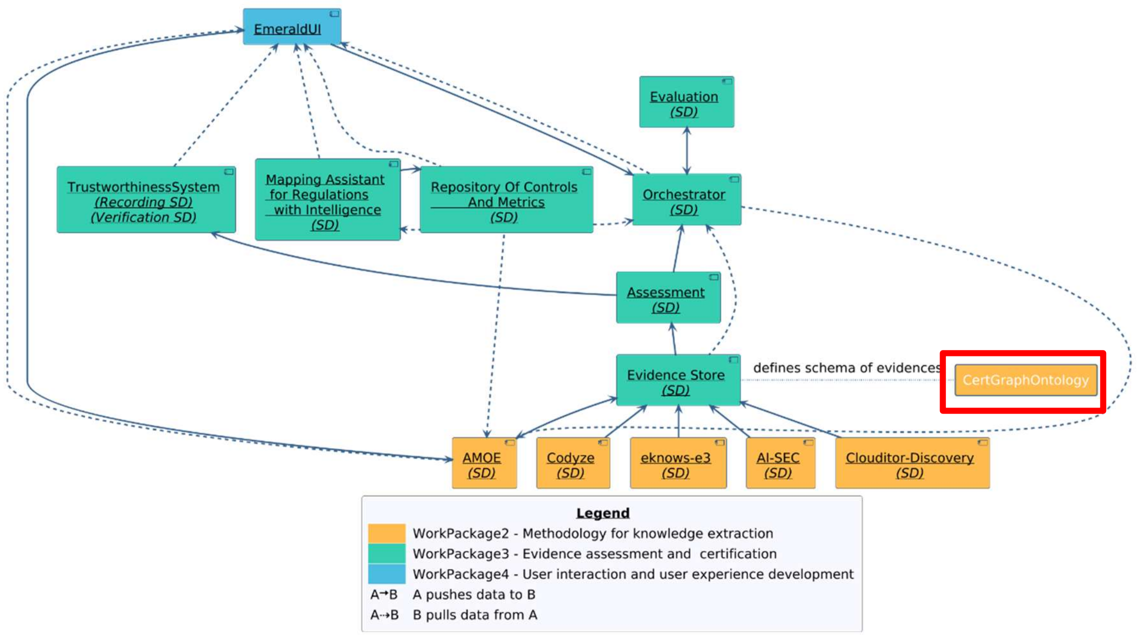
EmeraldUI
Evaluation (SD)
TrustworthinessSystem (Recording SD) (Verification SD)
Mapping Assistant for Regulations with Intelligence (SD)
Repository Of Controls And Metrics (SD)
Orchestrator (SD)
Assessment (SD)
Evidence Store (SD)
defines schema of evidences
CertGraphOntology
AMOE (SD)
Codyze (SD)
eknows-e3 (SD)
AI-SEC (SD)
Clouditor-Discovery (SD)
Legend
WorkPackage2 - Methodology for knowledge extraction
WorkPackage3 - Evidence assessment and certification
WorkPackage4 - User interaction and user experience development
A→B A pushes data to B
A⇢B B pulls data from A

Figure 2. EMERALD component overview diagram [12]

Usage. In EMERALD, the *CertGraph* ontology will be used for:

- Defining the schema of evidence to be stored in the *Evidence Store* (see D3.4 [13]).
- Preparing suitable evidence by the extraction components according to the ontology terms. The *tool Owl2proto*[5] may be used to convert the modelled ontology to an appropriate *protobuf* schema, which can be directly used in different programming languages (see D2.1 [1]).
- Defining required fields for security metrics (i.e., *Resource Type* and *Security Feature*) in the RCM (see D3.4 [13]).

3.2 Technical description

The following subsections describe the technical details of the *CertGraph* ontology.

3.2.1 Architecture

The *CertGraph* ontology consists of multiple smaller ontologies. As shown in Figure 3, five ontologies form the *Core*: *Evidence*, *Framework*, *Functionality*, *Properties,* and *Security*. Extensions are built on top of the *Core* and hook into the *Resource* taxonomy, starting at the *Resource* class defined in the *Evidence* ontology. We propose four extensions, each covering its own domain:

- A source code taxonomy (*Application*) to categorize and organize code elements based on their characteristics and functionalities.
- An organizational taxonomy (*Document*) to categorize and organize textual information from policy documents.
- An AI taxonomy (*ML*) to categorize and organize information extracted from ML models based on selected criteria.
- A cloud resource taxonomy (*Cloud*) to categorize and organize information extracted from cloud resources with application-specific runtime information.

[5] https://github.com/oxisto/owl2proto

This approach also allows for further extension of the ontology by developing new extensions for other domains, if needed. For all sub-ontologies and new extensions, customized URIs are used to avoid interoperability issues when working with multiple namespaces and combing evidence information.

