How Do I Share My IoT Forensic Experience With the Broader Community? An Automated Knowledge Sharing IoT Forensic Platform

Xiaolu Zhang, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, and Nicole Lang Beebe, *Senior Member, IEEE*

***Abstract*—It is challenging for digital forensic practitioners to maintain skillset currency, for example knowing where and how to extract digital artifacts relevant to investigations from newer, emerging devices (e.g., due to the increased variety of data storage schemas across manufacturers and constantly changing models). This paper presents a knowledge sharing platform, developed and validated using an Internet of Things dataset released in the DFRWS 2017–2018 forensic challenge. Specifically, we present an automated knowledge-sharing forensic platform that automatically suggests forensic artifact schemas, derived from case data, but does not include any sensitive data in the final (shared) schema. Such artifact schemas are then stored in a schema pool and the platform presents candidate schemas for use in new cases based on the data presented. In this way, investigators need not learn the forensic profile of a new device from scratch, nor do they have to manually anonymize and share forensic knowledge obtained during the course of an investigation.**

***Index Terms*—Digital forensics, forensic knowledge as a service (ForKaS), Internet of Things (IoT), IoT forensics, knowledge-sharing, ontology.**

I. Introduction

As INTERNET of Things (IoT) devices are becoming commonplace in our society (e.g., in environmental monitoring, smart cities, smart business/inventory and product management, smart homes/smart building management, health-care, security and surveillance, and battlefields such as Internet of Battlefield Things) [1]–[3], they are also a potential source of evidence in criminal investigations and civil litigations [4].

One of the many challenges in a digital forensic investigation is the lack of information and knowledge-sharing between investigators and cases, particularly those involving contemporary technologies, such as newer IoT devices. For example, in a typical investigation process, two or more investigators located in different cities and/or countries may be forensically examining the same (type of) device at the same time [see Fig. 1(a)]. As the experience and background of both investigators are likely to vary, the outcomes of the forensic investigations may also differ (e.g., in terms of the types and extent of artifacts being recovered).

As the diversity of devices increases (e.g., IoT, wearable, embedded, and other digital devices), so does the challenge in the forensic examination of such devices [4]. For example, an investigator who is unfamiliar with a particular device that (s)he has not previously examined, say a Makerbot Replicator Z18 Large 3-D Printer, may need to research the device before undertaking any forensic examination. The amount of time required in acquiring new knowledge could be prohibitive to smaller or regional forensic investigation teams with limited resources. The challenge is compounded with the advent of the IoT. With an estimated 20 billion devices connected to the IoT by 2020, forensic investigators are increasingly challenged to extract and analyze forensic artifacts effectively and efficiently. Automated, or semiautomated, knowledge sharing platforms that protect case sensitive data are critically needed.

In addition, due to the sensitive nature of data acquired from forensic examination, data is seldom shared. Such challenges can have an impact on the timing, efficiency and consistency of the investigations and findings. For example, Miller *et al.* [5] noted that:

> Practitioners tasked with investigating new device types, be it for forensic or security related purposes, however, do not have the luxury of relying on previous work and, with the variety present in AM [additive manufacturing] device architectures, examination of only the device may be insufficient to discover the entirety of the residual data footprint left by its printing process. Additionally, acquiring a representative sample of devices for study may be cost prohibitive or the exploratory analysis of a particular device may be inadvisable due to operational, evidentiary, or resource constraints.

Hence, we posit the importance of knowledge sharing among the forensic community—particularly technology enabled automated, or semiautomated, sharing platforms that

Manuscript received December 12, 2018; revised February 19, 2019 and April 6, 2019; accepted April 16, 2019. Date of publication April 22, 2019; date of current version July 31, 2019. This work was supported in part by the National Science Foundation CREST under Grant HRD-1736209. *(Corresponding author: Kim-Kwang Raymond Choo.)*

X. Zhang and N. L. Beebe are with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: xiaolu.zhang@utsa.edu; nicole.beebe@utsa.edu).

K.-K. R. Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249 USA, and also with the Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Digital Object Identifier 10.1109/JIOT.2019.2912118

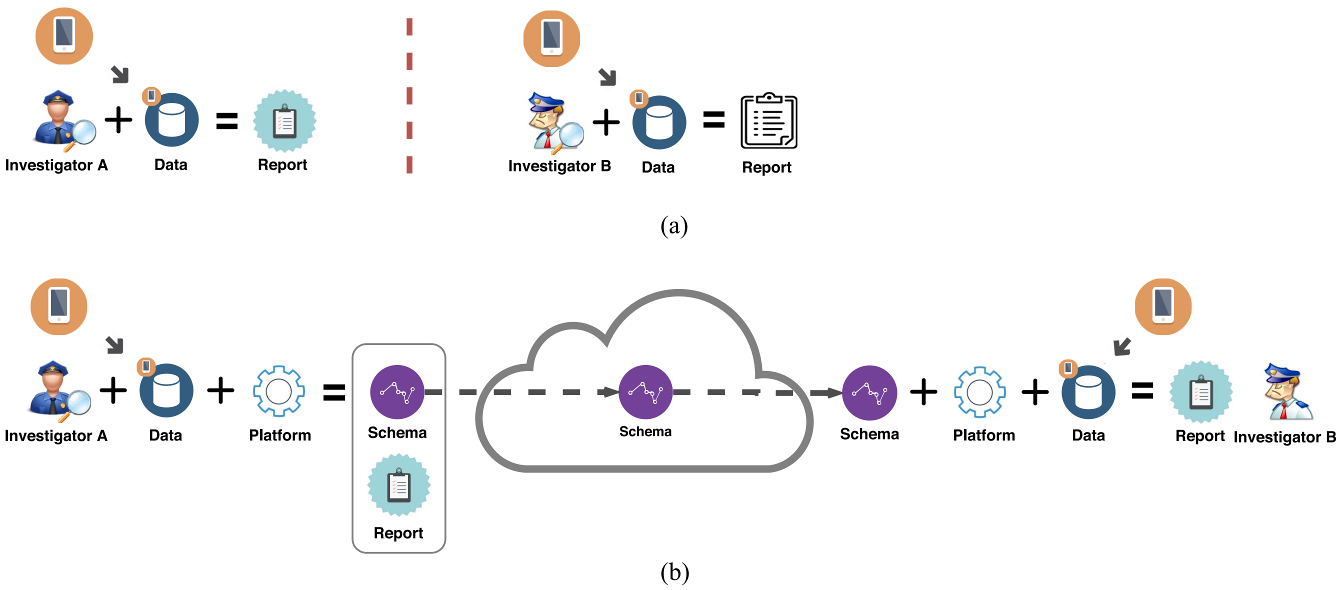


Fig. 1. Overview of the proposed ForKaS platform. (a) Traditional investigation process. (b) Proposed investigation process.

strictly protect sensitive case data. Investigators need mechanisms for efficiently and effectively sharing knowledge *about* forensic artifacts—their location, their forensic implications, their data structures, etc.—without sharing the artifacts themselves. Specifically, in this paper we present a forensic knowledge-sharing platform [see Fig. 1(b)], designed to facilitate the sharing of knowledge/experience via a schema generated from another investigation of a similar device. Such a schema captures the knowledge of an investigator who had previously examined a particular device, and contains information, such as type and nature of artifacts that could be acquired, to guide another investigator's forensic examination.

Our proposed forensic knowledge as a service (ForKaS) platform builds on the conventional four-layer digital forensic framework (collection, extraction, analysis, and visualization), by introducing the "abstraction" layer where the schema can be generated and shared. We also implement a proof of concept and evaluate the prototype using the publicly available smart home dataset released for the DFRWS 2017–2018 forensic challenge. Specifically, we demonstrate how one can efficiently generate a schema based on the examination of the smart home dataset (comprising a number of IoT devices) and then share the generated schema via the platform; thereby, increasing other investigators' efficient and effective extraction of evidence on devices they may or may not have analyzed before. We also demonstrate how the schema can then be utilized by a forensic investigator examining a different Android device and an iOS device.

We regard the key benefits of ForKaS to include the following.

1) Allow the sharing of knowledge and experience between forensic investigators without the need to share the sensitive data acquired from the forensic investigation or dataset.
2) The inconsistency and time required of a forensic investigation finding could be minimized, since investigators can draw on prior experience and findings from the international forensic community (e.g., INTERPOL and EUROPOL) to benefit their own investigations. This also minimizes the steep learning curve associated with newer technologies or technologies that the investigator is unfamiliar with.
3) Forensic investigators can provide other stakeholders (e.g., investigation officers and prosecutors) an expedited, high-level overview of the case, using existing schema.

