A methodology to identify identical single-board computers based on hardware behavior fingerprinting

Sánchez Sánchez, Pedro Miguel ; Jorquera Valero, José María ; Huertas Celdran, Alberto ; Bovet, Gérôme ; Gil Pérez, Manuel ; Pérez, Gregorio Martínez

DOI: https://doi.org/10.1016/j.jnca.2022.103579

Journal of Network and Computer Applications 212 (2023) 103579

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A methodology to identify identical single-board computers based on hardware behavior fingerprinting

Pedro Miguel Sánchez Sánchez [a,*], José María Jorquera Valero [a], Alberto Huertas Celdrán [b], Gérôme Bovet [c], Manuel Gil Pérez [a], Gregorio Martínez Pérez [a]

[a] Department of Information and Communications Engineering, University of Murcia, Murcia 30100, Spain
[b] Communication Systems Group (CSG), Department of Informatics (IfI), University of Zurich UZH, 8050 Zürich, Switzerland
[c] Cyber-Defence Campus, armasuisse Science & Technology, 3602 Thun, Switzerland

ARTICLE INFO

Keywords:
Device behavior fingerprinting
Device identification
Cyberattack detection
Behavioral data
Hardware fingerprinting

ABSTRACT

The connectivity and resource-constrained nature of single-board devices open the door to cybersecurity concerns affecting Internet of Things (IoT) scenarios. One of the most important issues is the presence of unauthorized IoT devices that want to impersonate legitimate ones by using identical hardware and software specifications. This situation can provoke sensitive information leakages, data poisoning, or privilege escalation in IoT scenarios. Combining behavioral fingerprinting and Machine/Deep Learning (ML/DL) techniques is a promising approach to identify these malicious spoofing devices by detecting minor performance differences generated by imperfections in manufacturing. However, existing solutions are not suitable for single-board devices since they do not consider their hardware and software limitations, underestimate critical aspects such as fingerprint stability or context changes, and do not explore the potential of ML/DL techniques. To improve it, this work first identifies the essential properties for single-board device identification: uniqueness, stability, diversity, scalability, efficiency, robustness, and security. Then, a novel methodology relies on behavioral fingerprinting to identify identical single-board devices and meet the previous properties. The methodology leverages the different built-in components of the system and ML/DL techniques, comparing the device internal behavior with each other to detect variations that occurred in manufacturing processes. The methodology validation has been performed in a real environment composed of 15 identical Raspberry Pi 4 Model B and 10 Raspberry Pi 3 Model B+ devices, obtaining a 91.9% average TPR with an XGBoost model and achieving the identification for all devices by setting a 50% threshold in the evaluation process. Finally, a discussion compares the proposed solution with related work, highlighting the fingerprint properties not met, and provides important lessons learned and limitations.

1. Introduction

The diversity of IoT devices in modern scenarios is huge, but single-board devices, such as Raspberry Pi, have gained enormous prominence due to their flexibility, reduced price, broad support, and peripherals availability (Fayos-Jordan et al., 2020). Unfortunately, the connectivity and resource-constrained nature of single-board devices, and IoT in general, open the door to numerous cybersecurity concerns affecting heterogeneous platforms (Perales Gómez et al., 2019). One of the most important cybersecurity concerns affecting IoT is the presence of unauthorized devices with the same hardware and software configuration as authorized nodes. Some real attacks based on unauthorized devices have caused big impacts in areas such as Industry 4.0 (Jagdale, 2022) or mobile phones (Montalbano, 2020). These malicious devices can be articulated by several well-known cybersecurity threats (Liu et al., 2020), such as *device spoofing* (Nosouhi et al., 2022), occurring when an attacker replaces a legitimate sensor or actuator with a malicious device using the same identity; *unauthorized device deployment*, related to the installation of a new device in the platform which is using an unregistered identity; and *Sybil attack*, referring to a malicious device (or many) using numerous identities to simulate being several devices. After that, other cybersecurity threats, such as *sensitive information leakage*, *data poisoning*, or *privilege escalation and lateral movements*, might arise as a consequence of spoofed devices. Besides, modern attacks exploit evasion techniques in order to be undetected by software-based security methods.

* Corresponding author.
E-mail address: pedromiguel.sanchez@um.es (P.M. Sánchez Sánchez).

https://doi.org/10.1016/j.jnca.2022.103579
Received 22 June 2022; Received in revised form 13 October 2022; Accepted 28 December 2022
Available online 3 January 2023

Traditional identification solutions rely on names, identifiers, certificates, or tags in order to perform the identification tasks. However, these solutions can be cloned or modified if the software of the device is completely replicated (Yousefnezhad et al., 2020). Hardware behavior fingerprinting is a potential solution for the identification of single-board devices with identical hardware and software, but still an emergent and open challenge. In such a context, there is no work focused on identical single-board device identification (Sabhanayagam, 2022). However, for other devices without component and resource limitations, the literature has proposed the usage of hardware *behavioral fingerprinting* as a promising solution to detect minor performance differences generated by imperfections occurred during the devices manufacturing process (Al-Omary et al., 2018).

In particular, existing work focuses on crystal oscillator impurities and cut variations that generate imperfect frequency outputs in components such as CPU or GPU to detect performance differences in identical devices (Polcák and Franková, 2015). Current solutions consider dimensions such as clock-skew analysis, intrinsic Physical Unclonable Functions (PUFs), or execution time and performance analysis (Sanchez-Rola et al., 2018). However, despite their benefits, the following challenges are still open: (i) many solutions require additional components or modifications in the devices, which is not possible in some IoT scenarios (Babaei and Schiele, 2019); (ii) there is no solution for identifying identical single-board devices based on their hardware (Babun et al., 2021); (iii) existing solutions are designed for traditional computers, being not suitable for IoT environments with single-board devices with software and hardware restrictions (Sanchez-Rola et al., 2018); (iv) most of the existing identification solutions have been tested missing essential properties and requirements affecting the identification performance (Rührmair et al., 2012); and (v) despite Machine and Deep Learning (ML/DL) techniques have gained enormous importance for the last years, they have not yet been widely applied in the individual device identification field (Sánchez Sánchez et al., 2021).

In order to improve the previous challenges, the main contributions of the present work are:

- The definition of a set of properties that should be fulfilled by any fingerprinting solution in charge of identifying identical single-board devices. These properties are uniqueness, stability, diversity, scalability, efficiency, robustness, and security.
- A novel methodology that leverages hardware behavioral fingerprinting to identify identical single-board devices, solving the problems and drawbacks of previous solutions. Some of these problems are the need for additional hardware, chip modifications, or physical access to the device to perform the identification. The proposed methodology creates unique device behavioral fingerprints measuring the impact that insignificant hardware differences, happened during the manufacturing process of identical devices, have on the device performance when a given task is executed.
- The validation of the proposed methodology, as a Proof-of-Concept (PoC) available on Sánchez Sánchez (2021), in a scenario composed of 25 identical Raspberry Pi 3 and 4 devices used in IoT scenarios. After testing different ML/DL algorithms, 91.9% average TPR was achieved by XGBoost, and a perfect identification was carried out by setting a 50% threshold in the assigned classes.
- A detailed analysis and comparative of existing device fingerprinting solutions for individual device identification, focusing on their suitability for IoT environments with single-board devices. It highlights which fingerprint properties are not met in each solution, rising issues such as reproducibility and solution stability.

The remainder of this paper is organized as follows. Section 2 reviews the main solutions for identical device identification and discusses why these approaches are not appropriate for IoT environments based on single-board devices. Section 3 describes the problem to be solved by the present methodology. It details a short threat model for the single-board device identification scenario. Then, it details the set of properties required in a fingerprinting solution to make it appropriate for individual device identification, together with the limitations found in the literature works. The design of the proposed device identification methodology is explained in Section 4, verifying how each fingerprint property is accomplished. Section 5 acts as validation of the present methodology, implementing it as a PoC that verifies its applicability in a realistic use case. Section 6 compares the literature works with the proposed methodology and depicts several lessons learned and limitations. Finally, Section 7 shows the conclusions extracted from the present work and future steps in the research.

2. Related work

This section gives the main insights of the related work dealing with unique device identification, paying special attention to device identification without additional external hardware requirements.

As a main remark, it is worth mentioning that, to the best of our knowledge, there is no methodology for individual fingerprinting of IoT devices based on hardware performance behavior. In fact, the same happens in the field of traditional devices such as personal computers. In this regard, the closest work is the one proposed by Babun et al. (2021), in which a fingerprinting framework for identifying classes of Cyber-Physical Systems (CPSs) was presented. This solution employed hardware and OS/kernel characteristics following a challenge/response mechanism for performance and system calls fingerprinting. During the validation, a set of single-board computers were employed. Nevertheless, the objective of Babun et al. (2021) is device type (class) fingerprinting and identification, not individual device fingerprinting when hardware and software are identical. Therefore, following this approach, identical devices would generate the same fingerprints, as the data sources leveraged are based on OS/kernel or component-related data and do not seek to identify fabrication variations or imperfections.

Although not defined in the form of a methodology, it is essential to analyze existing work focused on hardware-based individual device fingerprinting and device type identification, discussing the limitations of each work when applied to single-board devices. In this context, traditionally, Physical Unclonable Functions (PUFs) have been one of the main methods for unique device identification. PUFs are hardware elements that generate a unique physically-defined fingerprint for a given output based on the manufacturing characteristics of the physical chips. PUFs have been employed in IoT from several perspectives (Babaei and Schiele, 2019), differentiating between strong and weak PUFs depending on the number of Challenge-Response Pairs (CRPs). Strong PUFs are the ones most used for authentication protocols in IoT (Babaei and Schiele, 2019). However, the majority of strong PUFs require additional dedicated hardware elements that have to be attached to the device. This fact makes this solution not scalable in large environments, as costs per device are increased and commercial-off-the-shelf (COTS) devices have to be modified, or where direct access to the device is not possible. In contrast, most intrinsic PUFs in the literature require hardware modifications (Kong and Koushanfar, 2013) or components such as SRAM not present in IoT devices due to cost restrictions (Gao et al., 2019). Besides, some works using DRAM chips, present in IoT devices, require power-up chip status analysis (Yue et al., 2020; Tehranipoor et al., 2016), which is not straightforward to be done from the device itself.

From crystal oscillator analysis, Salo (2007) exploited differences in Real-Time Clocks (RTCs) and sound card Digital Signal Processors (DSPs) based on the drift between these chips and the CPU cycle counter (TSC in Intel processors). RTC-based and DSP-based differentiation achieved 98.7% and 93.3% of uniqueness when 703 computer pairs were evaluated. However, this method involves the use of components that, although common in computers, are not often available in single-board devices. Also leveraging oscillators, Sanchez-Rola et al.

