IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 30, 2021, accepted August 9, 2021, date of publication August 13, 2021, date of current version August 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3104525

Electromagnetic Side-Channel Analysis for IoT Forensics: Challenges, Framework, and Datasets

ASANKA P. SAYAKKARA AND NHIEN-AN LE-KHAC, (Member, IEEE)
School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland
Corresponding author: Nhien-An Le-Khac (an.lekhac@ucd.ie)

ABSTRACT Electromagnetic (EM) side-channel radiation from Internet of Things (IoT) devices are shown to be effective at acquiring forensic insights during digital investigations. These EM radiation patterns can be analysed with the help of machine learning algorithms to detect internal behaviours of IoT devices, which can be relevant to an investigation. However, the real-world application of EM side-channel analysis for digital forensic purposes is obstructed by the lack of suitable tools and the technical expertise among law-enforcement communities. Although certain frameworks, such as EMvidence, exist to cater this requirement, the sheer diversity of the IoT ecosystem makes it difficult to support a sufficiently large collection of devices that are commonly encountered in forensic investigations. The work presented in this paper makes multiple contributions towards addressing this problem. Initially, a detailed discussion on the challenges of applying EM side-channel analysis in practical digital forensic purposes is provided, where the practical difficulties are illustrated. Then, it was shown that the existing EM side-channel analysis frameworks, such as EMvidence, can be used to overcome the diversity of IoT devices in forensics by equipping them with extensible plug-ins targeting the internal system-on-chips (SoC) of each device type. These plug-ins are expected to incorporate trained machine learning models, which are capable of recognising patterns of specific IoT device SoCs. However, the development of such plug-ins requires sufficiently diverse EM datasets from IoT devices. Facilitating this requirement, this work presents a comprehensive EM side-channel dataset representing a diverse collection of popular IoT devices and smartphones. The presented dataset is used to demonstrate the potential usage of machine learning models to recognise device behaviour.

INDEX TERMS IoT forensics, electromagnetic side-channel, IoT devices, datasets, software behaviour detection, machine learning.

I. INTRODUCTION

With the increasing use of Internet of Things (IoT) devices and smartphones in the day-to-day life, it is inevitable to increasingly encounter them in legal and corporate investigations [1]. This problem domain currently combines a range of existing forensic techniques, such as network forensics, file system forensics, and mobile forensics, to acquire forensic evidence from IoT devices and smartphones [2]. The large diversity of IoT devices and smartphones currently in the market and the rapid changes occur in software and hardware of existing devices, makes it challenging for law-enforcement agencies to maintain a capability to use them in investigations. This situation demands for alternative approaches that are more effective at IoT and smartphone forensics.

The electromagnetic (EM) radiation of computers is known to leak information. When program instructions are being executed on the central processing unit (CPU) of a device, the instructions themselves and the data being handled by the execution of the instructions get reflected in the EM radiation patterns [3]. This information leakage is exploited by various types of EM side-channel analysis (EM-SCA) attacks. Identifying software behaviour, detecting malicious modifications to both hardware and software, and retrieving critical data handled by a software — such as cryptographic keys — are to name a few of such EM-SCA attacks [4]. It is an active research area in the field of computer security with a large body of literature.

In addition to the information security perspective of these EM-SCA attacks, it has recently been shown that EM-SCA attacks can be used in the field of digital forensics to extract forensic insights from computing devices [5]. In contrast to the general purpose computers, such as desktop and laptop

The associate editor coordinating the review of this manuscript and approving it for publication was Guido Lombardi.

computers, smart devices have drawn a major interest in EM-SCA research in the recent times [6]. An EM-SCA attack is performed by capturing the EM radiation of a target device — usually referred to as a device-under-test (DUT) — on a particular signal frequency for a predefined period of time, and later analysing the captured EM data. Unlike the security application of EM-SCA, the forensic application is more feasible since the DUT will be under total control of the attacker, i.e., forensic investigator. While being a promising approach for IoT forensics, the application of EM-SCA techniques in the domain is still not realised due to an important reason.

In order to perform EM-SCA for acquiring forensic insights from a particular IoT device or a smartphone, the EM radiation patterns of the target device have to be profiled and incorporated into EM-SCA methods. The profiling of a particular device starts with the required specialised hardware, including DUTs and data acquisition equipment [7]. The actual work on developing the EM-SCA technique for analysing EM data of the particular DUT comes as the second step. Due to the sheer diversity of the potential DUTs that should be profiled, it is not possible for law-enforcement agencies or EM-SCA researchers to get access to a sufficiently wide collection of devices to begin with [8]. Due to this reason, the development of EM-SCA techniques targeting IoT devices and smartphones has become a slow process [9]. This situation has lead to the lack of EM-SCA method implementations to support IoT devices and smartphones that are commonly encountered in forensic investigations.

A potential approach to overcome this barrier and enable the real-world use of EM-SCA for IoT forensics is to facilitate the process by providing tools or frameworks. For example, the EMvidence framework proposes to facilitate the development of EM-SCA techniques to acquire forensic insights from IoT devices and smartphones by a large and independent community of developers [10]. When individual developers focus on specific DUTs to acquire EM radiation data and develop EM-SCA techniques, such implementations can be distributed — as extensible plug-ins to the EMvidence framework — to potential users who are in need of acquiring forensic insights from those types of device. In that way, the burden of possessing a large collection of IoT devices and smartphones in one organisation is minimised.

While this approach is promising, there is a lack of sufficiently implemented EM-SCA methods to the EMvidence framework for it to be widely adopted among law-enforcement agencies. This work focuses on this problem and offers solutions to enable the real-world use of EM-SCA in IoT and smartphone forensics. Along this avenue, a diverse set of IoT devices and smartphones were used in this work to acquire EM radiation data representing their internal behaviour. Those data are thoroughly validated to ensure that they sufficiently captures the leakage information of each considered device. Later, the acquired EM datasets are used to demonstrate the possibility of building machine learning models as plug-ins for the EMvidence framework.

This work makes the following contributions to the domain of IoT Forensics:

- Identifies and discusses the challenges associated with applying EM-SCA for gaining forensic insights from IoT devices.
- Enables and accelerates EM-SCA for IoT forensics research by presenting an EM side-channel dataset covering a broad range of IoT devices and smartphones.
- Demonstrates the use case of EM dataset for building plug-ins for the EM-SCA framework, which can be followed to create new EM datasets and EMvidence plug-ins by third parties.

The rest of this paper is organised as follows. Section II discusses the related work on EM-SCA on smart devices and the use of EM datasets in the domain. Section III provides a technical background related to EM-SCA for forensics, including the nature of EM side-channel radiation, the procedure of acquiring EM radiation data, and the format of captured EM trace data files. The Section IV introduces the EMvidence framework for IoT forensics using EM-SCA and the process of building plug-ins for it. Section V describes the details of producing and pre-processing the EM dataset presented in this work. The use of the presented EM dataset to build machine learning models is demonstrated in Section VI. Finally, Section VII concludes the paper highlighting the future work directions.

II. RELATED WORK

Electromagnetic side-channel analysis (EM-SCA) and the power side-channel analysis are highly related side-channel attacks due to the correlation between power consumption of a device and its EM radiation. Therefore the same methods that are used for extracting side-channel information from power consumption trace data are applicable to EM trace data as well [11], [12]. These techniques have been widely explored in the domain of information security for several decades. Among various potential use cases of EM-SCA, the most widely explored objective is the retrieval of cryptographic keys from computing devices while they are performing data encryption operations [13], [14]. Additionally, large scale monitoring of IoT devices; detecting deviations of program behaviour; and identification of malicious modifications to software and firmware are examples of other application of EM-SCA [9], [15].

In the recent years, it has been shown that deep learning algorithms are highly effective at extracting leaked information in EM radiation patterns from computers [16]. While classical EM-SCA algorithms face difficulties in dealing with subtle changes in EM data, such as the effect of external EM noise sources and the misalignment of EM trace files due to timing errors in data acquisition, deep learning models are shown to be resilient to such disparities. As a result, most of the traditional EM-SCA attacks are rapidly being replaced with their deep learning equivalents [17]. Due to this reason,

the availability of large EM datasets is a necessary condition for the rapid progress in the field of EM-SCA.

Due to the need of dealing with computers in a manner that either makes absolutely no changes or causes only clearly explainable changes to a computer, digital forensics can largely benefit from EM-SCA techniques [4], [18]. The ability to detect internal states and current behaviour of a device can assist an investigator or a law-enforcement officer to quickly assess a situation in an investigation. Since EM-SCA works when the target DUT is currently powered on, it suits well to the triage examination phase of an investigation. It has been shown experimentally that various forensic insights can be acquired through EM-SCA techniques from embedded devices, such as Arduino and Raspberry Pi, and smartphones [5], [10], [19].

The side-channel analysis dataset of the National Agency for the Security of Information Systems (ANSSI), commonly referred to as ASCAD, is a dataset that provides a collection of power traces for benchmarking cryptographic key retrieval attacks [20]. The power traces in the ASCAD database are acquired from an ATMega8515 microcontroller unit (MCU) running two implementations of advanced encryption standard (AES) algorithm. The dataset uses the hierarchical data format 5 (HDF5) [21] to distribute power traces and metadata related to each power trace, such as plaintext, cryptographic key, and ciphertext. In addition to that, the authors of the ASCAD dataset provide Python functions to facilitate the reading and the processing of the dataset for machine learning purposes.

