UNIVERSIDAD DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

Identical IoT device identification via hardware performance fingerprinting and Machine Learning

Identificación de dispositivos IoT idénticos mediante fingerprinting del rendimiento del hardware y Machine Learning

D. Pedro Miguel Sánchez Sánchez

2024

UNIVERSIDAD DE MURCIA
ESCUELA INTERNACIONAL DE DOCTORADO

TESIS DOCTORAL

Identical IoT device identification via hardware performance fingerprinting and Machine Learning

Identificación de dispositivos IoT idénticos mediante fingerprinting del rendimiento del hardware y Machine Learning

Autor:
Pedro Miguel Sánchez Sánchez

Directores:
Dr. **Alberto Huertas Celdrán**, *Ph.D.*
Dr. **Gregorio Martínez Pérez**, *Ph.D.*

Murcia, 2024

UNIVERSIDAD DE MURCIA

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA EN MODALIDAD DE COMPENDIO O ARTÍCULOS PARA OBTENER EL TÍTULO DE DOCTOR

Aprobado por la Comisión General de Doctorado el 19-10-2022

D./Dña. Pedro Miguel Sánchez Sánchez

doctorando del Programa de Doctorado en

Informática

de la Escuela Internacional de Doctorado de la Universidad Murcia, como autor/a de la tesis presentada para la obtención del título de Doctor y titulada:

Identical IoT device identification via hardware performance fingerprinting and Machine Learning / Identificación de dispositivos IoT idénticos mediante fingerprinting del rendimiento del hardware y Machine Learning

y dirigida por,

D./Dña. Alberto Huertas Celdrán

D./Dña. Gregorio Martínez Pérez

D./Dña.

DECLARO QUE:

La tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la Ley de Propiedad Intelectual (R.D. legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, modificado por la Ley 2/2019, de 1 de marzo, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita, cuando se han utilizado sus resultados o publicaciones.

Además, al haber sido autorizada como compendio de publicaciones o, tal y como prevé el artículo 29.8 del reglamento, cuenta con:

- *La aceptación por escrito de los coautores de las publicaciones de que el doctorando las presente como parte de la tesis.*
- *En su caso, la renuncia por escrito de los coautores no doctores de dichos trabajos a presentarlos como parte de otras tesis doctorales en la Universidad de Murcia o en cualquier otra universidad.*

Del mismo modo, asumo ante la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad del contenido de la tesis presentada, en caso de plagio, de conformidad con el ordenamiento jurídico vigente.

En Murcia, a 21 de Diciembre de 2023

Fdo.: Pedro Miguel Sánchez Sánchez

Esta DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD debe ser insertada en la primera página de la tesis presentada para la obtención del título de Doctor.

Información básica sobre protección de sus datos personales aportados	
Responsable:	Universidad de Murcia. Avenida teniente Flomesta, 5. Edificio de la Convalecencia. 30003; Murcia. Delegado de Protección de Datos: dpd@um.es
Legitimación:	La Universidad de Murcia se encuentra legitimada para el tratamiento de sus datos por ser necesario para el cumplimiento de una obligación legal aplicable al responsable del tratamiento. art. 6.1.c) del Reglamento General de Protección de Datos
Finalidad:	Gestionar su declaración de autoría y originalidad
Destinatarios:	No se prevén comunicaciones de datos
Derechos:	Los interesados pueden ejercer sus derechos de acceso, rectificación, cancelación, oposición, limitación del tratamiento, olvido y portabilidad a través del procedimiento establecido a tal efecto en el Registro Electrónico o mediante la presentación de la correspondiente solicitud en las Oficinas de Asistencia en Materia de Registro de la Universidad de Murcia

Para ti mamá,
te quiero

The following PhD Thesis is a compilation of the next published articles, being the PhD student the main author in all of them:

- Pedro Miguel Sánchez Sánchez, José María Jorquera Valero, Alberto Huertas Celdrán, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez. "**A Survey on Device Behavior Fingerprinting: Data Sources, Techniques, Application Scenarios, and Datasets.**", *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1048-1077, 2021.
  DOI: 10.1109/COMST.2021.3064259
  JIF 2021: 33.84 (D1)