Table 1
Individual device identification solutions based on device behavior fingerprinting.

Work	Year	Device type	Algorithms	Behavior source	Features	Results
Salo (2007)	2007	General computers	Statistical	Processors and oscillators	RTC and DSP drift compared to the TSC	98.5% and 93.3% of differentiation by RTC and DSP in 38 PCs, respectively.
Jana and Kasera (2009)	2009	Wireless access points	Expectation Maximization	Clock skew	Wi-Fi beacons timestamps	Clock skew is a robust method and can detect different WLAN APs.
Sharma et al. (2012)	2012	General computers	Statistical	Clock skew	TCP and ICMP timestamp	Both identical and different devices correctly identified.
Wang et al. (2012)	2012	General computers	Correlation coefficient	Flash memory	Bit partial programming	Estimated false positive chance of 4.52×10^{-815}, and a false negative chance of 2.65×10^{-539}.
Radhakrishnan et al. (2014)	2014	Wireless devices	ANN	Clock skew + Network	Communication skew and patterns	From 99 to 95% accuracy and 74% recall on individual classification.
Nakibly et al. (2015)	2015	General computers	Entropy	GPU	Frames per second	Graphic rendering show differentiation capabilities on 9 identical PCs, but no advanced tests were performed.
Sanchez-Rola et al. (2018)	2018	General computers	Statistical (Mode)	System processors	Matrix of code execution times	100% host-based and +80% web-based device identification in two sets of 89 and 176 PCs.
Jafari et al. (2018)	2018	Wireless devices	MLP, CNN, LSTM	Electromagnetic signals	Radio frequency IQ samples	96.3% accuracy for MLP, 94.7% for CNN and 75% for LSTM when identifying 6 identical ZigBee devices.
Riyaz et al. (2018)	2018	Wireless devices	CNN	Electromagnetic signals	Raw frequency IQ samples	98% accuracy is achieved when identifying 5 identical devices.
Dong et al. (2019)	2019	General computers	Dynamic Time Warping	Resource usage	CPU usage-based graph	93.43% of uniqueness in the generated fingerprints of 10 identical devices.
Babun et al. (2021)	2021	CPSs	Correlation-based (Own)	Hardware and OS/kernel	Syscalls, Memory, CPU, Time	Device type (model/OS version) identification, not individual identification.
This Work	2022	Single-board devices	XGBoost	Hardware cycle counter skew	Window-based GPU/CPU features	91.9% average TPR when identifying 15 RPi4 and 10 RPi3 devices.

(2018) proposed a fingerprinting method based on execution time. The authors cyclically executed a simple function to generate a time matrix, and then they calculated the statistical mode of each matrix row to generate the fingerprint. Then, matching values in the fingerprints were compared according to a similarity threshold. The authors were able to identify two computer sets of 176 and 89 devices, and achieved 85% on a web-based implementation. Compared to the work at hand, single-board devices do not include an RTC with which to compare CPU time (two different clocks are required to analyze their deviation) and usually only contain one physical oscillator. Furthermore, after practically experimenting with this approach on single-board devices (see Section 6), it has been found that the resolution when measuring time on single-board devices does not allow this solution to be applied.

Additionally, some works have addressed identical device identification based on clock-skew calculated from network packets (Kohno et al., 2005; Sharma et al., 2012; Radhakrishnan et al., 2014) or wireless beacons (Jana and Kasera, 2009). However, they have shown scalability issues when the number of devices increases and require a common observer in the fingerprint and identification process; if the observer changes, the identification is no longer possible (Radhakrishnan et al., 2014; Lanze et al., 2012; Polcák and Franková, 2015). Besides, raw radio frequency measurements (Jafari et al., 2018; Riyaz et al., 2018) and Bluetooth transmissions (Huang et al., 2014) have also been used to identify devices uniquely, but these methods, as other wireless-based methods, require a near physical location to the fingerprinted device.

Based on hardware performance behavior, Wang et al. (2012) analyzed the differences that occur when writing a page in a Flash chip based on manufacturing variations. To evaluate different fingerprints of the same page, the authors used Pearson correlation coefficient. Based on their experiments on 24 chips, the authors showed an estimated false positive chance of 4.52×10^{-815}, and a false negative chance of 2.65×10^{-539}. However, not every device includes a Flash chip to apply the technique and its usage requires knowledge of low-level hardware. Recently, Dong et al. (2019) developed a fingerprinting method based on the CPU usage graph generated while the device executes a cyclical task. The authors achieved a 93.43% uniqueness in generated fingerprints when comparing them using the Dynamic Time Warping algorithm. However, the authors did not take into consideration critical aspects such as variable frequency or process scheduling between device cores affecting the identification stability. Regarding GPU, Nakibly et al. (2015) exploited GPU frequency and skew by using CPU clock as reference. Statistical fingerprints generated while rendering complex graphics show differentiation capabilities on 9 identical desktop computers, but no advanced tests were performed regarding fingerprint reliability and stability. In fact, the authors conclude that other factors to the GPU clock skew should be considered in a successful fingerprinting method.

Table 1 compares the main characteristics of the previous works. After reviewing these related works, the following points are extracted as conclusions. It is critical to develop modern fingerprinting mechanisms taking into account IoT device capabilities and constraints, as no

previous work considered this application scenario. Besides, to ensure that the fingerprinting mechanisms are fully operative, they should be defined through a methodology able to verify that the solution is reliable and applicable in real word scenarios.

3. Problem statement

This section presents the particularities of single-board devices to later illustrate the threat model of single-board device identification. After that, it describes the properties that identification solutions based on behavioral fingerprinting should meet in the presented scenario. Finally, it highlights the limitations of existing work and motivates the necessity of novel solutions.

3.1. Single-board device description

Although single-board devices offer great flexibility in terms of applications and operating systems, there are essential characteristics to consider before dealing with their identification. The main one is that all processing, memory, input/output, and other components are integrated into a single circuit board. In contrast, standard computers have several circuit boards for different components. This fact brings the following special aspects to consider:

- **Reduced number of crystal oscillators.** Due to the objective of reducing costs, single-board computers usually dispense with components that are not critical. Thus, most devices eliminate the RTC and other physical oscillators, simulating their presence through software or using another oscillator as source frequency. The most common is to have only one or two oscillators, one for the base frequency of the processing components and another for USB and network interfaces.
- **Many processing components integrated into a System on a Chip (SoC).** SoCs integrate microcontrollers with more advanced processing units such as CPUs, GPUs, or memory circuits in a single chip. As each of these components uses a different frequency to operate, it is common to use Phase-Locked Loops (PLLs) in the SoC (Pawar and Mane, 2017), circuits that multiply a base frequency depending on the voltage they receive as input.
- **Constrained processing power**. Although single-board computers offer increasingly higher computing capabilities, they also aim to maintain low resource consumption and low price. For these reasons, the performance of single-board computers is not comparable to that of today's computers or servers. This is important and should be taken into account when generating the fingerprint.

3.2. Threat model

The main threat against the single-board device identification scenario is an adversarial actor trying to introduce a illegitimate device in a critical environment, such as an industry, by impersonating or spoofing a legitimate one. This attack could be tackled from several perspectives:

- *TH1. Device spoofing* (Marabissi et al., 2022). The main security threat to solve is an adversarial entity replacing a legitimate device with a software identical malicious device. Here, the adversary uses the same legitimate software identifiers, but including malicious processes and functionality.
- *TH2. Sybil* (Rajan et al., 2017). A single device (or many) may try to generate multiple identities to send fake data from many simulated devices. The threat of a system to Sybil attacks depends on (i) how easy the generation of identities is; (ii) whether the system treats all entities identically, and (iii) the degree to which the system accepts entries from entities that do not have a trust chain that links them to a trusted entity.
- *TH3. Advanced persistent threat* (Chen et al., 2022). This threat arises as a consequence of the previous one. A malicious device deployed in the environment might be able to collect data from the scenario itself and from other devices, or perform further attacks such as vulnerability scan and or Denial of Service (DoS) attacks. Besides, modern attacks usually include evasion techniques that hide their activities to software-based behavior monitoring security solutions (Li and Li, 2020).

In order to solve the threats identified in this work, it is assumed that even if the device is malicious, the control over it is maintained by its legitimate administrator and the identification tasks can be executed. This condition guarantees that device management is maintained during a possible attack. Therefore, if this control is lost, it would be directly assumed that the device is infected or there is some error.

3.3. Device identification properties

In order to solve the previous threats, it is needed a proper identification mechanism able to meet properties that guarantee a consistent and reliable verification process, without forgetting the threat model depicted in Section 3.2. Similar properties have been defined before (Rührmair et al., 2012), but some of them are not suitable for IoT and single-board devices. These characteristics encompass from the fingerprint generation method to the morphology of the data generated and its manipulation. Thus, they are essential metrics to evaluate the performance of a device fingerprinting solution. In case one of them is no longer met, the solution will be severely affected in real-world deployments, limiting its usability when it comes to uniquely identify each device in the scenario.

Uniqueness (Sembiring et al., 2021). An efficient fingerprinting method should be able to uniquely identify its associated device. In other words, a fingerprint should not be generated by two different devices.

Stability (Hamza et al., 2018). The fingerprint generated by a device should be consistent in time. It means that a new fingerprint of a given device should be similar enough to the previous ones of the same device.

Diversity (Ahmed et al., 2022). The data sources and data format used to generate the fingerprint should be varied enough, so different devices generate different fingerprints. This characteristic is intrinsically related to stability, as increasing too much fingerprint diversity can affect its stability, and vice versa.

Scalability (Arellanes and Lau, 2020). The fingerprint should continue being unique as the number of devices to be identified increases. This can be achieved by adding additional features to the fingerprint or by looking for features that ensure uniqueness. Thus, this characteristic is very closely related to the uniqueness property discussed before.

Efficiency (Peng et al., 2018). To have a fingerprint useful for a live identification process, the generation and evaluation should not consume excessive resources, either in processing power or time.

Robustness (Zhou et al., 2019). The generation of the fingerprint must be immune to changes in the context that may affect the data used in the fingerprinting process. These changes in the context may include elements such as temperature, time synchronization, or resource exhaustion, among others.

Security (Lu and Da Xu, 2018). The fingerprint should be secure to tackle device unauthorized access or adversarial attacks. This property implies a complete fingerprint life cycle, from its generation to storage and comparison in future identification processes.