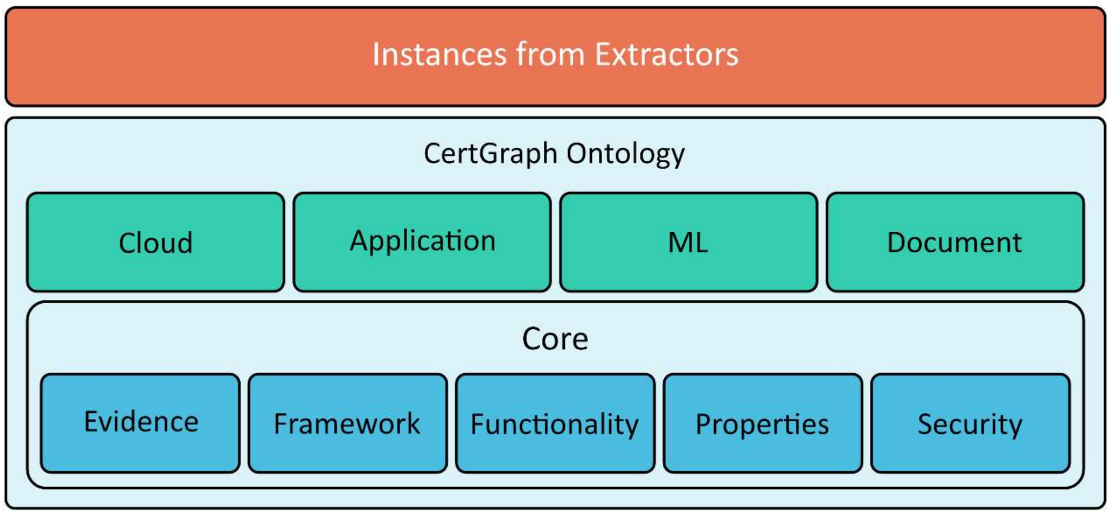

Figure 3. Updated design of the CertGraph ontology

3.2.2 Subcomponents description

This section presents the current state of the ontologies. They are modelled using Protégé[6] and stored as OWL/XML files.

3.2.2.1 *Core* – a base ontology

Core forms the base of the *CertGraph* ontology and is composed of five ontologies. From a technical perspective, this ontology is just a wrapper around the five sub-ontologies by importing them. There are two use cases for this ontology: First, when developing extensions, this ontology must be imported into the extension to reuse concepts from *Core*. Second, when instantiating the ontology, *Core* describes the (base) structure of the resulting knowledge graph.

3.2.2.1.1 *Evidence* – linking resources with security features

This sub-ontology models detected or extracted evidence regardless of the actual source. Each ***Evidence*** is connected to a ***SecurityFeature***, to a ***Tool*** (to link the extraction tool for traceability), to a ***Resource*** (to store the detection point for traceability), and to a ***CertificationTarget*** (to link to the related cloud service, for example). Furthermore, *Resource* has a connection to ***ResourceType*** (modelled as an enumeration type), to distinguish between specified and implemented behaviour.

Figure 4 shows an excerpt of the *Evidence* sub-ontology, i.e. the class hierarchy. Therefore, *SecurityFeature* is not shown in this figure (details can be found in Section 3.2.2.1.5). Further, connections (object and data properties) between classes are not shown in this diagram. An example of connecting various instances of those classes can be found in Section 4.

[6] https://protege.stanford.edu/

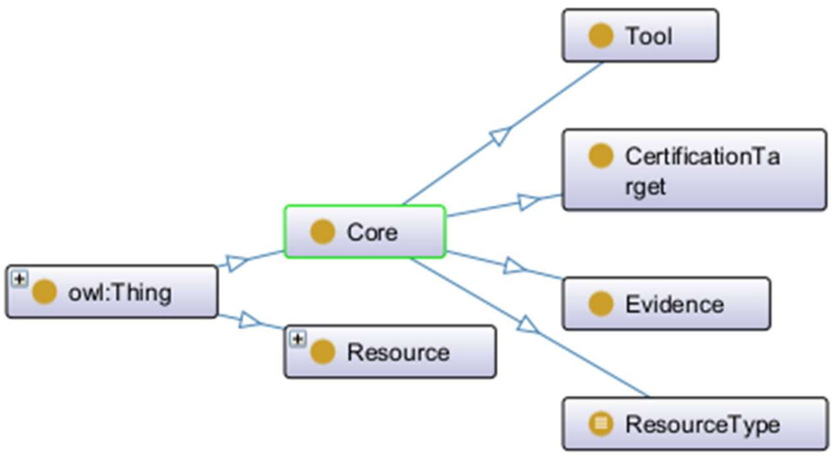

Figure 4. Excerpt of the Evidence sub-ontology

3.2.2.1.2 *Framework* – containing common types of software components

Common (high-level) types of software components are modelled in the *Framework* ontology (see Figure 5), which can be reused across different resources. This includes, for example, a ***HttpServer*** or a ***Logging*** component. This ontology is based on the taxonomy with the same name from the *Cloud Property Graph* [7].