- Pedro Miguel Sánchez Sánchez, José María Jorquera Valero, Alberto Huertas Celdrán, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez. "**A methodology to identify identical single-board computers based on hardware behavior fingerprinting.**", *Journal of Network and Computer Applications*, vol. 212, pp. 103579, 2023.
  DOI: 10.1016/j.jnca.2022.103579
  JIF 2022: 8.7 (D1)

- Pedro Miguel Sánchez Sánchez, José María Jorquera Valero, Alberto Huertas Celdrán, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez. "**LwHBench: A low-level hardware component benchmark and dataset for Single Board Computers.**", *Internet of Things*, vol. 22, pp. 100764, 2023.
  DOI: 10.1016/j.iot.2023.100764
  JIF 2022: 5.9 (Q1)

- Pedro Miguel Sánchez Sánchez, Alberto Huertas Celdrán, Gérôme Bovet, Gregorio Martínez Pérez, Burkhard Stiller. "**SpecForce: A Framework to Secure IoT Spectrum Sensors in the Internet of Battlefield Things.**", *IEEE Communications Magazine*, vol. 61, no. 5, pp. 174-180, 2023.
  DOI: 10.1109/MCOM.001.2200349
  JIF 2022: 11.2 (D1)

- Pedro Miguel Sánchez Sánchez, Alberto Huertas Celdrán, Gérôme Bovet, Gregorio Martínez Pérez. "**Adversarial attacks and defenses on ML- and hardware-based IoT device fingerprinting and identification.**", *Future Generation Computer Systems*, vol. 152, pp. 30-42, 2024.
  DOI: 10.1016/j.future.2023.10.011
  JIF 2022: 7.5 (D1)

- Pedro Miguel Sánchez Sánchez, Alberto Huertas Celdrán, Gérôme Bovet, Gregorio Martínez Pérez. "**Single-board Device Individual Authentication based on Hardware Performance and Autoencoder Transformer Models.**", *Computers & Security*, vol. 137, 103596, 2024.
  DOI: 10.1016/j.cose.2023.103596
  JIF 2022: 5.6 (Q2)

Contents

Acknowledgements

It has been six years since Chema and I, while looking for a Bachelor's Thesis on cybersecurity, stumbled upon Gregorio's office by chance. Unbeknownst to me, my life was about to change with that small decision. From the Bachelor's Thesis, I moved on to the Master's Thesis, and after giving it some thought, I decided to continue with the PhD thesis. I believe that has been one of the most accurate decisions of my life, both for the "academic" growth and for the personal growth I take away from these years. Therefore, I can only start this doctoral thesis by thanking my advisors, Alberto and Gregorio, for all the support, help, and work they have dedicated to me during these years. There have been many hours spent working on articles and figuring out how to guide this research, especially at the beginning of the thesis when not everything was going as we expected. But there have been even more good hours during these years, both on the trips we have been able to go on and in the small details of the day-to-day work.

I extend these thanks to the rest of the members of the research group that has been formed over these years, the CyberDataLab (CDL). I am proud to see how the group has evolved over the years and how it will continue to advance in the following years, thanks to the effort of those of us who form it. I would especially like to highlight Manuel because, although he has not been a tutor, he has always been there throughout this process, and Chema, for all the days of work since the early years of the Bachelor's degree.

I also want to thank Gérôme, and the rest of the team at the CyberDefence Campus in armasuisse S&T, for the opportunity to carry out this thesis in collaboration with their team. As well as for the good treatment during the two stays I have had the opportunity to do in Thun.

On a personal level, I want to first thank my father, Ginés, for his unconditional love, support, and sacrifices. Also to my mother, Salvadora, who left us just when this academic journey began, but has been present every day. I know you would be as proud of me as I am to have been your son.

Finally, I want to thank all my friends for your friendship, for the moments of disconnection, and for all the times I have been able to talk your ears off about my thesis during these years. You are an important pillar in my life.