Table 2
Limitations found in each literature work.

Work	Type	No hardware modification	IoT suitable	Tested stability	Remote/ Self-contained
Salo (2007)	Oscillator-based	✓	✗	✓	✓
Jana and Kasera (2009)	Clock skew	✓	✓	✗	✗
Sharma et al. (2012)	Clock skew	✓	✓	✗	✗
Wang et al. (2012)	Flash chip-based	✓	✗	✗	✓
Radhakrishnan et al. (2014)	Clock skew	✓	✓	✗	✗
Nakibly et al. (2015)	Oscillator-based	✓	✓	✗	✓
Sanchez-Rola et al. (2018)	Oscillator-based	✓	✗	✓	✓
Jafari et al. (2018)	Radio-based	✓	✓	✗	✗
Riyaz et al. (2018)	Radio-based	✓	✓	✗	✗
Dong et al. (2019)	CPU usage-based	✓	✓	✗	✓
Babaei and Schiele (2019)	PUF	✗	✓	✓	✓
Kong and Koushanfar (2013)	CPU PUF	✗	✓	✓	✓
Gao et al. (2019)	SRAM Intrinsic PUF	✓	✗	✓	✓
Tehranipoor et al. (2016)	DRAM Intrinsic PUF	✓	✓	✓	✗
Yue et al. (2020)	DRAM Intrinsic PUF	✓	✓	✓	✗
This work	Oscillator-based	✓	✓	✓	✓

3.4. Limitations of existing work

Although a good number of solutions are present in the literature, as reviewed in Section 2, they are not suitable for single-board devices and the device identification task that this work pretends to fulfill. The conditions identified, which have not been covered altogether in a single work, can be summarized as:

- No additional hardware or device component modification is required. In this sense, no previous solution intends to design a solution for COTS IoT devices, where devices already available in the market do not need any modification to be physically fingerprinted.
- Suitable for IoT devices, specially single-board devices. Many solutions for performance-based identification leverage components such as RTCs, which are not usual in IoT devices due to their reduced price. Besides, some authors proposed intrinsic PUFs that do not require additional hardware components. However, most IoT devices include DRAM chips due to their cheaper cost. There is a reduced number of works dealing with these components, and they require power-up chip analysis (Yue et al., 2020; Tehranipoor et al., 2016) which is complicated to be done from the device itself where the DRAM is deployed without affecting the PUF results.
- Tested stability and robustness. Some solutions in the literature show favorable identification results. However, they do not analyze critical factors affecting performance-based identification, such as the impact of device rebooting, temperature, etc.
- Remote and self-contained identification. The identifying entity does not need to be physically close to the identified device or in the same local network, as in the case of clock skew-based identification. Besides, no external component analysis is needed, so the device itself can execute the identification process.

Table 2 evaluates the conditions correctly accomplished in each one of the works reviewed in the literature regarding individual device identification (Section 2. As it can be seen, no work meets the three characteristics wanted in the present work.

4. Methodology definition

This section presents a novel methodology to identify identical single-board devices using behavioral fingerprints. It focuses on measuring the impact that insignificant hardware differences, which happened during the device manufacturing process, have on the device performance to create unique and stable behavior fingerprints. These differences are recognized by analyzing the performance of several heterogeneous components, according to parameters such as execution time or number of cycles. Thus, it is worth noting that this methodology could be applied to other types of devices containing at least two components to compare their behavior. Besides, ML/DL techniques are applied as processing tool following the best practices in the area of ML application for cybersecurity (Arp et al., 2022), but other statistical methods could also be suitable.

As shown in Fig. 1, the proposed methodology follows a client/server model and is composed of two fundamental phases: a first one of generation and a second of evaluation. During the fingerprint generation phase, the objective is the creation of a fingerprint per device by training ML/DL models for later device identification. During the fingerprint evaluation phase, new fingerprints per device are generated to be evaluated with the ML/DL models trained in the previous phase, giving a final identification output for the device. The methodology consists of the next seven fundamental steps, which can be repeated in both phases depending on the tasks to be carried out:

- (A) *Hardware Component Selection*. Select the device components whose behavior is going to be analyzed.
- (B) *Component Isolation and Stability Assurance*. Establish stable conditions for the components, reducing external inferences to a minimum.
- (C) *Data Gathering*. Measure the behavior of device hardware components.
- (D) *Data Preprocessing and Feature Extraction*. Remove erroneous measurements, normalizes them, and extracts new significant values.
- (E) *Evaluation Approach Selection*. Decide between classification or anomaly detection depending on the environment properties.
- (F) *Model Generation and Evaluation Design*. Train ML/DL algorithms, select performance metrics, and establish model thresholds.
- (G) *Device Evaluation and Identification Decision*. Repeat steps B, C, and D to perform device identification.

Fig. 2 shows the relationship between the different steps detailed above and the properties desired in an individual device identification solution, as introduced in Section 3.3.

4.1. Hardware component selection

The first step is to analyze the hardware of the device where the fingerprint needs to be generated. The goal is to identify components with potential manufacturing variations whose performance can be accurately measured and compared.

In this sense, since the fingerprint will be based on device self-contained hardware, it is necessary to identify at least two components to be used, as their behavioral performance will be compared to each other since one component cannot notice its own performance imperfections without a reference point, although to improve the *scalability*

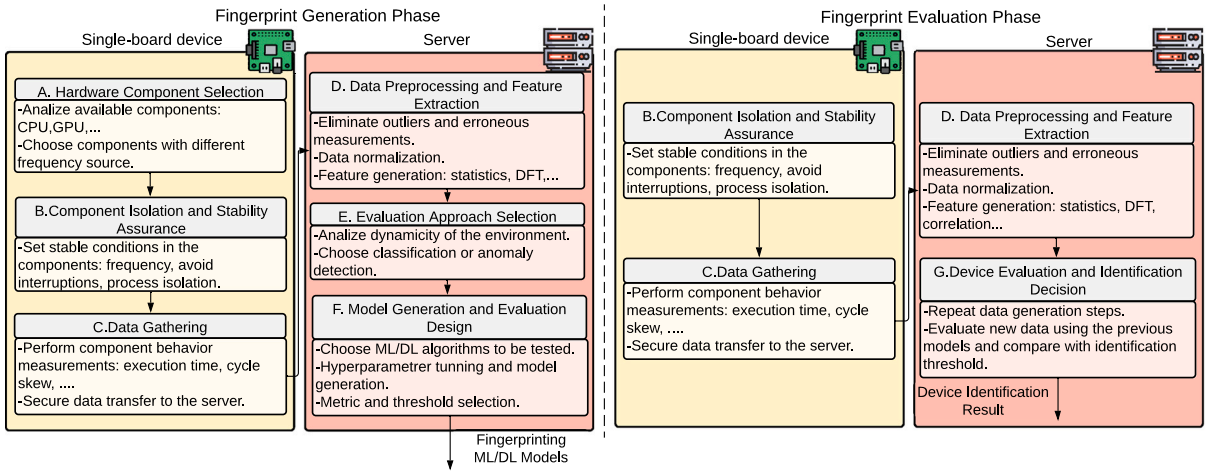


Fig. 1. Graphical representation of the proposed methodology for device fingerprinting and identification.

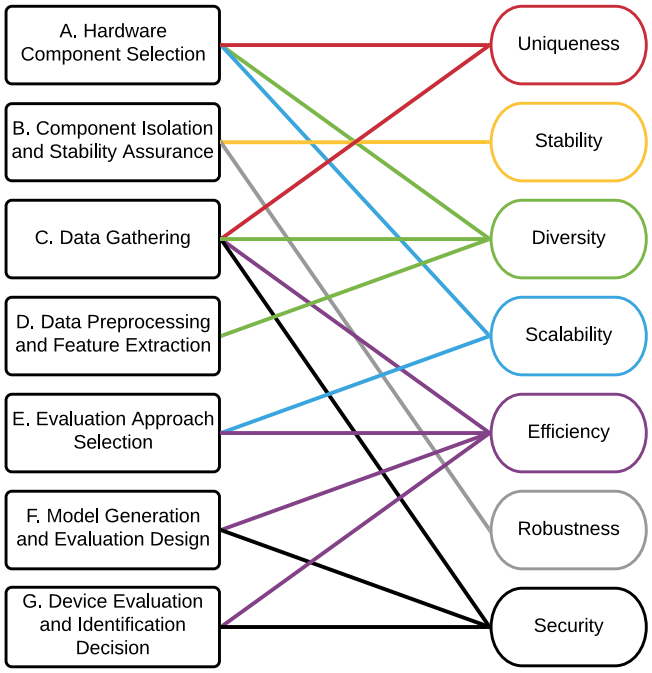


Fig. 2. Association between methodology steps and fingerprint properties.

and *diversity* of the fingerprint more could be added if available. The preference here is to select components whose frequency is based on different physical oscillators, as their differences will be larger, although components with different frequencies sharing one oscillator as the base frequency can also be compared. Examples of components to consider in single-board devices: CPU, GPU, memory, network controllers, USB controllers, or time control oscillators.

4.2. Component isolation and stability assurance

Once the hardware components to be monitored are chosen, the next step is to establish a configuration that ensures the *stability* of the behavioral measurements. This step seeks to ensure a stable and identical condition in the device configuration during the generation of the fingerprint, both for training and testing phases. At this point, it is critical to guarantee that there are no external elements, such as other processes, introducing noise or variability.

With that goal in mind, one of the key factors to take into account is the frequency at which the component is operating, since in single-board computers it is common for the operating system to establish some adaptability according to the load on the system or the need to save energy. Thus, it is necessary to ensure that the fingerprint will be generated under identical frequency conditions. Otherwise, it would be impossible to compare the variation in performance between various components. In this sense, components such as the CPU or GPU are the ones that can have more variability in their operating frequency, ranging from some MHz when are in power-save mode to several GHz when they are under high-performance requirements. Another aspect to take into account is the isolation of the software that performs the measurements with respect to other programs running on the system. The measures to guarantee this isolation include the separation of some of the CPU cores from the general process scheduler, the use of transactional memory (Harris et al., 2010), the disabling of interrupts by the kernel or isolating the GPU. Note that the exact actions may vary according to the components chosen. Moreover, it is also important to control external conditions such as temperature to the extent possible, since it can influence the performance variation of some components. In the case of using CPU timers, time synchronization made by services such as NTP should be also considered. These considerations seek to improve the *robustness* of the fingerprinting solution.