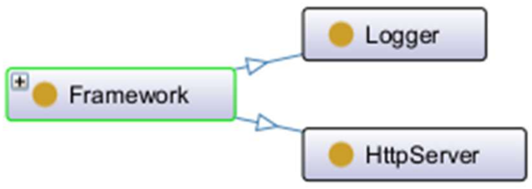

Figure 5. Excerpt of the Framework sub-ontology

3.2.2.1.3 *Functionality* – containing common data types

In addition to the high-level types (defined in *Framework*), smaller parts of software must be modelled. Also, in many parts of the *CertGraph* ontology, simple record types are needed. The *Functionality* ontology (see Figure 6) models all needed types, without restriction to a specific domain. For example, ***HttpEndpoint*** or ***HttpRequests*** are two classes in this ontology, which model smaller parts of software. On the contrast, ***CipherSuite***, for example, is used as a record type and wraps the respective properties. This ontology is based on the taxonomy with the same name from the *Cloud Property Graph* [7].

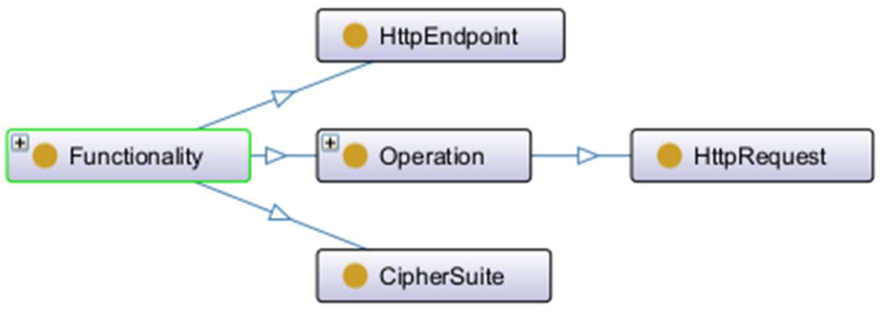

Figure 6. Excerpt of the Functionality sub-ontology

3.2.2.1.4 *Properties* – containing common object and data properties

Within the whole *CertGraph* ontology, classes are connected by object and data properties. Often, these connections are quite similar or have similar semantics. The *Properties* ontology (see excerpt in Figure 7) defines a common set of object and data properties, which can be reused across the whole ontology. This includes generic properties to model *-to-one and *-to-many relationships like ***has*** and ***hasMultiple***, and specific ones like ***filename*** or ***filetype*** to connect the respective properties to file-like classes and instances. Properties contained in this ontology are based on the ones from the *Cloud Property Graph* [7].

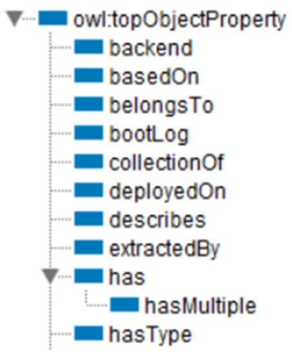

Figure 7. Excerpt of the Properties sub-ontology

3.2.2.1.5 *Security* – containing *Security Feature*

Security models security properties for all kind of domains (see Figure 8) and is based on the taxonomy with the same name from the *Cloud Property Graph* [7].

Concepts in this ontology include:

- ***Auditing*** – including anomaly detection or logging, for example.
- ***Authenticity*** – including different types of authentications like passwords, OTP or SSO, for example**.**
- ***Authorization*** – including firewalls or access control, for example.
- ***Availability*** – including backups or redundancy, for example.
- ***Confidentiality*** – including transport encryption or encryption at rest, for example.
- ***Integrity*** – including signatures or hashes, for example.
- ***Reliability*** – including ML-related scores for robustness or explainability, for example.