Agradecimientos

Hace ya seis años desde que, buscando un TFG sobre ciberseguridad, Chema y yo dimos de casualidad con el despacho de Gregorio. Sin saberlo, mi vida estaba a punto de cambiar con esa pequeña decisión. Del TFG pasé al TFM y después de darle unas cuantas vueltas, decidí seguir con la tesis doctoral. Creo que esa ha sido una de las decisiones más acertadas de mi vida, tanto por el crecimiento "académico" como por el crecimiento personal que me llevo de estos años. Por lo tanto, solo puedo empezar esta tesis doctoral agradeciendo a mis directores, Alberto y Gregorio, por todo el apoyo, ayuda y trabajo que han dedicado a mí durante estos años. Han sido muchas las horas trabajando en los artículos y viendo cómo encaminar esta investigación, sobre todo al principio de la tesis cuando no todo salía como esperábamos. Pero más han sido los buenos momentos durante estos años, tanto en los viajes a los que hemos podido ir como en los pequeños detalles del trabajo día a día.

Extender estos agradecimientos al resto de miembros del grupo de investigación que se ha creado durante estos años, el CyberDataLab (CDL). Es un orgullo ver como el grupo ha evolucionado estos años, y lo que va a seguir avanzando en los siguientes, gracias al esfuerzo de quienes lo formamos. En especial me gustaría remarcar a Manuel porque, aunque no ha sido director, siempre ha estado ahí en todo este proceso, y a Chema, por todos los días de trabajo desde los primeros años del grado.

También agradecer a Gérôme, y al resto del equipo del CyberDefence Campus en armasuisse S&T, por la oportunidad de realizar esta tesis en colaboración con su equipo. Así como por el buen trato durante las dos estancias que he tenido oportunidad de hacer en Thun.

A nivel personal, quiero dar las gracias en primer lugar a mi padre, Ginés, por su amor incondicional, apoyo y sacrificios. También a mi madre, Salvadora, que aunque nos dejó justo cuando empezaba todo este trayecto académico, siempre ha estado presente cada día. Sé que estarías tan orgullosa de mí como yo lo estoy de haber sido tu hijo.

Por último, agradecer a todos mis amigos por vuestra amistad, por los momentos de desconexión y por todas las veces que os he podido calentar la cabeza con mi tesis durante estos años. Sois un pilar importante en mi vida.

Abstract

I Introduction and motivation

By 2025, nearly 64 billion Internet-of-Things (IoT) devices are expected to be connected across various cutting-edge environments, such as Smart Cities, Industry 4.0, and crowdsensing [1]. The growth of IoT promises not only to enhance service quality and accessibility but also to revolutionize the user's experience, as it fosters intelligent ecosystems that significantly reduce human intervention, thus streamlining processes, cutting costs, and mitigating errors. However, the unique objectives of these environments also increase the complexity of optimizing device and service performance.

From a scenario perspective, the vast array of IoT devices employed nowadays includes Single-Board Computer (SBC) devices like Raspberry Pi (RPi), which have gained popularity due to their flexibility, affordability, extensive support, and available peripherals [2]. However, the connectivity and resource constraints of SBCs, and IoT devices, create numerous cybersecurity concerns for diverse platforms [3]. A significant problem is the presence of unauthorized devices with identical hardware and software configurations as authorized nodes, launching attacks impacting application areas such as Industry 4.0 [4], smartphones [5], or Internet of Battlefield Things (IoBT) [6]. These malicious devices can be present as a consequence of various cybersecurity threats [7], including i) device spoofing, where an attacker replaces a legitimate device with a malicious one using the same identity; ii) unauthorized device deployment, involving the installation of a new device with an unregistered identity; and iii) Sybil attack, where a malicious device uses multiple identities to mimic numerous devices. Consequently, other threats like sensitive information leakage, data poisoning, and privilege escalation and lateral movements may emerge from spoofed devices.

