Facilitating Electromagnetic Side-Channel Analysis for IoT Investigation: Evaluating the `EMvidence` Framework

Asanka Sayakkara, Nhien-An Le-Khac, Mark Scanlon

Forensics and Security Research Group, University College Dublin, Ireland

Abstract

The Internet of Things (IoT) has opened up new opportunities for digital forensics by providing new sources of evidence. However, acquiring data from IoT is not a straightforward task for multiple reasons including the diversity of manufacturers, the lack of standard interfaces, the use of light-weight data encryption, e.g. elliptic curve cryptography (ECC), etc. Electromagnetic side-channel analysis (EM-SCA) has been proposed as a new approach to acquire forensically useful data from IoT devices. However, performing successful EM-SCA attacks on IoT devices requires domain knowledge and specialised equipment that are not available to most digital forensic investigators.

This work presents the methodology behind and an evaluation of a framework, `EMvidence`, that enables forensic investigators to acquire evidence from IoT devices through EM-SCA. This framework helps to automate and perform electromagnetic side-channel evidence collection for forensic purposes. An evaluation of the framework is performed by applying it to multiple realistic digital investigation scenarios. In the case of attacking ECC cryptographic operations, the evaluation demonstrates that the volume of EM data that needs to be stored and processed can be significantly reduced using the framework's machine learning based approach.

Keywords:
Digital Forensics, Electromagnetic Side-Channels, Elliptic Curve Cryptography, Internet-of-Things (IoT), Machine Learning.

1. Introduction

Digital forensics involves data acquisition from digital devices in order to help progress corporate, civil and legal investigations. Traditionally, digital forensic investigators deal with personal computers or mobile devices as the principal digital evidence sources [1]. However, the emergence of the Internet of Things (IoT) has revolutionised the potential for digital forensics by opening up new sources of evidence [2]. For example, wearable fitness tracker data can enable the precise reconstruction of a person's movements. Similarly, smart home data can reveal the exact time a person entered or left a premises.

While IoT devices can provide invaluable data for digital investigations, acquisition of data from IoT devices is not a straightforward task. An IoT device is a special purpose device designed to perform a specific task. Several manufacturers produce these devices often with bespoke hardware and software designs. As a result, IoT devices lack standard interfaces and forensic acquisition methods. This can often result in a device requiring a memory chip-off procedure in order to access its data [3]. However, with the increasing application of lightweight cryptographic algorithms in IoT devices, such physical access into a device may not be a viable way to acquire forensic data [4].

All running electronic devices emit electromagnetic (EM) noise in various frequencies. This is due to the time-varying electrical currents that are used. Similarly, running computers and mobile devices are sources of strong EM noise. CPUs are considered to be one of the strongest EM noise sources in computing due to fast clock pulses used on them. The pattern of electrical pulses sent through the CPU of a computer depends on the software instructions being executed and data being handled. Consequently, CPU EM emissions has been shown to leak information about both software activities and data [5].

EM side-channel analysis (EM-SCA) is a branch in information security, which eavesdrops on these EM emissions [6]. EM-SCA techniques have been used for various purposes including software behaviour detection, software modification detection, malicious software identification, and data extraction. Among different data that can be extracted through EM-SCA techniques, cryptographic keys are of significant forensic interest. Various techniques have been developed for this purpose including simple electromagnetic analysis (SEMA), differential electromagnetic analysis (DEMA) and correlation electromagnetic analysis (CEMA).

The possibility of applying EM-SCA in digital forensic investigation scenarios involving IoT devices has been recently proposed [7]. When it is difficult or impossible to acquire forensic evidence from an IoT device, observing EM emissions of the device can provide valuable information to an investigator. Clues taken through EM-SCA may not be directly considered as court-admissible digital evidence, they can still provide use-

Email addresses: `asanka.sayakkara@ucdconnect.ie` (Asanka Sayakkara), `an.lekhac@ucd.ie` (Nhien-An Le-Khac), `mark.scanlon@ucd.ie` (Mark Scanlon)

ful insights for an investigator to find court-admissible evidence elsewhere. However, EM-SCA requires specialised equipment and technical expertise that many digital forensic investigators may lack, which poses a considerable barrier to its adoption.

This work addresses the challenge of making EM-SCA a practical reality to digital forensic investigators by outlining and evaluating a software framework called `EMvidence`. A preliminary, proof-of-concept of the framework is outlined in [8]. This work focuses on evaluating the `EMvidence` framework under multiple application scenarios. The framework is designed to facilitate extensiblity through an EM processing plug-in model. Digital investigators can enable and use such extended capabilities in practical cases, making a sustainable ecosystem.

Contributions of this work:

- Presentation of a methodology to extract forensically useful insights from IoT devices through EM-SCA.
- Demonstration and evaluation of multiple IoT device investigation scenarios where an investigator can benefit by applying EM-SCA methodology.
- Outline of an investigator-friendly open source software framework that incorporates EM side-channel analysis capability for digital forensic purposes.
- Implementation and evaluation of a methodology to automatically separate EM traces of elliptic curve cryptography (ECC) being performed on IoT devices, thus saving the storage space required to store raw EM traces.

2. Background

2.1. Related Work

Analysis of unintentional EM radiation has been identified as a method to eavesdrop on electrical and electronic devices for decades [7]. It has been demonstrated that cathode ray tube (CRT) computer displays leak sufficient amount of information to reconstruct the content being displayed through their EM radiation [9]. With the availability of off-the-shelf hardware capable of capturing weak EM signals and computers with sufficient processing power to analyse data, EM side-channel analysis on all kinds of computing devices became more viable. As a result, a multitude of research has been conducted on various EM-SCA techniques including software anomaly detection and cryptographic key recovery [7].

Kocher et al. showed that power consumption of a computing device can be used as a side-channel to extract cryptographic keys [6, 10]. When a CPU performs cryptographic operations, each value assigned to its registers gets reflected in the power consumption. By collecting a sufficient number of power consumption traces during cryptographic operations with the same key, it is possible to reveal the key using algorithms such as differential power analysis (DPA). Power consumption of the CPU is directly associated with EM radiation of the device, which opens up the EM side-channel. Therefore, variants of power analysis algorithms, such as DEMA, were introduced later in order to recover cryptographic keys using EM traces [11, 12].