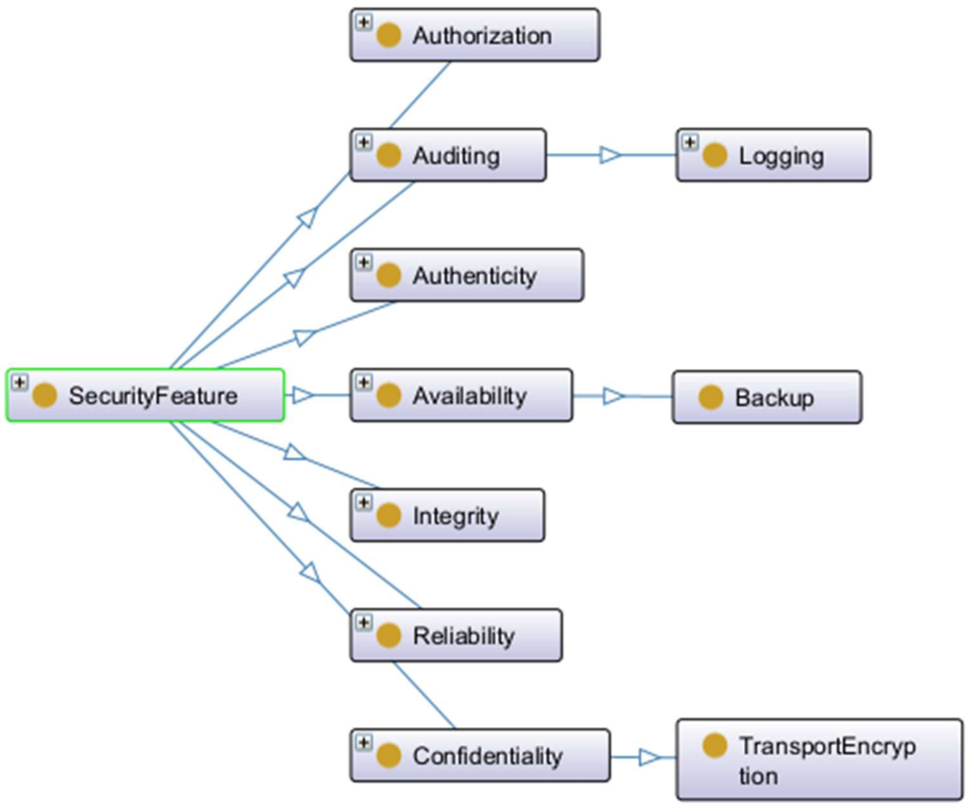

Figure 8. Excerpt of the Security sub-ontology

3.2.2.2 *Cloud* – an ontology extension for cloud resources

Cloud models cloud resources (see Figure 9), and this extension is based on the *CloudResource* taxonomy from the *Cloud Property Graph* [7].

Currently, this ontology extension is already the most developed one. High-level concepts in this ontology include, among others:

- ***CICDService*** – including jobs, and workflows.
- ***Compute*** – including different types of compute resources like containers, functions, and virtual machines.
- ***Credential*** – including certificates, keys, and secrets.
- ***Networking*** – including virtual networks, and load balancers.
- ***Storage*** – including file and data base storage.

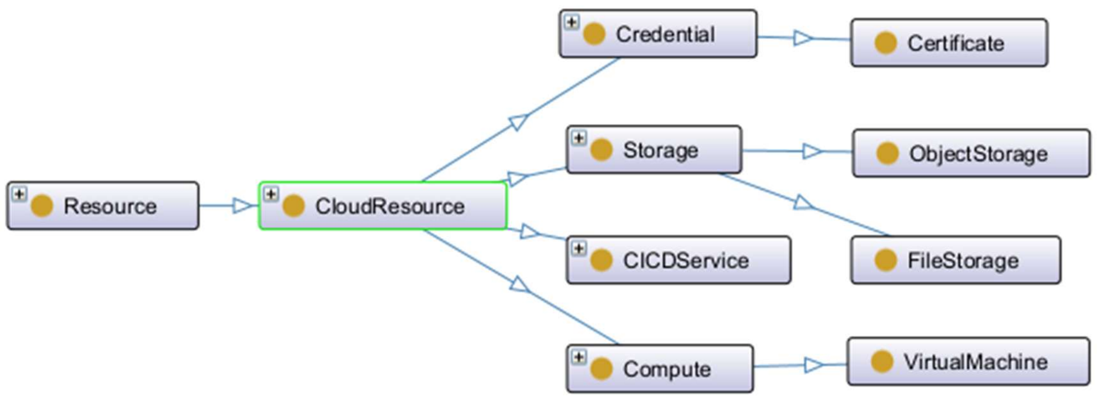

Figure 9. Excerpt of the Cloud ontology extension

3.2.2.3 *Application* – an ontology extension for source code

Application models source code and code-like artifacts (see Figure 10). A first draft of this extension, based on the genera idea of [11], is included in the current version. Still, further refinement of this extension is needed. This includes extending links to other classes and refining the abstraction level, as just storing the syntax tree would be far too detailed.

High-level concepts in this ontology include:

- ***Component*** – models large software components and forms the base class of *Application* and *Library.*
- ***Module*** – models small software components like source code files.
- ***Application*** – models source software applications and stores properties like the programming language.
- ***Library*** – describes dependencies of components.

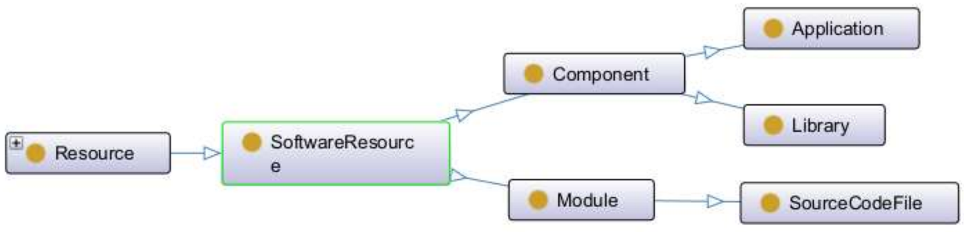

Figure 10. Excerpt of the Application ontology extension

3.2.2.4 *ML* – an ontology extension for AI/ML models

A taxonomy for assessing security-related criteria for ML models deployed in the cloud serves as a structured framework to evaluate, categorize, and mitigate potential threats and security vulnerabilities (see Figure 11). Whereas scientific work (e.g. [14]) provides a comprehensive taxonomy on deep learning techniques, in EMERALD we focus on required data to assess key criteria such as robustness against adversarial attacks and explainability (transparency and interpretability of decisions). We base our work on existing research [15] to be able to provide a generic approach that can be applied to various types of ML models.