4.3. Data gathering

When the desired stability conditions have been achieved, it is necessary to define the functions to be performed on the components (selected in phase A) to measure their behavior in parallel and determine the possible skew between them. In this sense, the measurements must allow the comparison of the performance of two different components from the same device, avoiding executing the operations and measuring the deviation using a unique component.

Choosing the functions to run on each component to compare their behavior is a critical task during the fingerprinting solution design and must be carefully studied to ensure the *efficiency*, *diversity*, and *uniqueness* of the fingerprint. Due to the fact that functions taking longer times to execute may better show the variance between components, but may make the fingerprint generation process take too long. In addition, the created approach should not consume too many resources as it could slow down the normal operation of the system and affect other tasks. For example, the authors of Sanchez-Rola et al. (2018) decided to measure functions that take a short time to execute using the RTC, comparing CPU, and RTC oscillators. Besides, the authors of Salo (2007) measured the clock cycles in one second compared with the RTC and when processing one second of audio using the DSP. Moreover,

to fulfill the *security* property, the data collection process should be executed using Trusted Execution Environments (TEEs) (Lee and Park, 2020), if available, to isolate the fingerprint generation task from the rest of the device processes. This avoids possible data leaks caused by attacks based on memory vulnerabilities. Finally, the generated behavioral data is sent to a server, where it will be processed to generate the fingerprint. This sending should be done over a secure communications channel, such as SSH or TLS, avoiding possible interceptions of data transmissions.

4.4. Data preprocessing and feature extraction

Once the server receives the behavioral data, the next step is to preprocess the data to eliminate possible erroneous measurements and extract new information. Here, note that the data gathering step could be done several times before going to data preprocessing and feature extraction. The final objective is the generation of a set of feature vectors that will act as the generated fingerprint data, guaranteeing the *diversity* in the values. These vectors are the ones that will later be used to feed the ML/DL algorithms and generate the models.

To start the preprocessing part, it is necessary to remove outliers, constant, corrupted, or missing values that may be in the dataset by scanning over the dataset. For that, it is useful to plot each set of values collected and remove values that are more than 3 standard deviations away from the average. Afterwards, it is highly recommended to scale the data and have it in the same data range. In this sense, the most common scaling algorithms are min–max and standard normalization.

Once errors have been removed and the values scaled, the next step is feature extraction. It is possible to extract different features from the series of values by grouping them together and calculating different characteristics of the resultant series. One of the most typical values is the extraction of statistical values such as mean, median, deviation, max, min, or mode. However, applying more advanced calculations can provide even more relevant information about possible latent features in the values. In this sense, Discrete Fourier Transform (DFT) or Discrete Wavelet Transform (DWT) can be applied to extract features related to the time and order of the values. In addition, it is also possible to calculate features based on the correlation between the available values using algorithms like the Pearson correlation. Associating this step with works in the literature, the authors of Sanchez-Rola et al. (2018) used the mode of a series of 1000 values, and the authors of Babun et al. (2021) and Wang et al. (2012) employed the average of the measurements taken and the correlation in the generated values.

4.5. Evaluation approach selection

Once the features that will generate the fingerprint have been obtained, it is necessary to define the ML/DL approach to be followed (Usuga Cadavid et al., 2020). There are two possibilities here: a classification-based approach, in which each device in the environment will be associated with a label and one classifier is trained to recognize these labels; and an anomaly detection-based approach, where the data from each device is labeled as “normal” and a separate model is generated for each of them.

This decision must be made taking into account both the scenario (number of devices, variety of devices, possibility of adding or removing devices) and the features that have been collected (similarity of values between devices, number of features, etc.). Thus, an environment with a low number of devices may benefit from the use of classification algorithms, while more dynamic environments with a large number of devices will need more varied features and will benefit from anomaly detection algorithms. Here, the *scalability* and *efficiency* of the approach are better if no retraining is needed each time a device joins or leaves the scenario. In the literature, solutions have been found with both approaches, applying classification perspectives (Babun et al., 2021) or generating a statistical model per device and confronting the new fingerprints to it when identification is to be performed (Sanchez-Rola et al., 2018).

4.6. Model generation and evaluation design

Once the desired approach has been selected, either classification or anomaly detection, it is necessary to train ML/DL algorithms and define the metrics that will be used in the identification. This step should be carried out considering the *efficiency* in the evaluation process and the *security* against possible data-based attacks to the models.

There is a wide variety of algorithms that can be considered in this step, differentiating between traditional ML algorithms and DL algorithms based on neural networks (Usuga Cadavid et al., 2020). Starting from classification, algorithms such as Random Forest, k-Nearest Neighbors, eXtreme-Gradient Boosting (XGBoost), Support Vector Machines (SVM), or Multi-Layer Perceptron (MLP) can be used. From the anomaly detection prism, Isolation Forest, Local Outlier Factor (LOF), One Class-SVM, or Autoencoders are good alternatives as well. At this stage, it is also worth considering the application of algorithms focused on time series (Usuga Cadavid et al., 2020), depending on whether there are time-based dependencies between the values. Once the algorithms to use have been selected, it will be necessary to train and fine-tune the hyperparameters that give the best results in each of them. Note that these hyperparameters will vary according to the selected algorithms. In addition, the model predictions are usually one per vector, so they cannot be used directly to give a decision during the evaluation and identification of the device. In this sense, it is common to determine a *threshold* based on the model performance from which the device under evaluation will be accepted as the legitimate one. This threshold can be defined using numerous equations or conditions, such as defining the 50% of the values being recognized as legitimate, as done by the authors in Sanchez-Rola et al. (2018). Common metrics to consider on this step are *accuracy*, *true positive rate (TPR)*, *false positive rate (FPR)*, or *F1-Score*, among others (Usuga Cadavid et al., 2020).

At this point, it is worth noting that although this methodology has been designed primarily for ML/DL algorithms due to their current prominence in many research fields, it could be possible to include in this step other statistical algorithms, or even some self-developed algorithms as in Babun et al. (2021).

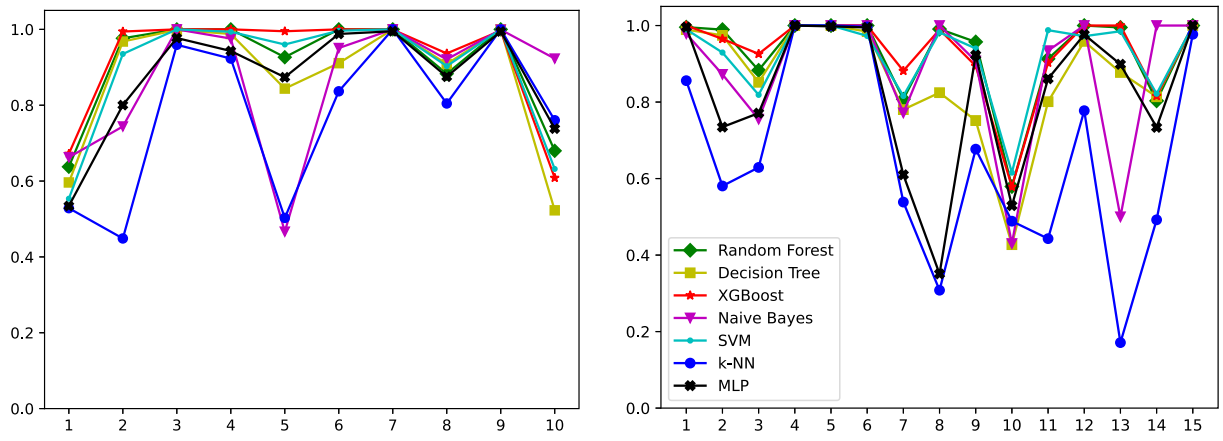
4.7. Device evaluation and identification decision

This step is only carried out in the evaluation phase and involves generating new behavioral data of the device following the same methodology as during the training phase, repeating steps B, C, and D.

Once the new dataset is generated, it is used to identify the device, determining whether it is the same device used during training or not. To this end, data will be evaluated using the ML/DL models previously generated, so that one result per vector is obtained. Then, the rule determined in the previous step will be applied, either based on a threshold or another equation to give a final decision on the device identification.

5. Methodology validation

This section validates the suitability of the proposed methodology by implementing a Proof-of-Concept (PoC) on a realistic scenario composed of 20 identical single-board devices. In particular, the devices are 15 Raspberry Pi 4 Model B 2 GB (RPi 4) and 10 Raspberry Pi 3 Model B+ (RPi3) running identical software images, with Raspbian 10 (buster) as OS and 5.4.83 as Linux kernel version. The operating systems ran in head-less mode, i.e., without a graphical environment or output to a display, a common configuration in SOC devices deployed in IoT. Next, it is detailed how the methodology has been implemented in the previous scenario, describing the decisions made in each of the defined steps. The language used has been Python and the code is available in Sánchez Sánchez (2021), for reproducibility sake.



(a) Average TPR for each RPi3.

(b) Average TPR for each RPi4.

Fig. 3. Fingerprint evaluation TPR during validation per device and model.

A. Hardware Component Selection. As a starting point, the physical oscillators available in the RPi4 and RPi3 were analyzed. The result of this study concluded that one oscillator is shared between the SoC components, running at 54 MHz in RPi4 and 19.2 MHz in RPi3, and the USB controller running at 25 MHz in both models (Embedded Linux Wiki, 2021). Since accessing the frequency of the USB oscillator from the device is not simple, the selected components were the VideoCore VI GPU and the ARM Quad-core Cortex-A72 CPU for RPi4 and VideoCoreIV GPU and the ARM Quad-core Cortex-A53 for RPi3. Although they share the base oscillator (GPU and CPU), their frequencies are given by different PLLs.

B. Component Isolation and Stability Assurance. Both the CPU and GPU work at varying frequencies depending on the load on the device. So, to guarantee the stability of the signatures, it is needed to ensure that frequency is fixed. For the validation, the RPi4 CPU frequency was set to 1.5 GHz and the GPU one to 500 MHz, while the RPi3 CPU frequency was set to 1.4 GHz and the GPU one to 400 MHz, the maximum values of both by default (without overclock). To do this, the Turbo Mode was enabled by adding *force_turbo=1* in */boot/config.txt*. After that, one of the CPU cores was isolated to be used in the fingerprint generation, using the options in */boot/cmdline.txt*, preventing processes from being assigned to it.