The remainder of this paper is organized as follows. The literature review is presented in Section II. Section III introduces the structure of the proposed ForKaS platform. In Section IV, we present two key algorithms required in the prototype implementation. We then present the case study in Section V and the discussion in Section VI. Lastly in Section VII, we conclude this paper and discuss future work.

II. Related Work

To keep pace with the constant evolution of consumer technologies [6], [7], there has been an increased focus on digital forensic research, such as IoT forensics (see Section II-A). Since the definition of "IoT device" is relatively broad, in this section we will focus on a smart home scenario where an IoT device usually refers to a device that is not conventionally "smart," or one that has an independent IP address (e.g., smart LED light bulb, IP camera, smart watch/TV, and a motion detector), and an Internet-connected device that can be used to control the smart device (e.g., a smart phone).

We will also briefly review prior attempts to share forensic knowledge/experience, for example as forensic taxonomies (see Section II-B).

A. IoT Forensics

IoT forensics can be broadly categorized into device level forensics, network forensics [8], [9], and cloud/server forensics [10], [11]. In the context of a smart home, for example, forensic data artifacts could be acquired from IoT devices, IoT controller/IoT hubs (e.g., Samsung SmartHub), a voice assistant device (e.g., Amazon Echo and Google Home), the home router and the cloud server [12]. In addition, the data can also be acquired from the mobile device or application (we refer interested reader to [13] for a comprehensive review of mobile forensics and [14] for Android application forensics), as demonstrated in the study of Amazon Echo by Chung *et al.* [15] and Li *et al.* [16].

In 2017, Meffert *et al.* [17] conducted a forensic examination of openHAB, an open source IoT controller dominating an IP camera and two sensors. In the study, openHAB was deployed on an Ubuntu server and the historical states and changes of the IoT devices were acquired from the local log file. Both Oriwoh and Sant [18] and Zawoad and Hasan [19] proposed a conceptual smart home forensics system and a conceptual IoT forensic model, without evaluating the proposed systems. Goudbeek *et al.* [12] proposed a forensic investigation framework for the smart home environment and demonstrated how it can be used to guide a smart home forensic investigation using three case studies, involving a do-it-yourself home automation system and a managed home automation system.

Baggili *et al.* [20] forensically examined an LG smart watch, and demonstrated how data (e.g., timestamps of updates performed, and voice memos were recorded on the watch) could be acquired from a Samsung Galaxy S4 smart phone paired with a smart watch. In another study, Do *et al.* [21] examined Android Wear, an Android-based Operating System for smart watch. The authors explained how sensitive data of the user on a Samsung Gear Live smart watch (e.g., SMS messages, contact information, and biomedical data) could be acquired.

Other areas of IoT forensics include unmanned aerial vehicle (UAV; also known as drones) forensics. For example, Horsman [22] conducted a forensic analysis of a Parrot Bebop UAV via its Telnet protocol. The flight data acquired was in structured files, which are formatted with JavaScript Object Notation (JSON). Later works include the study of DJI Phantom III by Clark *et al.* [23], in which the acquired flight data was found in the embedded microSD card and the memory of DJI Vision application.

B. Forensic Schema, Ontology, and Taxonomies

There have been efforts by the forensic community to standardize the format of evidence presentation in order to facilitate exchange-ability and comparability of the evidence. Most of these works focused on having a universal data format, where the raw evidence data and the associated meta-data are included. For example, Turner [24] proposed a wrapper of disk image [i.e., digital evidence bags (DEBs)] to provide investigators the capability to store data from disparate sources. Similarly, Garfinkel [25] and Levine and Liberatore [26] represented all partitions and files of the disk image using eXtensible Markup Language (XML) files. The Advanced Forensics Format (AFF4) is another example of a forensic format built for representation of meta-data and Zip64 compression [27]. Last, Casey *et al.* [28] developed Digital Forensic Analysis eXpression (DFAX), using Cyber Observal eXpression (CybOX), to represent digital trace information in an exchangeable format.

There has also been focus on the artifacts that can be forensically recovered, in order to create reusable knowledge for different evidence sources. For example, using semantic Web and ontology, one could add rich semantics to the forensic evidence so that investigators can easily select and query artifacts of interest from the evidence semantically. Brady *et al.* [29] proposed the digital evidence semantic ontology (DESO), which represents evidence in two general classes, namely "artifact location" and "artifact type." Once the semantic Web is built, an investigator can query artifacts generated by certain operating system since "operating systems" is considered a subclass of *artifact location* in DESO. Other ontologies proposed in the literature include those designed for Windows Registry [30], network traffic [31], Android System [32], abstract forensic-relevant actions [28], and various log files [33]. Generally speaking, semantic Web facilitates knowledge-driven digital forensics [34].

Perhaps one of the most recent and most integrative contributions is Cyber-investigation Analysis Standard Expression (CASE) [35], an open community-developed specification language that extends DFAX and is built upon the Unified Cyber Ontology (UCO). It addresses the challenge caused by the large number of specification languages and ontologies introduced previously. It addresses the challenge of merging data sources from many organizations or handling large datasets from a variety of tools.

However, there are limitations in existing ontology-based works. For example, one assumption in these works is for a human specialist to generate a universal ontology based on his/her expertise. However, in a real world setting, the creation of the ontology relies on the accumulation of extensive experience. Experienced forensic investigators lack experience in building ontologies and are discouraged from creating them given their constantly high case load and the manual and laborious nature of ontology development.

A number of forensic taxonomies that provide a systematic classification of forensic artifacts from different Android mobile app categories [36]–[38] and Windows Phone mobile app categories have been presented in the literature [39] and [40]. However, existing ontology-based approaches and (mobile app artifact) forensic taxonomies do not provide a knowledge "template" that can be used to facilitate knowledge sharing (we also refer interested reader to [41] for a comparison of some of these approaches).

The Artifact Genome Project (AGP) [42] makes important strides in cataloging digital forensic artifacts by providing "an online system for uploading and viewing digital forensic artifacts." The benefit of the proposed system in contrast, however, is that the proposed system provides a technology enabled mechanism for ontological schema development, by deriving proposed schema entities and relationships from the data extracted and analyzed in actual investigations. Whereas investigators must manually enter artifact metadata into AGP, the

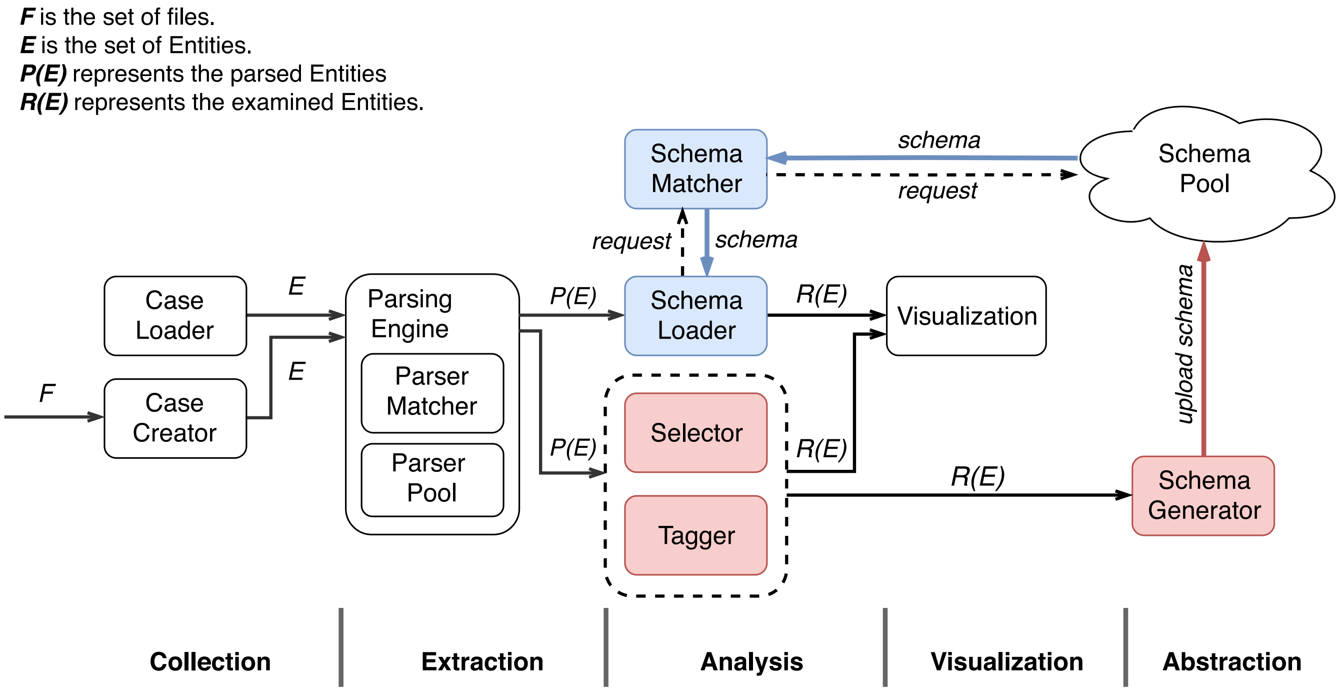


Fig. 2. Structure of the knowledge-sharing-based forensic analysis platform.

proposed platform automates knowledge generation, sharing, and use to a large extent.