To assess the robustness and explainability scores of a ML model, the following information is typically required in the taxonomy on a high conceptual level:

- ***MLModel*** – representing the model information, that is the model architecture (including model parameters, hyperparameters, loss function, etc.), the respective task (image recognition, NLP, etc.), the required input and output data types and formats, and evaluation metrics (such as accuracy, response time, confidence, etc.)
- ***Dataset*** – specifying the data actually used by the model, or its subset.

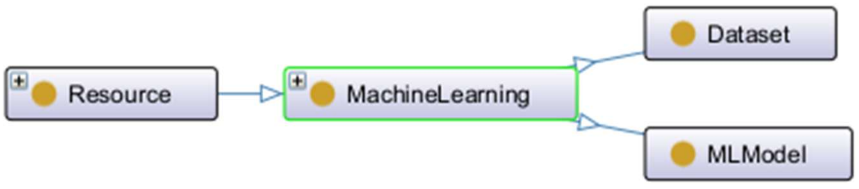

Figure 11. Excerpt of the ML ontology extension

Currently, classes contained in this ontology extension can be used to represent ML models and their context in a very high-level and abstract way. More details (e.g., properties) need to be elaborated and included into the taxonomy according to the needs and scope in EMERALD and will be documented in in the final deliverable D2.11 [2] due in month 27. Also, frameworks for Machine Learning need to be modelled. Here it still must be decided whether they are parts of the ML extension or fit better in the *Framework* sub-ontology of *Core*.

3.2.2.5 *Document* – an ontology extension for security-related documents

Creating a taxonomy for documents, which primarily contains human-readable text (see Figure 12), for automatically assessing security policies and standards involves organizing content into hierarchical or categorized groups that reflect the nature, purpose, and context of the documents [16] [17] [18] [19].

High-level concepts in this taxonomy include:

- ***PolicyDocument*** – documenting policies regarding information security, acceptable use, data protection, password, encryption, authentication, etc.
- ***SecurityAdvisoryDocument*** – documenting regulators requirements, internal guidelines, etc.
- ***ServiceMetadataDocument*** – documenting information on network security, application security, secure software development lifecycle, etc.
- ***GenericDocument*** – representing a placeholder for additional documents that are not yet modelled separately.

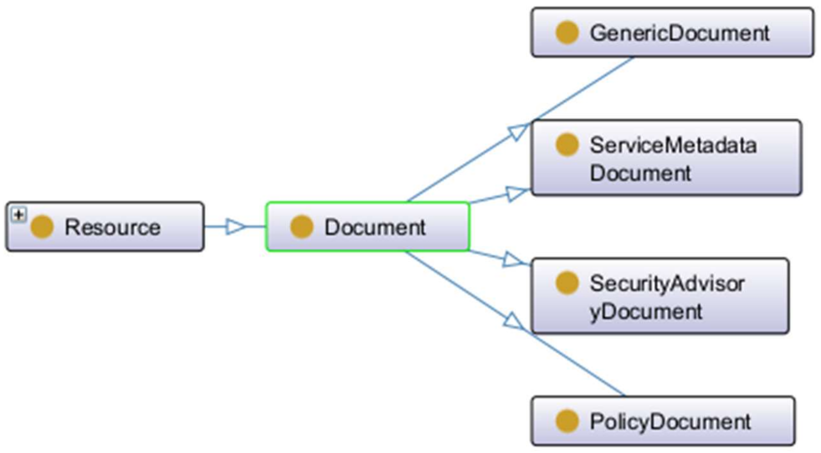

Figure 12. Excerpt of the Document ontology extension

At the time of writing, the taxonomy for security-related documents is only designed at a high level of abstraction, which will be adjusted in the final deliverable D2.11 [2] due in month 27 depending on specific needs and scope in EMERALD.

3.2.3 Technical specifications

The *Protégé*[6] and *Git*[7] tools are used to develop the *CertGraph* ontology in EMERALD. *Protégé* is a desktop application developed by Stanford university that enables the modelling of ontologies using OWL concepts. It supports the splitting of the ontology into multiple files for better structuring and linking of concepts using different namespaces, i.e., through different sub-ontologies and extensions. A reasoning component can derive new information based on rules, which is very useful for the fusion of multiple evidence parts and can detect inconsistencies in ontologies.

All sub-ontologies and extensions are saved as OWL/XML[8]. Changes are checked into the *Git* repository. The discussion and review of these changes occur via pull requests on *GitLab*, before the changes are merged into the main branch. This process ensures that the created ontologies are secured in the sense of allowing for version control, to make sure that the newly developed ontologies are discussed, and only corrected and accepted versions are merged into the main branch.

3.3 Delivery and usage

The following sub-sections detail the delivery and usage of the *CertGraph* ontology. The provided information is currently work in progress and may change.

3.3.1 Download

The *CertGraph* ontology is available from the public EMERALD GitLab repository[9] hosted by TECNALIA. The repository will host all sub-ontologies and extensions in the OWL/XML ontology format (*.owx).