To solve these arising cybersecurity problems, behavioral data science has expanded from studying human behaviors [8] to modeling device behaviors [9], with a focus on creating device behavior patterns (*fingerprints*) to optimize performance and detect potential issues early [10]. Two primary IoT application scenarios for constructing device fingerprints are i) device identification and authentication at different granularity levels [11], and ii) detecting cyberattacks, malfunction, or misbehavior [12, 13]. Various studies have applied device fingerprinting to both scenarios [14, 15]. In the identification scenario, behavioral data science has significantly enhanced device identification capabilities, transcending the constraints associated with conventional methodologies that predominantly rely on the utilization of names, identifiers, labels, or tags for device recognition [16]. A critical short-

coming of traditional strategies is their susceptibility to alterations and duplications. These conventional techniques often lack the dynamism to adapt to the rapidly evolving landscape of IoT scenarios, where many devices are being deployed with high mobility. These issues are especially relevant in scenarios with an exponential increase in the quantity of devices, such as smart industries or agriculture. Therefore, device identification based on behavior fingerprinting has significantly improved upon traditional solutions by focusing on type [11], model [17], and individual [18] granularity levels. On the other hand, detecting misbehavior or malfunction due to cybersecurity issues has seen the rise of device fingerprinting as a promising solution. Numerous works create "normal" behavioral fingerprints to detect changes caused by issues such as cyberattacks, malware execution, or device malfunctioning [12, 19].

At the individual identification granularity level, hardware behavior fingerprinting stands out as a promising avenue for uniquely identifying identical devices, a necessity in today's interconnected technological landscape. Despite its potential, this domain is still burgeoning, characterized by open challenges and a lack of dedicated research specifically targeting the identification of identical single-board devices [20]. The complexity of this task is amplified by the inherent similarities between such devices, necessitating innovative and precise identification methodologies. In the broader context, for devices that are not constrained by components and resources, the existing literature advocates for the adoption of *hardware behavioral fingerprinting* [21]. This technique is instrumental in discerning minor performance variances that are a byproduct of manufacturing imperfections [22]. By meticulously analyzing these subtle differences, hardware behavioral fingerprinting provides a granular level of device characterization, paving the way for accurate device identification. However, it is important to distinguish between identification and authentication in this context. While identification involves recognizing a device from a set based on its unique characteristics, authentication goes a step further by verifying the legitimacy of the device. Authentication addresses the question of whether a device is who it claims to be, offering a layer of security that mere identification does not.

The hardware behavior analysis for device identification has been partially performed in the literature but without following a clear methodology on the steps to be done in order to achieve a successful solution [20]. Therefore, there exists a necessity for this specific methodology, rooted in the critical need to bolster cybersecurity measures in network infrastructures that incorporate identical single-board computer devices, such as Raspberry Pi. The set of these devices working in a coordinated manner is what allows to offer IoT services based on crowdsourcing or joint data collection/processing. These devices, often deployed extensively and in settings where resources are scarce, are vulnerable to a myriad of security threats [23, 24] that affect the trustworthiness of the offered services. These range from attempts at malicious device impersonation to the introduction of unauthorized devices into a network. Ensuring the ability to identify and authenticate devices precisely becomes a non-negotiable requirement, pivotal for maintaining the integrity and resilience of the service. Therefore, a consistent security level is essential in order to have trustworthy services. Implementing hardware behavior fingerprinting in these contexts demands a level of precision and innovation that transcends conventional identification methods. The challenges stem from the subtle differences between identical devices and the constraints imposed by their limited resources [25]. Addressing these challenges necessitates a concerted research effort, aimed at unraveling the complexities of hardware behavior fingerprinting, selecting the most promising hardware performance characteristics, developing robust identification algorithms, and establishing best practices for practical application.

Moreover, having a complete identification solution based on hardware is critical for

IoT security. However, in real-world scenarios, individual identification solutions do not work isolated from other cybersecurity applications. They should be integrated with other network and behavior solutions that aim to cover heterogeneous cybersecurity issues such as malware. In this context, the integration of hardware behavior fingerprinting for single-board devices becomes a strategic component of a larger security framework. Uniquely identifying and verifying each device enables a solid foundation for secure communication and data integrity. However, other tools are still required to achieve a more complete security level. This level of security is not just about protecting information; it is about ensuring the operational effectiveness and success of IoT endeavors.

