

The Design and Implementation of a Semantic Web Framework for the Event-Centric Digital Forensics Analysis

Pavel Chikul¹^a, Hayretdin Bahşi^{1,2}^b and Olaf Maennel¹^c

¹Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia

²School of Informatics, Computing, and Cyber Systems, Northern Arizona University, U.S.A.

Keywords: Digital Forensics, Event Reconstruction, Knowledge Extraction, Forensic Timeline, Forensic Ontology, IoT.

Abstract: In the era of interconnected devices, digital crime scenes are characterized by their complexity and voluminous data from a plethora of heterogeneous sources. Addressing these twin challenges of data volume and heterogeneity is paramount for effective digital forensic investigations. This paper introduces a pioneering automated approach for the nuanced analysis of intricate cyber-physical crime environments within distributed settings. Central to our method is an event-centric ontology, anchored on the globally recognized UCO/CASE standard. Complementing this ontology is a robust software framework, designed to expedite data extraction processes, and ensure seamless interfacing with the knowledge repository. We demonstrate the usage of the framework on a public dataset, encapsulating a realistic crime scenario populated with diverse IoT devices.

1 INTRODUCTION

Digital forensics is a domain that continually faces challenges arising from the increasing complexity and volume of data across heterogeneous sources. Traditionally, experts in this field grapple with vast amounts of data, using many extraction and analysis tools to weave together insights. These challenges are only compounded by a shortage of experts who possess the necessary skills to navigate this intricate landscape.

With their advanced sensing capabilities, Internet of Things (IoT) systems gather, transmit, and process a significant amount of data related to various physical phenomena. Consequently, the data from these systems can be invaluable not just for cybercrimes but for conventional crime investigations as well. However, the inherent interconnectedness of IoT devices and the vast volume of data they produce intensify traditional forensic challenges. Valuable forensic data is often dispersed across multiple system components, necessitating sophisticated correlation analyses of artifacts obtained from diverse sources.


Ontologies present a promising solution. They encapsulate expert knowledge in a semantic representa-


tion, comprising concepts and their interrelationships. This representation facilitates human-machine interaction, enabling both semi- and fully automatic inference of new knowledge. When paired with machine-assisted pre-processing, ontologies can assimilate raw data from diverse sources, streamlining forensic investigations.


In an IoT environment, it is vital to consider and correlate the pieces of evidence obtained from different devices and other system components such as hubs, edge devices, mobile devices, conventional computers, and cloud sources. There exists a line of research regarding the event reconstruction based on the time attributes (Debinski et al., 2019; Esposito and Peterson, 2013; Hargreaves and Patterson, 2012), however, they usually concentrate on time information without using other semantic relations and they do not provide an extensible and well-structured framework when compared to ontologies.

Our contribution, through this paper, addresses these gaps. We tackle the source heterogeneity and data volume problems within IoT environments, proposing a unified framework that facilitates automatic data extraction from various sources and representing the extracted data in a standardized ontology format, aligned with the UCO/CASE specification to ensure robust interoperability and flexibility. Our system comprises three integral steps:

1. **Data Extraction.** Efficiently gleaning data from

^a <https://orcid.org/0000-0002-2846-9391>

^b <https://orcid.org/0000-0001-8882-4095>

^c <https://orcid.org/0000-0002-9621-0787>

evidence sources.

2. **Knowledge Aggregation.** Accumulating and preprocessing the collected data.
3. **Interface Layer.** Querying and visualizing the acquired knowledge.

To demonstrate the usage of the proposed method, we apply our system to the IoT forensic dataset from the DFRWS 2018 challenge. Our results demonstrate the efficacy of our solution, emphasizing how semantic relations between diverse IoT device evidence can be leveraged for efficient forensic analysis.

The structure of this paper is as follows: Section 2 reviews the literature regarding the application of ontologies to the digital forensic area. We then follow with the methodology and discussion of the proposed framework in Section 3, and the demonstration of the application is presented in Section 4. Section 5 concludes the study.

2 RELATED WORK

A typical process of a digital forensic investigation follows the steps of identification/seizure, acquisition/preservation, analysis, and presentation of evidence data. The analysis step retrieves forensics artifacts from the preserved copies of evidence and tries to validate or invalidate the hypothesis developed for explaining the crime or incident details. Event information, which represents an abstract single action of a crime actor at a given time with a duration (e.g., accessing the webpage, editing a document, sending an email), is deduced from the artifacts and incorporated into a timeline analysis (Chabot et al., 2015a). Event heterogeneity is defined under three criteria, the varying formats of data source, assigned temporal value, and its semantic property that changes depending on the context (Chabot et al., 2015a). Thus, a forensic approach that deals with heterogeneity is required to address these three criteria coherently.

The event analysis can be conducted in varying time horizons (Debinski et al., 2019), micro or nano timelines cover shorter time periods (Carvey, 2015) whereas the super timeline encompasses a wider period (Esposito and Peterson, 2013). The events can be created at different abstract levels. For instance, a high-level event such as a USB connection is identified via pre-determined rules and visualized in (Hargreaves and Patterson, 2012).

A literature review of the studies regarding the ontologies in the digital forensics domain is given in (Sikos, 2020). Although standardization of ontologies is considered a future research direction (Javed

et al., 2022), various taxonomies and ontologies are proposed for different purposes.

The categorization of the forensic techniques in the form of taxonomies is provided for the identification, acquisition, and analysis steps of the digital forensic investigation process (Ellison et al., 2019). Technological and professional aspects are covered in another taxonomy (Brinson et al., 2006). Forensic disciplines and sub-disciplines are categorized with the corresponding evidence resources in (Karie and Venter, 2014). An ontology for the categorization of digital forensics tools and exploration of their relations with others is proposed in (Wimmer et al., 2018). Although the tooling and forensic technique aspects are not addressed in the present paper, our ontology can be extended with the proposal of cited studies to give more insight into the traceability of the investigation.