[7] https://www.git-scm.com/
[8] https://www.w3.org/TR/owl-xmlsyntax/
[9] https://git.code.tecnalia.dev/emerald/public/certgraph

3.3.2 Package information

Table 8 shows the structure of the Gitlab repository[9] and its contents.

Table 8. Overview and description of package structure for the CertGraph ontology

Folder / File	**Description**
`emerald.owx`	Entry point of the whole ontology, which can be used to open it in *Protégé*, for example (used for convenient development within the EMERALD project)
`core.owx`	*Core* ontology. In addition, it imports the five core ontologies below
`core/evidence.owx`	*Evidence* Ontology
`core/framework.owx`	*Framework* Ontology
`core/functionality.owx`	*Functionality* Ontology
`core/properties.owx`	*Properties* Ontology
`core/security.owx`	*Security Feature Ontology*
`resource.owx`	Wrapper ontology, which imports the four extension ontologies below (used for convenient development within the EMERALD project)
`resource/infrastructure.owx`	Ontology extension for cloud resources
`resource/application.owx`	Ontology extension for source code
`resource/ml.owx`	Ontology extension for machine learning models
`resource/document.owx`	Ontology extension for documents

3.3.3 Instructions for use

Instructions for use are provided as part of the README in the public GitLab repository[9].

In summary, following requirements must be met before using the *CertGraph* ontology:

- To explore the ontology, an ontology modelling tool that supports the OWL ontology format, such as *Protégé*, must be installed.
- If the ontology should be extended or changed, a version control tool, such as Git, is recommended.

The whole ontology can be viewed in *Protégé* by opening `emerald.owx`. In addition, if only parts of the ontology are to be viewed, the respective owx file can also be opened on its own.

To instantiate the ontology, the following workflow is recommended:

1. Create a new ontology file.
2. Import `core.owx` using the *Imported Ontologies* view.
3. Import relevant extensions from the resource folder or all of them by importing `resource.owx`.
4. Created instances will be stored in the newly created ontology file.

Note that the *CertGraph* ontology will not be visualized in the EMERALD UI.

3.3.4 Licensing information

The *CertGraph* ontology and its sub-ontologies and extensions are licensed as open source under Apache License, Version 2.0.

3.4 Limitations and future work

The *CertGraph* ontology with its sub-ontologies and extensions will continuously be extended in the course of the project. Furthermore, some design decisions are not final and are still under discussion. This includes, but is not limited to, connections between classes in general or new classes required for describing the extension domains. The ontology is constantly being further developed, in particular, the *Application* and *ML* extensions require more extensive refinement.

To meaningfully fuse the knowledge, which is provided by the evidence extraction tools, we have discussed several ideas on how to accomplish this. One idea is to use SWRL[10] or similar languages to describe rules, which are used to derive new knowledge from gathered evidence, thus new edges are added to the graph, which in turn leads to denser interlinking of data. In this context, it has already become apparent that a unique ID is necessary to identify service instances (i.e., each service can be referenced by a unique URI across extractors). Another idea is to use SPARQL[11] to query the graph and in this way to link the information in the graph and receive it as a query result. Currently, we are evaluating what can be implemented, which libraries are available, and what is supported by the used graph database.

At the time of writing, the implications of each decision cannot yet be entirely estimated, and the structure of the *CertGraph* ontology will continue to evolve. The results will be reported in the upcoming deliverable D2.11 [2].

[10] https://www.w3.org/submissions/SWRL/
[11] https://www.w3.org/TR/sparql11-query/

4 Refined illustrative example of modelling and combining evidence information for the used TLS Version

For better illustration, we base the idea for modelling and combining evidence information from different sources on an example (see Figure 13), which uses a selected security criteria: “Encryption of data for transmission”, which is specified in the BSI C5:20201 (CRY-02). In this example we model the used TLS (*Transport Layer Security*) version from different views.

This is a refinement of the first idea drafted in D2.1 [1]. The focus is on illustrating interconnectivity between selected sub-ontologies and extensions, and not on contained details. The restructuring and extension of the *CertGraph* ontology is reflected in Figure 13. In the diagram, classes are visualized as rectangles and instances as hexagons. Open-headed arrows with a filled line (→▹) represent “subclass of” relations, which connect subclasses to their parent class, and open headed arrows with a dashed line (┈▹) represent “instance of” relations, which connect instances to their class. Simple arrows (→) represent data and object properties. These arrows are used between classes to define the schema, as well as between instances in their materialized form.