To bolster the effectiveness of hardware behavior fingerprinting for identical single-board devices, it is imperative to identify the most relevant data sources to solve this issue. Then, it is required to have a comprehensive benchmark and a well-curated dataset in place [26]. These resources serve as critical tools in the verification and validation of the methodology, ensuring its accuracy, reliability, and applicability in real-world scenarios. A precise benchmark is necessary to systematically evaluate the performance of the devices, providing a standardized measure that can be used to discern the subtle variances in hardware behavior [27]. This becomes particularly crucial when dealing with identical devices, where the differences are minuscule yet significant for accurate identification. However, there are no benchmarking applications available in the literature for low-level hardware behavior fingerprinting [26]. In tandem with a robust benchmark, a rich dataset plays a vital role in the process. It acts as a repository of hardware device behavior profiles, capturing the intricacies and unique characteristics of each device. This dataset becomes a reference point, aiding in the training of algorithms and serving as a benchmark for validation. The combination of a comprehensive benchmark and a detailed dataset ensures a holistic approach to device identification, fortifying the hardware behavior fingerprinting methodology. The motivation for establishing such rigorous verification tools stems from the evolving needs of the IoT and Edge computing paradigms, where devices are increasingly interconnected, and environment integrity is paramount. In these scenarios, the ability to precisely identify and authenticate devices is not just a security measure; it is a necessity for ensuring the seamless operation of the network and the trustworthiness of the data being exchanged. By investing in the development of precise benchmarks and comprehensive datasets, the hardware behavior fingerprinting methodology can be empowered, enhancing its precision and reliability. This, in turn, paves the way for a future where identical single-board devices can be seamlessly integrated into complex networks, operating securely and efficiently, and contributing to the robustness of the digital infrastructure.

Upon acquiring the fingerprints, a riveting domain of research emerges, centered on employing optimal techniques for processing and evaluating them. While statistical methods have held a dominant position in this field for decades, the advent of Artificial Intelligence (AI), specifically Machine Learning (ML) and Deep Learning (DL), has precipitated a paradigm shift, gaining the most prominence in the solutions being developed and deployed today [28]. The complexity and diversity of IoT device behaviors necessitate sophisticated DL models capable of capturing intricate patterns and variances. Techniques like Convolutional Neural Networks (CNNs) are being adapted to discern spatial relationships in hardware fingerprints [29], while Recurrent Neural Networks (RNNs), including their more sophisticated variant, Long Short-Term Memory (LSTM) networks, are being deployed for temporal data analysis, crucial for understanding device behavior over time [17]. Autoencoders have also carved a niche in this sector, enabling dimensionality reduction for high-volume fingerprint data and anomaly detection by reconstructing normal device behavior and highlighting deviations [30]. Additionally, recent advances in Graph Neural

Networks (GNNs) allow for the incorporation of topological data, facilitating the modeling of complex relationships within networks of interconnected IoT devices. Finally, attention-based transformer models are achieving extraordinary success in language-focused tools, such as ChatGPT [31], and can also be applied to other time series data. However, the applicability of all these modern techniques is conditioned by the lack of available datasets, as they require the existence of exhaustive datasets capable of training the required models [32]. Therefore, after having enough data, the next challenge is to explore the available ML/DL methods to find which one could provide the best performance in individual device identification. Here, new network architectures can be created to better model device behavior and enhance identification and authentication capabilities.