The terms related to incident response in SCADA systems are introduced in (Eden et al., 2015). Other conceptualization studies in the form of proper ontology structure have addressed the investigation process (Park et al., 2009). A more complete digital forensic ontology extending CybOX (Barnum et al., 2020), a language for representing the digital artifacts in a wider domain including intrusion detection, cyber threat intelligence, and incident handling, is proposed in (Casey et al., 2015). It is important to note that the ontologies addressing different aspects of the problem domain can be integrated with each other in the form of a meta-model to handle the domain complexity. The discussion regarding which meta-modeling approach would be much more suitable for the digital forensics domain is given in (Ameerbakhsh et al., 2021).

The contribution of all the papers given above remains at a conceptual level rather than a complete implementation of a reasoning system. On the other side, despite being limited, there exist other studies that propose semantic web implementations evolved around their ontologies. A comprehensive ontology that covers the technical and process aspects of digital investigations is given in (Chabot et al., 2015b). This study also proposes a semantic web framework that performs event reconstruction and enhances knowledge. The limitation of this study is that the ontology and the semantic framework conduct the correlation among the various data sources belonging to one evidence source. Our study aims to accumulate knowledge by correlating different devices in an IoT environment.

A general ontology for digital investigation is proposed in (Kahvedžić and Kechadi, 2009). This study details only the analysis of the Windows registry. The

same ontology is utilized for analyzing files and their metadata in (Kahvedžić and Kechadi, 2010). Another framework implementation addressed the analysis of files with NLP techniques (Amato et al., 2020a). Investigation ontologies are formed for the analysis of data obtained from online social networks (Elezaj et al., 2019; Turnbull and Randhawa, 2015). A noteworthy study applies ontology and knowledge representation framework for malware detection (Ding et al., 2019). This study uses the data collected from sandboxes (i.e., the platforms that can be used for collecting data about the behavior of malware) to populate the framework. The ontological framework has been applied for other similar problem domains such as cyber security threat modeling (Välja et al., 2020).

Of all these works the one that stands out is the Cyber Investigation Analysis Standard Expression (CASE)(Casey et al., 2017; Casey et al., 2018). It builds on top of and extends the UCO (Casey et al., 2015) and provides a standardized ontology form for managing cyber investigations. CASE defines domain-centric components that are on a higher level of abstraction than the UCO itself and can be thought of as the next step in the evolution of cyber ontologies. The major definitions that CASE provides are investigative actions, different investigation Action Lifecycles, and provenance records. These concepts allow the investigative entities to describe the course of investigation in a forensically sound manner. The wide community support of this ontology makes it a prominent candidate for our system development.

From the current studies in the field, it becomes evident that operating on unstructured data like plain text or CSV spreadsheets will become more tedious with time due to growing volumes and complexity of evidence source interconnections. Several works (Wimmer et al., 2018; Casey et al., 2017; Chabot et al., 2015b) note that traditional methods often lack a formal structure, making it difficult to organize and represent the data in a meaningful way. This can lead to difficulties in understanding and interpreting the information. Plain text and CSV files have limited expressiveness, which means they may not be able to capture the full complexity and richness of the data. This can result in a loss of important details and nuances during the reconstruction and analysis process. One other drawback of unstructured data is its lack of interoperability making it difficult to integrate data from different sources or collaborate with other investigators. This can hinder the overall efficiency and effectiveness of the investigation. On the other hand, "ontologies provide a formally explicit specification for the ontology as well as a rich and exten-

sive ecosystem of technology support for serialization, transformation, semantic mapping and semantic querying" (Casey et al., 2017). Chabot et al. state that "unlike more rudimentary data formats, ontology can represent relationships between entities in addition to the underlying logic of data. The explicit and formal nature of ontology facilitates the design and the use of interpretation and analysis tools" (Chabot et al., 2015a).

Although various ontologies are proposed in the digital forensics domain and the need for them is identified by the studies, there are limited works that implement a working semantic web-based solution (Chabot et al., 2015b; Casey et al., 2015; Casey et al., 2017; Chikul et al., 2021). Although the work by Chabot et al. aims to deal with data source heterogeneity, its focus is on one evidence source (e.g., a hard disk image) with the scope of correlating logs originating from the various file system and OS components (e.g., web history, event logs, or volatile memory content). This study also offers a way of standardizing the unified representation form for such evidence.

In (Chikul et al., 2021), challenges related to the heterogeneity of data sources and the reduction of data volume were partially addressed. The system in focus, ForensicFlow, introduces an automated methodology for extracting artifacts from various sources, including both volatile and non-volatile memory. This system also aids in the reconstruction of event-artifact graphs. However, there are limitations in its ontological representation, particularly in terms of flexibility. It falls short in accommodating the diverse range of artifacts, artifact families, associated actions, and the additional metadata that surrounds them.

On the other hand, the works by Casey et al. provide a sophisticated standard ontology to describe complex knowledge stores in virtually any cyber-related domain but do not standardize the ways of automatic artifact extraction and analysis (Casey et al., 2015; Casey et al., 2017).

3 METHODOLOGY AND DESIGN

In this section, we provide a detailed overview of the system design and implementation specifics that address the major challenges identified in Section 2. The system we present in this paper consists of four major blocks, namely the extraction layer, knowledge aggregation, ontology, and the communication interface. The schematics representing the high-level overview of the system can be observed in Figure 1.