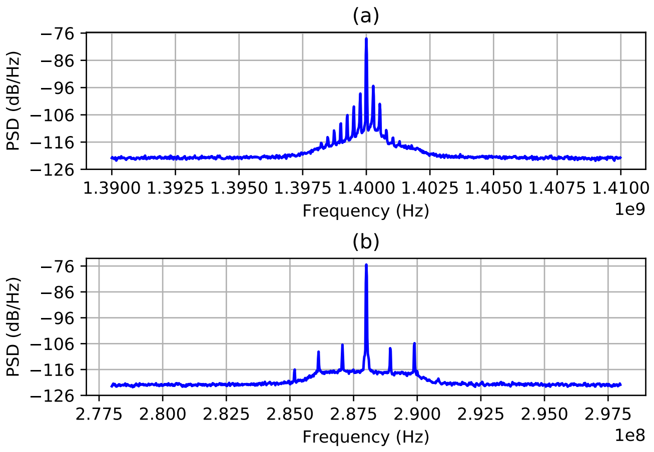

Figure 1: Leakage signals of two representative IoT devices - (a) Raspberry Pi 3 B+ at $1.4GHz$ and (b) Arduino Leonardo at $288MHz$ (18th harmonic).

In addition to the information security aspects of EM-SCA techniques, digital forensics is another field that can benefit from EM-SCA techniques. Souvignet and Frinken suggested that power analysis attacks can be used to extract data from smart-cards used by malicious skimmer devices as a method to identify victims in a forensic investigation [13]. However, it requires physically tapping into the device being investigated leading to potential tampering of evidences. In contrast, EM-SCA techniques can be more suitable for digital evidence acquisition as it does not require any physical alterations to the device being investigated [14]. However, cryptographic key recovery is still a challenging task with EM-SCA due to the fact that EM traces has to be acquired with precise alignment, which requires physical instrumentation of the device [15].

Carrying out EM-SCA attacks require precise acquisition and analysis of EM traces. In order to ease the job of information system security professionals to assess side-channel vulnerabilities of embedded systems, it is necessary to have tools. ChipWhisperer [16, 17] is a widely used tool among security professionals and academic researchers to perform cryptographic key recovery attacks. It consists of a collection of open-source trace acquisition hardware and data analysis software components. Similarly, Riscure Inspector [18] is a fully fledged commercial product that comes with software and hardware components to perform various power and EM side-channel attacks to embedded devices including smart-cards. While these tools are focused on information security objectives, we optimise the *EMvidence* framework for the specific needs of digital forensic use cases.

2.2. Observation of Electromagnetic Side-Channels

EM waves can be generated from electrical and electronic systems without the intention of the designers when conductors in a circuit accidentally behave as antennas [19, 20]. Electronic circuits that perform high-speed switching operations are especially susceptible to generating unintentional EM noise due to their higher frequencies. Among them, digital electronic components used on computers are well known sources of EM noise since they employ high-speed clocks to carry out their internal operations [5]. CPU, memory chips, data and address bus lines, various ports such as USB and Ethernet are examples of EM radiation sources on a typical computer system. Among these, EM radiation from the CPU is well known to leak information about the internal activities of the CPU including data being handled [15].

When performing an EM-SCA attack, the information about internal operations of the device being attacked are modulated into the emission signal in various ways. This exact method of leaking data through an EM side-channel is called the *leakage model*. It is necessary to assume a specific leakage model when performing an EM-SCA. From the inception of side-channel cryptographic key recovery attacks, the major leakage model that has been explored is *Hamming weight* leakage model. In multiple works by Kocher et al., it has been shown that the Hamming weight of the data being handled by a CPU gets modulated into the side-channel - hence the name of the leakage model [6, 10]. Another closely associated model that often gets considered is the *Hamming distance* leakage model where the number of bits that gets flipped is assumed to be modulated into the EM radiation [21]. Further improvements have even lead to the modelling of the exact bit transitions, which can be either $0 \rightarrow 1$ or $1 \rightarrow 0$, called *switching distance* leakage model [22].

$$A_{leak\ t} = \alpha HW(P_t \oplus K_t) + \eta_t \tag{1}$$

$$F_{observe} = F_{clock} \pm F_{leak} \tag{2}$$

When attacking a cryptographic algorithm with the intention of retrieving the encryption key under Hamming weight model, it is assumed that in a specific point in the execution of the algorithm, the Hamming weight of the cryptographic key bits are exposed [23]. For example, consider a simple cipher where a plaintext, P, is associated with a key, K, through `XOR` operations to generate the ciphertext. The amplitude of the information leaking EM signal $A_{leak\ t}$ can be modelled with the Hamming weight leakage model as shown the Equation (1). HW is the Hamming weight function while P_t and K_t are the plaintext and key bytes `XOR`-ed at the time instance t. α is a positive integer used as a scaling factor while η_t is the noise at time t. When a signal gets modulated with a carrier wave such as CPU clock or on-board radio transmitter signal through the amplitude, it causes side-bands to occur between the carrier wave [24]. For example, consider the clock frequency of a CPU to be F_{clock} and the frequency of the leakage signal to be F_{leak}. This causes the observation of a frequency bandwidth $F_{observe}$ that spans from $(F_{clock} - F_{leak})$ to $(F_{clock} + F_{leak})$ as illustrated in the Equation (2).

Observation of unintentional EM emissions from computing devices can be made using traditional signal analysis hardware, such as oscilloscopes and spectrum analysers. As an alternative, software defined radios (SDR) where fast analogue-to-digital converters (ADC) are used to digitise EM signals and feed into software for processing and visualisation. Compared to the traditional options, SDRs provide more flexibility and ease of use to non-signal analysis professionals. Figure 1 illustrates the EM signals observed at clock frequencies of two representative IoT devices using an SDR device setup, i.e., a Raspberry Pi

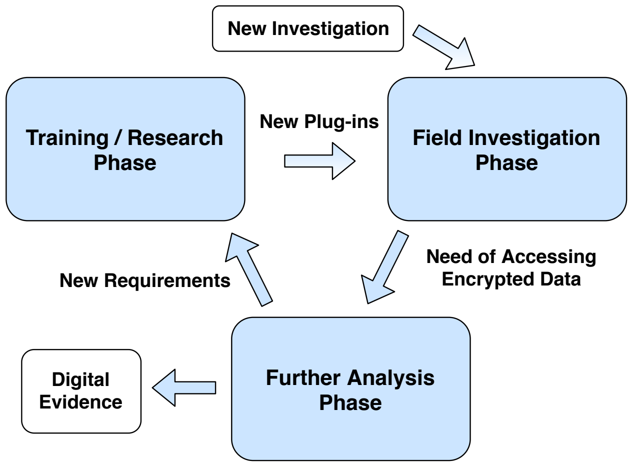

Figure 2: The three phases of EMvidence framework usage.

($1.4GHz$ CPU) and an Arduino Leonardo (running at $16MHz$). In both cases, an H-loop antenna is placed slightly above the processor chip of the devices. The antenna is connected to a *HackRF* SDR device [25] that is finally connected to a host computer running *GNURadio* software library [26] to process data. The frequency region of the Arduino was too noisy to observe directly. Therefore, it was empirically identified that the 18^{th} harmonic of the clock frequency, i.e., $288MHz$, is most appropriate to observe EM emissions of the Arduino device.