While the ASCAD dataset is useful to test new EM-SCA methods for cryptographic key retrieval attacks, multiple factors make it unsuitable for IoT forensics research using EM-SCA. Firstly, this dataset only includes power consumption traces of the target IoT hardware platform. The use of the power side-channel requires an attacker to physically tamper the target device for capturing power measurements, thus making it less forensically sound. Secondly, the ASCAD dataset only includes the data representing a single SoC, which is not widely used in off-the-shelf IoT devices. Thirdly, the ASCAD dataset is limited to AES cryptographic operations occurring in the considered device. These factors are too limited for the broad application of EM-SCA in IoT forensics scenarios.

III. ELECTROMAGNETIC SIDE-CHANNEL FORENSICS

In order to understand the specific details of the EM side-channel dataset, it is necessary to describe the nature of the EM side-channel radiation from DUTs, and the methods and equipment that are used to capture EM radiation into EM trace files. This section illustrates the necessary technical background of those aspects.

A. NATURE OF ELECTROMAGNETIC SIDE-CHANNEL RADIATION

Time varying electrical currents are known to generate EM radiation. Therefore, any device that uses electricity as the power source, generates EM radiation. Digital electronic circuits consist of fast clock pulse generators to carry out their internal functionalities. These fast clocks easily generate EM radiation.

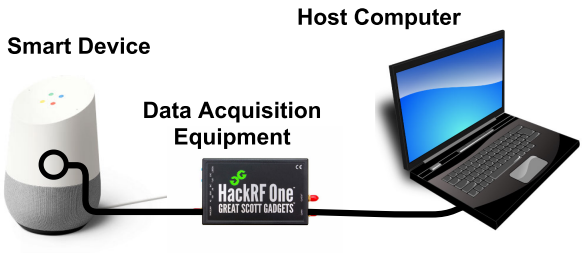

FIGURE 1. The hardware setup for acquiring EM radiation from a smart device. The data acquisition equipment – an SDR in this case – is connected with an H-loop near-field antenna from one end and with a host computer from the other end. Near-field probe is placed over the smart device, i.e., DUT, in order to capture and save EM trace files on the host computer.

A computer consists of various internal components that use fast clock pulses, such as processor, RAM, data bus lines on the motherboard, video graphics card, display units, and many more. Each of these components are handling various critical information that may have security and privacy concerns. Therefore, any potential information leakage through the EM radiation of any of those components can be interesting to attackers. Modern smart devices, such as smartphones and IoT devices, tend to employ system-on-chips (SoC), which includes many peripheral components inside a single chip. Therefore, the CPU cores of the SoC of a smart device becomes the key focus of EM-SCA attacks [22].

Whenever a CPU core is executing a program, the instructions have to be fetched, decoded and executed. These operations causes the CPU registers to be loaded and unloaded, changing the bit patterns in them. Flipping bits in CPU registers, i.e., $0 \rightarrow 1$ or $1 \rightarrow 0$, affects the energy consumption of the CPU core, which consequently results in EM radiation. Therefore, the EM signals radiated from the CPU cores correlate with the instructions and data that were handled in registers. In particular, the pattern of the EM radiation across time domain is shaped by the sequence of instructions being executed and the pattern of data loading and unloaded on registers. This means that the EM radiation patterns are influenced by both the running programs and the data being handled.

B. ACQUISITION OF ELECTROMAGNETIC RADIATION

The acquisition of EM radiation from a computing system requires multiple hardware and software components (see Figure 1). The computing system that is subjected to the EM-SCA attack is the DUT. In the case of this work, the DUT is either a smartphone or an IoT device. In order to capture the EM radiation of the DUT, a signal capturing equipment is used. This component can be either an oscilloscope, signal analyser or a software-define radio (SDR). Depending on the distance to the DUT from the signal capturing equipment, a near-field or a far-field antenna have to be used [23]. The

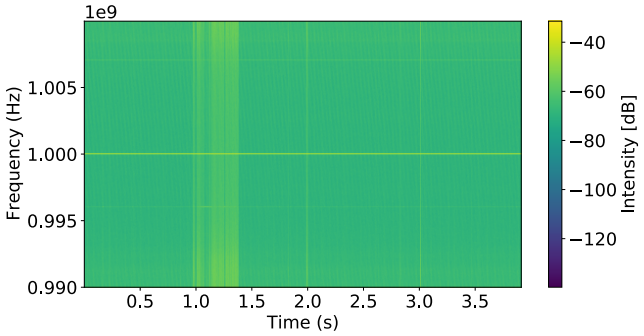

FIGURE 2. The spectrogram visualisation of EM radiation generating from an IoT device at 1 GHz device processor clock frequency, captured with a sample rate of 20 MHz using an SDR hardware.

signal acquisition equipment is connected to a host computer that runs the necessary software to read the EM data samples and save them into trace files.

While there exists multiple types of signal acquisition equipment, SDRs are preferred in this work. On SDR platforms, the hardware layer is kept to a bare minimum and all the important signal processing functionalities are handled on software [24]. The hardware layer only performs signal amplification and digitisation. Due to this reason, SDRs provide great flexibility in processing acquired EM data according to the application requirements. In this work, a HackRF One SDR is used, which supports a data acquisition frequency range from 1 MHz to 6 GHz [25]. The device has a maximum sampling rate of 20 MHz. In order to configure the SDR device and to process the data produced by it, GNU Radio library is employed on the host computer [26]. The GNU Radio library provides a graphical interface called GNU Radio Companion (GRC), which facilitates creating visual flow graphs to build EM data processing pipelines.

Since the EM radiation of interest is emerging from the SoC processor of the DUT, being closer to the SoC during data acquisition increases the strength of the signal reception. Therefore, it is advantageous to use a near-field probe for the data acquisition. In this work, an RF Explorer near-field H-loop antenna with a diameter of 25 mm was connected to the HackRF One device for the acquisition of data from the DUT within close proximity [27]. Identification of the ideal location over a DUT that maximises signal reception has been explored in the literature with various tools and algorithms [7]. For the purpose of this work, the optimum signal reception position for each considered DUT is identified by manually moving the near-field antenna across the device while plotting the spectrogram of receiving signal at the CPU clock frequency (see Figure 2). The antenna position where the signal is the strongest was fixed for the subsequent EM trace acquisition for creating the dataset.

C. FORMAT OF ELECTROMAGNETIC TRACE DATA

When capturing EM radiation through a signal acquisition equipment, there exists two potential sampling methods to use. The first is *real-valued sampling* where the signal amplitude across time domain is sampled at a fast rate. When this method is used, the sampling rate has to be at least twice as high as the frequency of the signal being observed, i.e., Nyquist sampling theorem [28]. For example, observing an EM signal at 1 GHz frequency requires at least 2 GHz sample rate. Achieving such high sampling rates is challenging due to the high cost of data acquisition equipment, and the overhead of storing and processing large amounts of data.

The second sampling method is *complex In-phase and Quadrature-phase (I/Q) sampling*, where the amplitude and the phase of a signal is captured in each sample. As a result, each sample becomes a complex number where the real component contains the signal amplitude and the imaginary component contains the signal phase for the considered time instance. While complex I/Q sampling complicates the nature of individual samples, it helps to overcome the limitation imposed to real-valued sampling by the Nyquist sampling theorem [29]. Consequently, a sampling rate much smaller than the frequency of the signal being observed can be used in the signal capturing equipment.

The HackRF One SDR device used in this work along with the GNU Radio library produces two 32 bit floating-point values for each complex I/Q sample. Consequently, each sample produced by the data acquisition setup is 8 bytes long. The HackRF One device supports sample rates up to 20 MHz and can be tuned to any frequency within the range of 1 MHz to 6 GHz. Furthermore, the device provide three signal amplifiers, namely radio frequency (RF) power amplifier (0 or 14 dB), low-noise amplifier (IF) (0 to 40 dB in 8 dB steps), and variable-gain amplifier (BB) (0 to 62 dB in 2 dB steps), which can be adjusted according to the requirements during data EM acquisition. The produced samples can be saved by default as raw data files with *.cfile* file extension. Additionally, the absolute values of the complex samples can be stored in NumPy arrays and saved into files with *.npy* file extension.

D. ELECTROMAGNETIC SIDE-CHANNEL ANALYSIS FOR FORENSICS

Whenever an IoT device is encountered in an investigation, it is highly likely that the device is powered-on. Most of these devices tend to store a minimum amount of data on-board, rendering forensic analysis of the device storage less useful. Furthermore, the devices can be employing data encryption, which makes the retrieval of stored data impossible. Therefore, most of the forensically-interesting information about such IoT devices are only available when the device is inspected alive. Turning the device off and moving it to a forensic laboratory destroys the opportunity to acquire forensic insights from the running device.