Besides, a fully functional and trustworthy solution should consider the possible security issues intrinsic to the hardware-based individual device identification [33]. Adversarial attacks arise as one of the main threats targeting identification solutions in IoT. These sophisticated attacks present a formidable challenge, as they are specifically crafted to manipulate or evade the security mechanisms in place, potentially leading to unauthorized access, data breaches, and compromised network integrity [34, 35]. Adversarial attacks in the context of single-board device identification exploit the mechanisms designed to ensure device authenticity and network security. By subtly altering device behavior or mimicking the characteristics of legitimate devices, adversaries can deceive identification systems, resulting in misclassification or false acceptance of rogue devices. This not only undermines the trustworthiness of the network but also opens the door to further malicious activities, jeopardizing the confidentiality, integrity, and availability of data and services. Delving deeper into the nature of these adversarial attacks, there is a rising trend in context-based and ML/DL-based evasion tactics. Context-based attacks cleverly manipulate the environmental conditions or operational context of devices, aiming to distort the data used for identification and thereby mislead the system [36]. These attacks are particularly insidious as they exploit the natural variability in device behavior due to changes in external factors, making them harder to detect and counteract. On the other hand, ML/DL-based evasion attacks represent a sophisticated and calculated assault on single-board device identification systems. Adversaries employing these techniques utilize advanced ML models to learn the patterns and characteristics of legitimate devices. Armed with this knowledge, they craft adversarial samples that closely mimic authentic devices, effectively blurring the lines between legitimate and rogue devices [37]. These samples are then used to probe and deceive the identification system, leading to erroneous classifications and potentially granting unauthorized access to the network. These attacks exploit the inherent complexities and variabilities in device behavior and the advanced capabilities of ML/DL models to deceive and compromise network security. Addressing these challenges requires a comprehensive and adaptive security strategy capable of anticipating, detecting, and mitigating the myriad of threats posed by adversarial attacks.

Apart from all the work to be done in the individual identification of IoT devices context, the authentication problem is an open challenge on its own due to its more complex requirements [38]. While hardware behavioral fingerprinting offers a robust foundation for identification, extending this approach to include authentication mechanisms is vital for ensuring a higher degree of security in interconnected systems. However, the devices from the same model can exhibit very similar or even overlapping behavioral characteristics due to standardized production processes. This similarity in hardware behavior makes it challenging to distinguish between legitimate and unauthorized devices based solely on hardware behavioral data, as data distributions merge together. Therefore, authentication requires a much finer granularity of data analysis. Here, traditional processing

approaches and ML/DL techniques have not achieved remarkable results. However, new ML/DL algorithms with advanced pattern recognition capabilities, such as attention-based transformers, could improve authentication performance based on hardware performance behavior.

As a summary of the open challenges, the literature suggests that while hardware behavior fingerprinting presents a promising solution, the field is still nascent, with considerable research gaps in methodology and practice for single-board devices. The individual identification of these devices requires innovative techniques to distinguish subtle manufacturing differences, as well as precise benchmarks and extensive datasets for validating new algorithms. The evolution of AI, particularly ML and DL, is shifting the paradigm for processing and evaluating device fingerprints, and new techniques are required in this area. Besides, there remains a critical need for the integration of hardware fingerprinting with broader network security measures to address various cybersecurity threats. Next, adversarial attacks pose a formidable risk, leveraging context-based tactics or ML/DL models to craft samples that mimic legitimate devices, necessitating a robust, adaptive security strategy that encompasses the comprehensive identification and mitigation of such threats. Finally, it is vital to explore the device authentication problem leveraging more complex ML/DL solutions. Authentication solutions have to generate advanced patterns in the fingerprint, generating unique models for each device based on its intrinsic characteristics.

Based on the previous considerations, this PhD Thesis explores the feasibility of individual device identification and authentication based on hardware performance behavior fingerprinting. It should investigate the methodology definition, its application in real frameworks and tools, their integration in real-world devices and scenarios, and the associated security threat analysis and mitigation. In this sense, several research questions arose from the previous challenges, guiding the research process of this PhD Thesis, and are presented as follows:

- RQ1: What is the current status of device behavior fingerprinting solutions applied in individual device identification and which data sources, techniques, application scenarios and datasets are present in the literature?

- RQ2: Which methodology should be followed to uniquely identify IoT devices in a scalable manner while leveraging on hardware behavior and ML/DL techniques?

- RQ3: Which hardware metrics can be extensively collected for measuring hardware performance and generating the required datasets for late behavior fingerprinting?