3. EMvidence Forensic Framework

This section presents the design and functionality of the *EMvidence* software framework for EM-SCA for digital forensic purposes. The key principle behind the design of EMvidence framework is to make it easy to use for digital forensic investigators who are generally non-experts of EM-SCA. Furthermore, due to the rapid changes that occur in IoT device ecosystem, it is intended to make the framework easily extendable by third parties - especially with new ML models to support new IoT devices and new information gathering.

3.1. High-level Architecture

In order to gather forensically useful information using EM-SCA techniques, a software tool should facilitate three main phases. The first is the *training/research* phase where IoT devices and their forensically useful software activities are profiled and added into the framework. Once the framework with such analysis capability is ready, it can be used in the second phased called the *field investigation* phase. There, an investigator can collect EM data from a suspect IoT device, referred to as "device under test" (DUT), in a real-world scenario and gather insights about the device on-the-spot. The third phase is *further analysis* where the device is taken into a forensic laboratory and used to perform advanced EM-SCA methods such as cryptographic key recovery attacks. Figure 2 illustrates these three phases of EM-SCA based digital forensics.

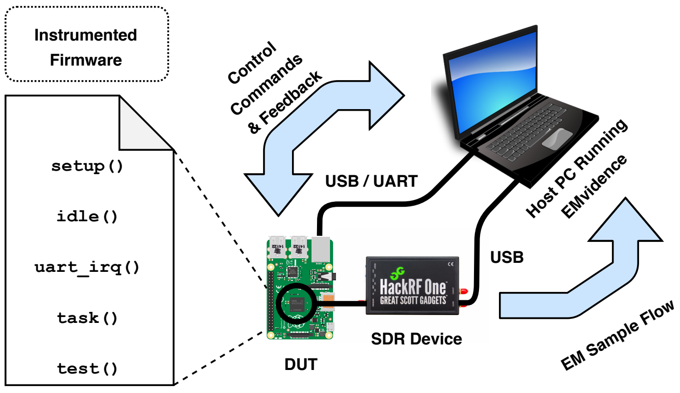

Figure 3: Controlled/Instrumented EM signal Acquisition.

EMvidence framework consists of a main core with multiple default modules and facility to add third-party plug-ins depending on future requirements. Main component of EMvidence is its core GUI that provides the default interface to a user. It also manages the modules and plug-ins by establishing communication between them in a coordinated fashion. Together with core GUI, the framework comes with three default software modules that are essential to the normal operations of the framework namely, data acquisition module, data visualisation module, and report generation module. Furthermore, depending on the requirements, third-party users can develop and add plug-ins to the core GUI of the EMvidence framework. Such plug-ins may provide various data analysis capabilities such as software behaviour detection, cryptographic key recovery, etc. The source code of the EMvidence framework and it's associated plug-ins are available at a *Github* repository[1].

3.2. Data Acquisition Module

This module facilitates the acquisition of EM data for analysis. For this purpose, a SDR device needs to be connected to the host computer where the EMvidence framework is running. Since the framework makes use of GNURadio software library [26] to handle SDR interfaces, any SDR device supported by GNURadio can be used. This module supports two types of data acquisition methods. Firstly, observation of EM emission signal can be made for a predefined period of time without any interaction or communication with the DUT from few centimetres away from it. This is the approach used in a digital forensic investigation scenario. Secondly, EM signals can be acquired while actively interacting with the DUT in scenarios where it is safe to communicate with the device through an interface such as universal serial bus (USB), universal asynchronous receiver/transmitter (UART), or Ethernet.

While interactive EM data acquisition can be used to profile a new type of IoT, it requires precise coordination between the SDR, DUT, and the host computer. Consider a scenario where it is required to build a ML model to detect a specific piece of software code running on an IoT processor/microcontroller. As the training samples for ML, individual EM traces are needed – each representing the exact time period where particular software is executed. EMvidence provides functionality to send commands to the target device to start and stop performing a particular task through USB or UART interfaces. Users can program target devices to run associated programs upon receiving these commands. With this setup, EMvidence can command the target device to start the required software operation and start saving EM data from the SDR device. Once EMvidence receives feedback that software operation is complete, it immediately ceases data acquisition. By repeating this process, a large number of EM trace files can be acquired representing a specific software behaviour on the target device. Figure 3 illustrates the acquisition of EM traces in a coordinated fashion from a target device. Once such data are used to build ML models, they can be incorporated into EMvidence framework to inspect similar IoT devices in investigative scenarios.

3.3. Data Visualisation Module

In the simplest form, an EM signal can be visualised as a time domain signal; where the x-axis represents time and the y-axis represents amplitude. In order to see the individual frequency components of a signal, it can be converted to the frequency domain using Fast Fourier Transform (FFT). Furthermore, FFT can be applied in short time intervals, i.e., Short-Time Fourier Transform (STFT), to generate a spectrogram – a graph where the x-axis represents observation time and the y-axis represents the frequency. The colour codes are used within a spectrogram to represent the amplitude of the signal at a particular time instance in a particular frequency.

There are two methods for EM data visualisation. The first is reading data from the SDR device and real-time visualisation thereof. This approach is useful for visually identifying the frequencies where information leakages occur from the DUT. A spectrogram is generally used to observe such suspicious EM signals for information leaking patterns. The second method is visualising EM data taken from saved files. Such off-line EM data visualisations are useful for EM traces acquired outside of EMvidence. Users can interpret such an externally-captured EM trace file by observing it visually with EMvidence. An FFT can be used in this scenario (see Figure 4).

3.4. Third-Party Plug-ins

The IoT ecosystem is highly dynamic in nature. New IoT devices enter into the market continuously while existing IoT devices can change frequently due to software updates. Furthermore, various novel EM-SCA analysis techniques are released time to time that can be used to inspect IoT devices. Keeping up with these changes is not an easy task for any digital forensic investigation tool. This is where the necessity arises to support development of third-party plug-ins to the software tool depending on user requirements. When a large community of users continuously develop and provide new plug-ins to the platform, it can keep up with the dynamic IoT ecosystem.

EMvidence plug-ins enable extra functionality to analyse EM data. EMvidence provides an application programming in-

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

terface (API) for plug-ins to take services from the core framework, such as providing access to real-time EM signal samples or saved EM trace files. Furthermore, a plug-in can provide information back to the core framework. Consider a situation where a user develops a new ML model to recognise a particular software activity running on a specific IoT device. This trained model can be easily integrated into EMvidence by wrapping it with API calls to the framework. Once developed, such plug-ins can be either distributed and used among the users of EMvidence framework or submitted to the source code repository of the framework in order to be distributed officially. When an investigator enables this particular plug-in during an investigation, EMvidence delivers EM signal data to the plug-in, lets the investigator interact with it, and receive ML classification results back.