III. Proposed Platform

In this section, we present our proposed forensic platform. As illustrated in Fig. 2, the platform has five layers, namely collection, extraction, analysis, visualization, and abstraction. The fifth layer, Abstraction, extends the conventional four-layer approach (i.e., collection, extraction, analysis, and visualization), by leveraging the experience of other forensic investigators as they learn from their analyses (typically in the analysis layer). In other words, our fifth layer abstracts knowledge obtained by forensic investigators, enabling it to be shared with other investigators. The description for each layer of our platform and the modules can be found in Sections III-A–III-E.

A. Collection

The Collection layer deals with the loading and encapsulating of the data source(s) to a *forensic case* (e.g., a murder investigation). This layer is also known as the interface between the platform and the data source. Suppose that set F refers to the data source, where $F = \{f|f$ is a valid file of the data source$\}$. To ensure that the loaded data source is ready for the next layer, the *case loader* and *case creator* modules will be responsible for achieving the encapsulation and reloading the case to the platform, respectively.

When a forensic case is created, the data source F within the case is logically separated into one or multiple *entities*. In this platform, an *entity* is a unit with a group of files, where artifacts may be found that are logically independent (see the circled “SmartThings_DFRWS C:Users: ...” in Fig. 5). For example, if the investigator is examining a Samsung Thing, then the investigator can load up the “SmartThings_DFRWS C:Users: ...” entity.

Entities can exist at different granularities, for example at an application level (e.g., a specific mobile app), and a device level (e.g., a specific IoT device). Investigators examining “complex” devices, such as mobile devices (e.g., Android and iOS devices), should consider using application-specific entities (e.g., Facebook iOS app entity) rather than a device-level entity (in this context, iOS entity).

In Fig. 2, E denotes the set of entities of a forensic case, $E = \{e|e \subseteq F\}$.

B. Extraction

In the second layer (i.e., extraction), the file format is automatically recognized by the platform, the data are retrieved and parsed from the files (f) standardized for further investigation. Specifically, the entities (obtained from the first layer) are processed through a parsing engine, which consists of two modules, namely *parser matcher* and *parser pool*. Parser matcher is responsible for assigning the most appropriate parser within the parser pool to each file of an entity. Given that $P(E)$ is the set of parsed entities, then $P(E) = \{P(e)|e \in E\}$ and $P(e) = \{P(f)|f \in e\}$. So that, the assigned parser can parse a file f to $P(f)$.

Here, in order to further process the parsed file $P(f)$ in the next layer, all $P(f)$ are tabulated to a n-column table and each column has some associated data. In other words, we can represent $P(f)$ via a set of an ordered two-tuple—(c, d), where c is the identifier of the column, and d is an ordered n-tuple including the corresponding data.

C. Analysis

The analysis layer can be considered a knowledge implementer. Through this layer, a parsed file $P(f)$ can be examined using the digital forensic investigator's experience. If the schema for the particular device or application does not exist and the investigator wishes to contribute to the schema pool, then (s)he can use the *selector* and *tagger* modules to select and tag the relevant parsed files. On the other hand, if the schema for the particular device or application exists, then the investigator can use the *schema loader* module to load the relevant schema to work on the parsed file $P(f)$.

In either of the above approaches (i.e., select and tag, or load), data of interest/relevance from $P(f)$ will be identified and tagged. Since $P(f)$ is a set of (c, d), the tagged data D will be a subset of $P(f)$. Let $R(f)$ denotes the examined data of $P(f)$, i.e., the output of this layer that contains the investigator's annotations, and hence $R(f)$ is a set of (t, d).

D. Visualization

In the visualization layer, the *visualization* module allows investigators (or any user) to view the forensic artifacts graphically. The module can recommend a suitable visualization type (e.g., timeline), based on the schema type. This module also allows the user to choose parts of the data under investigation [denoted as $R(E)$] to be visualized. In order to do so, an investigator needs to assign a set of tags for visualization on the visualization panel. The tags would then be shown to the investigator during the investigation process. As long as a column is labeled with any of these tags, the related data will be obtained and visualized.

From the investigator's perspective, labeling the data with an appropriate tag is important. There are a number of commonly used tags, such as "device ID," "device name," "action," "value," and "time." Some of these values can be further broken down into subtags, such as *time* can be separated into "start time" and "end time," or "last accessed time," "last modified time," and so on.

E. Abstraction

The last layer is abstraction, which provides the knowledge extraction capacity for investigators. As the name implies, the extracted knowledge/schema can be shared between investigators, say from different agencies or countries, without the case-specific, sensitive data associated with the case. This way, *device metadata*, rather than *case data* can be shared.

Suppose that a schema s is a set of $S(f)$ ($f \in e$, e is an entity). To generate the element $S(f)$, both $P(f)$ (the parsed file) and $R(f)$ (the parsed data and investigated data of file f) are needed. In other words, the targeted entity where the schema was generated should have processed by the parsing engine and the selector and tagger modules in the analysis layer. Specifically, in order to obtain the knowledge from each investigated file in the entity, we first extract c out of every (c, d) tuple ($(c, d) \in P(f)$) and also t out of every (t, d) tuple ($(t, d) \in R(f)$), before assembling the extracted c and t to a new element of $S(f) - (c, t)$ ($(c, t) \in S(f)$). Thus, a schema of the entity can be created.

In this layer, the schema generator module allows us to create a schema, which will be stored in a schema pool. The latter can reside in a cloud, such as an INTERPOL/EUROPOL cloud. For example, when some investigator works on a case involving a new or unfamiliar device, the investigator can make a request to the schema pool. Upon receipt of such a request, the schema matcher module attempts to find Schemas that match the request. If one or more matching schemas are found, the schema loader module will retrieve and load the schemas to $P(E)$, so that the requesting user/investigator can apply it(them) to his/her case.

In $P(E)$, the first step is to search through the data from $P(f)$. For each element of $P(f)$, it will be retained as long as column c ($c \in (c, d)$) matches the element (c, t) ($(c, t) \in S(f)$). The second step is to replace the column name c of the searched elements with the associated tag t. Note that the schema matcher module conducts a scoring approach (see Section IV-B) in order to find the most appropriate schema from the schema pool. It also provides a reference to the investigator who intends to load a schema manually.

IV. Key Algorithms

In this section, we will describe the two key algorithms required in the implementation of the prototype, namely file parsing and schema recommendation.

A. File Parsing

Essentially, parsing file f to a set of (c, d) (see Section III-B) is equivalent to converting the file to a 2-D table, where c represents a column of the table and d denotes the corresponding data of the column. Due to the different format of the input files, different parsers are required. In other words, the proposed platform will require new parsers be built as needed, as significant device changes are introduced in the market. However, a benefit of the proposed approach is that it leverages common data structures across a large number of IoT devices. Only significantly divergent entrants to the IoT market will necessitate the development of new file parsers.

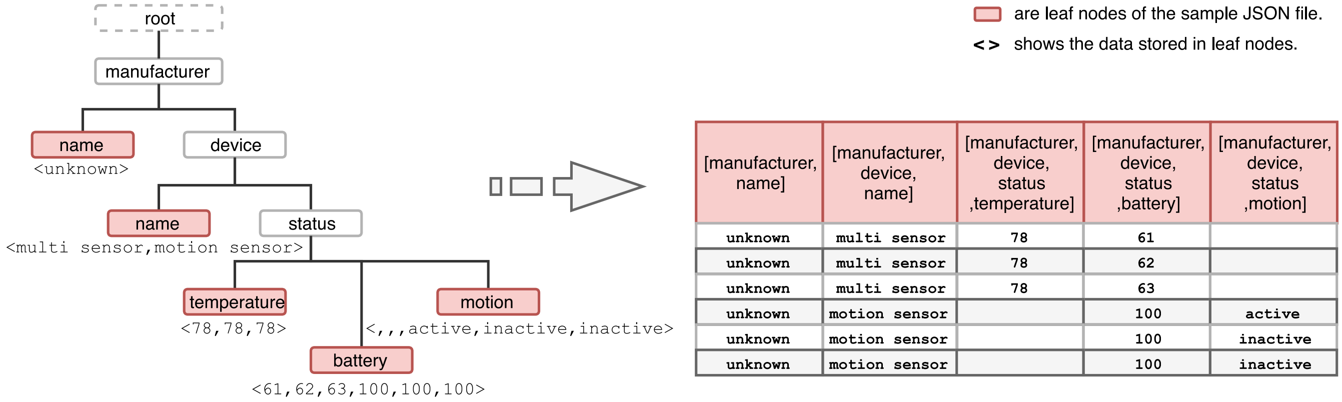
This update model is similar to how current commercial forensic tools function, where updates are being introduced to the forensic tool in order to acquire forensic artifacts from new devices. However, we contend that the frequency with which new parsers will need to be built will be less than the pace of forensic tool updates that would otherwise be needed to update forensic tools with device level artifact knowledge, since general file formats and general data structures stay much more constant across device models than specific artifact metadata.