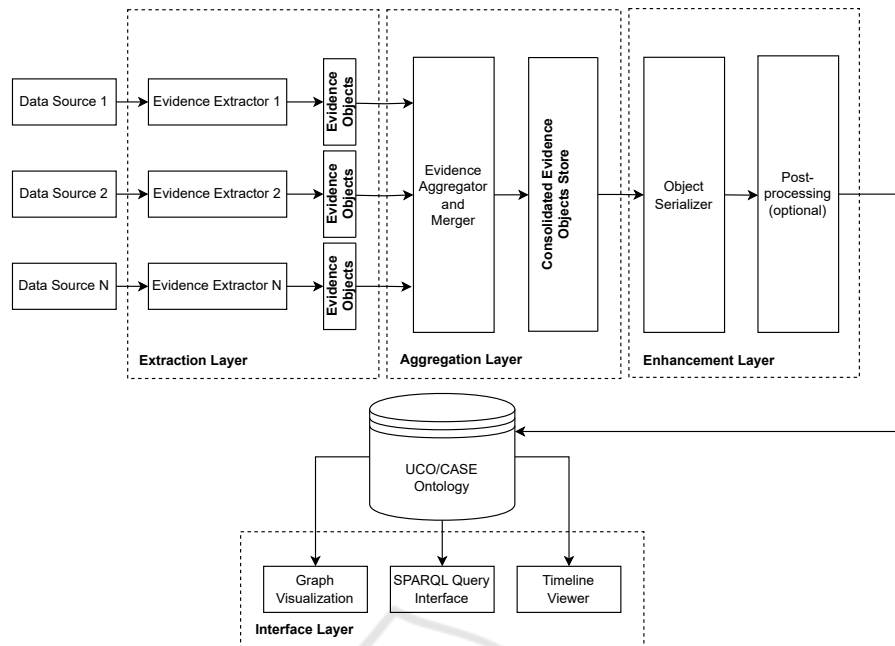


Figure 1: High Level System Overview.

3.1 Ontology Design

The core of the system is the ontology-based knowledge store that defines generic entity classes and their mutual connections. Considering the amount of previous work done in that domain, it was decided not to design the ontology from scratch but rather adopt an already existing solution. As mentioned in (Casey et al., 2018) the approach we exercise is to “integrate rather than duplicate: build on existing standardized representations, rather than create a separate one, to avoid redundancy and duplication of effort”. From the range of currently available solutions, we concentrated on the Unified Cyber Ontology (UCO) (Casey et al., 2015) and Cyber Investigation Analysis Standard Expression (CASE) (Casey et al., 2017) for storing the digital forensic entities (evidence, data sources, provenance information, etc.) and their relationships. As stated in (Casey et al., 2017) “UCO could be thought of as a collection of building blocks and parts, e.g., big blocks, little blocks, seats, tables, windows, wheels”. It is flexible enough to represent cyber environments in various domains such as incident handling, malware analysis, and security operations. CASE on the other hand takes these building blocks to build and operate in the domain of cyber-investigation. Currently, UCO provides five base ontologies with four of them meant for inter-domain foundation (`uco:core`, `uco:action`, `uco:observable`, and `uco:victim`) and one domain-

specific (`uco:investigation`). In this work, we utilize only two of them: `uco:core` and `uco:observable`, since we concentrate on the observable traces extraction and analysis. However, it is planned to expand the system to include more investigation-related facts in the future. Two key components that are strongly adopted in our system are `core:Relationship` and `core:ConfidenceFacet`. These are used to create the semantic linkage between the extracted observables and define the level of certainty for such links.

The serialization method was chosen in favor of RDF/XML syntax which is different from the Turtle format native to both UCO and CASE. This choice was made in order to more easily automate the extraction and preprocessing of the data in Python language. It is important to note here that since the ontology is fully Resource Description Framework/Web Ontology Language (RDF/OWL) specification compliant, the serialization format can be switched to any of the supported ones (JSON-LD, XML, protocol buffers, etc.).

3.2 Information Extraction

The extraction layer consists of an arbitrary number of distinct extraction modules that operate on a single source of evidentiary data. The data source can be a database, a log file, or a binary memory dump. These data sources are converted to a standardized evidence representation to be later combined into the ontolog-

ical form. To achieve a level of flexibility that could accommodate almost any type of data source and extract a wide range of entities in the cyber domain, a Python-based framework was developed.

The base of the framework consists of a selection of UCO/CASE class wrappers that are relevant to the framework specifics, such as *observable:EventRecord*, *observable:Device*, *identity:Person*, and many others. The class hierarchy of the wrappers represents a 1-to-1 mapping of the original UCO/CASE structure for consistency and ease of maintenance. Every class wrapper is derived from one base class named *RdfEntityBase* that populates two methods: *print()* that simply prints out an object's properties and *to_rdf()* that must return a serialized XML node with the object's data. It is worth mentioning that the *RdfEntityBase* class defines two functions that are used later on in the aggregation stage: an equality function (*__eq__()*) and a hash function (*__hash__()*). These functions define the comparison between the instances of the classes and can be overridden if needed by subclasses. We kept the UCO Facet extension approach to maintain the versatility of the original standard. UCO Facets represent groupings of properties that characterize a specific aspect of an object. An example of Facets can be a file entity that can have some basic file properties (name, path, size, etc.) described in an instance of *facet:FileFacet* and some content descriptive data (entropy, hashes, magic number, etc.) stored in a *facet:ContentDataFacet* structure. Facets can be optionally assigned to any object derived from the base UCO *core:UcoObject* class. If any specific data or object property is needed for some custom event a new class can be derived from the base one and the new custom field should be added to the initializer. After that, the only thing that is needed is to add a serialization of the field to the node populated by the parent. An example implementation of a custom Event Facet class that introduces a new string-based data property named *custom_field* is demonstrated in Figure 2.