4. Evaluation

The goal of this section is to evaluate the potential of using EMvidence framework in multiple practical aspects of information gathering from IoT devices. IoT devices that are in wide use in day-to-day life comes in all shapes and sizes. While each IoT device can consist of a unique combination of hardware and software, they can broadly be categorised into two classes based on their computing resources as *low-end* and *high-end* IoT devices. The design choice of choosing computing hardware resources to be included in an IoT device depends on multiple reasons. Among them, source of power is an important factor. A device that can be continuously mains powered can have more powerful hardware, and therefore, can be considered as a high-end IoT device. Meanwhile, a device that has to rely on a battery for a prolonged period of time needs to use an energy efficient processor such as micro-controllers or system-on-chips (SoC). Hence, such devices can be considered as low-end IoT devices and include health implants, fitness wearable devices, and smart light bulbs. Due to limited on-board storage, these devices generally do not store much data. They tend to either transmit data into an associated smart-phone app or hub, or directly to a cloud service. Therefore, even if a low-end IoT device contains some non-volatile data storage, e.g., an SD card, it is less likely that traditional digital evidence extraction methods would prove fruitful.

Meffert et al. highlighted the need for identifying the running forensic state of IoT devices in an investigation [27]. A forensic state is the state of hardware and software of an IoT device at the time it was seized by law enforcement. For example, an IoT smart lock can have two states, i.e., locked and unlocked, and its state could be a vital information in an investigation. Furthermore, IoT devices that are subject to investigation may have been tampered with intentionally or as a result of malware. Ronen et al. demonstrated that IoT smart bulbs can be infected over-the-air (OTA) and be controlled remotely [28]. Such maliciously modified devices are shown to be effective in causing harmful results to humans, such as creating epileptic seizures to vulnerable individuals by adjusting the frequency of LED smart bulbs [29]. Therefore, verifying whether the device is running its default firmware can useful. In case the device has been reprogrammed and protected with encryption, identifying the encryption algorithm is also forensically useful.

4.1. Processing Electromagnetic Trace Data

EM-SCA attacks require a sufficient number of target device EM traces. Furthermore, in order to train ML models using EM traces, it is necessary to have EM trace samples annotated with the specific software activities of the target device they represent. Three hardware components are necessary for the acquisition of EM traces. These are namely; a DUT, a signal capturing device, and a host computer. The signal capturing device is connected to the host computer via USB interface while the DUT may or may not be connected in a similar manner. The host computer runs EMvidence framework and stores the captured EM traces for analysis.

An EM trace is a vector that represents the amplitude variation of a signal over time. Due to fast sampling rates used by signal acquisition hardware, an EM trace with a duration of milliseconds can contain millions of data points. When these data are used directly as input to train and test machine learning and deep learning models, the highly dimensional data can negatively affect time and amount of computing resources they demand. As a result, EM traces acquired through the aforementioned hardware setup are not suitable to be directly used to train DL models. Therefore, EM traces are pre-processed in order to transform them from a continuous time domain signal into a format that has a manageable feature vector for DL. LSTM architecture was used for the deep neural network, which is suitable for the identification of patterns that occur in time series data, such as EM traces [30].

When attempting to classify software activity EM traces, labelled EM traces are needed. For this purpose, EM trace samples are acquired by running each software activity on the target device and collecting EM traces of a predefined length. Originally, each EM trace is in time domain. Time-domain signals are prone to fluctuations caused by external noise. Therefore, each trace is transferred to frequency domain using FFT with an overlapping sliding window. For each EM trace, this results in a collection of FFT vectors representing consecutive time steps. The dimensions of the FFT vectors are still considerably higher to be directly used as the feature vector for LSTM classifier, e.g., $200,000$ dimensions. Therefore, the dimensions of these FFT vectors are further reduced by dividing the elements of each FFT vector into $1,000$ equally spaced buckets. From each bucket, the maximum element is selected as the representative of the bucket without losing the generalisation. This results in a $1,000$ element long feature vector for each time step of EM traces.

4.2. Forensic State Detection

In order to illustrate the usefulness of detecting forensic state of low-end IoT devices through EM-SCA, the following hypothetical scenario was considered. An IoT device has been deployed in a building as a part of an intruder detection system. The device consists of a sensor that detects movements within a specified space of the premises. The device consists

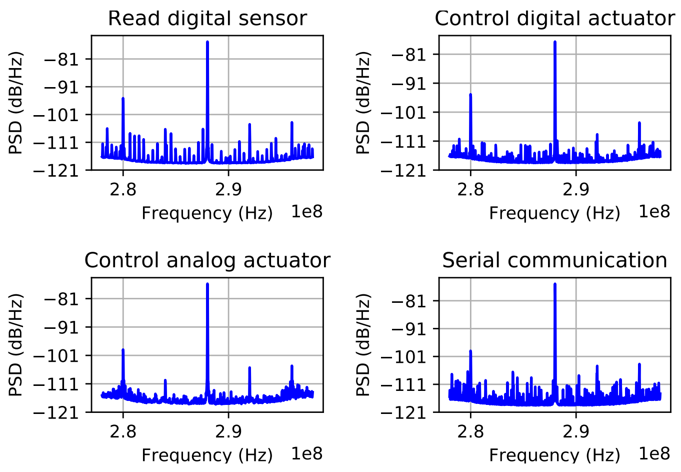

Figure 4: Power Spectral Density (PSD) of the IoT device's EM signals.

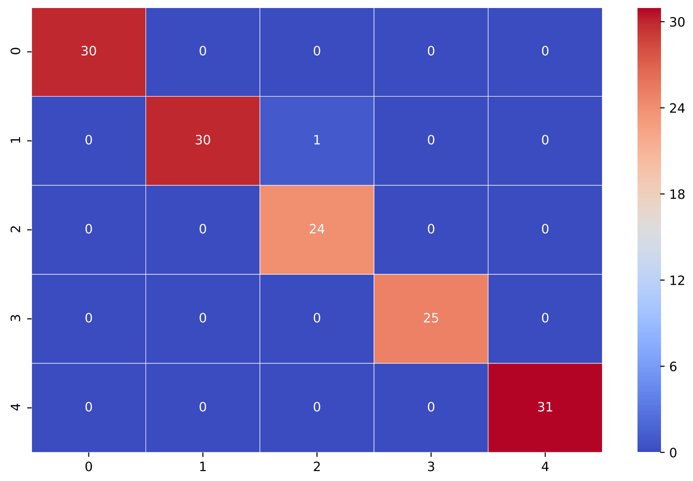

Figure 5: Confusion Matrix of the IoT Device State Classifier.

of two actuators, an alarm and a door lock, that it can control independently. Furthermore, the device is connected to a GSM module in order to send and receive SMS. The device firmware is programmed to continuously read the motion sensor to detect intrusions into the premises. Upon detection, it can perform one of three tasks – locking the door, firing an alarm, or sending a text message to the owner. At any time, the device can be disabled by pressing a physical button that puts the device into an idle state. The device does not have any other associated network servers that can log device states. Furthermore, the device does not switch internal states due to any other reason than the specified ones. The five states of the device that we are interested in are namely; (1) idle, (2) read digital sensor (reading motion sensor), (3) control digital actuator (firing the alarm), (4) control analogue actuator (turning door lock), and (5) serial communication (sending text message).