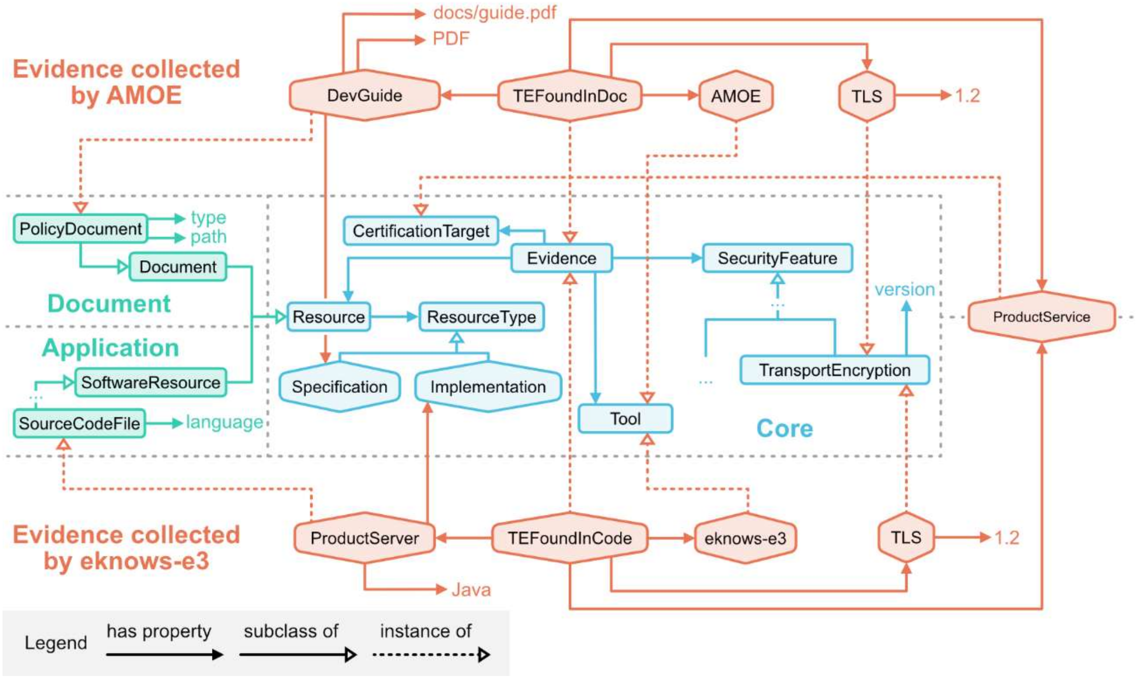

Figure 13. Classes (rectangles) and instances (hexagons) for the TLS example, showing evidence found in source code (implemented) and corresponding evidence in a document (specified) regarding transport encryption, which can be used to verify CRY-02 from BSI C5:2020 (adapted from [9])

As described in Section 3.2.2, the ontology ***Core*** forms the basis for the *CertGraph* ontology. It defines the metamodel for EMERALD evidence and uses the concepts defined in the *Security Feature* sub-ontology, which contains a variety of security features and data properties:

- ***Evidence*** is the central class and instances of it represent detected security evidence. Each evidence has connections to *SecurityFeature, CertificationTarget, Resource,* and *Tool*.
- ***Resource*** represents the source of a piece of evidence and stores relevant metadata for the location. Each *Resource* has a connection to an *ResourceType*.
- ***ResourceType*** classifies the role of resource within the system. *ResourceType* is modelled as an enumeration type in ontology terms. For this, a class is needed, and an

instance is created for each possible variant. Currently, we distinguish between these two possible variants:
  - The first variant, ***Specification***, is used for evidence found in resources, which describe how the system *should* behave. The main application for this variant is in human-readable documents which are not automatically processed for compilation, e.g. policy documents.
  - The second variant, ***Implementation***, is used for evidence found in resources, which describe, how the system *actually* behaves. This variant is mainly used for evidence found in machine-processed assets, e.g. source code, configuration files, or runtime information.
- ***CertificationTarget*** ties the evidence to a certain service. This connection enables the fusion of evidence from multiple sources using a unique identifier for each service, which will be used as URI for the service instance.
- ***Tool*** represents the extractor component that has collected the evidence.
- To keep things simple, only a single feature (***TransportEncryption*** class) is showcased in this example and the hierarchy has also been simplified to two levels. The base class of this hierarchy is ***SecurityFeature***. Also, for simplicity reasons, just one data property ***version*** is shown to store the TLS version.

Resource (defined in *Core*) is the starting point for ontology extensions. In this example, we used the ***Document*** and ***Application*** extensions, which are built on top of the *Core* ontology, and limit the scope to just one class per extension. As previously described, the classes in the extensions should model their respective domains. The following two classes are used in the example:

- ***PolicyDocument*** represents a human-readable textual document for policies. It is modelled as a sub-class of ***Document*** and includes (*has*) two shown data properties ***type*** and ***path***.
- ***SourceCodeFile*** represents a source code file which is compiled for a given service and is stored in a repository. It is modelled as a (indirect) sub-class of ***SoftwareResource*** and includes (*has*) a data properties ***language.***

Gathered evidence provided by tools is modelled by instantiating classes defined in *Core* and in extensions. In the example in Figure 13, evidence for the certification target ***ProductService*** is provided by two extraction components:

- ***AMOE*** scanned the ***DevGuide*** (a PDF document stored at *docs/guide.pdf*) and found that ***TLS*** version *1.2* is required to be used in development.
- ***eknows-e3*** scanned the ***ProductServer*** (written in *Java*) and found that ***TLS*** version *1.2* is used in the implementation.
- Found evidence is represented as the instances ***TEFoundInDoc*** and ***TEFoundInCode***, which have respective connections to the other instances. Please note that “TransportEncryption” is abbreviated as “TE” in the diagram.