The *RdfExportBase* is a common base class for all the extractor modules in the system. The instance of the extractor class receives a data source location, (a path to a database file, memory dump, etc.), and optional parameters configuration to filter the events (e.g. a time frame). All the derived classes are to implement a single method - *extract()*. This method is responsible for the extraction of the events and related artifacts, as well as the generation of the initial object relationships. It returns a list of *RdfEntityBase*-derived objects that represent the ontology for the scope of the processed data source. Each extractor

```
class CustomEventFacet(EventRecordFacet):
    def __init__(self, application: Application,
                 computer_name: str = None,
                 cyber_action: CyberAction = None,
                 event_id: str = None,
                 event_text: str = None,
                 event_type: str = None,
                 created_time: datetime = None,
                 custom_field: str = None) -> None:
        super().__init__(application, computer_name,
                         cyber_action, event_id, event_text,
                         event_type, created_time)

        self.custom_field = custom_field

    def to_rdf(self, root_node: et.Element) -> et.Element:
        node = super().to_rdf(root_node)

        custom_node = et.SubElement(
            node, "custom:field")
        custom_node.set(
            "rdf:datatype", "%xsd:string")
        custom_node.text = self.custom_field
        return node
```

Figure 2: Custom event facet class example.

can implement its own source-specific filtering to reduce the volume of output entities. As an example of such filtering, we may consider dropping regular health check events of a device that do not bring much value in terms of better understanding a crime scene but generate a lot of noise in the data. Additionally, the data source passed to the extractor gets populated into the ontology as well: it is added as a file object record (file path, data size, MD5/SHA hashes, etc.) that is linked as an evidence source to all extracted events. It is a matter of future work to add deeper integration of cyber investigation entities from CASE such as *investigation:ProvenanceRecord*, *investigation:Examiner*, and others. There is one predefined extractor that is supplied with the framework: *KnownFactsExtractor*. This module allows for the population of any known facts about context events or actors. This helps to enrich the timeline and supplement the ontology with additional crime scene context. Examples of such facts can be a list of suspects and some non-cyber events that are known for sure. For example, a call to the police is made (an event) from a specific phone number (an observable). Additionally, the framework provides a small library of tools to perform typical operations with different data stores like CSV, JSON, and SQLite databases making the effort to build new extractors minimal and ensuring the reusability of the code to match diverse data sources. Full code with samples can be found at <https://github.com/link/follows/here>.

3.3 Knowledge Aggregation

The knowledge aggregation layer is represented by a single module that handles all of the extractors and is responsible for consolidated data composition and initial knowledge preparation. This module first initializes and configures all present extractors and then calls their respective *extract()* methods to fetch individual sub-ontologies for every data source and add those to a unified data store. This data store then undergoes the initial normalization step which is entity merging. By utilizing the equality and hash functions of the *RdfEntityBase* class, the aggregator is able to quickly identify multiple representations of the same object. If a duplicate is found it gets deleted, however, its relationships are merged into the initial object thus creating an inter-source linkage. An example of such a merge can be an email address artifact extracted from an email client and the same address used as a username for the home automation system. In this case, the home automation account and the email communication will be automatically bound by the email address artifact (see Figure 3). Another case where merging is applied is the same user account extracted from two different data locations, e.g. cloud source dump and a mobile phone app. In this case, the artifacts and events bound to the user may differ by source but the user record will be the same so after merging the resulting ontology individual will have both contexts. The entity merging stage is followed by the timeline creation. The aggregator extracts all of the individuals that are derived from the *observable:EventRecord* class and arranges them chronologically. After extracting object relationships, they are placed into the knowledge store which is then passed on to the Object Serializer module for the final ontology instantiation in the RDF/XML format.

3.4 Post-Processing

There are unlimited possibilities for the post-processing of the generated ontology in order to find additional correlations. We implemented an example post-processor that goes over the application user accounts that were not previously linked to any person and by applying the string similarity algorithm described in (Myers, 1986) try to match the real name and a username by generating a similarity score. It is important to note that in forensic investigations, while such hints and deductions can guide inquiries, it's essential to remember that assumptions need to be validated with concrete evidence before reaching a definitive conclusion.

3.5 Knowledge Interfacing

To effectively assist the investigator in solving the crime, we propose three approaches to crime interpretation: graph visualization, timeline view, and a set of SPARQL queries to fetch the desired facts and their correlations conveniently. The ontology graph view can help in quickly identifying the underlying events and the context around them such as forensic artifacts involved or the interacting actors (see example in Figure 4). The visualization scope can be shrunk to a certain point of interest, e.g. a specific user and events surrounding it, or expanded instead to see a wider picture. In the current state, for the graph visualization, we utilize Protege's OntoGraph plug-in that is included in the standard installation package. For the purpose of timeline generation, our system provides a module capable of presenting the events in chronological order accompanied by any subset of the surrounding context. The user may select which fields should be included in the timeline view (origin source, associated users, confidence levels, etc.). At present this information is output as a CSV spreadsheet but a sophisticated GUI tool is being developed. Lastly, the SPARQL interface allows for complex knowledge querying. SPARQL is a query language similar to SQL but designed to extract data from knowledge bases instead of relational databases. Some examples of such queries can be found in Section 4.3.

4 SYSTEM DEMONSTRATION

This section covers the demonstration of the proposed method on a publicly available dataset to showcase the advantages of automated artifact extraction and interfacing with the ontology-based knowledge store.