Suppose that this building is subject to a legal investigation for a crime - assumed to be conducted by an intruder. Upon the arrival of law enforcement, one of the investigative questions is whether the intruder detection system functioned as expected or did an insider disable it before the crime occurred. The answer to this question can be found if the current internal state of the device is known. Turning the device off and moving it to a digital forensics laboratory destroys the current internal state of the device. Performing a live EM-SCA on the intrusion detection device and identifying its current software state may be the only viable approach to resolve this problem.

The IoT device was emulated by using an *Arduino Leonardo* device. It was programmed to run a software code that puts the device on each of the 5 states chosen by the user. While the device was running on each state, a 30-second long EM trace was captured per state with a sampling rate of $20MHz$ using HackRF SDR. The SDR was tuned into the 18^{th} harmonic of the Arduino's clock frequency, i.e., $288Mz$. The H-loop antenna of the SDR was placed over the processor of the device during data acquisition. Figure 4 illustrates the power spectral density (PSD) of the EM signals from IoT device in its different states. A neural network classifier based on multi-layer perceptron (MLP) architecture was selected to distinguish each state of the IoT device. A non-overlapping sliding window with a width of 250ms was used to extract EM trace segments; subsequently converted to the frequency domain. These windows were grouped and averaged to produce a vector of $1,000$ features that were considered as training and testing samples for the ML classifier.

Figure 5 illustrates the confusion matrix of the classification results. The classifier was able to achieve an average F1-score of 99% in distinguishing the 5 IoT device states. This indicates that a pre-trained model to identify internal software states of the IoT device. For example, if it was identified that the device is in the *idle* state at the time investigators arrived at the scene, it's clear that someone deliberately turned the device into idle state in order to stop it from triggering the intruder alarm. In that case, fingerprints on the button of the IoT device could potentially help to identify the insider. Once the machine learning classifier was built, it is integrated into the EMvidence as a third-party machine learning model for identifying internal state of the particular type of IoT devices.

4.3. Elliptic Curve Detection

Some cryptographic algorithms, e.g., RSA, demand reasonably high computational power making them unsuitable for low-powered computing devices. As a result, elliptic curve cryptography (ECC) has increasingly been deployed on these devices. ECC is a public key cryptography that requires a smaller key length compared to RSA. Numerous different elliptic curves can be used in ECC. Table 1 illustrates five major elliptic curves designed to run on low power devices (available in the *micro-ecc* library). These elliptic curves use different private and public key lengths with bespoke configuration settings.

Due to the differences in settings of each elliptic curve, the running time of ECC operations for each curve can vary. Figure 6 illustrates the average time different elliptic curves take to digitally sign a message and to verify the signature of a message (ECDSA). The measurements were calculated by running each ECDSA algorithm on an Arduino device with equally sized messages. For each curve, both signing and signature verifying operations take almost the same amount of average time.

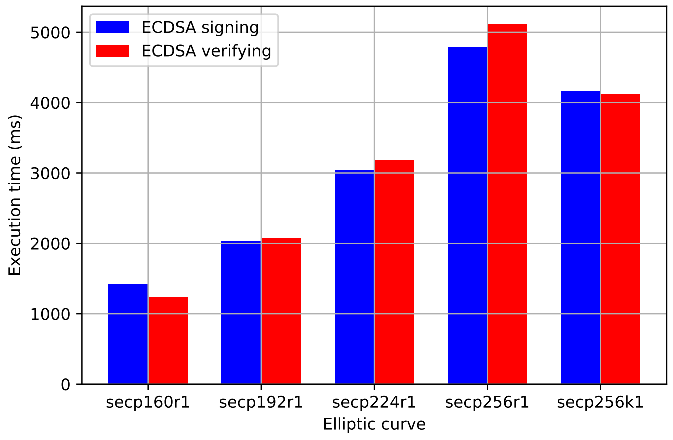

Figure 6: The time it takes to digitally sign and verify a message using different ECC curves on an Arduino device.

However, the average time taken by individual curves are distinguishably different from each other.

In order to investigate the possibility of detecting the presence of ECC cryptography through EM emissions, an experiment was conducted with an LSTM binary classifier. The objective is to train a model to distinguish between ECC cryptography operations and other software activities. For this purpose, an Arduino device was programmed to perform ECDSA signing operations that was controlled by sending commands through USB from the host computer. Each time an ECDSA signing operation is performed, a 100ms trace was captured through a H-loop antenna placed over the Arduino's microcontroller chip connected to a *HackRF* software defined radio.

For each of the five elliptic curves, 50 EM traces were acquired (totalling 250 traces) for the ECC cryptography class. For non-ECC cryptography operations class, 20 Arduino programs used that have varying complexities. From each Arduino program, 12 traces were acquired for non-ECC cryptography class (totalling 240 traces). A sliding window of 10ms was used with a 2ms step size (80% overlap) to collect segments from each trace and subsequently a feature vector for each window segment was calculated using FFT broken down into $1,000$ equally spaced buckets. Each bucket's maximum amplitude frequency component was selected as the representative signal for the bucket. This results in training sequences each having time steps where each time step consists of $1,000$ features. All the training samples were normalised to values between 0 and 1. Figure 7 illustrates examples of signals acquired from a DUT while running two curves of ECC and running two non-ECC programs.

Curve	Private Key (bytes)	Public Key (bytes)
secp160r1	21	40
secp192r1	24	48
secp224r1	28	56
secp256r1	32	64
secp256k1	32	64

Table 1: Private and Public Key Sizes of ECC Curves.

The LSTM classifier was implemented using *Keras* library with python. A single LSTM layer consists of 100 nodes and a fully connected layer with 1 node for binary classification. This last node uses a *sigmoid* activation function, while the model uses *binary crossentropy* as the loss function. The sequence data set was broken into 75% and 25% sets for training and testing purposes respectively. When using 5 epochs and 64 batch size, the LSTM classifier achieved an impressive 100% accuracy. In order to assess the effect from the sliding window length and EM trace length, further LSTM models were trained and tested varying those parameters. Figure 8 illustrates the variation of classification accuracy against both sliding window length and EM trace length. As is evident, the longer the EM traces, the higher the classification accuracy achieved. This is due to the fact that longer EM traces results in longer sequences with more information for the LSTM model to learn. In contrast, longer sliding window lengths negatively affected the classification accuracy, reducing it to 95% in the worst case. Again, the reason is behind the length of the sequences. Longer windows result in shorter sequences for a fixed length of EM traces.