C. Data Gathering. To measure the variation of behavior between components, it was compared how the cycle counters of each component (CPU and GPU) vary with respect to the other. To do this, *sleep*, *random number generation*, and *hash* functions were selected. As the validation prototype was implemented in Python, *time.sleep()* was used for *sleep* execution, *os.urandom()* was used for *random number generation*, and *hashlib.sha256()* was used to *hash* a string. In particular, these functions were sequentially executed in the CPU and the number of GPU cycles that occurred during each function execution was measured. To interact with the GPU, Idein's py-videocore6 library (Idein, 2021b) was used in RPi4. Concretely, the *CORE_PCTR_CYCLE_COUNT* GPU counter was the register monitored. In the case of RPi3, Idein's py-videocore (Idein, 2021a) library was used to monitor the *QPU_Total_idle_clock* counter (as the RPi3s were in headless mode). The data gathering procedure is summarized in Algorithm 1. For the data collection, the sleep function time *t* was set to 120 s, as the variations between CPU and GPU are presumably low, a fixed string was set for the hash function, and the number on measurements (*n_measurements*) was set to 400. It is important mentioning that these values were adjusted according to the results in later steps. Other configuration parameters such as *t=60* s were tested providing with slightly worse results. Additionally, the use of TEE to run the algorithm was considered, however the ARM TrustZone instance available in RPi is simulated only (TrustedFirmware.org, 2012).

Algorithm 1: CPU/GPU data acquisition algorithm

Result: Set of GPU/CPU performance measurements.

```
result_set={};
for n in n_measurements do
    #Sleep cycle counter
    GPU_CYCLE_COUNT=0;
    sleep(t);
    sleep_gpu_cycles=GPU_CYCLE_COUNT;

    #Random number generator cycle counter
    GPU_CYCLE_COUNT=0;
    random_number_gen();
    random_gpu_cycles=GPU_CYCLE_COUNT;

    #Hash cycle counter
    GPU_CYCLE_COUNT=0;
    hash("Test string");
    hash_gpu_cycles=GPU_CYCLE_COUNT;

    #Add measurements to result set
    result_set.append("sleep_gpu_cycles,
      random_gpu_cycles,hash_gpu_cycles");
end
```

D. Data Preprocessing and Feature Extraction. The data gathering process was repeated a total of ten times per device, for testing purposes, with different temperature conditions and performing several reboots between the generation of each fingerprint (set of measurements). Then, the 400 measurements of each fingerprint were grouped in different sliding windows ranging from 10 to 100 values in jumps of 10 values (10 different sliding windows in total). Afterwards, several statistical features were calculated for each window-based group and concatenated together. Concretely, the statistical values calculated were: *minimum*, *maximum*, *mean*, *median* and *sum*. Following this approach, the resultant vectors for training and evaluation have a size of 150 (3 data gathering functions * 10 different sliding windows * 5 statistical features). Table 3 depicts the final set of features extracted from this step.

E. Evaluation Approach Selection. Due to the staticity of the test environment, as the number of devices do not change in time, it was decided to follow an approach based on classification ML algorithms combined with a threshold on the True Positive Rate (TPR) that would

Table 3
Feature set extracted for validation.

Operation collected	Python code function	Sliding windows	Statistics extracted	No. features
Sleep 120 s	*time.sleep(120)*	10 Sliding windows.	Minimum, maximum,	50
Random number gen.	*os.urandom()*	Group sizes: 10, 20, 30, 40,	mean, median,	50
String hashing	*hashlib.sha256(str)*	50, 60, 70, 80, 90, 100	sum	50
Total				150

Table 4
Classification algorithms and hyperparameters tested.

Model	Hyperparameters tested	Avg TPR
Naive Bayes	No hyperparameter tuning required	87.29%
k-NN	$k \in [3, 20]$	71.40%
SVM	$C \in [0.01, 100]$, $gamma \in [0.001, 10]$ $kernel \in \{'rbf', 'linear', 'sigmoid', 'poly'\}$	89.65%
XGBoost	$lr \in [0.01, 0.3]$, $max\_depth \in [3, 15]$ $min\_child\_weight \in [1, 7]$, $gamma \in [0, 0.5]$, $colsample\_bytree \in [0.3, 0.7]$	91.92%
Decision Tree	$max\_depth \in [None, 5, 10, 15, 20]$ $min\_samples\_split \in [2, 3, 4, 5]$	86.47%
Random Forest	$number\_of\_trees \in [50, 1000]$ $max\_depth \in [None, 5, 10, 15, 20]$ $min\_samples\_split \in [2, 3, 4, 5]$	91.64%
MLP	$layers \in [1, 3]$, $neurons\_layer \in [10, 100]$, $batch\_size \in [32, 128, 256, 512]$	85.32%

delimit the minimum number of successfully classified vectors. Besides, F1-Score is also calculated to validate the classification performance of the models. (TP: True Positive, FP: False Positive, TN: True Negative, FN: False Negative).

$$TPR/recall = \frac{TP}{TP + FP} \tag{1}$$

$$F1 - Score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{2}$$

F. Model Generation and Evaluation Design. For the model generation, six fingerprints of each device were used as separate training in order to have cross-validation. The selected algorithms were Random Forest, Decision Tree, k-NN, XGBoost, Naive Bayes, SVM and MLP. After hyperparameter optimization (see Table 4), using cross-validation with the fingerprints used for training, the best performing algorithm was XGBoost (*lr=0.1, max_depth=20, gamma=0.01, colsample_bytree =0.5*), giving an average TPR of 91.92%, ranging from 100% in the best case to 55% in the worst (a random predictor would give 4% for each device, as the model can be easily identified based on device frequency). Fig. 3 shows the results per algorithm and device. This value varies highly, as some of them seem to be more similar between them. Based on the previous results, a threshold of 50% in the assigned classes in evaluation can be defined to give the identification decision, so that if half of the vectors are correctly evaluated, the device is recognized as legitimate.

As Fig. 3 depicts, the performance of the classifiers when identifying the devices is not homogeneous and they are able to classify better some devices than others. To explore more in detail this aspect, Fig. 4 shows the density plots for the CPU and GPU skew after the sleep function execution. In the vertical dotted line, the median of the distribution is shown. It can be appreciated how the skew of some devices varies, being more similar between some of them. Concretely, for the RPi4 number 10, the one with the worst classification performance in Fig. 3, it can be seen in Fig. 4 how the median of its distribution is almost identical to the RPi4 number 12. Although only one of the three functions executed is plotted due to space constraints, this analysis demonstrates how some devices are more similar to each other than others, a factor that influences the scalability of the solution, so that for larger deployments, a greater number of functions or data sources would be necessary.

G. Device Evaluation and Identification Decision. In the present PoC, this phase was performed with the four fingerprints of each device not used for the previous phase. In this step, the normalization was repeated with the same values used to generate the model, and the vectors containing the same features were evaluated using the XGBoost model trained previously. Using the 50% threshold as explained above, all the devices were correctly identified without any device erroneously identified as another one. Fig. 5 shows the average confusion matrix for the four fingerprints used for testing, using XGBoost as classifier. The labels are defined as the device model followed by its MAC address. The evaluation is done by grouping together devices within the same model, as RPi3 and RPi4 have different running frequencies in the components leveraged and they can be easily differentiated. ≈93% and ≈92% average F1-score is achieved for RPi4 and RPi3, respectively.

As conclusion, it has been demonstrated the performance of the proposed methodology in an environment with real devices. Still, this is only a PoC and its performance could be substantially improved by extracting other data from devices and generating more elaborate features.

6. Discussion, lessons learned and limitations

This section compares the proposed methodology with the solutions available in the literature. After that, it discusses the limitations of the proposed solution and provides some lessons learned.

6.1. Literature comparison

Despite the solutions discussed in Section 2 do not follow a common methodology, many of them implement certain steps proposed in this work. Table 5 compares the proposed methodology and related work using on-device components for identification. As can be seen, all works performing identification utilize a threshold (Step F), defined based on different statistical approaches. Besides, none of the approaches employed ML/DL algorithms (Step F) and many of them did not consider hardware isolation properly or the usage of fixed component frequencies (Step B).

After the theoretical comparison, it is relevant to analyze the most similar and comparable solutions from a common prism. Although most of the solutions analyzed in Section 2 use components that are not available on the RPi4 or RPi3, three solutions, Dong et al. (2019), Nakibly et al. (2015) and Sanchez-Rola et al. (2018), can be adapted to the present scenario and methodology. Table 6 compares the methodology approach and the results of its validation with four implementations inspired by the works found in Section 2. Besides, it highlights which fingerprint properties were not met, resulting in erroneous device identification.

The first of these approaches was inspired by Dong et al. (2019), only the CPU was selected as a component but making the fingerprint of each of its cores separately by using thread affinity. The features to be obtained were statistics based on the time taken to perform small operations (string hash and random number generation) on each of the cores. Using LOF as an anomaly detection algorithm and one model per device, the identification was possible by setting a threshold of 50%. However, the reboot of the devices caused the fingerprints to change and it was not possible to perform the identification due to the new kernel process scheduling, something that may also be affecting the proposed solution in Dong et al. (2019). The same problem occurred in a second tested approach inspired by Nakibly et al. (2015). In particular, each CPU core was compared with the GPU separately in a concurrent manner and executing short operations. Here, different operations of variable complexity were performed in the GPU while the execution time was measured using the CPU. In this case, the evaluation also followed an anomaly detection-based approach, being LOF the algorithm with the better results. Again, it was possible to identify the devices consistently, now using a threshold around 60%,

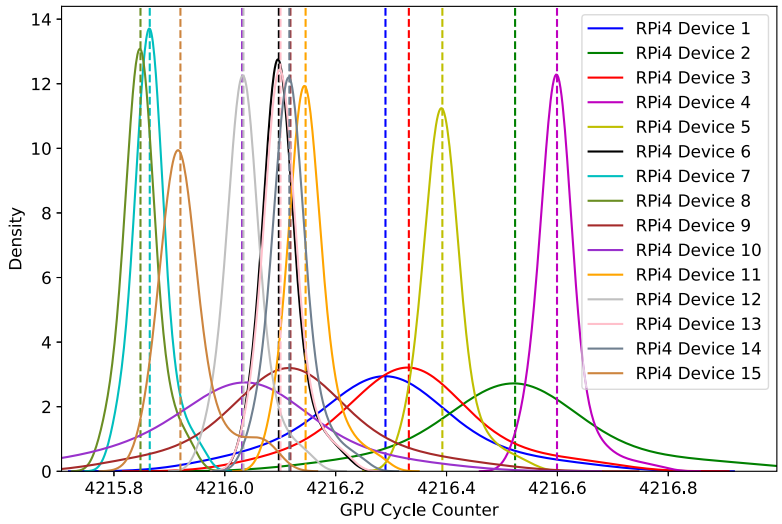


Fig. 4. Density plot for the GPU cycle counter in RPi4.