4.1 The Dataset and the Scenario

In many scientific fields, the repeatability of the experiments poses a serious challenge and digital forensics is no different. The vast majority of the works that we studied throughout this research were incorporating either private or irreproducible datasets. For other researchers to validate the results and what is more importantly to advance the research and build on top of these results the method and the data must be clearly defined. The DFRWS community introduced an IoT-oriented forensic challenge in 2018. In the scope of this challenge, a comprehensive dataset was introduced. Not only was it diverse by representing data extracted from different crime scene devices

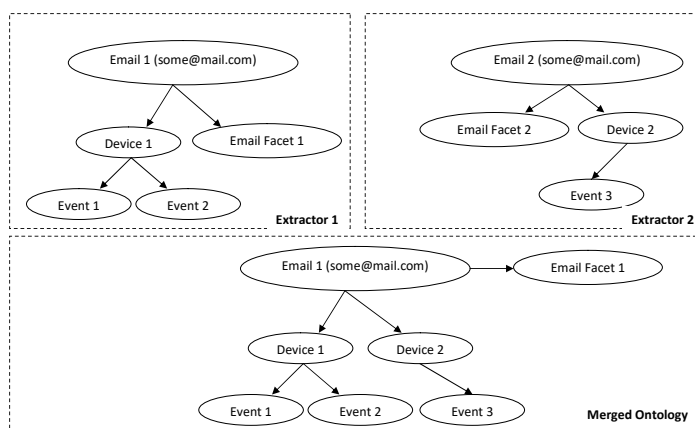


Figure 3: Entity Merging example.

but it came with a realistic scenario and a set of puzzles for the investigators to solve. The data presented in the dataset includes different logs, cache files, device memory dumps, disk snapshots, network traffic interceptions, cloud-extracted data, and more. All these facts made this dataset a perfect candidate for the demonstration of our method.

The scenario of the challenge centers around a situation in a drug-producing laboratory. The inception of the case starts with the police being alerted about an unsuccessful raid of the lab that ended up in an arson attempt. The forensic team is dispatched to find the lab heavily equipped with different IoT devices, such as cameras, different sensors, voice- and remote-controlled hubs, and network infrastructure equipment. In addition, a forgotten cell phone belonging to the lab owner Jessie Pinkman is found at the scene. All identified devices were seized and carefully analyzed in order to extract potential evidence data sources. Police officers interrogated two of Pinkman’s known associates, D. Pandana and S. Varga, who had access to the lab. Both of them deny any involvement in the raid.

There are two key questions for the investigators to answer: the time at which the lab was raided, if any of Pinkman’s friends could have been involved, and if yes with what confidence we can say so.

4.2 Extraction of the Evidence

For demonstration purposes, considering the wide range of evidentiary material at hand it was decided to concentrate on the following points of interest: the sensor data generated by the iSmartAlarm ecosystem (door sensor, motion sensor, and the hub), the NEST Protect system, and Amazon Echo voice control. The reasoning behind such selection is very practical:

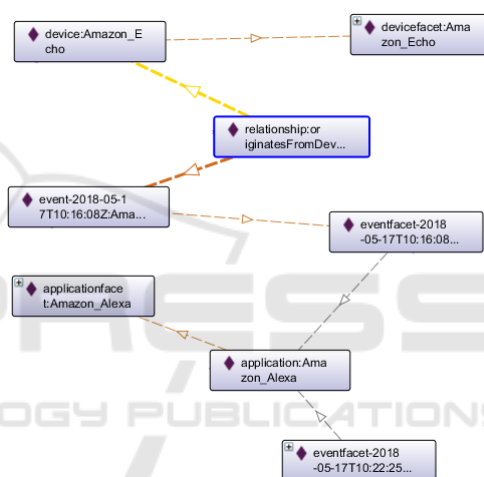


Figure 4: Instantiated ontology objects with a relationship (cropped).

from the forensic report, the range of selected interconnected devices covers most of the crime scene, including motion and smoke detection, as well as control points (hubs), and should provide an exhaustive overview of the events that took place.

iSmartAlarm-related artifacts were found on the phone of Jessie Pinkman inside a controller app’s local database, an SQLite file. The database provides valuable information such as devices connected to the hub (sensors), users having access to the system, events generated by the sensors, and user events executed on the hub itself. The evidence set related to Amazon Echo consists of JSON files, CSV sheets, sound files with voice commands, and SQLite database, representing different cloud-extracted artifacts. The main point of interest here is the database that provides all the major information in a consolidated manner, including event logs and voice com-

mand transcriptions. As for the NEST Protect data related to event tracking was found in the controller app cache extracted from Pinkman's phone. The data is stored in JSON format. With this information at hand, three extraction modules were created deriving from the *RdfExportBase* class as described in Section 3.2: *AlexaExtractor*, *IsaExtractor*, and *NestExtractor*. With the help of the framework's built-in functionality, the extraction modules' code did not exceed a hundred lines. One common filter that is derived by all extractors is the time frame that specifies the start and the end of the period that events should fall under. In our case, we limited the time frame to the day of the accident which is 17 May 2018.

The data extraction process resulted in 290 events placed chronologically. However, after quick observation that some of the extracted events turned out to be noise of little use. For example, for NEST devices there are two device maintenance event types that may be filtered out: *check-in* which is an online status check, and *promise* which is a NEST Nightly Promise mode (a quick check that all systems are operational). After applying a filter for those events the total number of meaningful events dropped by 86% effectively reducing to 41 (see Table 1).

One aspect discussed in Section 3.3 is the entity merging and it can be demonstrated here in the example of the email artifact `jpinkman2018@gmail.com`. This email was identified by two different extractors: the Amazon Echo extractor (as part of the Amazon ID) and the NEST Protect extractor (as the device registration email). Both these extractors assign different references to their copies of the artifacts that they find in their own limited scope. Later after being merged together in the knowledge aggregation stage these references will provide a deeper involvement of the artifact in the full scene environment. In this concrete case, the email in question becomes a linking point between the two devices and their operating accounts. Another instance of entity merging is the full customer name extracted from Amazon ID (Jessie Pinkman). This person observable is getting merged with the suspect person observable provided by the crime scene context from the *KnownFactsExtractor*. The simplified view of the extracted ontology can be viewed in Figure 5. Thin solid arrows represent object relations defined by CASE. Additionally, we create our own instances of *ObservableRelationship* to widen the semantic scope of the ontology (marked as thick solid lines). One example of those is *linkedEmail* which identifies the relation between an application account and an email. *ownedBy* relation specifies ownership of some entity by a specific person. In the schematic representation, dashed arrows are utilized

to symbolize potential relational linkages, as derived from the post-processing phase, each accompanied by a quantified confidence metric. Specifically, in the instance at hand, the prospective associations between the iSmartAlarm users denoted as JPinkman and Pandadodu, have been provisionally attributed to Jessie Pinkman and D. Pandana, respectively. As previously noted in Section 3.4, it is imperative to approach these inferred connections with an appropriate level of circumspection, acknowledging the inherent uncertainty in such algorithmically generated linkages.