The acquisition of information related to the internal behaviour of an IoT device through its EM radiation works only when the device is actively running. Therefore, EM-SCA techniques can be applied during the triage examination of IoT devices in forensic investigations at the scene where the device is found. Furthermore, EM-SCA techniques do not require any physical tampering to the device being inspected,

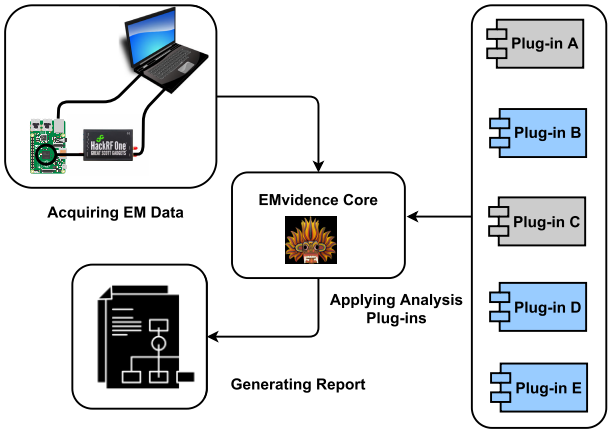

FIGURE 3. Major functional components of the EMvidence framework and their involvement in the workflow of analysing an IoT device. The plug-ins that are in gray color are disabled while the others are activated.

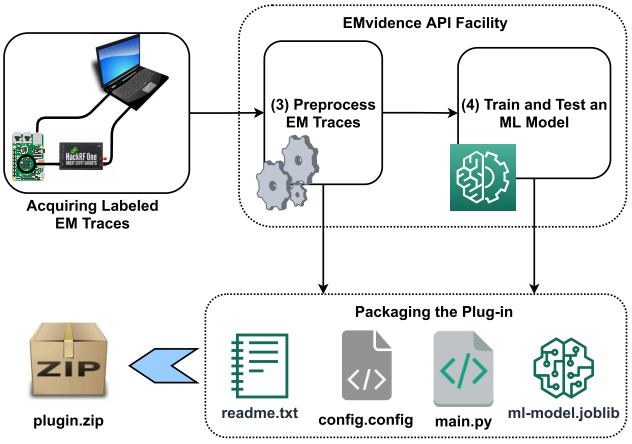

FIGURE 4. Main components of developing a plug-in for EMvidence framework by enclosing a trained machine learning (ML) model with EMvidence API facility.

making it forensic friendly. Under these circumstances, EM-SCA can be considered as an ideal approach to perform IoT forensics.

The practical use of EM-SCA in real-world IoT forensics is obstructed by two important challenges. The first is the lack of EM-SCA method implementations covering a sufficiently large ecosystem of IoT devices. It is currently not possible to perform EM-SCA in an investigation if the IoT device type in question has not been already tested for forensic insight-gathering, and if the software and hardware required for such an analysis are not readily available. As new IoT device models are entering the market rapidly, it is difficult for a single law-enforcement agency to come up with EM-SCA implementations for all of such devices. Adding to this problem, most digital forensic investigators lack the required technical expertise to implement and use EM-SCA methods for acquiring forensic insights from IoT devices.

It is agreeable that the solution to this problem is a comprehensive tool that provides the implementation of all the required EM-SCA methods for a diverse collection of IoT devices. With the need of dynamically adapting to new IoT devices with new EM-SCA implementations, such a tool needs to be highly extensible. The next section presents the architecture and the implementation of a framework called EMvidence that aims to cater these requirements in EM-SCA for IoT forensics.

IV. EMvidence FRAMEWORK FOR IOT FORENSICS

EMvidence[1] is an open-source framework, which is developed to facilitate the application of EM-SCA in digital forensics [10]. It follows an extensible architecture where only the core functionalities are included by default, such as capturing and managing EM data. It facilitates the extension of functionalities dynamically by third-party contributors. In order to facilitate the use of the EMvidence framework in a wide variety of environments and platforms, it is developed in Python language using various open-source libraries. Due to the use of GNU Radio library, it supports many SDR hardware platforms for EM data acquisition.

The Figure 3 depicts the major internal components of the EMvidence framework and their interactions. The acquisition of EM trace data from a target device and the compilation of analysis report are the two default functionalities provided by EMvidence framework. Plug-ins provide various EM data analysis capabilities individually to the framework. Meanwhile, the role of the EMvidence core is to coordinate the data acquisition, plug-ins and the report generation sequence from a central point. During an investigation, the forensic analyst can acquire EM radiation of an IoT device and enable a selection of plug-ins to engage in the data analysis. The EMvidence core applies the activated plug-ins to the acquired EM data and retrieves the results produced by them. Finally, the analysis results are handed to the report generation component to produce a comprehensive report on the leaked information through the EM radiation of the target device.

As it is evident from the design of the EMvidence framework, the plug-ins hold the key capability to perform EM-SCA on the EM radiation data of IoT devices and acquire forensic insights from them. A plug-in consists of the implementation of a specific EM-SCA method for detecting a known pattern from a specific IoT device or from a family of IoT devices that uses the same SoC. Therefore, a particular plug-in can be enabled when dealing with the IoT devices it is designed to analyse. In addition to the forensic insight acquisition, EMvidence plug-ins can be implemented to perform other tasks such as visualising or further pre-processing of data.

The Figure 4 depicts the key aspects of developing a plug-in for the EMvidence framework to acquire a specific forensic insight from an IoT device. As the first step, a specific IoT device type and the forensic insight that should be acquired from it need to be decided. Then, the SDR hardware setup can be used to acquire a labeled EM trace dataset from the target IoT device representing the specific internal behaviour of the device. The EM trace dataset presented in

[1]https://github.com/asanka-code/EMvidence

this work has already completed these initial steps of building an EMvidence plug-in. Once the dataset is ready, the API facility provided by the EMvidence framework can be used to pre-process the data and train a machine learning model to classify different labels of the dataset.

A plug-in in the EMvidence framework is basically a compressed ZIP file that encloses a collection of files. The trained machine learning model saved as a `joblib` file is the key ingredient of the plug-in ZIP file. Furthermore, a python script file needs to be included, which implements some of the important call-back functions for the plug-in. These functions are written by the plug-in developer with the help of EMvidence API to process and classify EM data, and produce results in a specific format. The EMvidence framework invokes a plug-in by calling those functions in the python script, and retrieve the results produced by the plug-in into the frameworks core for producing final analysis report. New plug-ins can dynamically be added to the framework as ZIP files and removed from the framework whenever required by the user. Currently, the EMvidence framework does not provide a plug-in management and distribution infrastructure. It is an important future work to implement such an infrastructure similar to the *apt* package manager for the Linux operating system, or the *pip* package manager for the python language.

Due to the dynamics of the IoT ecosystem, a limited number of plug-ins would not be sufficient for the EMvidence framework to be useful in real-world investigations. Whenever an investigation encounters a new IoT device or a previously studied device with a new forensic requirement, a plug-in has to be developed to fulfill this requirement. Developing new plug-ins is currently a slow and a challenging task because it requires the access to a large number of representative IoT devices for acquiring EM radiation data. Furthermore, it is necessary for the investigators to posses technical expertise to program and build plug-ins. This situation has caused the number of plug-ins being developed to be low and insufficient. The lack of plug-ins for most important IoT devices in the market needs to be addressed to make EMvidence usable in practical IoT forensics scenarios.

V. GENERATION OF ELECTROMAGNETIC DATASET

In order to overcome the limitations of using EM-SCA to acquire forensic insights, the facility to develop new plug-ins for the EMvidence framework should be improved. This work addresses this problem by providing an EM radiation dataset along with a demonstration on how to use it for building machine learning-based plug-ins for EMvidence.

A. CONSIDERED SMART DEVICES

When considering the deployment environment of an IoT application, it is important to recognise that both IoT devices and smartphones collectively play their own roles. This is because, most IoT devices typically do not consist of rich user interfaces to be interacted with. They tend to require a smartphone app, in order to provide a user interface. Therefore, the configuration and interaction with most IoT devices are performed through the users' smartphones. Due to this reason, this work considers smartphones along with IoT devices in data acquisition.

A wide variety of IoT devices demonstrate a practical forensic interest. Among this diversity of IoT devices, there exists a large consumer base for smart hubs/gateways. Regardless of the types of IoT devices and their manufacturers, modern IoT devices tend to provide application programming interface (API) to interact with common smart hubs/gateways. This means, most of the activities that occur within an IoT environment involves such intermediate devices. Due to this reason, smart hubs/gateways can be considered as the central focus of an IoT environment. Therefore, this work considers smart hubs/gateways for the dataset creation.

A total of 8 main smart device types were used for the creation of the dataset, including smart hubs/gateways – the most popular smart hubs/gateways currently in use – and smartphones. Two devices from each device type were used in order to ensure the cross-device portability of the EM data. Table 1 illustrates the technical specifications of these devices and the software activities selected from each of them to be included in the dataset. In addition to the main smart devices that were the focus of the dataset, two other supporting IoT devices were involved in the process. Those supporting devices were controlled by some of the main smart devices during data acquisition.

B. DATA ACQUISITION AND PRE-PROCESSING PROCEDURE

The data acquisition procedure with the hardware settings were mostly similar for all the devices, except for signal capturing frequency, which differs for each DUT. The system clock frequency of each DUT's CPU is considered as the information-leaking frequency for data acquisition. The other settings, which are fixed for the entire set of considered DUTs, are the sampling rate, bandwidth, and the gain values for the three signal amplifiers available on the HackRF SDR device. The sample rate and the bandwidth of the HackRF SDR were set to 20 MHz, which is the maximum capacity of the device. The higher the sample rate and bandwidth, the more information is captured and stored into EM trace files. Therefore, it was decided to use the maximum capability of the HackRF device in terms of sample rate and bandwidth.