4.4. Automated Trace Segmentation

When an IoT device is employing data encryption, EM-SCA techniques can assist an investigator to retrieve data decryption key. However, the success of cryptographic key recovery techniques depends on the number of EM traces acquired by observing a target device during cryptographic operations. These EM traces need to be acquired in a manner such that the cryptographic operation being observed must occur at the same point in time across all the traces. It has been shown that the accuracy of key recovery is considerably affected by misalignment of the EM traces [10]. The most common approach to extract sufficiently aligned EM traces is by instrumenting the target device through software or hardware. The purpose of this instrumentation is to identify the beginning and ending time where the cryptographic operation occurs so that the EM observation

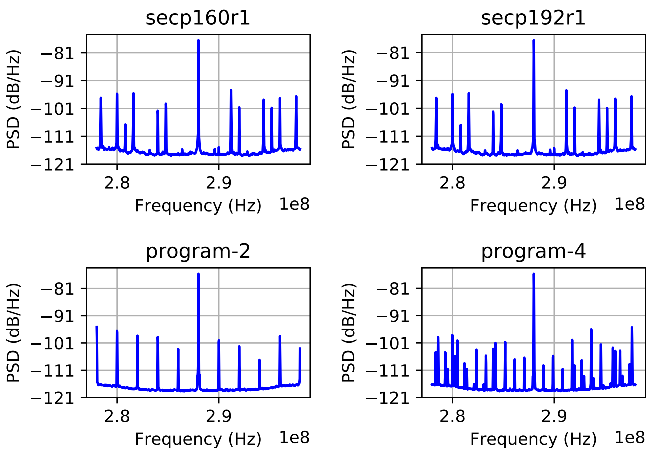

Figure 7: Example ECC and non-ECC Signals Acquired.

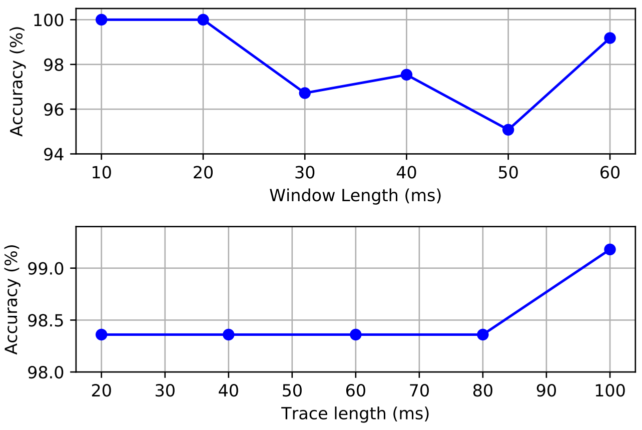

Figure 8: ECC Digitally Signing and Verification Time on an Arduino.

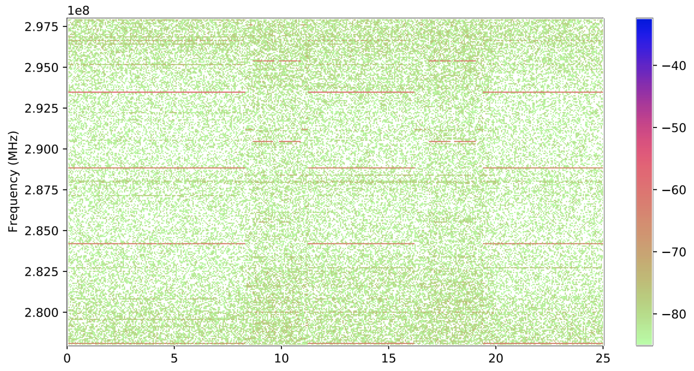

Figure 9: Signal Spectrogram of ECC Digital Signing Two Times.

hardware can be synchronised to extract EM traces within that precise time period.

Instrumentation of target devices can be used to demonstrate key recovery attacks, but it is not viable in most real-world application scenarios. Such scenarios occur when the attacker has no physical access to the device being attacked or when implementing such instrumentation is not possible. Under such circumstances, the EM traces have to be gathered without instrumentation and later be processed in order to realign them. Such post processing is a computationally costly task where the success of alignment is not sufficiently guaranteed.

When an Arduino device is performing a cryptographic operation, the experimental setup captures the data over a bandwidth of $20MHz$ centred at $288MHz$. Figure 9 illustrates the spectrogram of a signal that was observed while the device was performing two consecutive ECC digital signature operations. As can be seen, two distinctive regions across the time dimension have changed patterns corresponding to the two ECC operations. However, not all captured bandwidth leaks information about software activities. It is evident that some channels represent significant changes in signal amplitude compared to others in the two interested regions. Figure 10 illustrates two selected channels from the captured bandwidth where the first channel clearly represents two unique areas corresponding to two ECC operations and the second channel represents only a common region of change in pattern. This means that finding the right channel can help to identify the precise time instance where a particular software task has started and ended.

The automated separation of EM traces relevant to ECC operations was performed as follows. The target IoT device, i.e., Arduino, was programmed to run an ECC-related triggered by a command sent through USB. The exact task it excutes is using a private key generated using curve *secp160r1* to digitally sign a 32 byte long arbitrary message. The controlling host computer sends a command every 5 seconds in order to make the desired operation occur periodically at fixed intervals. Meanwhile, EM traces were acquired by observing for a predefined time period of 30 seconds. For such captured EM traces, a STFT was applied with a window size of 1ms. Due to the sampling rate of 20MHz, the STFT operation results in a spectrum dataset with $20,000$ channels. Finally, one manually selected channel from the EM data is applied to a change point detection algorithm, The Pruned Exact Linear Time (PELT) [31], in order to identify time instances where the significant pattern changes occur. Once the change points of an EM trace were identified, the trace was broken into it corresponding segments. Finally, a ML model trained to identify ECC curves was used to distinguish those EM trace segments with ECC operations. Keeping the EM segments of ECC operations for future use, such as cryptographic key recovery attacks, the rest are discarded.

4.5. Processing Overhead

High sampling rates are necessary to extract as much CPU data leakage information as possible. It is necessary to observe EM signals for a longer time duration in order to gather sufficient traces to attack cryptographic operations. However, on the down side, these two factors considerably increase the size of the resultant EM trace files saved on the host computer. A significant percentage of such saved data may not contain intended software activities, wasting storage space and processing time. As a result, it is useful to process EM signals in real-time and save traces only when activity of interest is detected.