Table 5
Analogy between hardware-based fingerprinting solutions in the literature and the proposed methodology.

Work	Step A	Step B	Step C	Step D	Step E	Step F	Step G
Salo (2007)	RTC, DSP, CPU	–	CPU cycles in one second (measured with DSP and RTC)	Raw values	–	Statistical (t-test), $p <= 0.05$ threshold	98.7%–93.3% identification
Sanchez-Rola et al. (2018)	RTC, CPU	Transactional memory	Execution time of short functions	Mode-based matrix	–	Statistical comparison, 50% similarity threshold	Correct identification
Wang et al. (2012)	Flash memory	Isolation of one page in flash memory	Bit programming errors (flip from 1 to 0)	Error order per bit	–	Pearson correlation, 0.5 threshold	Estimated 4.52×10^{-815} FPR and 2.65×10^{-539} FNR
Dong et al. (2019)	CPU	Thread affinity	CPU usage while executing a cyclical tasks	Raw values	–	Dynamic Time Warping algorithm, 0.3244 threshold	93.43% uniqueness (Shannon entropy)
Nakibly et al. (2015)	CPU, GPU	–	Number of frames per 5 s	Entropy and statistics	–	Statistical	No evaluation, partial differentiation capabilities
This work	CPU and GPU	Core isolation, Fixed frequency	Sleep for 120 secs, Random num. gen., hash	Sliding window-based statistical features	Classification	XGBoost, 50% threshold	Perfect Identification (91.92% avg. TPR)

Table 6
Comparison of the validation approaches implemented.

Approach	Step A	Step B	Step C	Step D	Step E	Step F	Step G	Properties not met
Dong et al. (2019)-inspired approach	CPU	Thread affinity	Short functions	Raw values	Anomaly Detection	LOF, 50% threshold	Identification until device reboots (69.4% avg. TPR)	Stability
Nakibly et al. (2015)-inspired approach	CPU and GPU	–	Different GPU operations	Raw values	Anomaly Detection	LOF, 60% threshold	Identification until device reboots (89.6% avg. TPR)	Stability
Sanchez-Rola et al. (2018)-inspired approach A	CPU	–	Short functions	Window-based statistical features	Classification	XGBoost, 50% threshold	No identification (27.5% avg. TPR)	Uniqueness, Diversity, Stability
Sanchez-Rola et al. (2018)-inspired approach B	CPU	–	Short functions	Window-based statistical features	Anomaly Detection	LOF, 50% threshold	No identification (19.8% avg. TPR)	Uniqueness, Diversity, Stability
This work (Section 5)	CPU and GPU	Core isolation, Fixed frequency	GPU-measured CPU operations	Window-based statistical features	Classification	XGBoost, 50% threshold	Perfect Identification (91.9% avg. TPR)	–

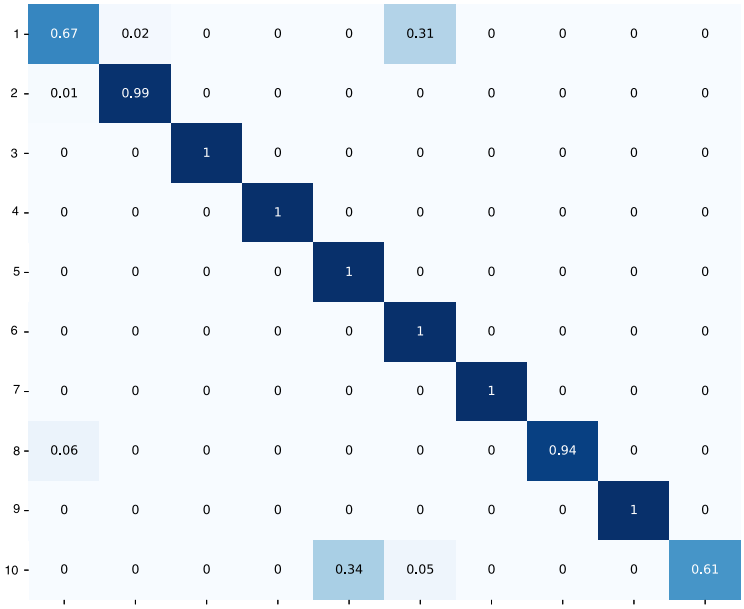
until they are rebooted. Finally, two different approaches were tested inspired by Sanchez-Rola et al. (2018). Both share the fact that the data collected was based on short functions executed in the CPU without considering stability measurements. They differ in the evaluation approach, one using anomaly detection and the other using classification. These approaches achieved the worse performance, as the solutions could not identify the devices even without rebooting them.

From these results, it can be concluded that the stability of these approaches is not sufficient for dynamic IoT scenarios where the devices operate in a typical way (i.e. devices are restarted from time to time and power can go out). In contrast, they would be useful in IoT environments where device reboots are not possible, such as in the control of electrical or security systems.

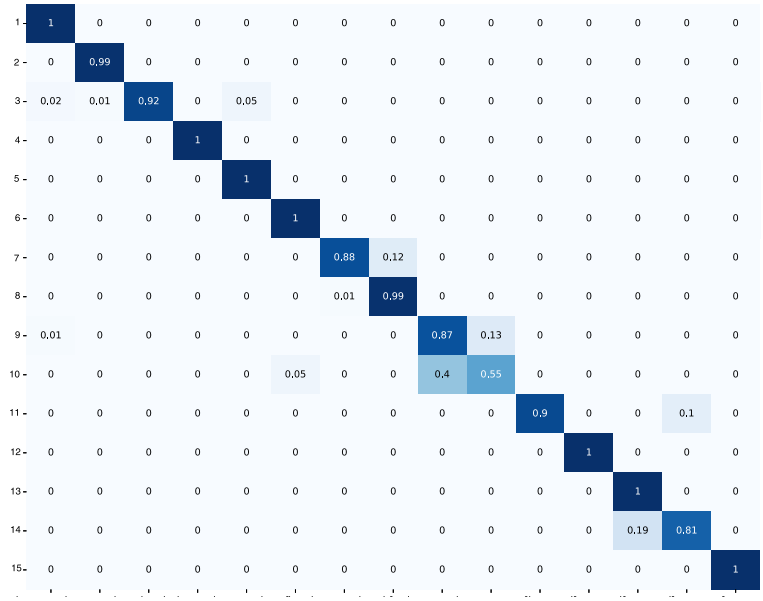
6.2. Lessons learned and limitations

From the above comparison and the tests performed, valuable conclusions are drawn, both in the form of lessons learned and possible limitations of the proposed methodology. Regarding lessons learned, the main ones are:

Component isolation is critical. As Table 6 shows, it can be seen that isolating the measurements from external processes is crucial to



(a) 10 RPi3 classification confusion matrix.



(b) 15 RPi4 classification confusion matrix.

Fig. 5. Test confusion matrix for device identification using XGBoost.

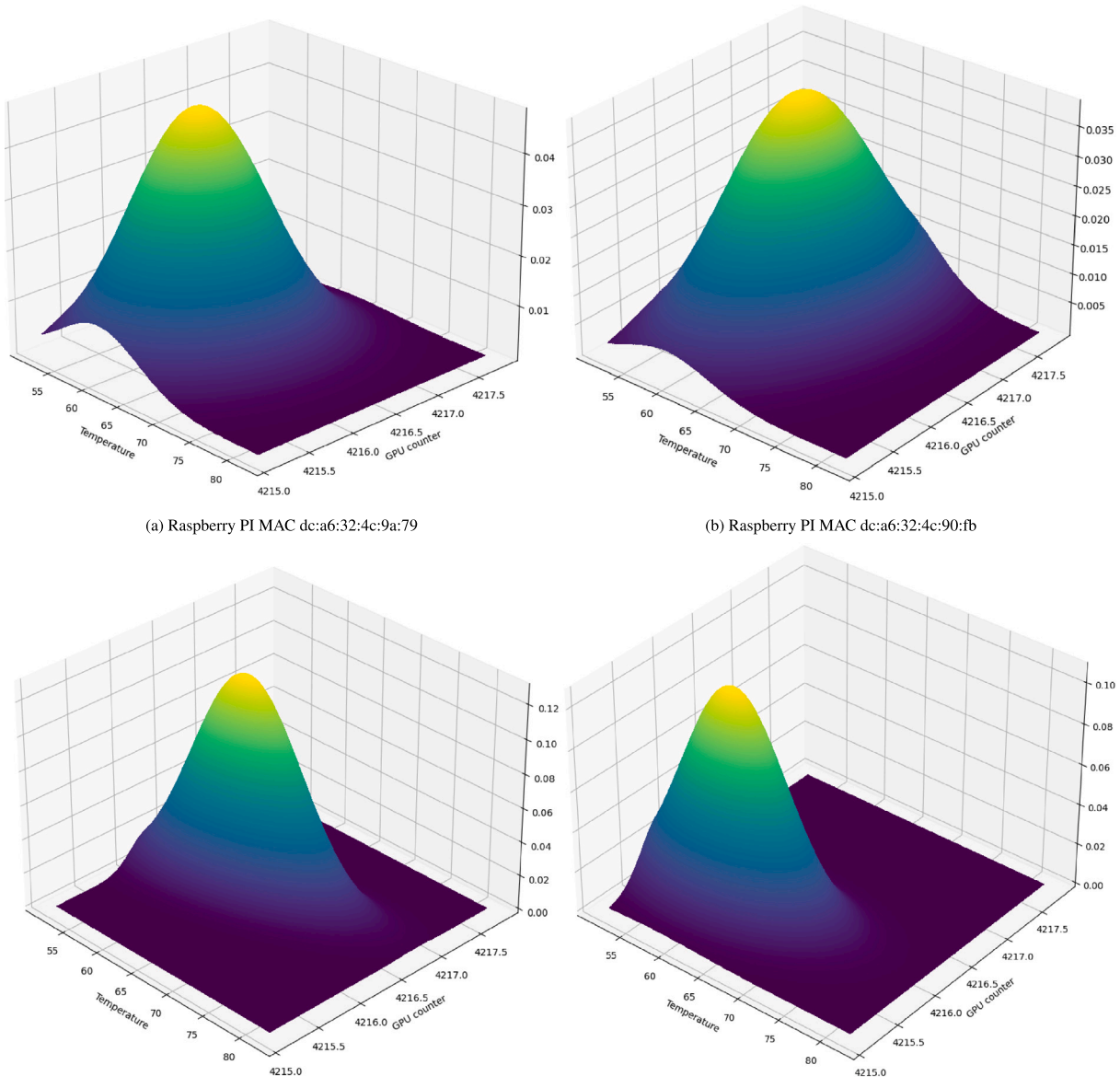
ensure the *stability* of the fingerprinting process. In this sense, in cases where the conditions of the components were not stable, it was not possible to reliably identify them after device rebooting.