We also note that devices of interest in most forensic cases tend to be popular consumer technologies, such as commonly used IoT devices and smart phones, and these commonly used devices store/exchange data in some standardized file formats such as JSON, XML, SQLite database, and comma-separated values (CSV). Hence, most of these formats can be captured in our platform [represented using (c, d) in a table].

Formats such as SQLite and other similar database formats are typically structured as a table. Element tags, column names, and row names can be extracted from JSON, XML, and

```
{"manufacturer":{"name":"unknown","device":[{"name":"multi sensor","status":[{"temperature":78,"battery":61},
{"temperature":78,"battery":62},{"temperature":78,"battery":63}]},{"name":"motion sensor","status":[{"motion"
:"active","battery":100},{"motion":"inactive","battery":100},{"motion":"inactive","battery":100}]}]}}
```

Fig. 3. Sample JSON file.


root
manufacturer
name
<unknown>
device
name
status
<multi sensor,motion sensor>
temperature
<78,78,78>
motion
<,,,active,inactive,inactive>
battery
<61,62,63,100,100,100>
🟥 are leaf nodes of the sample JSON file.
< > shows the data stored in leaf nodes.

[manufacturer, name]	[manufacturer, device, name]	[manufacturer, device, status ,temperature]	[manufacturer, device, status ,battery]	[manufacturer, device, status ,motion]
unknown	multi sensor	78	61	
unknown	multi sensor	78	62	
unknown	multi sensor	78	63	
unknown	motion sensor		100	active
unknown	motion sensor		100	inactive
unknown	motion sensor		100	inactive

Fig. 4. Convert the sample JSON file from a tree to a table.

CSV. JSON, XML, and CSV can be converted from one to another, so if one format is supported, then the other formats are guaranteed to be supported as well. Hence, for illustration, we only introduce the conversion process for JSON files in this section. Other format conversion can be performed in a similar fashion.

Fig. 3 illustrates how a JSON file can be converted to a 2-D table within two steps.

1) Traverse JSON file based on its natural tree structure; thus, all the leaf nodes are used to create the columns of the table.
2) Traverse JSON file again to fill the table with the data associated with the leaf nodes.

Since JSON files are naturally structured as a tree, there is only one root node[1] in the tree and the data is only stored on the leaf nodes. In terms of the JSON grammar: 1) each string before ":" is a node of the tree; 2) between "{" and "}" are child nodes; and 3) if neither "{" nor "[" appear before ":," then the string/value following the ":" is the node's data. Fig. 4 is the tree structure of the parsed sample JSON file in Fig. 3.

Since a leaf node and the associated data conform to the structure of (c, d), we assign each column of the table to a leaf node. If the name of a leaf node is n and the name of its furthest node is n_l (l refers to the level of the node), then the unique name of a leaf node can be created by a ordered tuple $(n_0, \ldots n_{l-1}, n_l, n)$, where n_0 is the root note. Note that the root node is an nonexistent node of the JSON file, and the name of the root node is always "null"[2] (e.g., node "manufacturer" in Fig. 4 is n_1 rather than n_0 if node "motion" is n). The empty table is created by the set of $(n_0, \ldots, n_{l-1}, n_l, n)$. Also, $c = (n_0, \ldots, n_{l-1}, n_l, n)$.

Once the table is created, we traverse the JSON file again to retrieve the data of the leaf nodes, using depth-first-search (DFS). Function setTableCell(..) in Algorithm 1 is responsible for feeding the data to the corresponding column of the table. In the algorithm, two leaf nodes may have different relations. The leaf nodes that have same depth are known as "brothers" if they have the same furthest node. For leaf nodes with different depths, the shallow node is the "uncle" of the deeper node. The deeper node is the "nephew" of the shallow node if the deeper node's father has the same furthest node with the shallow node. The right-hand side of Fig. 4 shows the tabulated table of the sample JSON file.

B. *Schema Recommendation*

In this section, we will introduce the recommendation algorithm in the schema adaptor module, which is tasked with the identification of the most appropriate schema(s) to the investigator. For optimized performance, we need to avoid searching the entire schema pool for each request and the module should filter out as many irrelevant schemas as possible. Therefore, metadata are embedded in both the schema and the entity to facilitate searching (this concept is similar to inserting relevant keywords to facilitate searching over encrypted data that are outsourced to a cloud—i.e., searchable encryption [43]).

Table I shows a simple example of using metadata to describe an entity or a schema. In the prototype, it is recommended that the metadata should be included as soon as a schema/entity is created, or wherever practical. To find the best match(es) from the schema pool, a scoring algorithm is executed and its output reflects how well a schema matches the entity. Usually, the investigator is recommended to load the schema with the highest score. Basically, the scoring algorithm (Algorithm 2) calculates the extent to which a schema can be applied to the entity.

For operational reasons, an organization may also choose to implement a scaled down version of the platform. For example,

[1] A JSON file consists of one JSON object or a JSON array of multiple JSON objects.

[2] The reason for introducing such a nonexistent root node is to present the structure of the JSON file with a single tree. Otherwise, the JSON file structured with multiple JSON objects would include more than one root nodes/trees.

Algorithm 1: Algorithm for Converting a JSON File to a Table

```
Function setTableCell (colName, value) is
    Data:
    colName: Column name of the table.
    datalist: List of data of a column.
    value: Value of a datalist.
    cell: Element of a datalist/a cell of the table. A cell
    has two status, writable and unwritable.
    if the last cell of the datalist of column ‘colName’ is
      writable & the datalist is not null then
        set the value to the last cell of the datalist;
        set the cell to unwritable;
    else
        set the value to a new cell;
        add the new cell to the end of the datalist;
        if the length of the datalist is the longest then
            for each column/node, excluding the column
              holding the datalist do
                if it is the uncle node of the node
                  representing the column ‘colName’ then
                    add the last value of the datalist to a
                      new cell;
                    set the new cell to unwritable;
                    add the new cell to the end of the
                      datalist;
                else if It is a nephew/brother node then
                    add a writable empty cell to the end
                      of the datalist;
                else
                    add a unwritable empty cell to the
                      end of the datalist;
            end
        end
    end
    return 0;
end
```

TABLE I
SAMPLE METADATA FOR ENTITY AND SCHEMA

Category	Options
Application	Operating system, developer, version
Operating system/firmware	developer, kernel version
IoT Device	name, model, manufacturer, firmware

Algorithm 2: Algorithm to Score a Schema for a Given Entity

```
Function scoringSchema (entity, schema) is
    Data:
    score: calculated score of the schema.
    entity: given entity.
    schema: target schema.
    totalColNumber: number of columns in each element
    of a schema.
    score = 0;
    for each table of the parsed file of the entity do
        hit = 0;
        for each column of the table do
            if the column can be found in the schema
              then
                hit ++;
            end
        end
        score = score+(hit/totalColNumber);
    end
    return score;
end
```

in Table I, if the entity can be confined to the data of a specific application rather than the entire IoT device, then searching for the most effective schema from the pool would be faster.