The overhead of processing EM signals in real-time is evaluated as follows. While EM signal capture device is set to 20MHz sampling rate, a sliding window with a fixed width

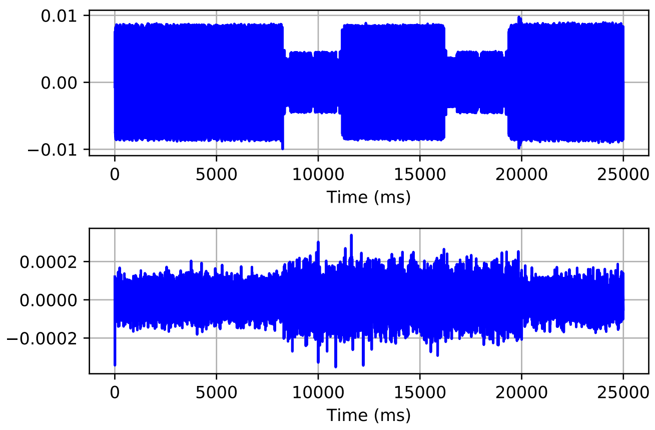

Figure 10: Two Arduino EM Channels when Digitally Signing with ECC.

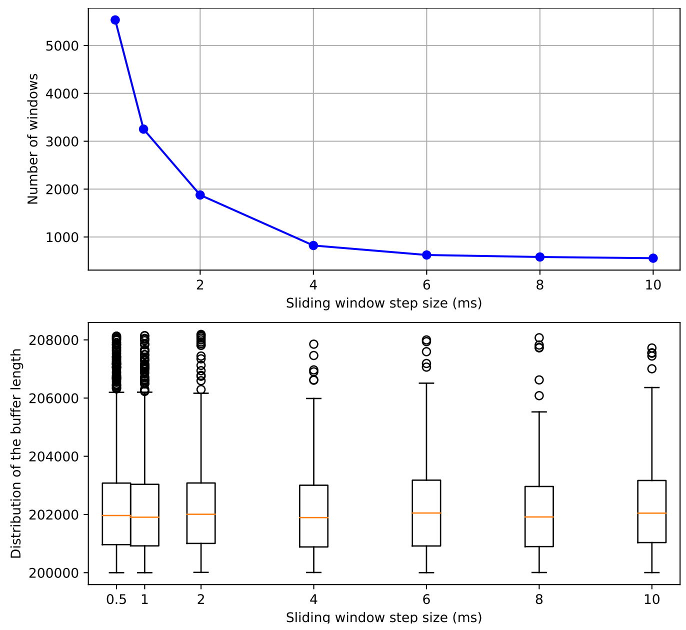

Figure 11: Effect of the Sliding Window Step Size to the Data Collection Buffer over 10ms.

of 10ms was used to slide through the real-time I-Q data feed. Each data window was then preprocessed in real-time to generate the features and fed to a neural network-based binary classifier to detect the presence of ECC cryptography operations. The step size of the sliding window, i.e., the amount of overlap between consecutive windows, was varied between 0.5ms and 10ms for different independent trials. In all experiments, the total signal capturing duration was fixed to 10 seconds.

In Figure 11, the graph on the top illustrates the number of windows produced against the sliding window step size. When reducing step size, the number of windows available to process for a given time period increases exponentially. The graph at the bottom of Figure 11 illustrates the statistics of the number of data samples waiting in the real-time buffer until the sliding window had processed them. As it can be seen, the median length of the I-Q signal buffer does not indicate any noticeable increase even for the smallest sliding window step size considered. This means, even though the number of sliding windows to process increases with the reduction of sliding window step size, it seems not to incur any considerable overhead to the real-time processing buffer. The production and consumption of the EM samples were in an equilibrium.

5. Discussion

In traditional digital forensics, the data acquired from devices under investigation are handled in a forensically sound way in order to make sure the court admissibility of evidence. For example, when a disk image is acquired from a computer, cryptographic hash values are calculated and stored along with the disk image. The hash values can later be used to verify the integrity of an image. Similarly, EM traces acquired from IoT devices has to be stored with a hash verification facility. The `EMvidence` framework supports hash calculation for EM traces acquired in real-time and stores them along with the traces. However, the patterns of the EM signals from an IoT device depends on its current internal states and external noise sources in the vicinity. Therefore, hash values are useful only to maintain the integrity of the originally acquired EM traces during subsequent analysis. Further steps to ensure forensic soundness of EM data acquisition and analysis require more studies that we hope to conduct in the future.

The experimental evaluations with the `EMvidence` framework indicate that EM-SCA based inspection of IoT devices can be useful for forensic investigations in a variety of scenarios. Detection of the forensic state, specific ECC curve, and automatically segmenting ECC-related EM traces are device-specific tasks. Performing each of those tasks with IoT devices by using tailor-made ML models requires profiling of each individual IoT device of interest. This cannot be done without the support of a large community due to the dynamics of IoT ecosystem. This need reinforces the necessity of open-source and extendable platforms like `EMvidence`.

The focus of EM-SCA techniques is to consider direct unintentional emissions from the processor as a side-channel. However, there can be useful alternative side-channels that are also highly relevant. For example, IoT devices are often equipped with System-on-a-Chip (SoC) processors that contains a CPU and a radio transceiver on the same chip. Due to the close proximity between the CPU (a digital component) and the radio transceiver (an analogue component), it has been shown that the CPU's digital operations can affect the analogue circuitry. This causes information leakage from the CPU to the on-chip radio transceiver [24]. Therefore it is possible to observe the transmissions of the radio from longer distances and extract CPU operation-related information. This is a less explored opportunity that can have digital forensic use cases, such as when a device needs to be inspected from a distance.

If the data stored on the IoT device are not encrypted and the device has a standard interface, the data can be extracted using existing forensic evidence acquisition methods [32]. However, many IoT devices lack these standard interfaces, often forcing investigators to take more risky approaches, such as chip-offs. Mistakes during such operations could inadvertently destroy useful evidence on a device. Therefore, in cases where a chip-off is being considered, it may be a good approach to first try an EM-SCA inspection on the device to gather as much information as possible before attempting a chip-off.

There are various hardware and software mitigation techniques available to counter EM information leakage [33]. It is possible that an IoT device may have applied EM side-channel mitigation techniques into its firmware in order to mislead EM-SCA attacks. It is possible that such mitigation techniques could cause ML classifiers to return inaccurate classification results. Further studies are required to assess the impact of such mitigation approaches to the EMvidence framework. However, due to the computational and energy cost of such mitigation techniques, they are rarely applied in low-powered devices making them less immediately threatening.

6. Conclusion and Future Work

With the ever-increasing applications of IoT systems in domestic and industrial environments, digital forensic investigations increasingly require the extraction of digital evidence from them. Most forensically useful information in IoT devices are currently extracted by intrusive inspections of hardware that makes them less forensically sound. This work presented the design of EMvidence, a framework for digital forensic investigators and researchers to leverage unintentional EM radiation from IoT devices as an information source. EMvidence is designed in a manner that it can be easily extended with new functionalities to keep up with the dynamism of IoT devices. Experimental demonstrations proved that ML classifiers can be used to gain useful insights in IoT investigative scenarios.