The signal amplification values on the HackRF device have to be set to the optimum amounts because, too low or too high amplification can negatively impact the quality of EM trace data files. Having a too low amplification makes it difficult to capture a weak EM radiation coming from a DUT. However, an unnecessarily high signal amplification can inevitably amplify external noise from various other sources, cluttering the captured EM trace file. Therefore, the amplification settings were empirically decided by trying different settings and observing the clearness of the signal across different devices. As a result of these empirical obser-

TABLE 1. Technical specifications of the smart devices used to create the EM dataset.

Smart Device	System-on-Chip	Architecture	CPU Frequency	Software Activities
Amazon Echo Show 5	MediaTek MT 8163	ARMv 8-A	1.5 GHz (4 cores)	(1) asking a definition, (2) asking for time, (3) asking to play radio, (4) controlling light-bulb, (5) device idle, (6) device resetting, (7) just wake up word, (8) powering off, (9) powering on.
Amazon Echo Dot (3rd Gen)	Mediatek MT 8516	ARMv 8-A	1.3 GHz (4 cores)	(1) asking a definition, (2) asking for time, (3) asking to play radio, (4) controlling light-bulb, (5) device idle, (6) device muted (7) device resetting, (8) just wakeup word, (9) powering on.
Google Home	Marvell 88DE3006 Armada 1500 Mini Plus	ARMv 7	1.2 GHz (2 cores)	(1) asking a definition, (2) asking for time, (3) asking to play radio, (4) controlling light bulb, (5) device idle, (6) device muted (7) device resetting, (8) just wake-up word, (9) powering on.
Samsung SmartThings Hub (v2)	MCIMX6L2DVN10AB	ARMv 7-A	1 GHz (1 core)	(1) controlling smart outlet, (2) device idle, (3) device powered off, (4) device powering on, (5) opening the app, (6) viewing arrival sensor, (7) viewing door sensor, (8) view motion sensor.
Apple iPhone 4S	Apple A5	ARMv 7-A	1 GHz (2 cores)	(1) calendar app, (2) camera photo, (3) camera video, (4) email app, (5) gallery app, (6) home screen, (7) device idle, (8) phone app, (9) SMS app, (10) web browser app.
Sony Xperia T	Qualcomm Snapdragon MSM8260A	ARM v7-A	1.5 GHz (2 cores)	(1) calendar app, (2) camera photo, (3) camera video, (4) email app, (5) gallery app, (6) home screen, (7) device idle, (8) phone app, (9) SMS app, (10) web browser app.
Samsung Galaxy Grand Prime	Qualcomm Snapdragon MSM8916	ARMv 8-A	1.2 GHz (4 cores)	(1) audio recording, (2) camera photo, (3) camera video, (4) email app, (5) gallery app, (6) home screen, (7) device idle, (8) phone app, (9) SMS app, (10) web browser app.
Nokia 4.2	Qualcomm Snapdragon SDM439	ARMv 8-A	1.95 GHz (4 cores), 1.45 GHz (4 cores)	(1) calendar app, (2) camera photo, (3) camera video, (4) email app, (5) gallery app, (6) home screen, (7) device idle, (8) phone app, (9) SMS app, (10) web browser app.

vations, the radio frequency power amplifier (RF) is set to 14 dB, the low noise amplifier (IF) is set to 40 dB, and the variable-gain amplifier (BB) is set to 18 dB throughout the experiments.

The Figure 5 illustrates the GNU Radio Companion (GRC) software that was ran on the host computer for the EM data acquisition. Each block in the flow graph visually represents either a physical hardware component or a virtual element that is necessary to deal with the EM signal data samples. The *Osmocom Source* block represents the HackRF SDR device on the flow graph where the data acquisition settings are defined. As can be seen, the signal acquisition frequency, bandwidth and amplifier gain settings are defied in this block. The sample rate and the signal acquisition frequency are defined as individual variable blocks in the flow graph, which are referred to by other blocks. The I/Q data samples produced by the Osmocom Source block are directed to three other blocks in the flow graph, namely the *Frequency Sink*, the *Waterfall Sink*, and the *File Sink*. The Frequency Sink and Waterfall Sink blocks are used to visualise the I/Q data stream in real-time; this is useful for verifying that the target signal is focused during the data acquisition. The File Sink block saves

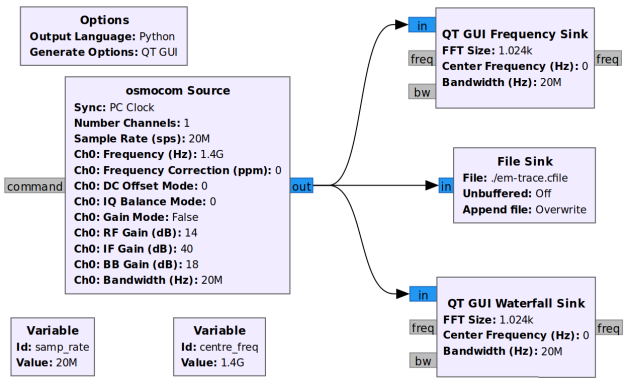

FIGURE 5. The GNU Radio Companion (GRC) flow graph for acquiring EM traces. The *Osmocom Source* represents the configuration of the HackRF SDR device producing I/Q data samples. The *Frequency Sink* and *Waterfall Sink* are used to visualise data while the *File Sink* write the I/Q data stream into a raw data file.

the I/Q data samples into a raw data file with the extension *.cfile* on the host computer.

The complete data acquisition and pre-processing pipeline for a single DUT is illustrated in the Figure 6. The first step is to prepare the hardware setup consisting of the DUT, the SDR

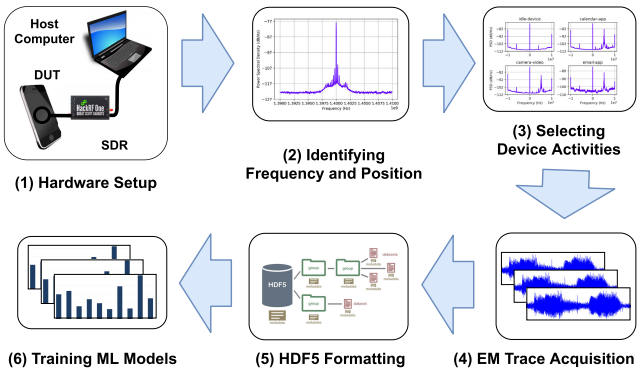

FIGURE 6. The sequence of main steps followed to produce EM dataset. Using the hardware setup, the information leaking position and frequency is identified. Then, the specific activities of each device to be recorded is shortlisted. Later, the EM traces for each activity are packaged using the HDF5 format for distribution. This dataset can be finally used to train and test ML models.

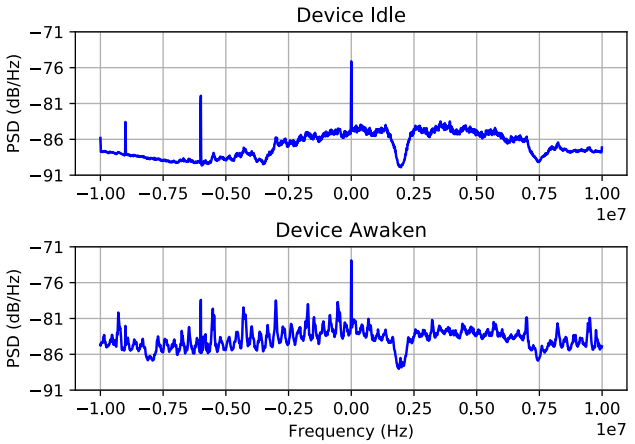

FIGURE 7. The power spectral density of the EM radiation from Amazon Echo Dot device when it is in idle state and when it was awaken by calling the wake-up word "Alexa". A drastic increase in the observed signal is visible from the device is awaken.

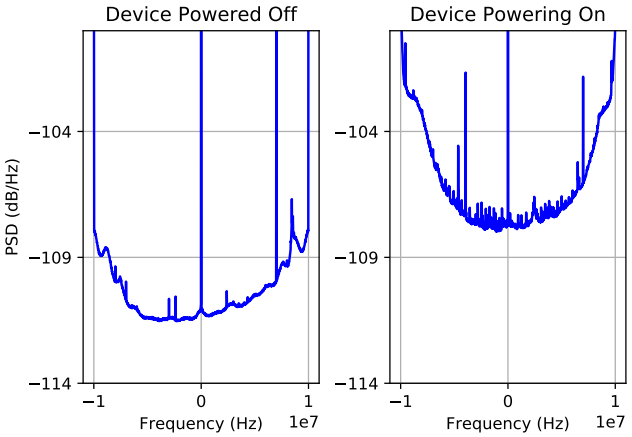

FIGURE 8. The power spectral density of the EM radiation from Samsung SmartThings Hub device when it was in powered off state and when it was being powered on by switching the power supply. A drastic increase in the observed signal is visible when the device was powered on.