V. Case Study

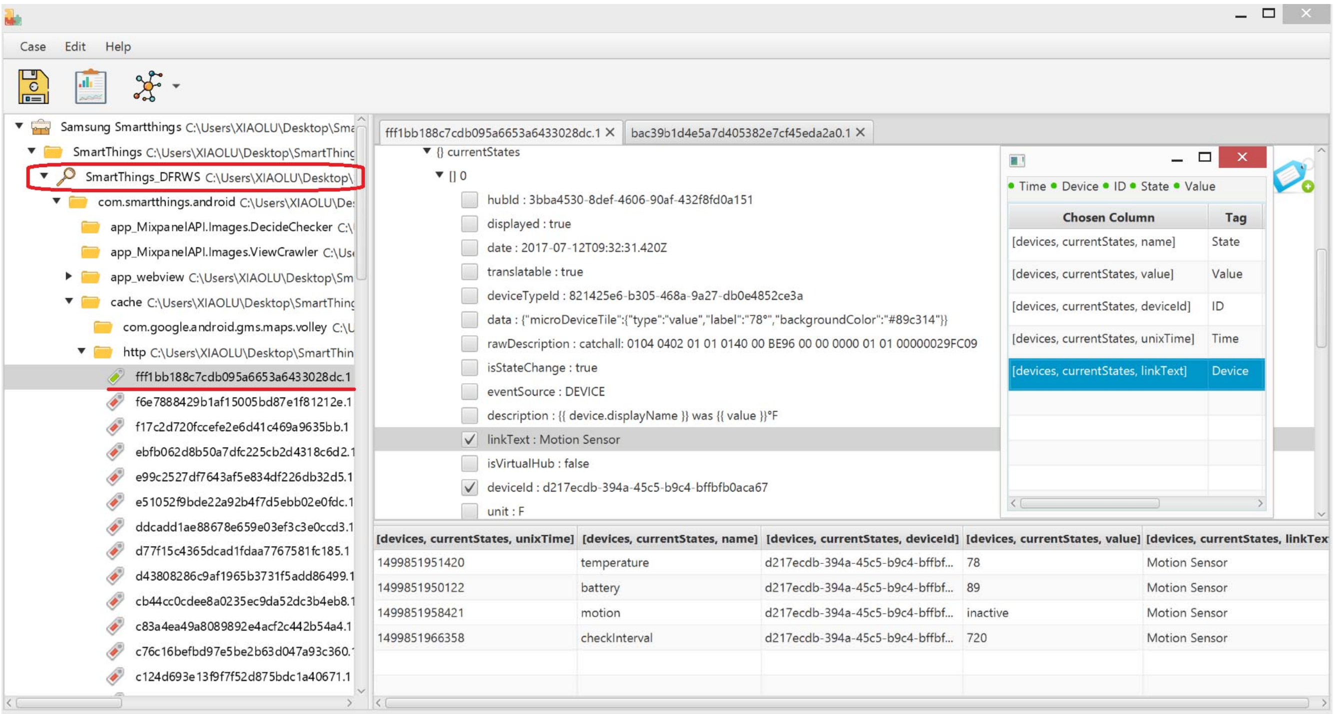
To evaluate the utility of the proposed ForKaS platform, we developed a prototype that has been evaluated using the IoT forensic dataset from the 2017–2018 DFRWS Forensic Challenge,[3] where Simon was been investigated for murdering his wife, Betty, whose dead body was found on the floor of their house. This dataset contains the forensic data obtained from a Rasberry Pi, two Samsung Note 2 (belonging to Simon and Betty), a smart watch, a sensor, a Samsung SmartHub, etc., as well as Google OnHub Diagnostic report, Amazon Echo Cloud Data, and MDS (Acme, Inc.) Smarthome Network Dump.

Using this dataset, we built a schema pool using the schema generator, comprising of a set of schemas, one for each Entity of interest (e.g., Samsung SmartThings application on Samsung Note 2). For example, when analyzing Simon’s Samsung Note 2, we found a number of files. One of the files was located at `com.smartthings.android\cache\http\fff1bb188c7cdb095a6653a6433028dc.1`, and we were also able to identify this file as a JSON file (see the underlined file in Fig. 5).

As the file was parsed using the ForKaS prototype, we found that the file stored the data of the sensors in the SmartHub network. Therefore, we selected the most important data of the file that represent time, device name, device ID, device state, and state value, respectively. We also set the visualization module to accept such data as long as the data is tagged with “time,” “device,” “ID,” “state,” and “value.”

Once all recognized files of the application have been investigated, the selected data will then be displayed on the timeline-based visualization panel (see Fig. 6). The visualization panel labeled each record with their “device state” label, grouped the data with their “device name,” and displayed the respective record with their “device ID” in different colors. The remainder of the selected data would be displayed when the mouse cursor hovers over it.

[3] [Online]. Available: https://jijames.github.io/DFRWS2018Challenge/



- Samsung Smartthings C:\Users\XIAOLU\Desktop\Sma
  - SmartThings C:\Users\XIAOLU\Desktop\SmartThing
    - SmartThings_DFRWS C:\Users\XIAOLU\Desktop\
      - com.smartthings.android C:\Users\XIAOLU\De
        - app_MixpanelAPI.Images.DecideChecker C:\
        - app_MixpanelAPI.Images.ViewCrawler C:\Us
        - app_webview C:\Users\XIAOLU\Desktop\Sm
        - cache C:\Users\XIAOLU\Desktop\SmartThing
          - com.google.android.gms.maps.volley C:\U
          - http C:\Users\XIAOLU\Desktop\SmartThin
            - fff1bb188c7cdb095a6653a6433028dc.1
            - f6e7888429b1af15005bd87e1f81212e.1
            - f17c2d720fccefe2e6d41c469a9635bb.1
            - ebfb062d8b50a7dfc225cb2d4318c6d2.1
            - e99c2527df7643af5e834df226db32d5.1
            - e51052f9bde22a92b4f7d5ebb02e0fdc.1
            - ddcadd1ae88678e659e03ef3c3e0ccd3.1
            - d77f15c4365dcad1fdaa7767581fc185.1
            - d43808286c9af1965b3731f5add86499.1
            - cb44cc0cdee8a0235ec9da52dc3b4eb8.1
            - c83a4ea49a8089892e4acf2c442b54a4.1
            - c76c16befbd97e5be2b63d047a93c360.1
            - c124d693e13f9f7f52d875bdc1a40671.1

fff1bb188c7cdb095a6653a6433028dc.1 ✕ | bac39b1d4e5a7d405382e7cf45eda2a0.1 ✕

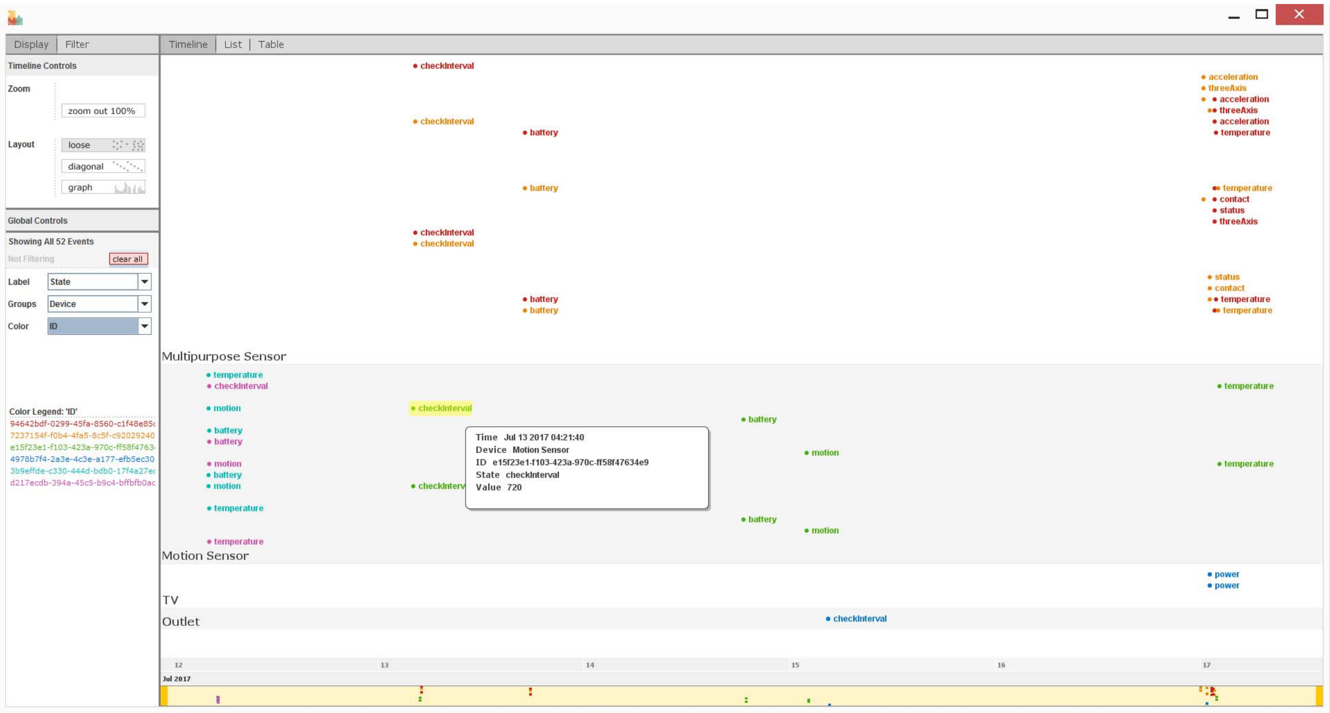
- {} currentStates
  - [] 0
    - □ hubId : 3bba4530-8def-4606-90af-432f8fd0a151
    - □ displayed : true
    - □ date : 2017-07-12T09:32:31.420Z
    - □ translatable : true
    - □ deviceTypeId : 821425e6-b305-468a-9a27-db0e4852ce3a
    - □ data : {"microDeviceTile":{"type":"value","label":"78°","backgroundColor":"#89c314"}}
    - □ rawDescription : catchall: 0104 0402 01 01 0140 00 BE96 00 00 0000 01 01 00000029FC09
    - □ isStateChange : true
    - □ eventSource : DEVICE
    - □ description : {{ device.displayName }} was {{ value }}°F
    - ☑ linkText : Motion Sensor
    - □ isVirtualHub : false
    - ☑ deviceId : d217ecdb-394a-45c5-b9c4-bffbfb0aca67
    - □ unit : F