To sum up, the example illustrates the key idea of the *Certification Graph* to represent security-related parts of a cloud service, e.g. of the source code, in a graph structure and provide additional context through the discovery of other related cloud resources, e.g. policy documents. Bridging different domains allows to combine evidence at a higher level of knowledge and enables a comparison, for example, of what is described in policy documents and what is actually implemented in software. An important point for maximising the potential of the *Certification Graph* is that evidence from other extraction components must link to the same service instance. In OWL, two instances are considered as *the same* if they are identified by the same URI. This enables knowledge fusion later on for the assessment in WP3.

5 Conclusions

In this deliverable, we described the interim version of the *CertGraph* ontology, which is the central schema for integrating evidence extracted from multiple cloud service layers (i.e., infrastructure, platform, and software), including policy documents, ML models, and runtime information, into a single graph-based structure (KR2-CERTGRAPH).

Based on the general idea of (harmonized) security metrics, the *CertGraph* ontology allows different evidence collection tools to gather and combine different kinds of evidence for the same metric, enhancing reuse of evidence collected, and providing answers to assess the metrics. We provided functional and technical descriptions of the ontology, including its sub-ontologies and extensions to support the holistic approach to evidence collection. These include security features to assess security metrics, as well as different domain concepts and relationships for extracting evidence from source code, from policy documents, from ML models, and from cloud runtime environments. We further discussed refinements and limitations of the current status of the *CertGraph* ontology and presented instructions for its delivery and usage. An example of modelling and combining evidence information for TLS encryption (specified in the BSI C5:20201 (CRY-02)) was refined and presented to illustrate the purpose and innovation of the ontology.

Next steps will include further formalization of concepts like the *Document* and *ML* extensions. We are also looking for collaborations with other domains that can be included in the ontology as well. Furthermore, the fusion of knowledge must be modelled and implemented in software, whereby it must be evaluated in advance, which formalism is supported by libraries and databases. Accordingly, the uniform schema of evidence information will be further refined and analysed. The final version of the *CertGraph* ontology will then be reported in D2.11 [2] due in month 27.

6 References

[1] EMERALD Consortium, “D2.1 Graph Ontology for Evidence Storage,” 2024.

[2] EMERALD Consortium, “D2.11 Certification Graph– v2,” 2026.

[3] EMERALD Consortium, “D2.2 Source Evidence Extractor – v1,” 2024.

[4] EMERALD Consortium, “D2.4 AMOE – v1,” 2024.

[5] EMERALD Consortium, “D2.6 ML model certification – v1,” 2024.

[6] EMERALD Consortium, “D2.8 Runtime evidence extractor – v1,” 2024.

[7] C. Banse, I. Kunz, A. Schneider and K. Weiss, “Cloud property graph: Connecting cloud security assessments with static code analysis,” in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD) (pp. 13-19)*, 2021.

[8] S. Schöberl, C. Banse, V. Geist, I. Kunz and M. Pinzger, “CertGraph: Towards a Comprehensive Knowledge Graph for Cloud Security Certifications,” in *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (pp. 76-77)*, 2024.

[9] L. Orue-Echevarria, J. Alonso and et al., “MEDINA: Improving Cloud Services trustworthiness through continuous audit-based certification,” in *CEUR Workshop Proceedings*, 2021.

[10] C. Banse, I. Kunz, N. Haas and A. Schneider, “A Semantic Evidence-based Approach to Continuous Cloud Service Certification,” in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (pp. 24–33)*, 2023.

[11] I. Kunz, K. Weiss, A. Schneider and C. Banse, “Privacy property graph: Towards automated privacy threat modeling via static graph-based analysis,” in *Proceedings on Privacy Enhancing Technologies*, 2023.

[12] EMERALD Consortium, “D1.1 Data modelling and interaction mechanisms - v1,” 2024.

[13] EMERALD Consortium, “D3.4 Evidence Assessment and Certification-Implementation - v2,” 2025.

[14] I. Sarker, “Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions,” *SN computer science,* vol. 2, no. 6, p. 420, 2021.

[15] Y. Cai and G. Wunder, "On Gradient-like Explanation under a Black-box Setting: When Black-box Explanations Become as Good as White-box. arXiv preprint arXiv:2308.09381," 2023.

[16] J. H. Eloff, R. Holbein and S. Teufel, “Security classification for documents,” *Computers & Security,* vol. 15, no. 1, pp. 55-71, 1996.

[17] S. W. Lee, R. Gandhi, D. Muthurajan, D. Yavagal and G. J. Ahn, "Building problem domain ontology from security requirements in regulatory documents," in *Proceedings of the 2006 international workshop on Software engineering for secure systems, pp. 43-50*, 2006.

[18] U. Garain and B. Halder, "Machine authentication of security documents," in *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition, pp. 718-722*, 2009.

[19] G. Liu and H. Zhang, "An ontology constructing technology oriented on massive social security policy documents," *Cognitive Systems Research,* vol. 60, pp. 97-105, 2020.