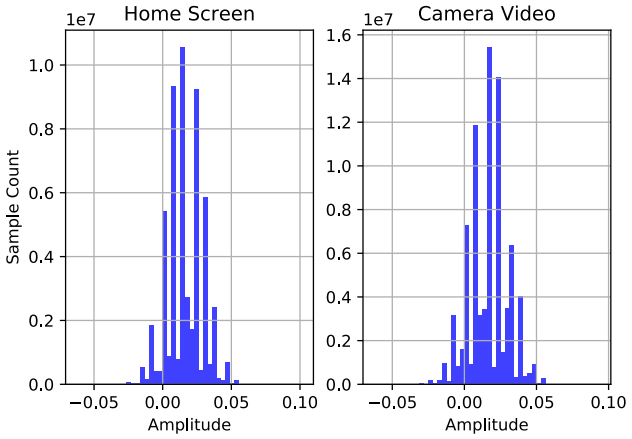

FIGURE 9. Histograms of two EM trace files representing two software behaviour of Nokia 4.2 smartphone. The distributions are not Gaussian for the EM trace data.

equipment with near-field H-loop antenna, and the host computer. The second step is to identify the information-leaking frequency and the optimum antenna position for the target DUT. Since this work considers the system clock frequency of the DUTs as the information-leaking frequency, this value is already known. However, the optimum positioning of the antenna over the DUT needs to be decided empirically by moving the antenna over the device while plotting data in real-time. The third step is the decision of target device activities to be included in the dataset. For each DUT, the set of activities were defined based on the type of the device and its real-world application scenarios.

Figure 7 depicts the PSD of the EM radiation captured from Amazon Echo Dot device while it was on two different internal states: device idle and device awaken. In the first state, the device is powered on and running, but not doing any specific interactive task (it can be running certain tasks in the background). In the second state, the device is woken up by calling the wake-up word, ''Alexa''. As it is evident from the PSD plot that the power of the observed signal on the clock frequency of the Amazon Echo Dot device considerably increased when switching from the idle state to the awaken state. Similarly, Figure 8 illustrates the PSD plots of the EM radiation observed at the clock frequency of Samsung SmartThings Hub device when it was powered off and being powered on. Similar to the previous device, the signal strength drastically increases in Samsung SmartThings Hub when it is powered on. Similar observations were made on the other considered smart devices for the EM dataset. The observation of a considerable increase in the observed EM signal at a particular frequency confirms that the observed signal is indeed originating from the target device.

In the fourth step, an EM trace was acquired for a pre-defined period of time while the device was set to run a particular desired activity, i.e., if there are n number of target device activities, an n number of EM traces were acquired separately for that device. Finally, the acquired EM trace files are packaged together in a hierarchical structure and stored for later use as the fifth step. Further details of the format of the dataset storage is described in the next section. When the EM dataset is ready, it can be used to train and test various machine learning models to identify device behaviour.

The distribution of the acquired EM data samples are not Gaussian as can be observed from histograms (see Figure 9). In order to ensure that the collection of EM trace files from each device was conducted correctly, a Wilcoxon rank test [30] was conducted among the EM trace files of each device. It can be concluded with a 95% confidence level ($p\text{-}value > 0.05$) that the samples of different EM trace files taken from the same device type has the same distribution.

C. STORAGE AND AVAILABILITY OF THE DATASET

In order to properly process EM trace files later, it is necessary to keep track of the settings of the SDR hardware during EM data acquisition. In addition to the SDR hardware settings of the raw I/Q trace files, any pre-processing settings such as the frequency domain conversion using STFT involves extra variables that should be stored as metadata. Furthermore, EM datasets can grow into a large collection that consists of hundreds of files and several gigabytes of data in size. Therefore, organisation and storage of such EM data files with the metadata associated with them is a challenge. For this purpose, this work uses hierarchical data format 5 (HDF5) standard to store a large collection of EM datasets representing various smart devices and their activities along with the EM trace metadata [21].

The Figure 10 depicts the structure of the dataset in HDF5 format. Under the root of the HDF5 hierarchy, there are two separate HDF5 categories for the IoT devices and smartphones. Under each of these device categories, further HDF5 subcategories are defined for the particular device models, e.g., Amazon Echo Dot, Google Home, etc. For each device model HDF5 subcategory, the EM traces representing the device's activities are added as HDF5 datasets. The entire HDF5 file is about 53 GB in size. When compressed using *gzip* (which is a command line-based utility program for Unix-like platforms) with a compression level of 6, the entire dataset file can be reduced to around 12 GB in size. Therefore, the compressed HDF5 dataset can be distributed, stored, and processed on a reasonably-resourced computer.

The dataset is available in the public domain, which can be downloaded along with instructions to use it .[2]

VI. MACHINE LEARNING ON THE DATASET

The EM trace datasets acquired from smart devices have to be useful for the research and development of methods for acquiring forensic insights. Among potential approaches, the use of machine learning to identify known patterns in the EM radiation is at the foremost. The objective of this section is to demonstrate how to train machine learning models in order to classify device behaviours using the EM dataset. A wide variety of machine learning techniques are applicable for this purpose. As it is impossible to test all kinds of machine learning algorithms, it was decided to use 3 machine learning algorithms: multi-layer perceptron (MLP), random forest (RF), and convolutional neural network (CNN). MLP and RF have already shown to be effective in classifying EM radiation data in previous work [5] and [31]. Therefore, they were selected to test the usefulness of the EM dataset produced in this work. In contrast to MLPs, CNNs have gained popularity in recent times for complex classification problems with highly dimensional data. Therefore, CNN was selected as the third machine learning algorithm for the evaluation with the dataset.

[2]http://aseados.ucd.ie/datasets/EMSCA

A. PREPARATION FOR MACHINE LEARNING

In order to be used with machine learning algorithms, the EM trace data need to be pre-processed to extract features. When EM side-channel radiation is generated from a DUT, the information of the internal device behaviour can leak through multiple EM frequencies around the system clock frequency of the DUT. Due to external noise sources, the captured time-domain signal buries the slight variations of the signal that are vital for accurate detection of device behaviour. In contrast, the frequency domain of a signal separates individual frequency components of the captured EM radiation, which allows the information-leaking frequency components to stand out. Therefore, it is more effective to convert the original time-domain signal into frequency domain to generate the feature vector. This can be done using short time Fourier Transform (STFT).

The time domain EM trace files were applied to STFT function with a window size, i.e., FFT size, of 2048 I/Q samples and an overlap of 256 I/Q samples. This means, each STFT sliding window has a 12% overlap. For each time domain EM trace file, the STFT operation produces a two-dimensional dataset, where the frequency dimension has 2048 columns (frequency channels). The time dimension has time points each representing a collection of 2048 samples in the original time domain signal. Each STFT-converted dataset bears the label of the original smart device's software activity. For example, for the Amazon Echo Dot device, the STFT-converted datasets bear the labels *asking-definition, asking-time, playing-radio, control-lamp, device-idle, device-muted, device-resetting, device-awaken*, and *powering-on*. Each of these labels is a class in the machine learning classification. Furthermore, the frequency dimension of each STFT-converted trace is considered as the feature vector for the machine learning algorithm, which consists of 2048 features.

B. METRICS FOR EVALUATING MODELS

When evaluating machine learning models, multiple accuracy metrics can be used. Among them, precision, recall, and F1-score are the most common. These measures are based on the number of true positive (TP), true negative (TN), false positive (FP), and the false negative (FN) predictions made by a trained model on a testing dataset.

The Equation 1 illustrates the calculation of precision of a machine learning model. It quantifies the accurate positive predictions made by a model in contrast to the total number

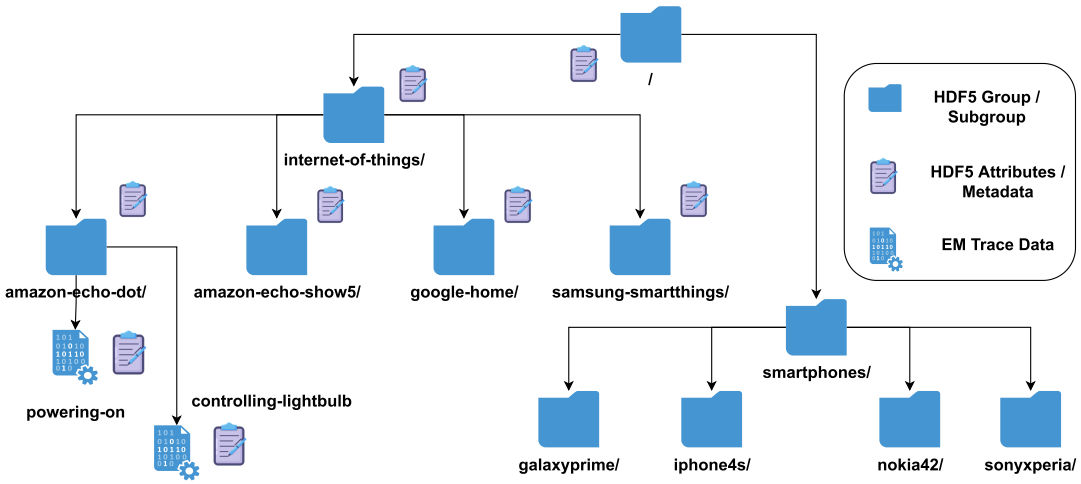

FIGURE 10. The structure of the EM dataset in HDF5 format. The HDF5 file consists of the EM trace files of smartphones and IoT devices along with their metadata in a hierarchical tree-like structure. The entire EM dataset is stored as a compressed single HDF5 file for efficient storage, distribution, and processing.