- RQ4: How can individual device identification be integrated with other behavior-based cybersecurity solutions deployed in real-world scenarios?

- RQ5: Which ML/DL techniques can achieve the best performance for individual device identification?

- RQ6: How can the resilience of hardware behavior-based individual device identification models be improved against context- and ML/DL-focused adversarial attacks?

- RQ7: Can attention-based ML/DL techniques, such as transformers, enable the individual identification of devices following an anomaly detection approach? Which resources are required?

II Objectives

The main goal of this PhD thesis is to advance the field of device behavior fingerprinting for improving the identification and security of IoT devices, with a particular focus on single-board computers and resource-constrained systems. The research aims to develop novel methodologies and frameworks for individual device identification, leveraging ML and DL techniques. By examining various application scenarios, including Smart Cities, Industry 4.0, and the Internet of Battlefield Things, the thesis aims to address the increasing cybersecurity threats, such as unauthorized device deployment, and other issues emerging due to the exponential growth of interconnected devices in the network-based computing world.

The thesis explores different aspects of device behavior fingerprinting, including data sources, techniques, application scenarios, and datasets. The research focuses on solving the unique problems faced when defining methodologies for identifying identical single-board computers based on hardware behavior fingerprinting and ML. Some of these open issues are the data availability and the solution integration in real-world scenarios. Therefore, they are explored as part of the research objectives. The research also investigates adversarial attacks and defenses in IoT fingerprinting, aiming to develop resilient architectures. Lastly, the work explores the authentication problem, where each device should be recognized without considering others and, therefore, more complex ML models are necessary. From the goal of covering the previous aspects, several specific objectives are derived as subsequently presented, indicating the research questions related to them:

O.1. Analyze the current state of the art regarding device behavior fingerprinting for individual identification through comprehensive review, analysis, and comparison of data sources, techniques, applications, and datasets to guide future research and solutions (RQ1).

O.2. Identify existing gaps in the literature in individual device identification based on hardware behavior fingerprinting and the required steps to cover them (RQ1).

O.3. Address single-board device identification challenges with a novel methodology leveraging hardware behavioral fingerprinting, ML/DL techniques, and essential properties for improved accuracy and robustness (RQ2).

O.4. Develop a low-level hardware benchmarking solution for Single-Board Computers to enable Edge-based AI solutions and versatile device management through extensive performance datasets (RQ3).

O.5. Integrate the single-board device identification solution into a security framework using behavioral fingerprinting and ML/DL techniques to accurately detect diverse cyber-attacks (RQ4).

O.6. Find the best ML/DL-based techniques for individual identification, iterating over the available datasets to achieve the best performance (RQ5).

O.7. Investigate the impact of context and ML/DL-focused attacks against hardware behavior-based device identification solutions leveraging ML/DL models (RQ6).

O.8. Implement and evaluate defense mechanisms to strengthen the solution resilience against adversarial attacks while maintaining its performance in the context of hardware behavior-based device identification (RQ6).

O.9. Explore the individual authentication problem to verify the capabilities of attention-based anomaly detection ML/DL models to detect advanced patterns in hardware behavior data (RQ7).

III Methodology

This PhD Thesis was conducted following a scientific approach based on the continuous study of the state of the art and the analysis of the results obtained during the different stages of the research. This thesis is defined as a set of six papers published in high-impact journals indexed in the Journal Citation Reports (JCR).

The first step was to solve RQ1 and provide a complete view of device behavior fingerprinting. A broad exploration of the existing literature was undertaken to provide a comprehensive understanding of the current status of device behavior fingerprinting solutions in cybersecurity. This exploration encompassed a variety of academic publications, spanning across journals and conference proceedings, ensuring a wide coverage of the topic. The gathered literature was meticulously categorized and analyzed based on specific criteria, such as the data sources used for fingerprinting, the techniques employed, the application scenarios addressed, and the datasets utilized. This granular categorization facilitated the identification of prevalent trends, commonly used methods, and potential gaps within the current body of knowledge. The analysis provided a nuanced understanding of how device behavior fingerprinting is being applied across different domains in cybersecurity, highlighting its versatility and the variety of ways it can be implemented. This extensive review culminated in a holistic view of the field, offering valuable insights and a solid foundation for future research endeavors in device behavior fingerprinting. All these considerations resulted in the first publication of this PhD Thesis, presented in the first chapter (A Survey on Device Behavior Fingerprinting: Data Sources, Techniques, Application Scenarios, and Datasets (Article 1–`IEEE_COMST`)), which covered the first two Objectives of the thesis.