4.3 SPARQL Querying

The SPARQL query language is a powerful tool to infer data from and manipulate RDF-based ontologies. It can help in determining some simple correlations as well as complex ones. An example of a moderately simple query shown is in Figure 6 It retrieves all the events associated with a person named "Jessie". It includes the event ID, the time the event was created, and the type of the event. The results are ordered chronologically based on the time of each event. This allows for a timeline view of events for the specified person.

A more comprehensive and practically useful case would be to retrieve all events for a person, including links from person to accounts and emails, and to display a flag indicating whether the relationship has a confidence facet (see Figure 7). It first retrieves the name of the person to check if it matches the condition. Then it finds all relationships that originate from this person, which can be either account or email linkages. For each relationship, it checks if there is an associated confidence facet and sets the *hasConf* flag accordingly. It then retrieves all event records linked through these relationships, including the event ID, time, and type, and orders the results by event time, providing a chronological view of events per person, including the presence of a confidence facet in their relationships. This query is particularly useful in scenarios where you need a comprehensive view of events associated with individuals, including the strength of the evidence (indicated by the presence of a confidence facet).

5 CONCLUSION AND FUTURE WORK

In this work, we proposed a system for automated extraction, ontological representation, and analysis of complex distributed crime scenes. The standard-based ontology provides semantic linkage of all the

Table 1: Consolidated events timeline.

#	Time	Source	Event	Additional Info	User
1	09:44:53	iSASensor	Door Opened		
2	09:45:22	iSAHub	Disarm		TheBoss
3	09:47:18	iSASensor	Door Closed		
4	09:47:50	iSAHub	Arm		JPinkman
5	10:09:52	iSASensor	Door Opened		
6	10:09:55	iSASensor	Motion Detected		
7	10:09:57	iSAHub	Disarm		TheBoss
8	10:16:08	AmazonEcho	History (Dialog)	alexa play led zeppelin	Jessie Pinkman
9	10:16:09	AmazonEcho	SalmonCard	Link Spotify	Jessie Pinkman
10	10:16:09	AmazonEcho	History	alexa play led zeppelin	Jessie Pinkman
11	10:16:09	AmazonEcho	History (Dialog)	To play Spotify, link your premium account first using the Alexa App.	Jessie Pinkman
12	10:16:20	AmazonEcho	History	Unknown	Jessie Pinkman
13	10:16:20	AmazonEcho	History (Dialog)	Unknown	Jessie Pinkman
14	10:22:08	AmazonEcho	History (Dialog)	alexa	Jessie Pinkman
15	10:22:09	AmazonEcho	History	alexa	Jessie Pinkman
16	10:22:12	AmazonEcho	History (Dialog)	tell i. smart alarm to arm my system	Jessie Pinkman
17	10:22:13	AmazonEcho	TextCard	Mode Changed (iSmartA-lArm)	Jessie Pinkman
18	10:22:13	AmazonEcho	History	tell i. smart alarm to arm my system	Jessie Pinkman
19	10:22:13	AmazonEcho	History (Dialog)	Your Door is open, Are you sure you want to arm your system?	Jessie Pinkman
20	10:22:19	AmazonEcho	History (Dialog)	yes	Jessie Pinkman
21	10:22:20	AmazonEcho	TextCard	Mode Changed (iSmartA-lArm)	Jessie Pinkman
22	10:22:20	AmazonEcho	History	yes	Jessie Pinkman
23	10:22:20	AmazonEcho	History (Dialog)	Your system will set to Arm in 30 seconds.	Jessie Pinkman
24	10:22:22	iSAHub	Arm		JPinkman
25	10:22:25	AmazonEcho	History	-	Jessie Pinkman
26	10:22:30	iSAHub	Disarm		TheBoss
27	10:34:15	iSASensor	Door Closed		TheBoss*
28	10:34:17	iSAHub	Home		TheBoss
29	10:34:31	iSAHub	Disarm		pandadodu
30	10:34:36	iSASensor	Door Opened		pandadodu*
31	10:35:54	NEST	Smoke Heads Up	Duration 16s	pandadodu*
32	10:36:11	NEST	Smoke Clear		pandadodu*
33	10:37:52	iSAHub	Disarm		pandadodu
34	10:40:00	Known Event	Police informed		
35	10:45:00	Known Event	Forensics arrive		
36	11:39:50	iSASensor	Door Closed		
37	14:52:10	iSASensor	Door Opened		
38	14:57:06	iSASensor	Door Closed		
39	14:58:03	iSASensor	Door Opened		
40	14:58:15	iSASensor	Door Closed		
41	17:50:55	NEST	Unknown (0204)		

entities that comprise the digital crime scene environment. The ontology is assisted by a Python-based software development framework that allows for evidentiary data extraction from arbitrary data sources and conversion of that data into a unified representation inside the ontology. The filtering mechanisms

that are part of the system allow for a great information volume reduction helping to overcome the investigation scope bloating with irrelevant facts.