● Time ● Device ● ID ● State ● Value

Chosen Column	Tag
[devices, currentStates, name]	State
[devices, currentStates, value]	Value
[devices, currentStates, deviceId]	ID
[devices, currentStates, unixTime]	Time
[devices, currentStates, linkText]	Device

[devices, currentStates, unixTime]	[devices, currentStates, name]	[devices, currentStates, deviceId]	[devices, currentStates, value]	[devices, currentStates, linkTex
1499851951420	temperature	d217ecdb-394a-45c5-b9c4-bffbf...	78	Motion Sensor
1499851950122	battery	d217ecdb-394a-45c5-b9c4-bffbf...	89	Motion Sensor
1499851958421	motion	d217ecdb-394a-45c5-b9c4-bffbf...	inactive	Motion Sensor
1499851966358	checkInterval	d217ecdb-394a-45c5-b9c4-bffbf...	720	Motion Sensor

Fig. 5. Screenshot of the prototype's main panel, where the circled "SmartThings_DFRWS C:Users: …" is the entity.



Timeline Controls

Zoom: zoom out 100%

Layout: loose / diagonal / graph

Global Controls

Showing All 52 Events

Not Filtering clear all

Label: State

Groups: Device

Color: ID

Color Legend: 'ID'

94642bdf-0299-45fa-8560-c1f48e85
7237154f-f0b4-4fa5-8c5f-c92029240
e15f23e1-f103-423a-970c-ff58f4763
4978b7f4-2a3e-4c3e-a177-efb5ec30
3b9effde-c330-444d-bdb0-17f4a27e
d217ecdb-394a-45c5-b9c4-bffbfb0ac

Time Jul 13 2017 04:21:40
Device Motion Sensor
ID e15f23e1-f103-423a-970c-ff58f47634e9
State checkInterval
Value 720

Fig. 6. Screenshot of the prototype's visualization panel.

In summary, as Fig. 6 shows, the records of the four smart devices (i.e., multipurpose sensor, motion sensor, TV, and outlet) were retrieved. In terms of *time*, all the records were distributed on the time axis. Note that, the time stamp shown on the figure was the current Unix epoch time (i.e., not affected by the time zone). These generated schemes are then being sent to the schema pool.

A. Data Analysis

In our case study, we also built our own SmartHub network, which includes a motion sensor, two multipurpose sensors, and one power outlet. We installed the SmartHub application—*SmartThings Classic* (v2.16.0) on a rooted Samsung Galaxy S7 (Android 6.0) and a jailbroken iPad mini (IOS 9.3.5). Once the SmartHub application on either device was paired with the

TABLE II
FINDINGS FROM THE ANALYSIS OF THE TEST ANDROID AND IOS DEVICES: A SNAPSHOT

Time	Device name	Device id	Action	Value
Thu May 17 14:05:53 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	temperature	70
Thu May 17 10:21:04 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	temperature	69
Thu May 17 09:32:52 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	status	closed
Thu May 17 09:30:46 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	status	open
Thu May 17 08:26:09 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	temperature	70
Thu May 17 07:51:10 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	temperature	71
Thu May 17 07:41:11 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	temperature	72
Thu May 17 06:26:13 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	temperature	71
Thu May 17 05:41:15 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	temperature	70
Thu May 17 05:16:16 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	temperature	69
Sat May 19 23:53:01 CDT 2018	Motion Sensor	dc398969-5c0b-428d-b43b-b1c8fde57377	temperature	67
Tue May 15 11:09:28 CDT 2018	Motion Sensor	dc398969-5c0b-428d-b43b-b1c8fde57377	battery	100
Thu May 17 09:33:01 CDT 2018	Motion Sensor	dc398969-5c0b-428d-b43b-b1c8fde57377	motion	inactive
Fri Nov 17 19:18:31 CST 2017	Motion Sensor	dc398969-5c0b-428d-b43b-b1c8fde57377	checkInterval	720
Tue May 15 20:04:22 CDT 2018	Outlet	d833119b-b4a5-4d83-9258-ab4d1502a2c3	switch	on
Tue May 15 12:07:15 CDT 2018	Outlet	d833119b-b4a5-4d83-9258-ab4d1502a2c3	power	0.0
Fri Nov 17 19:32:15 CST 2017	Outlet	d833119b-b4a5-4d83-9258-ab4d1502a2c3	checkInterval	720
Sat May 19 03:11:36 CDT 2018	Multipurpose Sensor 2	73661b88-061a-49df-a460-35db2cab2d7b	temperature	66
Sat May 19 19:12:51 CDT 2018	Multipurpose Sensor 2	73661b88-061a-49df-a460-35db2cab2d7b	battery	89
Tue May 15 14:14:54 CDT 2018	Multipurpose Sensor 2	73661b88-061a-49df-a460-35db2cab2d7b	contact	closed
Tue May 15 14:16:11 CDT 2018	Multipurpose Sensor 2	73661b88-061a-49df-a460-35db2cab2d7b	threeAxis	84,1027,3
Tue May 15 14:16:10 CDT 2018	Multipurpose Sensor 2	73661b88-061a-49df-a460-35db2cab2d7b	acceleration	inactive
Fri Nov 17 20:03:45 CST 2017	Multipurpose Sensor 2	73661b88-061a-49df-a460-35db2cab2d7b	checkInterval	720
Tue May 15 14:14:54 CDT 2018	Multipurpose Sensor 2	73661b88-061a-49df-a460-35db2cab2d7b	status	closed
Sat May 19 12:48:24 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	temperature	67
Tue May 15 23:49:51 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	battery	100
Thu May 17 09:32:52 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	contact	closed
Wed May 16 14:59:32 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	threeAxis	-5,1032,8
Fri Nov 17 19:20:51 CST 2017	Multipurpose Sensor	184fb490-1c61-49a6-bf75-31acd2db198d	checkInterval	720
Thu May 17 09:32:52 CDT 2018	Multipurpose Sensor 01	184fb490-1c61-49a6-bf75-31acd2db198d	status	closed
Thu May 17 06:57:28 CDT 2018	Motion Sensor	dc398969-5c0b-428d-b43b-b1c8fde57377	temperature	71
Thu May 17 05:57:27 CDT 2018	Motion Sensor	dc398969-5c0b-428d-b43b-b1c8fde57377	temperature	70
Thu May 17 00:02:24 CDT 2018	Motion Sensor	dc398969-5c0b-428d-b43b-b1c8fde57377	temperature	69
Wed May 16 21:52:23 CDT 2018	Motion Sensor	dc398969-5c0b-428d-b43b-b1c8fde57377	temperature	70
Wed May 16 20:41:54 CDT 2018	Motion Sensor	dc398969-5c0b-428d-b43b-b1c8fde57377	motion	inactive
Wed May 16 20:40:36 CDT 2018	Motion Sensor	dc398969-5c0b-428d-b43b-b1c8fde57377	motion	active
Wed May 16 20:40:15 CDT 2018	Motion Sensor	dc398969-5c0b-428d-b43b-b1c8fde57377	motion	inactive
...	...	...	...	