Rebooting can have impact on the fingerprints. During the testing of literature-based validation approaches (see Table 6), it was observed that the restart of the devices has an impact when the fingerprinting program is not isolated from other processes, probably due to the effect of the process scheduler. In contrast, this issue was not present in the approach of Section 5, as the data collection process was properly isolated from the noise introduced by other processes in the device. From this validation, it an be concluded that the *robustness* property against the negative effects of other processes running in the device is achieved.

Temperature does not seem to affect the components selected for validation. The above tests have been performed at different temperature conditions and this condition does not seem to affect the results, possibly because by using integrated components on the same chip, it affects the base frequency and overall performance equally. Therefore, the *robustness* property is met based on the temperature context. Actually, temperature was also measured during the data gathering of Section 5. Using it as a feature, the average TPR for XGBoost is increased from 91.92% to 93.46%. Therefore, this information can be added as a correlation feature, incorporating supplementary information to the identification process. For different devices, Fig. 6 shows the density plot of the correlation in different devices of the temperatures and the GPU counter value after a 120-second CPU sleep. It can be seen that each device has a different plot shape and temperature is not influencing that the devices generate a similar fingerprint.

In terms of limitations of the methodology, the following have been identified after its design and validation:

The methodology implementation is highly dependent on the hardware model. The implementation of the present methodology,



(a) Raspberry PI MAC dc:a6:32:4c:9a:79

(b) Raspberry PI MAC dc:a6:32:4c:90:fb

(c) Raspberry PI MAC dc:a6:32:4c:98:93

(d) Raspberry PI MAC dc:a6:32:4c:99:bf

Fig. 6. 3D density plot of temperature/GPU cycle counter value for different devices executing the 120 s-sleep function.

being based on the hardware components available in the devices, is highly dependent on the libraries needed to interact with them. Thus, implementations of the methodology may not be compatible between different models of single-board devices if their components are different, so it would be necessary to adapt the code.

Some steps might need an exploratory analysis. It is difficult to determine which hardware behavior measurements to take or which features to extract a priory. So, the implementation of the methodology may require several exploratory iterations to find a combination that meets all the properties needed in the generated fingerprint. This trial-and-error analysis can be highly reduced by analyzing the leveraged devices properties, different component and running frequencies. As every chip has imperfections, the challenge is how to measure them properly. In Section 5, a successful application of the methodology has been provided, which serves as a guide and recommendation for future applications.

Scalability in large deployments. Manufacturing errors and variations are within the accepted tolerance range accepted by the manufacturers. Therefore, using these variations for identification in large deployments makes a single source of data possibly not sufficient (Polcák and Franková, 2015). Thus, depending on the number of devices to be individually identified, a greater number of components and features should be employed to generate unique device fingerprints. Therefore, *scalability* property arises as one of the most difficult properties to be met.

6.3. Insights for real-world implementations

Based on the previous set of lessons learned and limitations, this section gives some implementation ideas for future researchers that may deploy the proposed methodology in real-world IoT scenarios based on SBCs. Examples of these scenarios can be spectrum crowd-sensing, with projects such as ElectroSense (Rajendran et al., 2017) or agriculture environments where SBCs are employed to control sensors and actuators. The main guidelines for these scenarios are:

1. Investigate how to get the hardware counters. After checking the available hardware components that might be used for fingerprinting, a critical step is to check the firmware managing them and how their performance counters can be gathered.

2. Use heterogeneous functions for data collection. As selecting the executed functions to perform the fingerprint can be seen as an exploratory step, selecting both long and short time execution functions is a good decision as the collected data can be later processed and compare the results from both approaches.
3. Expend time in feature extraction analysis. As the tolerance errors in the hardware components are between constrained limits, the collected performance values will be similar. Therefore, extracting useful metrics (such as the median in the validation of Section 5) is a critical step to separate the distributions of the collected data and perform the device identification.
4. Use tree-based ML algorithms as the initial evaluation approach. In the validation section, these methods provided good performance with relatively low complexity in the hyperparameter tuning. Therefore, before exploring more complex DL solutions, tree-based ML methods can give the desired performance keeping a lower complexity in the fingerprinting solution.

7. Conclusions and future work

This paper proposes a methodology composed of seven steps that allow identifying identical single-board devices (same hardware and software configuration) used in heterogeneous IoT scenarios. These seven steps are grouped into two main phases, one to generate a behavioral fingerprint and another to evaluate it and identify the device. This work also presents the threat model affecting single-board device identification and seven properties that solutions dealing with identical device identification based on behavioral fingerprint must consider: uniqueness, stability, diversity, scalability, efficiency, robustness and security. The proposed methodology has been successfully validated in a real environment composed of 25 identical Raspberry Pi 4 Model B and Raspberry Pi 3 Model B+ using ML techniques for data processing. These devices were perfectly identified using a XGBoost model trained using features derived from the variation in performance between their CPU and GPU by setting a 50% TPR threshold. Besides, this work compared the methodology identification performance with other implementations inspired in the literature works and provided some lessons learned and limitations.

As future work, it is planned to validate the methodology in larger scenarios with more devices and types, defining new features to be obtained and other ML/DL algorithms to evaluate the scalability of the solution in larger and real-world environments. The performance of the solution in a dynamic scenario is another key aspect to be researched. Furthermore, it is desired to explore the usage of TEEs when generating the fingerprint, guaranteeing the security of the measurements by isolating the fingerprinting program from the rest of the system processes. Besides, we also plan to perform adversarial attacks against the proposed validation PoC, improving its resilience and performance.

CRediT authorship contribution statement

Pedro Miguel Sánchez Sánchez: Writing – original draft, Software, Data curation, Formal analysis, Visualization, Writing – review & editing. **José María Jorquera Valero:** Writing – original draft, Validation, Visualization. **Alberto Huertas Celdrán:** Supervision, Writing – original draft, Resources, Writing – review & editing, Methodology. **Gérôme Bovet:** Funding acquisition, Project administration, Writing – review & editing. **Manuel Gil Pérez:** Supervision, Resources, Writing – review & editing. **Gregorio Martínez Pérez:** Funding acquisition, Project administration, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data and code available on GitHub. Link available in the article.

Acknowledgments

This work has been partially supported by *(a)* the Swiss Federal Office for Defense Procurement (armasuisse) with the TREASURE and CyberSpec (CYD-C-2020003) projects and *(b)* the University of Zürich UZH.

References

Ahmed, Dilawer, Das, Anupam, Zaffar, Fareed, 2022. Analyzing the feasibility and generalizability of fingerprinting Internet of Things devices. Proc. Priv. Enhanc. Technol. 2022 (2), 578–600.

Al-Omary, Alauddin, Othman, Ali, AlSabbagh, Haider M, Al-Rizzo, Hussain, 2018. Survey of hardware-based security support for IoT/CPS systems. KnE Eng. 52–70.

Arellanes, Damian, Lau, Kung-Kiu, 2020. Evaluating IoT service composition mechanisms for the scalability of IoT systems. Future Gener. Comput. Syst. 108, 827–848.

Arp, Daniel, Quiring, Erwin, Pendlebury, Feargus, Warnecke, Alexander, Pierazzi, Fabio, Wressnegger, Christian, Cavallaro, Lorenzo, Rieck, Konrad, 2022. Dos and don'ts of machine learning in computer security. In: Proc. of the USENIX Security Symposium.

Babaei, A., Schiele, G., 2019. Physical unclonable functions in the Internet of Things: State of the art and open challenges. Sensors 19 (14), 3208.

Babun, L., Aksu, H., Uluagac, A.S., 2021. CPS device-class identification via behavioral fingerprinting: From theory to practice. IEEE Trans. Inf. Forensics Secur. 16, 2413–2428. http://dx.doi.org/10.1109/TIFS.2021.3054968.

Chen, Zhiyan, Liu, Jinxin, Shen, Yu, Simsek, Murat, Kantarci, Burak, Mouftah, Hussein T., Djukic, Petar, 2022. Machine learning-enabled IoT security: Open issues and challenges under advanced persistent threats. ACM Comput. Surv..

Dong, S., Farha, F., Cui, S., Ma, J., Ning, H., 2019. CPG-FS: A CPU performance graph based device fingerprint scheme for devices identification and authentication. In: 4th IEEE Cyber Science and Technology Congress. pp. 266–270.

Embedded Linux Wiki, 2021. The undocumented Pi. https://elinux.org/The_Undocumented_Pi/. (Online; Accessed 8 June 2021).

Fayos-Jordan, Rafael, Felici-Castell, Santiago, Segura-Garcia, Jaume, Lopez-Ballester, Jesus, Cobos, Maximo, 2020. Performance comparison of container orchestration platforms with low cost devices in the fog, assisting Internet of Things applications. J. Netw. Comput. Appl. 169, 102788.

Gao, Yansong, Su, Yang, Yang, Wei, Chen, Shiping, Nepal, Surya, Ranasinghe, Damith C, 2019. Building secure SRAM PUF key generators on resource constrained devices. In: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops. PerCom Workshops, IEEE, pp. 912–917.

Hamza, Ayyoob, Ranathunga, Dinesha, Gharakheili, Hassan Habibi, Roughan, Matthew, Sivaraman, Vijay, 2018. Clear as MUD: Generating, validating and applying IoT behavioral profiles. In: Proceedings of the 2018 Workshop on IoT Security and Privacy. pp. 8–14.

Harris, T., Larus, J., Rajwar, R., 2010. Transactional memory. In: Synthesis Lectures on Computer Architecture, vol. 5, (no. 1), Morgan & Claypool Publishers, pp. 1–263.

Huang, J., Albazrqaoe, W., Xing, G., 2014. BlueID: A practical system for Bluetooth device identification. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications. IEEE, pp. 2849–2857.

Idein, 2021a. py-videocore. Python library for GPGPU on Raspberry Pi. https://github.com/nineties/py-videocore/. (Online; Accessed 8 June 2021).