For the demonstration, we applied the proposed method against a publicly available dataset representing a crime scene in a distributed environment of In-

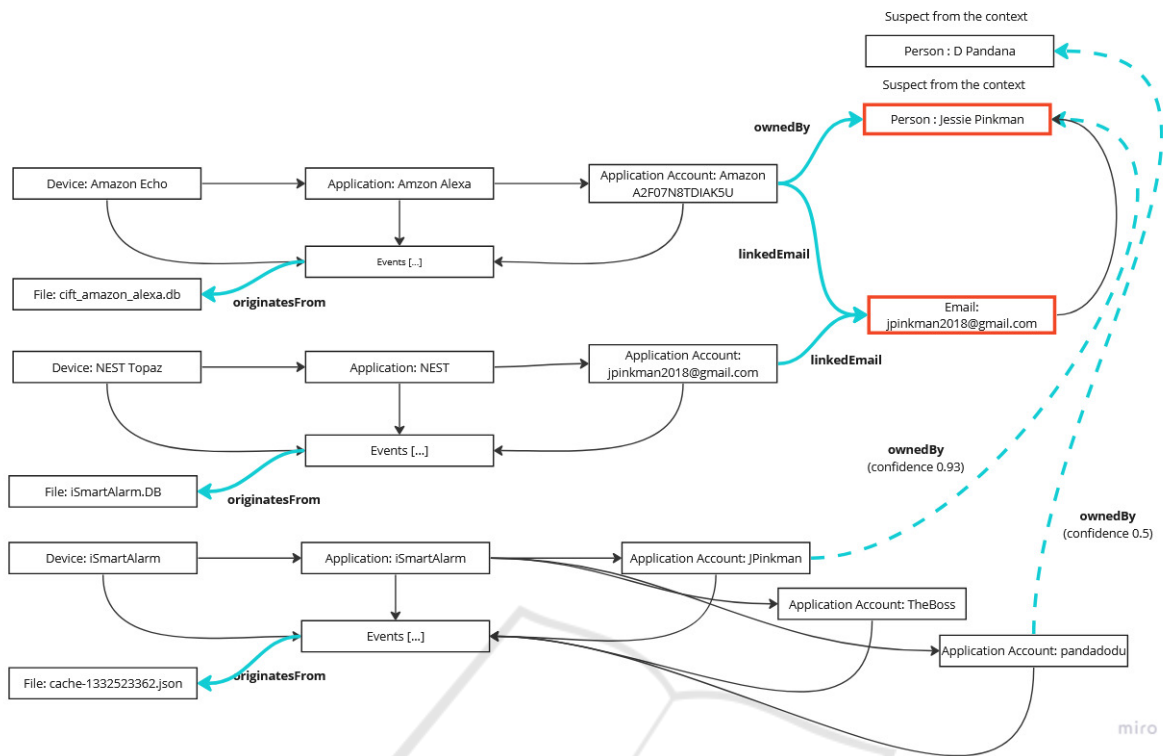


Figure 5: Simplified view of the extracted ontology.

```

SELECT ?evt ?evtTime ?evtType
WHERE {
  ?p rdf:type uco-identity:Person.
  ?p uco-identity:hasFacet ?nameFacet.
  ?nameFacet identity:givenName "Jessie".
  ?rel uco-observable:source ?p.
  ?evtRec uco-observable:hasFacet ?rel.
  ?evtRec rdf:type uco-observable:EventRecord.
  ?evtRec uco-observable:observableCreatedTime
  ?evtRec uco-observable:eventType ?evtType.
}
ORDER BY ASC(?evtTime)
    
```

Figure 6: An example of SPARQL query to retrieve events related to NEST Protect and Amazon Echo.

ternet of Things devices to showcase how investigators can quickly and efficiently approach a very diverse evidence data set. One of the advantages here is the ability to easily plug in any new data source to enrich an already populated knowledge base about a crime scene. The newly added data will be organically embedded into existing ones providing new correlations or refining the existing ones. The standardized ontological representation allows the populated knowledge to be easily integrated into any compatible data store from a different domain.

As part of future work, we plan to integrate pattern matching based on NLP techniques similar to those

```

SELECT ?pName ?evt ?evtTime ?evtType
(BOUND(?confFacet) AS ?hasConf)
WHERE {
  # Person details
  ?p rdf:type uco-identity:Person.
  ?p uco-identity:hasFacet ?nameFacet.
  ?nameFacet identity:lastName ?pName.
  FILTER(?pName = "Pinkman")

  # Link person to accounts and emails
  ?rel uco-observable:source ?p.
  OPTIONAL { ?rel core:hasFacet ?confFacet. }
  {
    ?rel uco-observable:target ?acc.
    ?acc rdf:type uco-observable:ApplicationAccount.
  } UNION {
    ?rel uco-observable:target ?email.
    ?email rdf:type uco-observable:EmailAddress.
  }

  # Fetch related events
  ?evtRec uco-observable:hasFacet ?rel.
  ?evtRec rdf:type uco-observable:EventRecord.
  ?evtRec uco-observable:observableCreatedTime ?evtTime.
  ?evtRec uco-observable:eventType ?evtType.
}
ORDER BY ASC(?evtTime)
    
```

Figure 7: A complex SPARQL query example.

described in (Amato et al., 2020b) to enrich the fact enhancement phase of post-processing with more data correlation capabilities. To continuously support new UCO/CASE releases, we will develop an automated class generator from the JSON-LD ontology representation. This will allow for hassle-free adoption of any future iteration of the specification.

Ontologies play a crucial role in the realm of artificial intelligence, especially in automating analysis and facilitating the deduction of new knowledge. By structuring data in a standardized, machine-readable format, ontologies enable AI systems to interpret complex relationships and extract insights that might not be readily apparent. Our current project exemplifies this, as we are actively engaged in processing the provided ontology using advanced Large Language Models (LLMs). This approach not only enhances the depth and accuracy of analysis but also paves the way for uncovering new patterns and connections within the data, showcasing the powerful synergy between ontology structures and AI capabilities.