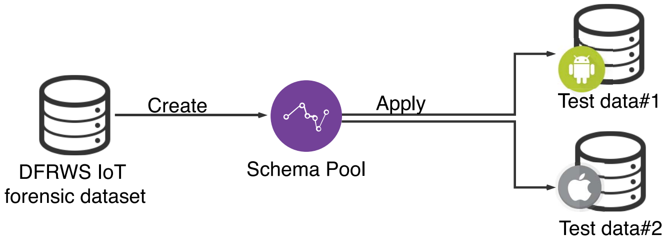


Fig. 7. Deploying the ForKaS platform in our case study.

SmartHub network, the data would be synchronized between the device(s) and the SmartHub. After a week of activities, we extracted the data of the SmartHub application from both devices.

To extract the data from the iOS device, we opened the SSH service on the device and located the folder of the SmartThing application at `/var/mobile/Containers/Data/Application/307CAC89-E4A8-4021-9D91-AB17EECF88E6/`. Data from the Android device was located at `/data/data/com.smartthings.android` (see also our Github repository).[4] Note, the platform does not need metadata (the directory/name of the file) for recognizing and extracting the artifacts. The extracted folder and files do not have to maintain their original metadata. While the current prototype does not hide such metadata from investigators, it easily could for a further privacy preserving design.

B. Schema Application

We then made a request to the panel pool and the schema with the highest score was recommended (see Fig. 5). Once we were presented with the most appropriate schema, we loaded the data acquired from the analysis of the Android and the iOS devices and applied the schema to these data. In total, the schema retrieved 348 and 118 historical records from both test data, respectively. Table II presents the identical data extracted using the schema from both devices. In addition, we also verified that the folder `com.smartthings.android\cache\http` of the Android application and the folder `\307CAC89-E4A8-4021-9D91-AB17EECF88E6\Library\Caches\com.smartthings\fsCachedData` of the iOS application stored the files with the artifacts.

VI. Discussion

In order to verify the findings from the application of the schema, we manually compared the findings of the test data with the records on the applications of both devices. We were able to determine that the test data was retrieved correctly. However, we also remarked that there are a number of factors can affect the completeness of the data, such as historical records being cached differently between operating systems and different users having varying browsing habits (hence, different amount of data on the application was captured).

In addition, the schema recommended from the schema pool was generated from a Samsung Note 2. However, we were able to apply this schema on both our test devices of different make and model (i.e., a rooted Samsung Galaxy S7 and a jailbroken iPad mini). This demonstrated the utility of the proposed ForKaS platform, where schema generated from the analysis of a device/version may be applicable for other devices running different operating systems and versions. This is the case because, to ensure backward compatibility, developers do not usually drastically change the approach for data storage/transmission on another version of the same product or a different product (e.g., Samsung Note 2 and Samsung Galaxy s7). Furthermore, developers often use the same data structures for different products. For example, the schema, in this case, was successfully applied on a iOS device. Meanwhile, using a schema, the investigator may discover new artifacts of interest when applied on a different device. For instance, even though the schema was generated based on the analysis of an Android-based device, the location storage could facilitate investigators examining an iOS device.

There are, however, limitations in our current implementation, as shown in Fig. 7. For example, the selector and tagger modules in the current prototype are incapable of processing unstructured/encrypted files. Therefore, such files will need to be preprocessed. However, this also explains the need for knowledge sharing. For example, if there is information on the ForKaS platform that tells us user data from Samsung SmartThings, Android device, and iOS device have a similar structure, then we will be able to use a schema for Android device or iOS device from the platform to facilitate the investigation of Samsung SmartThings (as discussed in the preceding paragraph).

VII. Conclusion

Digital forensics will be increasingly important as more devices in our environment become digitalized and are capable of collecting, storing, and disseminating data.

In this paper, we proposed a knowledge sharing platform that allows forensic investigators to generate schemas from the findings of their forensic examination of devices. The generated schemas can then be shared with the broader forensic community, without the need to share sensitive data acquired from the forensic examination. We then demonstrated the potential for deploying such a platform using a case study.

Such a platform can be especially useful in time-sensitive cases. For example, a law enforcement investigator may be required to complete the forensic examination of a device or system within 72 h or a couple of days. Hence, having access to schemas generated by other experienced and trusted member of the forensic community would expedite the forensic investigation.

Future research includes the following.

1) Extending the prototype to support unstructured formats, encrypted data, and other data formats (e.g., as discussed in the preceding section) from both IoT devices and other computing systems.
2) Surveying the forensic community on the design requirements (e.g., what interfaces should be incorporated to

[4] [Online]. Available: https://github.com/xiaoluzhang1985/Samsung-Smartthings-testdata

ease the sharing of knowledge, and what modifications are required to be made in platform to meet the community's requirements).

3) Integrating ForKaS into existing forensic platforms to ease the entity extraction and schema development steps in the context of current forensic workflows.
4) Integrating ForKaS into CASE and UCO to truly provide an end-to-end, artifact-to-ontology information exchange ecosystem.
5) Developing privacy preserving data validation mechanisms to guarantee sensitive case data is not incorporated into the schema and is not transferred to the schema pool, to increase confidence and model/tool adoption.
6) Exploring the potential of using blockchain to facilitate the sharing of schemas and other data in the platform/ecosystem.
7) Leveraging machine/deep learning approaches to facilitate the forensic analysis and extraction of key artifacts, particularly from newer and emerging consumer technologies, to automatically generate the schemas.

References

[1] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of Things: Vision, applications and research challenges," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1497–1516, 2012.

[2] I. U. Din, M. Guizani, B.-S. Kim, S. Hassan, and M. K. Khan, "Trust management techniques for the Internet of Things: A survey," *IEEE Access*, vol. 7, pp. 29763–29787, 2018.

[3] I. U. Din *et al.*, "The Internet of Things: A review of enabled technologies and future challenges," *IEEE Access*, vol. 7, pp. 7606–7640, 2019.

[4] Q. Do, B. Martini, and K.-K. R. Choo, "Cyber-physical systems information gathering: A smart home case study," *Comput. Netw.*, vol. 138, pp. 1–12, Jun. 2018.

[5] D. B. Miller, W. B. Glisson, M. Yampolskiy, and K.-K. R. Choo, "Identifying 3D printer residual data via open-source documentation," *Comput. Security*, vol. 75, pp. 10–23, Jun. 2018.

[6] W. Z. Khan, M. Y. Aalsalem, and M. K. Khan, "Communal acts of IoT consumers: A potential threat to security & privacy," *IEEE Trans. Consum. Electron.*, vol. 65, no. 1, pp. 64–72, Feb. 2019.

[7] W. Z. Khan, M. Y. Aalsalem, M. K. Khan, and Q. Arshad, "Data and privacy: Getting consumers to trust products enabled by the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 8, no. 2, pp. 35–38, Mar. 2019.

[8] S. Khan, A. Gani, A. W. A. Wahab, M. Shiraz, and I. Ahmad, "Network forensics: Review, taxonomy, and open challenges," *J. Netw. Comput. Appl.*, vol. 66, pp. 214–235, May 2016.

[9] M. S. Pour *et al.*, "Understanding the IoT cyber threat landscape: A data dimensionality reduction technique to infer and characterize Internet-scale IoT probing campaigns," *Digit. Invest.*, vol. 28, suppl. S40–S49, 2018.

[10] M. M. Ahsan *et al.*, "CLASS: Cloud log assuring soundness and secrecy scheme for cloud forensics," *IEEE Trans. Sustain. Comput.*, to be published.

[11] S. Khan *et al.*, "Cloud log forensics: Foundations, state of the art, and future directions," *ACM Comput. Surveys*, vol. 49, no. 1, 2016, Art. no. 7.

[12] A. Goudbeek, K.-K. R. Choo, and N.-A. Le-Khac, "A forensic investigation framework for smart home environment," in *Proc. 17th IEEE Int. Conf. Trust Security Privacy Comput. Commun. (TrustCom)*, 2018, pp. 1446–1451.

[13] K. Barmpatsalou, T. Cruz, E. Monteiro, and P. Simoes, "Current and future trends in mobile device forensics: A survey," *ACM Comput. Surveys*, vol. 51, no. 3, 2018, Art. no. 46.

[14] X. Zhang, I. Baggili, and F. Breitinger, "Breaking into the vault: Privacy, security and forensic analysis of Android vault applications," *Comput. Security*, vol. 70, pp. 516–531, Sep. 2017.

[15] H. Chung, J. Park, and S. Lee, "Digital forensic approaches for Amazon Alexa ecosystem," *Digit. Invest.*, vol. 22, pp. S15–S25, Aug. 2017.

[16] S. Li *et al.*, "IoT forensics: Amazon Echo as a use case," *IEEE Internet Things J.*, to be published.