Several future work directions exist. The experimental demonstrations provided in this paper used an Arduino and a Raspberry Pi as representative IoT devices. It is necessary to evaluate EMvidence with the most commonly encountered IoT devices in real-world digital forensic scenarios. Currently, the framework is tested only with HackRF SDR as the EM signal acquisition device. It is necessary to test the inseparability between different SDR devices. For example, a ML model trained from EM traces from one specific SDR device must be accurate in classifying EM signals captured with other SDR devices.

References

1. Soltani S, Seno SAH. A survey on digital evidence collection and analysis. In: *7th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE; 2017:247–253.
2. Lillis D, Becker B, O'Sullivan T, Scanlon M. Current Challenges and Future Research Areas for Digital Forensic Investigation. In: *The 11th ADFSL Conference on Digital Forensics, Security and Law (CDFSL 2016)*. Daytona Beach, FL, USA: ADFSL; 2016:9–20.
3. Watson S, Dehghantanha A. Digital forensics: the missing piece of the internet of things promise. *Computer Fraud & Security* 2016;2016(6):5–8.
4. Casey E, Stellatos GJ. The impact of full disk encryption on digital forensics. *ACM SIGOPS Operating Systems Review* 2008;42(3):93–98.
5. Getz R, Moeckel B. Understanding and eliminating emi in microcontroller applications. *National Semiconductor* 1996;.
6. Kocher P, Jaffe J, Jun B. Differential power analysis. In: *Advances in Cryptology (CRYPTO '99)*. Springer; 1999:789–789.
7. Sayakkara A, Le-Khac NA, Scanlon M. A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics. *Digital Investigation* 2019;29:43 – 54. URL: `http://www.sciencedirect.com/science/article/pii/S1742287618303840`. doi:`https://doi.org/10.1016/j.diin.2019.03.002`.
8. Sayakkara A, Le-Khac NA, Scanlon M. EMvidence: A Framework for Digital Evidence Acquisition from IoT Devices through Electromagnetic Side-Channel Analysis. *Forensic Science International: Digital Investigation* 2020;doi:`https://doi.org/10.1016/j.fsidi.2020.300907`; proceedings of DFRWS EU 2020.
9. Van Eck W. Electromagnetic radiation from video display units: An eavesdropping risk? *Computers & Security* 1985;4(4):269–286.
10. Kocher P, Jaffe J, Jun B, Rohatgi P. Introduction to differential power analysis. *Journal of Cryptographic Engineering* 2011;1(1):5–27.
11. Quisquater JJ, Samyde D. Electromagnetic Analysis (EMA): Measures and counter-measures for smart cards. *Smart Card Programming and Security* 2001;:200–210.
12. Gandolfi K, Mourtel C, Olivier F. Electromagnetic analysis: Concrete results. In: *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. Springer; 2001:251–261.
13. Souvignet T, Frinken J. Differential power analysis as a digital forensic tool. *Forensic science international* 2013;230(1-3):127–136.
14. Sayakkara A, Le-Khac NA, Scanlon M. Leveraging Electromagnetic Side-Channel Analysis for the Investigation of IoT Devices. *Digital Investigation* 2019;29:S94 – S103. URL: `http://www.sciencedirect.com/science/article/pii/S1742287619301616`. doi:`https://doi.org/10.1016/j.diin.2019.04.012`.
15. Sayakkara A, Le-Khac NA, Scanlon M. Electromagnetic side-channel attacks: Potential for progressing hindered digital forensic analysis. In: *Companion Proceedings for the ISSTA/ECOOP 2018 Workshops*. ISSTA '18; New York, NY, USA: Association for Computing Machinery. ISBN 9781450359399; 2018:138–143. URL: `https://doi.org/10.1145/3236454.3236512`. doi:`10.1145/3236454.3236512`.
16. ChipWhisperer embedded hardware security toolchain. `https://newae.com/tools/chipwhisperer/`; 2019. Accessed: 2019-10-01.
17. O'Flynn C, Chen ZD. ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research. In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer; 2014:243–260.
18. Riscure . Inspector SCA: Side-Channel Analysis Tool for Embedded Systems. `https://www.riscure.com/security-tools/inspector-sca/`; 2019. Accessed: 2019-10-01.
19. Maxwell JC. A dynamical theory of the electromagnetic field. *Philosophical transactions of the Royal Society of London* 1865;155:459–512.
20. Jabbar M, Rahman MA. Radio frequency interference of electric motors and associated controls. *IEEE Transactions on Industry Applications* 1991;27(1):27–31.
21. Brier E, Clavier C, Olivier F. Correlation power analysis with a leakage model. In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer; 2004:16–29.
22. Peeters E, Standaert FX, Quisquater JJ. Power and electromagnetic analysis: Improved model, consequences and comparisons. *Integration, the VLSI journal* 2007;40(1):52–60.
23. Robyns P, Quax P, Lamotte W. Improving cema using correlation optimization. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019;:1–24.
24. Camurati G, Poeplau S, Muench M, Hayes T, Francillon A. Screaming channels: When electromagnetic side channels meet radio transceivers. In: *Proceedings of the 25th ACM conference on Computer and communications security (CCS)*. CCS '18; ACM; 2018:.
25. Ossmann M. Software defined radio with hackrf. *Great Scott Gadgets, https://greatscottgadgets com/sdr* 2016;.
26. Blossom E. GNU radio: tools for exploring the radio frequency spectrum. *Linux Journal* 2004;2004(122):4.
27. Meffert C, Clark D, Baggili I, Breitinger F. Forensic State Acquisition from Internet of Things (FSAIoT): A general framework and practical approach for IoT forensics through IoT device state acquisition. In: *Proceedings of the 12th International Conference on Availability, Reliability and Security*. ACM; 2017: 56.
28. Ronen E, Shamir A, Weingarten AO, O'Flynn C. IoT goes nuclear: Creating a ZigBee chain reaction. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE; 2017:195–212.
29. Ronen E, Shamir A. Extended functionality attacks on IoT devices: The case of smart lights. In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE; 2016:3–12.
30. Sayakkara A, Miralles L, Le-Khac NA, Scanlon M. Cutting through the Emissions: Feature Selection from Electromagnetic Side-Channel Data for Activity Detection. *Forensic Science International: Digital Investigation* 2020;doi:`https://doi.org/10.1016/j.fsidi.2020.300927`; proceedings of DFRWS EU 2020.
31. Killick R, Fearnhead P, Eckley IA. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association* 2012;107(500):1590–1598.
32. MacDermott Á, Lea S, Iqbal F, Idowu I, Shah B. Forensic analysis of wearable devices: Fitbit, garmin and hetp watches. In: *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE; 2019:1–6.
33. Zankl A, Seuschek H, Irazoqui G, Gulmezoglu B. Side-channel attacks in the internet of things: Threats and challenges. In: *Solutions for Cyber-Physical Systems Ubiquity*. IGI Global; 2018:325–357.