Idein, 2021b. py-videocore6. Python library for GPU programming on Raspberry Pi 4. https://github.com/Idein/py-videocore6/. (Online; Accessed 8 June 2021).

Jafari, H., Omotere, O., Adesina, D., Wu, H., Qian, L., 2018. IoT devices fingerprinting using deep learning. In: 2018 IEEE Military Communications Conference. pp. 1–9. http://dx.doi.org/10.1109/MILCOM.2018.8599826.

Jagdale, Saumitra, 2022. The role of hardware root of trust in edge devices. https://www.eetimes.eu/the-role-of-hardware-root-of-trust-in-edge-devices/. (Online; Accessed 21 June 2022).

Jana, S., Kasera, S.K., 2009. On fast and accurate detection of unauthorized wireless access points using clock skews. IEEE Trans. Mob. Comput. 9 (3), 449–462.

Kohno, T., Broido, A., Claffy, K.C., 2005. Remote physical device fingerprinting. IEEE Trans. Dependable Secure Comput. 2 (2), 93–108.

Kong, Joonho, Koushanfar, Farinaz, 2013. Processor-based strong physical unclonable functions with aging-based response tuning. IEEE Trans. Emerg. Top. Comput. 2 (1), 16–29.

Lanze, F., Panchenko, A., Braatz, B., Zinnen, A., 2012. Clock skew based remote device fingerprinting demystified. In: 2012 IEEE Global Communications Conference. pp. 813–819.

Lee, U., Park, C., 2020. SofTEE: Software-based trusted execution environment for user applications. IEEE Access 8, 121874–121888.

Li, Deqiang, Li, Qianmu, 2020. Adversarial deep ensemble: Evasion attacks and defenses for malware detection. IEEE Trans. Inf. Forensics Secur. 15, 3886–3900.
Liu, Yongxin, Wang, Jian, Li, Jianqiang, Song, Houbing, Yang, Thomas, Niu, Shuteng, Ming, Zhong, 2020. Zero-bias deep learning for accurate identification of Internet-of-Things (IoT) devices. IEEE Internet Things J. 8 (4), 2627–2634.
Lu, Yang, Da Xu, Li, 2018. Internet of Things (IoT) cybersecurity research: A review of current research topics. IEEE Internet Things J. 6 (2), 2103–2115.
Marabissi, Dania, Mucchi, Lorenzo, Stomaci, Andrea, 2022. IoT nodes authentication and ID spoofing detection based on joint use of physical layer security and machine learning. Future Internet 14 (2), 61.
Montalbano, Elizabeth, 2020. Bluetooth spoofing bug affects billions of IoT devices. https://threatpost.com/bluetooth-spoofing-bug-iot-devices/159291/. (Online; Accessed 21 June 2022).
Nakibly, G., Shelef, G., Yudilevich, S., 2015. Hardware fingerprinting using HTML5. arXiv preprint arXiv:1503.01408.
Nosouhi, Mohammad Reza, Sood, Keshav, Grobler, Marthie, Doss, Robin, 2022. Towards spoofing resistant next generation IoT networks. IEEE Trans. Inf. Forensics Secur..
Pawar, S.N., Mane, P.B., 2017. Wide band PLL frequency synthesizer: A survey. In: 2017 International Conference on Advances in Computing, Communication and Control. IEEE, pp. 1–6.
Peng, Limei, Dhaini, Ahmad R., Ho, Pin-Han, 2018. Toward integrated Cloud–Fog networks for efficient IoT provisioning: Key challenges and solutions. Future Gener. Comput. Syst. 88, 606–613.
Perales Gómez, A.L., Fernández Maimó, L., Huertas Celdran, A., García Clemente, F.J., Cadenas Sarmiento, C., Del Canto Masa, C.J., Méndez Nistal, R., 2019. On the generation of anomaly detection datasets in industrial control systems. IEEE Access 7, 177460–177473.
Polcák, L., Franková, B., 2015. Clock-skew-based computer identification: Traps and pitfalls. J. UCS 21 (9), 1210–1233.
Radhakrishnan, S.V., Uluagac, A.S., Beyah, R., 2014. GTID: A technique for physical device and device type fingerprinting. IEEE Trans. Dependable Secure Comput. 12 (5), 519–532.
Rajan, Anjana, Jithish, J., Sankaran, Sriram, 2017. Sybil attack in IOT: Modelling and defenses. In: 2017 International Conference on Advances in Computing, Communications and Informatics. ICACCI, IEEE, pp. 2323–2327.
Rajendran, Sreeraj, Calvo-Palomino, Roberto, Fuchs, Markus, Van den Bergh, Bertold, Cordobés, Héctor, Giustiniano, Domenico, Pollin, Sofie, Lenders, Vincent, 2017. Electrosense: Open and big spectrum data. IEEE Commun. Mag. 56 (1), 210–217.
Riyaz, S., Sankhe, K., Ioannidis, S., Chowdhury, K., 2018. Deep learning convolutional neural networks for radio identification. IEEE Commun. Mag. 56 (9), 146–152.
Rührmair, Ulrich, Devadas, Srinivas, Koushanfar, Farinaz, 2012. Security based on physical unclonability and disorder. In: Introduction to Hardware Security and Trust. Springer, pp. 65–102.
Sabhanayagam, T., 2022. A comparative analysis to obtain unique device fingerprinting. In: Proceedings of International Conference on Deep Learning, Computing and Intelligence. Springer, pp. 349–354.
Salo, T.J., 2007. Multi-factor fingerprints for personal computer hardware. In: MILCOM 2007-IEEE Military Communications Conference. pp. 1–7.
Sanchez-Rola, I., Santos, I., Balzarotti, D., 2018. Clock around the clock: Time-based device fingerprinting. In: 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 1502–1514. http://dx.doi.org/10.1145/3243734.3243796.
Sánchez Sánchez, P.M., 2021. identification_methodology. https://github.com/sxz0/identification_methodology. (Online; Accessed 8 June 2021).
Sánchez Sánchez, P.M., Jorquera Valero, J.M., Huertas Celdrán, A., Bovet, G., Gil Pérez, M., Martínez Pérez, G., 2021. A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets. IEEE Commun. Surv. Tutor. 23 (2), 1048–1077. http://dx.doi.org/10.1109/COMST.2021.3064259.
Sembiring, Rivaldo Ludovicus, Pahlevi, Rizka Reza, Sukarno, Parman, 2021. Randomness, uniqueness, and steadiness evaluation of physical unclonable functions. In: 2021 9th International Conference on Information and Communication Technology. ICoICT, IEEE, pp. 429–433.
Sharma, S., Hussain, A., Saran, H., 2012. Experience with heterogenous clock-skew based device fingerprinting. In: 2012 Workshop on Learning from Authoritative Security Experiment Results. pp. 9–18.
Tehranipoor, Fatemeh, Karimian, Nima, Yan, Wei, Chandy, John A, 2016. DRAM-based intrinsic physically unclonable functions for system-level security and authentication. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 25 (3), 1085–1097.
TrustedFirmware.org, 2012. OP-TEE documentation. Raspberry Pi 3. https://optee.readthedocs.io/en/latest/building/devices/rpi3.html/. (Online; Accessed 8 June 2022).
Usuga Cadavid, J.P., Lamouri, S., Grabot, B., Pellerin, R., Fortin, A., 2020. Machine learning applied in production planning and control: A state-of-the-art in the era of industry 4.0. J. Intell. Manuf. 1–28.
Wang, Y., Yu, W., Wu, S., Malysa, G., Suh, G.E., Kan, E.C., 2012. Flash memory for ubiquitous hardware security functions: True random number generation and device fingerprints. In: 2012 IEEE Symposium on Security and Privacy. pp. 33–47.
Yousefnezhad, Narges, Malhi, Avleen, Främling, Kary, 2020. Security in product lifecycle of IoT devices: A survey. J. Netw. Comput. Appl. 171, 102779.
Yue, Michael, Karimian, Nima, Yan, Wei, Anagnostopoulos, Nikolaos Athanasios, Tehranipoor, Fatemeh, 2020. DRAM-based authentication using deep convolutional neural networks. IEEE Consum. Electron. Mag. 10 (4), 8–17.
Zhou, Xinyu, Hu, Aiqun, Li, Guyue, Peng, Linning, Xing, Yuexiu, Yu, Jiabao, 2019. Design of a robust RF fingerprint generation and classification scheme for practical device identification. In: 2019 IEEE Conference on Communications and Network Security. CNS, IEEE, pp. 196–204.

Pedro Miguel Sánchez Sánchez received the M.Sc. degree in computer science from the University of Murcia. He is currently pursuing his Ph.D. in computer science at University of Murcia. His research interests are focused on continuous authentication, networks, 5G, cybersecurity and the application of machine learning and deep learning to the previous fields.

José María Jorquera Valero is a Ph.D. student in Computer Science at Murcia University. Jorquera Valero received the M.Sc. degree in Computer Science from the University of Murcia, Spain. His scientific interests include cybersecurity, data privacy, continuous authentication, computer networks, and 5G.

Alberto Huertas Celdrán received the M.Sc. and Ph.D. degrees in computer science from the University of Murcia, Spain. He is currently a postdoctoral fellow associated with the Communication Systems Group (CSG) at the University of Zurich UZH. His scientific interests include medical cyber–physical systems (MCPS), brain–computer interfaces (BCI), cybersecurity, data privacy, continuous authentication, semantic technology, context-aware systems, and computer networks.

Gérôme Bovet is the head of data science for the Swiss DoD, where he leads a research team and a portfolio of about 30 projects. His work focuses on machine/deep learning approaches applied to cyber-defence use cases, with emphasis on anomaly detection, adversarial and collaborative learning. He received his Ph.D. in networks and systems from Telecom ParisTech, France, in 2015.

Manuel Gil Pérez is Associate Professor in the Department of Information and Communication Engineering of the University of Murcia, Murcia, Spain. His scientific activity is mainly devoted to cybersecurity, including intrusion detection systems, trust management, privacy-preserving data sharing, and security operations in highly dynamic scenarios. Gil Pérez received M.Sc. and Ph.D. degrees (latter with distinction) in Computer Science from the University of Murcia.

Gregorio Martínez Pérez is Full Professor in the Department of Information and Communications Engineering of the University of Murcia, Spain. His scientific activity is mainly devoted to cybersecurity and networking, also working on the design and autonomic monitoring of real-time and critical applications and systems. He is working on different national (14 in the last decade) and European IST research projects (11 in the last decade) related to these topics, being Principal Investigator in most of them. He has published 160+ papers in national and international conference proceedings, magazines and journals.