REFERENCES

- Amato, F., Castiglione, A., Cozzolino, G., and Narducci, F. (2020a). A semantic-based methodology for digital forensics analysis. *Journal of Parallel and Distributed Computing*, 138:172–177.
- Amato, F., Castiglione, A., Cozzolino, G., and Narducci, F. (2020b). A semantic-based methodology for digital forensics analysis. *Journal of Parallel and Distributed Computing*, 138:172–177.
- Ameerbakhsh, O., Ghabban, F. M., Alfadli, I. M., AbuAli, A. N., Al-Dhaqm, A., and Al-Khasawneh, M. A. (2021). Digital forensics domain and metamodeling development approaches. In *2021 2nd International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, pages 67–71. IEEE.
- Barnum, S., Martin, R., Worrell, B., and Kirillov, I. (2020). Cyber observable expression (cybox™) archive website.
- Brinson, A., Robinson, A., and Rogers, M. (2006). A cyber forensics ontology: Creating a new approach to studying cyber forensics. *digital investigation*, 3:37–43.
- Carvey, H. (2015). Micro- & mini-timelines. *Windows Incident Response*.
- Casey, E., Back, G., and Barnum, S. (2015). Leveraging cybox™ to standardize representation and exchange of digital forensic information. *Digital Investigation*, 12:S102–S110.
- Casey, E., Barnum, S., Griffith, R., Snyder, J., van Beek, H., and Nelson, A. (2017). Advancing coordinated cyber-investigations and tool interoperability using a community developed specification language. *Digital Investigation*, 22:14–45.
- Casey, E., Barnum, S., Griffith, R., Snyder, J., van Beek, H., and Nelson, A. (2018). *The Evolution of Expressing and Exchanging Cyber-Investigation Information in a Standardized Form*, pages 43–58. Springer International Publishing, Cham.
- Chabot, Y., Bertaux, A., Kechadi, T., and Nicolle, C. (2015a). Event reconstruction: A state of the art. *Handbook of Research on Digital Crime, Cyberspace Security, and Information Assurance*, pages 231–245.
- Chabot, Y., Bertaux, A., Nicolle, C., and Kechadi, T. (2015b). An ontology-based approach for the reconstruction and analysis of digital incidents timelines. *Digital Investigation*, 15:83–100.
- Chikul, P., Bahsi, H., and Maennel, O. (2021). An ontology engineering case study for advanced digital forensic analysis. In Attiogbé, C. and Ben Yahia, S., editors, *Model and Data Engineering*, pages 67–74. Cham. Springer International Publishing.
- Debinski, M., Breiting, F., and Mohan, P. (2019). Timeline2gui: A log2timeline csv parser and training scenarios. *Digital Investigation*, 28:34–43.
- Ding, Y., Wu, R., and Zhang, X. (2019). Ontology-based knowledge representation for malware individuals and families. *Computers & Security*, 87:101574.
- Eden, P., Blyth, A., Burnap, P., Cherdantseva, Y., Jones, K., and Soulsby, H. (2015). A forensic taxonomy of scada systems and approach to incident response. In *3rd International Symposium for ICS & SCADA Cyber Security Research 2015 (ICS-CSR 2015) 3*, pages 42–51.
- Elezaj, O., Yayilgan, S. Y., Kalemi, E., Wendelberg, L., Abomhara, M., and Ahmed, J. (2019). Towards designing a knowledge graph-based framework for investigating and preventing crime on online social networks. In *International Conference on e-Democracy*, pages 181–195. Springer.
- Ellison, D., Ikuesan, R. A., and Venter, H. S. (2019). Ontology for reactive techniques in digital forensics. In *2019 IEEE Conference on Application, Information and Network Security (AINS)*, pages 83–88. IEEE.
- Esposito, S. and Peterson, G. (2013). Creating super timelines in windows investigations. In *IFIP International Conference on Digital Forensics*, pages 135–144. Springer.
- Hargreaves, C. and Patterson, J. (2012). An automated timeline reconstruction approach for digital forensic investigations. *Digital Investigation*, 9:S69–S79.
- Javed, A. R., Ahmed, W., Alazab, M., Jalil, Z., Kifayat, K., and Gadekallu, T. R. (2022). A comprehensive survey on computer forensics: State-of-the-art, tools, techniques, challenges, and future directions. *IEEE Access*, 10:11065–11089.
- Kahvedžić, D. and Kechadi, T. (2009). Dialog: A framework for modeling, analysis and reuse of digital forensic knowledge. *digital investigation*, 6:S23–S33.
- Kahvedžić, D. and Kechadi, T. (2010). Semantic modelling of digital forensic evidence. In *International Conference on Digital Forensics and Cyber Crime*, pages 149–156. Springer.

- Karie, N. M. and Venter, H. S. (2014). Toward a general ontology for digital forensic disciplines. *Journal of forensic sciences*, 59(5):1231–1241.
- Myers, E. W. (1986). An o(nd) difference algorithm and its variations. *Algorithmica*, 1(2):251–266.
- Park, H., Cho, S., and Kwon, H.-C. (2009). Cyber forensics ontology for cyber criminal investigation. In *International Conference on Forensics in Telecommunications, Information, and Multimedia*, pages 160–165. Springer.
- Sikos, L. F. (2020). Ai in digital forensics: Ontology engineering for cybercrime investigations. *Wiley Interdisciplinary Reviews: Forensic Science*, page e1394.
- Turnbull, B. and Randhawa, S. (2015). Automated event and social network extraction from digital evidence sources with ontological mapping. *Digital Investigation*, 13:94–106.
- Välja, M., Heiding, F., Franke, U., and Lagerström, R. (2020). Automating threat modeling using an ontology framework. *Cybersecurity*, 3(1):1–20.
- Wimmer, H., Chen, L., and Narock, T. (2018). Ontologies and the semantic web for digital investigation tool selection. *Journal of Digital Forensics, Security, and Law*, 13(3):21.