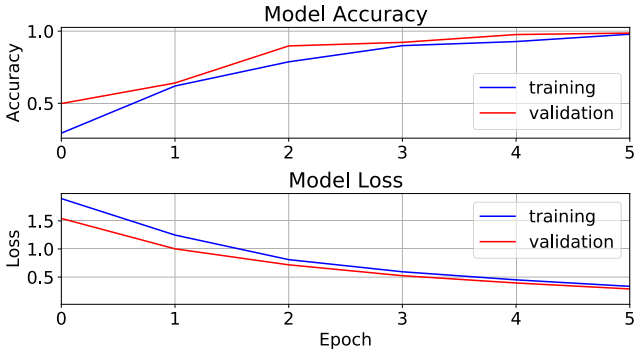

FIGURE 11. Learning curve for the MLP model that was trained using the EM trace data of Apple iPhone 4S device.

of positive predictions it has made.

$$precision = \frac{TP}{TP + FP} \tag{1}$$

The Equation 2 illustrates the calculation of recall of a machine learning model. The recall quantifies the ratio of accurate positive predictions made by a model in contrast to the total number of actual positive samples available in the testing dataset.

$$recall = \frac{TP}{TP + FN} \tag{2}$$

The two metrics, precision and recall attempts to quantify two different aspects of a trained machine learning model. By combining the two measures to calculate a harmonic mean of the values, the F1-score is defined as illustrated in the Equation 3.

$$F_1 = 2 \times \left(\frac{precision \times recall}{precision + recall}\right) \tag{3}$$

When reporting the performance of a trained machine learning model, a classification report is usually used to illustrate all the important measures for each class in the testing data. The classification report shows the precision, recall and F1-score for each class in the testing data individually. Furthermore, it shows macro average (the mean values of each measure for each class), the weighted average (the mean values of each measure calculated by weighting them with the support values), and the final accuracy, which is calculated as shown in the Equation 4. As it is evident, the final accuracy measure shows a ratio between all the correct predictions made (both positive and negative) against the total number of samples in the testing dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

In evaluations of machine learning models, two types of visualisations are used to better illustrate the outcome: confusion matrix and classification report. In a confusion matrix, the column indexes are representing the predicted labels of the classes while the row indexes are representing the true labels of the classes. Every element in the matrix shows the number of samples with a true label that got classified as a particular label. Meanwhile, classification report illustrates the precision, recall, and the F1-score values for each class of the testing data, and the accuracy for the entire testing data. The number of samples that ware used from each class for the testing is depicted by the *support* column. Macro average in a classification report represents the average of each of those parameters for all the classes. Similarly, weighted average represents the average of the parameters calculated by taking the available number of support for each class into account.

C. MULTI-LAYER PERCEPTRON

A multi-layer perceptron neural network was trained per each device for classifying its software behaviour as captured by EM trace files. The Table 2 illustrates the architecture of the MLP neural networks that were used to train classifiers. The input layer consists of 2048 features, while the number of output layer nodes depends on the number of classes for each device type. There are 6 hidden layers in this network where

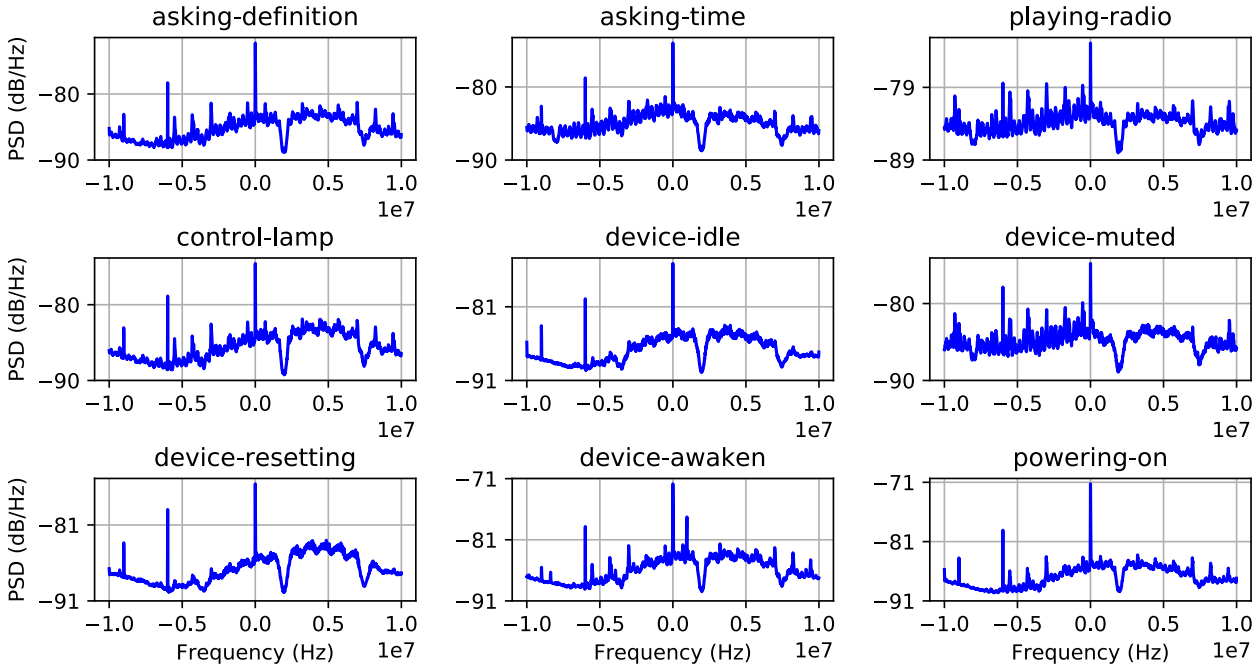

FIGURE 12. The PSD plots of the EM trace data of 9 internal behaviour classes of the Amazon Echo Dot device that were used to train classifiers using deep neural networks and random forest.

TABLE 2. The hypterparameters of the deep neural network architecture for devices with data of 10 internal software behaviours.

Layer Type	Output Shape	# of Parameters
Dense (ReLU)	1400	2868600
Dense (ReLU)	800	1120800
Dense (ReLU)	500	400500
Dense (ReLU)	200	100200
Dense (ReLU)	100	20100
Dense (Softmax)	10	1010
Total Parameters		4,511,210

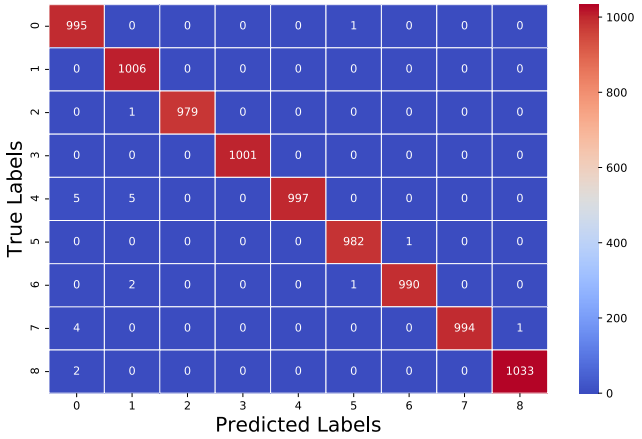

FIGURE 13. The confusion matrix of the classifier built using deep neural networks to distinguish between 9 device activity classes of the Amazon Echo Dot device.

5 of them use *ReLU* activation function and the 6th layer uses *Softmax* activation function. From each class, 10,000 samples were used for each deep learning model, where training and testing data were separated in 9:1 ratio. Each model was trained for 5 epochs (see Figure 11) with the *sparse categorical cross-entropy* loss function. From the total training samples of all the classes, a validation data set was created in each epoch by randomly selecting a 10% of the training data. For developing and testing deep neural networks, the Keras API, provided by the Tensorflow library, was used. The PSD plots of the EM traces from Amazon Echo Dot device that were used to train a machine learning models are shown in the Figure 12. The Figure 13 illustrates the confusion matrix of the machine learning classifier to distinguish between 9 different classes of the Amazon Echo Dot device. This classifier achieved an accuracy of 99.68% on the test data. The Table 3 shows the classification report of the model.

TABLE 3. Classification report of the muti-layer perceptron model for Amazon Echo Dot device.

Class	Precision	Recall	F1-score	Support
asking-definition (0)	1.00	1.00	1.00	996
asking-time (1)	0.98	1.00	0.99	1006
playing-radio (2)	1.00	1.00	1.00	980
control-lamp (3)	1.00	1.00	1.00	1001
device-idle (4)	1.00	0.99	1.00	1007
device-muted (5)	1.00	1.00	1.00	983
device-resetting (6)	1.00	1.00	1.00	993
device-awaken (7)	1.00	1.00	1.00	999
powering-on (8)	1.00	1.00	1.00	1035
Macro Avg	1.00	1.00	1.00	9000
Weighted Avg	1.00	1.00	1.00	9000
Accuracy			1.00	9000

D. RANDOM FOREST

A random forest classifier works by aggregating the results of a large collection of decision trees. When developing random forest classifiers for the devices, the same data pre-processing

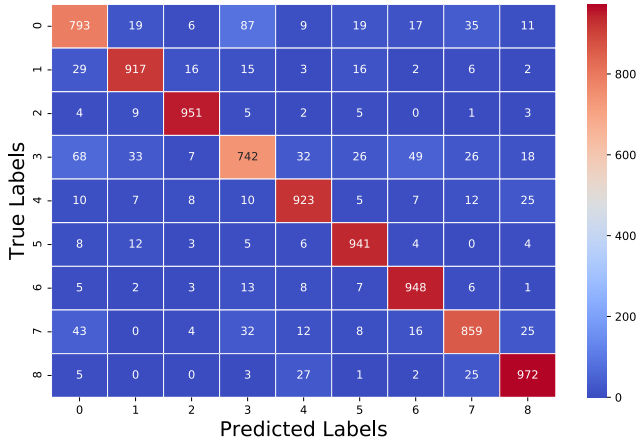

FIGURE 14. The confusion matrix of the classifier build using random forest to distinguish between 9 device activity classes of the Amazon Echo Dot device.