[17] C. Meffert, D. Clark, I. Baggili, and F. Breitinger, "Forensic state acquisition from Internet of Things (FSAIoT): A general framework and practical approach for IoT forensics through IoT device state acquisition," in *Proc. ACM 12th Int. Conf. Availability Rel. Security*, 2017, p. 56.

[18] E. Oriwoh and P. Sant, "The forensics edge management system: A concept and design," in *Proc. IEEE 10th Int. Conf. Auton. Trusted Comput. Ubiquitous Intell. Comput. (UIC/ATC)*, 2013, pp. 544–550.

[19] S. Zawoad and R. Hasan, "FAIoT: Towards building a forensics aware eco system for the Internet of Things," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, 2015, pp. 279–284.

[20] I. Baggili, J. Oduro, K. Anthony, F. Breitinger, and G. McGee, "Watch what you wear: Preliminary forensic analysis of smart watches," in *Proc. IEEE 10th Int. Conf. Availability Rel. Security (ARES)*, 2015, pp. 303–311.

[21] Q. Do, B. Martini, and K.-K. R. Choo, "Is the data on your wearable device secure? An Android wear smartwatch case study," *Softw. Pract. Exp.*, vol. 47, no. 3, pp. 391–403, 2017.

[22] G. Horsman, "Unmanned aerial vehicles: A preliminary analysis of forensic challenges," *Digit. Invest.*, vol. 16, pp. 1–11, Mar. 2016.

[23] D. R. Clark, C. Meffert, I. Baggili, and F. Breitinger, "DROP (drone open source parser) your drone: Forensic analysis of the DJI phantom III," *Digit. Invest.*, vol. 22, pp. S3–S14, Aug. 2017.

[24] P. Turner, "Unification of digital evidence from disparate sources (digital evidence bags)," *Digit. Invest.*, vol. 2, no. 3, pp. 223–228, 2005.

[25] S. Garfinkel, "Digital forensics XML and the DFXML toolset," *Digit. Invest.*, vol. 8, nos. 3–4, pp. 161–174, 2012.

[26] B. N. Levine and M. Liberatore, "DEX: Digital evidence provenance supporting reproducibility and comparison," *Digit. Invest.*, vol. 6, pp. S48–S56, Sep. 2009.

[27] M. Cohen, S. Garfinkel, and B. Schatz, "Extending the advanced forensic format to accommodate multiple data sources, logical evidence, arbitrary information and forensic workflow," *Digit. Invest.*, vol. 6, pp. S57–S68, Sep. 2009.

[28] E. Casey, G. Back, and S. Barnum, "Leveraging CybOX^{TM} to standardize representation and exchange of digital forensic information," *Digit. Invest.*, vol. 12, pp. S102–S110, Mar. 2015.

[29] O. Brady, R. Overill, and J. Keppens, "Addressing the increasing volume and variety of digital evidence using an ontology," in *Proc. IEEE Joint Intell. Security Informat. Conf. (JISIC)*, 2014, pp. 176–183.

[30] D. Kahvedžić and T. Kechadi, "DIALOG: A framework for modeling, analysis and reuse of digital forensic knowledge," *Digit. Invest.*, vol. 6, pp. S23–S33, Sep. 2009.

[31] S. Dosis, I. Homem, and O. Popov, "Semantic representation and integration of digital evidence," *Procedia Comput. Sci.*, vol. 22, pp. 1266–1275, Oct. 2013.

[32] M. Alzaabi, A. Jones, and T. A. Martin, "An ontology-based forensic analysis tool," in *Proc. Conf. Digit. Forensics Security Law*, 2013, p. 123.

[33] P. Nimbalkar, V. Mulwad, N. Puranik, A. Joshi, and T. Finin, "Semantic interpretation of structured log files," in *Proc. IEEE 17th Int. Conf. Inf. Reuse Integr. (IRI)*, 2016, pp. 549–555.

[34] A. Cuzzocrea and G. Pirrò, "A semantic-Web-technology-based framework for supporting knowledge-driven digital forensics," in *Proc. ACM 8th Int. Conf. Manag. Digit. EcoSyst.*, 2016, pp. 58–66.

[35] E. Casey *et al.*, "Advancing coordinated cyber-investigations and tool interoperability using a community developed specification language," *Digit. Invest.*, vol. 22, pp. 14–45, Sep. 2017.

[36] A. Azfar, K.-K. R. Choo, and L. Liu, "Forensic taxonomy of Android productivity apps," *Multimedia Tools Appl.*, vol. 76, no. 3, pp. 3313–3341, 2017.

[37] A. Azfar, K.-K. R. Choo, and L. Liu, "Forensic taxonomy of Android social apps," *J. Forensic Sci.*, vol. 62, no. 2, pp. 435–456, 2017.

[38] A. Azfar, K.-K. R. Choo, and L. Liu, "An Android communication app forensic taxonomy," *J. Forensic Sci.*, vol. 61, no. 5, pp. 1337–1350, 2016.

[39] N. D. W. Cahyani, K.-K. R. Choo, N. H. Ab Rahman, and H. Ashman, "An evidence-based forensic taxonomy of windows phone dating apps," *J. Forensic Sci.*, vol. 64, no. 1, pp. 243–253, 2018.

[40] N. D. W. Cahyani, B. Martini, K.-K. R. Choo, N. H. Ab Rahman, and H. Ashman, "An evidence-based forensic taxonomy of windows phone communication apps," *J. Forensic Sci.*, vol. 63, no. 3, pp. 868–881, May 2018.

[41] O. Brady. *Exploiting Digital Evidence Artefacts*. Accessed: Apr. 25, 2019. [Online]. Available: https://kclpure.kcl.ac.uk/portal/files/95696488/2018_Brady_Owen_0966104_ethesis.pdf

[42] *AGP—Artifact Genome Project*. Accessed: Apr. 25, 2019. [Online]. Available: https://agp.newhaven.edu/

[43] G. S. Poh *et al.*, "Searchable symmetric encryption: Designs and challenges," *ACM Comput. Surveys*, vol. 50, no. 3, 2017, Art. no. 40.

Xiaolu Zhang received the Ph.D. degree in computer science from Jilin University, Changchun, China, in 2016.

He was a Visiting Ph.D. Student with the University of New Haven, West Haven, CT, USA. He is currently an Assistant Professor with the University of Texas at San Antonio, San Antonio, TX, USA. His current research interests include cyber security and digital forensics.

Dr. Zhang was the recipient of a China Scholarship Council Scholarship for his doctoral work.

Kim-Kwang Raymond Choo (SM'15) received the Ph.D. degree in information security from the Queensland University of Technology, Brisbane, QLD, Australia, in 2006.

He currently holds the Cloud Technology Endowed Professorship with the University of Texas at San Antonio (UTSA), San Antonio, TX, USA, and has a courtesy appointment with the University of South Australia, Adelaide, SA, Australia.

Dr. Choo was a recipient of the Cybersecurity Educator of the Year—APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), in 2016, the Digital Forensics Research Challenge organized by Germany's University of Erlangen–Nuremberg, in 2015, the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, the Outstanding Associate Editor of 2018 for IEEE ACCESS, the British Computer Society's 2019 Wilkes Award Runner-Up for his paper published in the 2018 volume of *The Computer Journal* (Oxford University Press), the 2019 EURASIP *Journal on Wireless Communications and Networking* Best Paper Award, the *Journal of Information Processing Systems* Survey Paper Award (Gold) in 2019, the IEEE TrustCom 2018 Best Paper Award, the ESORICS 2015 Best Research Paper Award, the 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, the Fulbright Scholarship in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award in 2008. He is also a Fellow of the Australian Computer Society, the Co-Chair of the IEEE Multimedia Communications Technical Committee's Digital Rights Management for Multimedia Interest Group, and an Honorary Commander of the 502nd Air Base Wing, Joint Base San Antonio–Fort Sam Houston.

Nicole Lang Beebe (SM'19) received the B.S. degree in electrical engineering from Michigan Technological University, Houghton, MI, USA, the M.S. degree in criminal justice from Georgia State University, Atlanta, GA, USA, and the Ph.D. degree in information technology from the University of Texas at San Antonio (UTSA), San Antonio, TX, USA.

She is the Lachman Distinguished Professor and an Associate Professor of Cybersecurity with the UTSA. She has over 20 years of experience in information security and digital forensics, from both commercial and government sectors, is a Certified Information Systems Security Professional, and holds three professional certifications in digital forensics. She has authored or co-authored several journal articles related to information security and digital forensics in *Decision Support Systems*, the IEEE TRANSACTIONS OF INFORMATION FORENSICS AND SECURITY, *Digital Investigation*, and several other journals. Her current research interests include digital forensics, information security, and data analytics.