TABLE 4. Classification report of the Random Forest model for Amazon Echo Dot device.

Class	Precision	Recall	F1-score	Support
asking-definition (0)	0.82	0.80	0.81	996
asking-time (1)	0.92	0.91	0.91	1006
playing-radio (2)	0.95	0.97	0.96	980
control-lamp (3)	0.81	0.74	0.78	1001
device-idle (4)	0.90	0.92	0.91	1007
device-muted (5)	0.92	0.96	0.94	983
device-resetting (6)	0.91	0.95	0.93	993
device-awaken (7)	0.89	0.86	0.87	999
powering-on (8)	0.92	0.94	0.93	1035
Macro Avg	0.89	0.89	0.89	9000
Weighted Avg	0.89	0.89	0.89	9000
Accuracy			0.89	9000

procedure was followed, hence the feature vector consisted of 2048 features. The scikit-learn machine learning library was used to implement and test the random forest classifiers. Through empirical trials, the random forest classifiers were designed to have 5 estimators and a maximum depth of 10. Similar to the case of MLP, 10,000 samples from each class were taken from the dataset and were separated by 9:1 ratio as training and testing data. The Figure 14 illustrates the confusion matrix of the random forest classifier designed to distinguish between 9 different device behaviour classes of the Amazon Echo Dot device. The classifier for this device achieved an accuracy of 89.4% on test data. The Table 4 shows the classification report of the model.

E. CONVOLUTIONAL NEURAL NETWORK

A convolutional neural network (CNN) differs from the traditional multi-layer perceptron due to the use of one or many special layers called convolutions and pooling. During the training phase, convolutional layers learn their kernel parameters, while pooling layers help to drop the unnecessary features captured by the convolutional layers. The Table 5 illustrates the architecture of CNN used in this work to train and test classifiers using the EM trace dataset. It consists of, most importantly, two convolutional layers and two dense layers, which have trainable parameters. The two convolutional layers and the intermediate dense layer uses *ReLU*

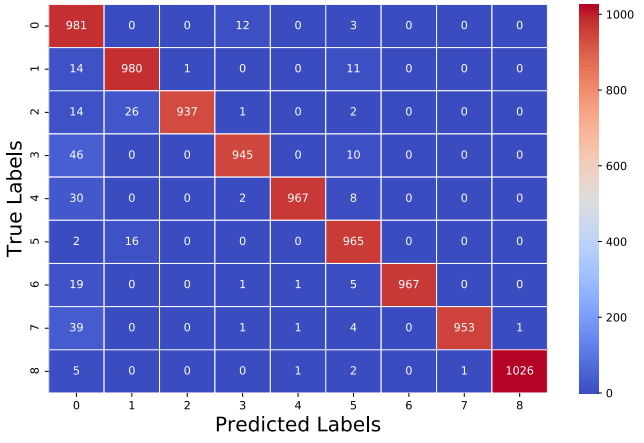

FIGURE 15. The confusion matrix of the classifier build using CNN to distinguish between 9 device activity classes of the Amazon Echo Dot device.

TABLE 5. The hyper-parameters of the convolutional neural network architecture for devices with data of 10 internal software behaviours.

Layer Type	Output Shape	# of Parameters
Conv1D (ReLU)	(None, 1021, 32)	288
MaxPooling1D	(None, 255, 32)	0
Conv1D (ReLU)	(None, 62, 64)	16448
MaxPooling1D	(None, 15, 64)	0
Dropout	(None, 15, 64)	0
Flatten	(None, 960)	0
Dense (ReLU)	(None, 64)	61504
Dropout	(None, 64)	0
Dense (Softmax)	(None, 9)	585
Total Parameters		78,825

TABLE 6. Classification report of the Convolutional Neural Network model for Amazon Echo Dot device.

Class	Precision	Recall	F1-score	Support
asking-definition (0)	0.85	0.98	0.91	996
asking-time (1)	0.96	0.97	0.97	1006
playing-radio (2)	1.00	0.96	0.98	980
control-lamp (3)	0.98	0.94	0.96	1001
device-idle (4)	1.00	0.96	0.98	1007
device-muted (5)	0.96	0.98	0.97	983
device-resetting (6)	1.00	0.97	0.99	993
device-awaken (7)	1.00	0.95	0.98	999
powering-on (8)	1.00	0.99	1.00	1035
Macro Avg	0.97	0.97	0.97	9000
Weighted Avg	0.97	0.97	0.97	9000
Accuracy			0.97	9000

activation function, while the last dense layer uses *Softmax* activation. Similar to the MLP models, unique CNN models were trained using 10,000 samples from each class for every device in the dataset. The Figure 15 illustrates the confusion matrix of the CNN model trained using Amazon Echo Dot device to detect its 9 device activities, which are included in the EM dataset. The CNN classifier for this device achieved an accuracy of 96.89% on the test data. The Table 6 shows the classification report of the model.

F. SUMMARY OF THE RESULTS

The length of the feature vector, i.e., 2048, and the size of the dataset of this work has made the use of n-fold cross-validation challenging. It is because, the computational

TABLE 7. The summary of classifier performance. The best performing classifier is highlighted in bold text for each device model.

Device Model	Machine Learning Technique	Accuracy
Amazon Echo Dot	**Multi-layer Perceptron**	**99.68%**
	Random Forrest	89.40%
	Convolutional Neural Network	96.89%
Amazon Echo Show 5	**Multi-layer Perceptron**	**99.66%**
	Random Forrest	91.64%
	Convolutional Neural Network	98.39%
Google Home	**Multi-layer Perceptron**	**99.76%**
	Random Forrest	82.95%
	Convolutional Neural Network	99.17%
Samsung SmartThings	**Multi-layer Perceptron**	**99.96%**
	Random Forrest	85.76%
	Convolutional Neural Network	97.12%
Apple iPhone 4S	Multi-layer Perceptron	98.16%
	Random Forrest	98.90%
	Convolutional Neural Network	**99.36%**
Sony Xperia T	Multi-layer Perceptron	99.62%
	Random Forrest	98.68%
	Convolutional Neural Network	**99.67%**
Galaxy Grand Prime	Multi-layer Perceptron	99.63%
	Random Forrest	99.40%
	Convolutional Neural Network	**99.71%**
Nokia 4.2	Multi-layer Perceptron	99.32%
	Random Forrest	98.36%
	Convolutional Neural Network	**99.45%**

overhead is too high for this type of data on n-fold cross-validation, causing the experiments to take a prolonged period of time. However, due to the use of a large number of training and testing samples, and also due to the random shuffling of samples, the fixed train and test data split can be considered as a sufficient approximation in the evaluation.

The summary of the results of the machine learning models that were trained to detect activities of the smartphones and IoT devices is shown in the Table 7. In overall, the three machine learning methods – multi-layer perceptron, random forest, and convolutional neural networks – perform well in classifying device activities. Various other machine learning methods may be used for the purpose based on the specific requirements of an IoT forensics scenario.

The availability of EM datasets and the possibility of building machine learning models to accurately classify different IoT device behaviors is evident from the evaluation results. However, it is necessary to be able to use such trained models in real-world digital forensic scenarios. For this purpose, tools, such as the EMvidence framework, should incorporate the capability to acquire forensic insights with the assistance of machine learning algorithms. This can be done on the EMvidence framework by developing plug-ins by third parties for acquiring specific forensic insights from specific IoT devices. A large collection of such plug-ins can enable investigators to use EMvidence framework across a diverse ecosystem of IoT devices.

VII. CONCLUSION AND FUTURE WORK

While EM-SCA offers a promising opportunity to the field of IoT forensics, the lack of tools that can tackle the diversity and the rapid changes in the IoT system obstructs its practical use. Highlighting this challenge, this work presented an approach to mitigate it by enabling IoT forensics community to collaboratively develop and share EM-SCA forensic capabilities. This is achieved by creating independent plug-ins for the open-source EMvidence framework, where each plug-in encloses a machine learning model trained to detect a specific forensic insight from a specific type of IoT devices.

For developing plug-ins for the EMvidence framework, the developers should have access to IoT devices, which can facilitate the acquisition of EM radiation. This work further facilitated this requirement by providing a rich EM side-channel radiation dataset representing a large collection of IoT devices. The potential usage of the dataset for developing plug-ins for EMvidence is demonstrated by developing and testing machine learning models, which includes both deep neural networks and random forests. The contributions of this work open up the door to apply EM-SCA in practical IoT forensics domains and furthermore, encourages further research into the topic with the help of the presented dataset and the framework.

A. FUTURE WORK

When training machine learning models to detect specific activities of a computing systems, it is necessary to ensure that the models are portable across all devices of the same type. For example, a machine learning model trained to detect software activities of an Amazon Echo Dot device needs to be compatible with any Amazon Echo Dot device found in forensic investigations. Although multiple existing research indicate that EM radiation patterns of the same device model or component are mostly consistent, it is necessary for future research to explore the cross-device portability of machine learning models [32], [33].

The devices that are powered by batteries have to consume energy wisely for a better endurance. Therefore, modern SoC processors that are included in smartphones and IoT devices tend to use various techniques for performance improvement while being efficient at energy consumption. One such common technique is the dynamic voltage-frequency scaling (DVFS) [34] where the CPU cores of a processor dynamically adjusts its clock frequency according to the workload; the higher the workload, the faster the CPU cores are running. Another techniques is the use of multiple CPU core clusters that are fixed to run at different clock frequencies, e.g., ARM's big.LITTLE technology [35]. Depending on the workload, program threads are allocated to different sets of CPU clusters; lower workloads call low-frequency CPU clusters to operation, while higher workloads force the high-frequency CPU clusters to function. Since EM-SCA attacks generally rely on the system clock frequency as the information-leaking frequency to observe, such frequency dynamics can cause the attacker to miss vital information-leaking signals. Therefore, it is necessary for future research to explore suitable methods to adjust to such dynamics.

REFERENCES

[1] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," *J. Cleaner Prod.*, vol. 140, no. 3, pp. 1454–1464, 2017.

[2] M. Chernyshev, S. Zeadally, Z. Baig, and A. Woodward, "Internet of Things forensics: The need, process models, and open issues," *IT Prof.*, vol. 20, no. 3, pp. 40–49, May/Jun. 2018.
[3] E. Peeters, F. X. Standaert, and J. J. Quisquater, "Power and electromagnetic analysis: Improved model, consequences and comparisons," *Integr. VLSI J.*, vol. 40, no. 1, pp. 52–60, Jan. 2007.
[4] A. Sayakkara, N.-A. Le-Khac, and M. Scanlon, "A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics," *Digit. Invest.*, vol. 29, pp. 43–54, Jun. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287618303840, doi: 10.1016/j.diin.2019.03.002.
[5] A. Sayakkara, N.-A. Le-Khac, and M. Scanlon, "Leveraging electromagnetic side-channel analysis for the investigation of IoT devices," *Digit. Invest.*, vol. 29, pp. S94–S103, Jul. 2019, doi: 10.1016/j.diin.2019.04.012.
[6] R. Spreitzer, V. Moonsamy, T. Korak, and S. Mangard, "Systematic classification of side-channel attacks: A case study for mobile devices," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 465–488, 1st Quart., 2017.
[7] J. Danial, D. Das, S. Ghosh, A. Raychowdhury, and S. Sen, "SCNIFFER: Low-cost, automated, efficient electromagnetic side-channel sniffing," *IEEE Access*, vol. 8, pp. 173414–173427, 2020.
[8] J. Longo, E. De Mulder, D. Page, and M. Tunstall, "SoC it to EM: Electromagnetic side-channel attacks on a complex system-on-chip," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Springer, 2015, pp. 620–640.
[9] M. A. N. Abrishamchi, A. H. Abdullah, A. D. Cheok, and K. S. Bielawski, "Side channel attacks on smart home systems: A short overview," in *Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2017, pp. 8144–8149.
[10] A. Sayakkara, N.-A. Le-Khac, and M. Scanlon, "Facilitating electromagnetic side-channel analysis for IoT investigation: Evaluating the EMvidence framework," *Forensic Sci. Int., Digit. Invest.*, vol. 33, Jul. 2020, Art. no. 301003.
[11] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology*. Springer, 1999, p. 789.
[12] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *J. Cryptograph. Eng.*, vol. 1, no. 1, pp. 5–27, Apr. 2011.
[13] Y.-I. Hayashi, N. Homma, T. Mizuki, H. Shimada, T. Aoki, H. Sone, L. Sauvage, and J.-L. Danger, "Efficient evaluation of EM radiation associated with information leakage from cryptographic devices," *IEEE Trans. Electromagn. Compat.*, vol. 55, no. 3, pp. 555–563, Jun. 2013.
[14] Y.-I. Hayashi, "State-of-the-art research on electromagnetic information security," *Radio Sci.*, vol. 51, no. 7, pp. 1213–1219, Jul. 2016.
[15] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, "Watch me, but Don't touch me! Contactless control flow monitoring via electromagnetic emanations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1095–1108.
[16] X. Wang, Q. Zhou, J. Harer, G. Brown, S. Qiu, Z. Dou, J. Wang, A. Hinton, C. A. Gonzalez, and P. Chin, "Deep learning-based classification and anomaly detection of side-channel signals," *Proc. SPIE Cyber Sens.*, vol. 10630, May 2018, Art. no. 1063006.
[17] L. Masure, C. Dumas, and E. Prouff, "A comprehensive study of deep learning for side-channel analysis," in *Proc. IACR Cryptograph. Hardw. Embedded Syst.*, Nov. 2019, pp. 348–375.
[18] T. Souvignet and J. Frinken, "Differential power analysis as a digital forensic tool," *Forensic Sci. Int.*, vol. 230, nos. 1–3, pp. 127–136, Jul. 2013.
[19] A. P. Sayakkara and N.-A. Le-Khac, "Forensic insights from smartphones through electromagnetic side-channel analysis," *IEEE Access*, vol. 9, pp. 13237–13247, 2021, doi: 10.1109/ACCESS.2021.3051921.
[20] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Study of deep learning techniques for side-channel analysis and introduction to ascad database," ANSSI, Paris, France, Tech. Rep. 22, 2018. [Online]. Available: https://eprint.iacr.org/2018/053.pdf
[21] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, "An overview of the HDF5 technology suite and its applications," in *Proc. EDBT/ICDT Workshop Array Databases*, 2011, pp. 36–47.
[22] R. Callan, F. Behrang, A. Zajic, M. Prvulovic, and A. Orso, "Zero-overhead profiling via EM emanations," in *Proc. 25th Int. Symp. Softw. Test. Anal.*, Jul. 2016, pp. 401–412.
[23] B. B. Yilmaz, E. M. Ugurlu, M. Prvulovic, and A. Zajic, "Detecting cellphone camera status at distance by exploiting electromagnetic emanations," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2019, pp. 1–6, doi: 10.1109/milcom47813.2019.9021060.
[24] A. M. Wyglinski, R. Getz, T. Collins, and D. Pu, *Software-Defined Radio for Engineers*. Norwood, MA, USA: Artech House, 2018.
[25] M. Ossmann. *HackRF*. Accessed: Aug. 29, 2020. [Online]. Available: https://greatscottgadgets.com/hackrf/
[26] E. Blossom, "GNU radio: Tools for exploring the radio frequency spectrum," *Linux J.*, vol. 2004, no. 122, p. 4, 2004.
[27] R. Explorer, "RF explorer near field antenna kit datasheet," RF Explorer, Madrid, Spain, Tech. Rep., 2017. [Online]. Available: http://j3.rf-explorer.com
[28] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. Pasadena, CA, USA: California Institute of Technology, 1997, ch. 28.
[29] C. D. O'Connell, "Exploiting quasiperiodic electromagnetic radiation using software-defined radio," Ph.D. dissertation, Comput. Lab., Girton College, Univ. Cambridge, Cambridge, U.K., 2019.
[30] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.
[31] A. Sayakkara, L. Miralles-Pechuán, N.-A. Le-Khac, and M. Scanlon, "Cutting through the emissions: Feature selection from electromagnetic side-channel data for activity detection," *Forensic Sci. Int., Digit. Invest.*, vol. 32, Apr. 2020, Art. no. 300927. http://www.sciencedirect.com/science/article/pii/S2666281720300226, doi: 10.1016/j.fsidi.2020.300927.
[32] F. T. Werner, B. B. Yilmaz, M. Prvulovic, and A. Zajic, "Leveraging EM side-channels for recognizing components on a motherboard," *IEEE Trans. Electromagn. Compat.*, vol. 63, no. 2, pp. 502–515, Apr. 2021.
[33] E. J. Jorgensen, F. T. Werner, M. Prvulovic, and A. Zajić, "Deep learning classification of motherboard components by leveraging em side-channel signals," *J. Hardw. Syst. Secur.*, vol. 4, pp. 1–13, Jun. 2021.
[34] N. Chawla, A. Singh, M. Kar, and S. Mukhopadhyay, "Application inference using machine learning based side channel analysis," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2019, pp. 1–8, doi: 10.1109/ijcnn.2019.8852124.
[35] A. Holdings. *ARM Big LITTLE Technology*. Accessed: Dec. 9, 2020. [Online]. Available: https://www.arm.com/why-arm/technologies/big-little

ASANKA P. SAYAKKARA received the B.Sc. degree from the School of Computing, University of Colombo (UCSC), Sri Lanka, in 2013, and the Ph.D. degree in computer science from the University College Dublin (UCD), Ireland, in 2020. He is currently a Research Assistant with the School of Computer Science (CS), UCD. His research interests include the Internet of Things, digital forensics, and security.

NHIEN-AN LE-KHAC (Member, IEEE) received the Ph.D. degree in computer science from the Institut National Polytechnique Grenoble (INPG), France, in 2006. He is currently a Lecturer with the School of Computer Science, University College Dublin (UCD), Ireland. He is also the Programme Director of the M.Sc. Programme in Forensic Computing and Cybercrime Investigation and the Co-Founder of UCD-GNECB Postgraduate Certificate in fraud and e-crime investigation. Since 2013, he has been collaborated on many international and national research projects, as the Principal/CO-PI/Funded Investigator. He has published more than 150 scientific articles in peer-reviewed journal and conferences in related research fields. His research interests include cybersecurity and digital forensics, machine learning for cyber security, fraud and criminal detection, cloud security and privacy, high performance computing, and knowledge engineering.