After performing the state-of-the-art analysis and identifying the current gaps in IoT individual identification solutions, the focus moved to solve RQ2 and address Objective 3, which addresses the unique IoT device identification problem. In order to uniquely identify IoT devices based on their hardware behavior while leveraging ML and DL techniques, a structured approach was employed. The process commenced with the collection of a diverse and extensive set of hardware behavior data from a variety of RPi models. This data served as the foundation for the subsequent analysis. Following the data collection phase, pre-processing techniques were applied to refine and prepare the data for the ML and DL models. Various ML/DL models were then selected and trained on the pre-processed data, resulting in the development of identification algorithms. The performance of these algorithms was rigorously evaluated using common ML/DL classification metrics and real SBC devices, ensuring their accuracy and effectiveness in real-world scenarios. Afterward, the methodology was compared with other approaches existing in the literature, although they did not present a structured set of steps in order to develop an identification solution. In this comparison, the methodology performance improvement was contrasted practically. This comprehensive approach facilitated a clear and practical methodology for the unique identification of IoT devices based on their hardware behavior, leveraging the latest advancements in ML and DL. The proposal and validation of the individual device identification methodology resulted in the second chapter (A methodology to identify identical single-board computers based on hardware behavior fingerprinting (Article 2–`JNCA`)).

At this point, a limitation in the research line arose regarding data availability and how to collect it. A specialized benchmarking tool and dataset were developed as part

of this PhD thesis and utilized to address the challenge of extensive data collection for measuring hardware performance and generating the required datasets for later behavior fingerprinting. This tool was meticulously designed to measure and record the performance of various hardware components across a range of IoT devices, ensuring a standardized and comprehensive data collection process. The tool was executed then in some SBC devices to extract a complete and realistic dataset. The collected data was then subjected to rigorous validation and quality assurance processes, verifying its accuracy and reliability. In addition to the data collection, clear guidelines were provided on how to structure and store the data, facilitating its integration into comprehensive datasets. This meticulous approach ensured that the data collected was not only extensive but also of high quality, providing a solid foundation for subsequent behavior fingerprinting applications and analyses. The benchmarking application, together with the collected dataset, are publicly available for other researchers in the area. This work resulted in the third chapter of this thesis (LwHBench: A low-level hardware component benchmark and dataset for Single Board Computers (Article 3–IoT)), answering RQ3 and completing Objective 4.

A holistic and interoperable framework was developed once the individual identification viability was verified with the exhaustive dataset collected using the benchmark application. It sought to integrate individual device identification with other behavior-based cybersecurity solutions. This framework was based on an extensive analysis of device behavior fingerprinting and existing cybersecurity solutions, ensuring a comprehensive understanding of the necessary components and processes. The designed framework facilitated seamless integration, allowing for easy sharing and utilization of device fingerprints and behavior profiles across different security solutions. To achieve this, the framework was designed to be highly modular and scalable, allowing for easy integration with a variety of behavior-based cybersecurity solutions. The identification framework could work with various security systems, including intrusion detection systems, security information and event management solutions, and advanced threat protection tools. It could enhance their effectiveness and add an extra layer of security. To validate the framework suitability, extensive simulations, and real-world tests were conducted, evaluating its functionality in diverse scenarios and against various threat models. The real-world validation was performed considering an IoBT scenario based on the ElectroSense platform [39]. Then, a kernel event and syscall monitoring solution was integrated together with the individual identification solution to provide a complete security approach for the platform. The results of these evaluations confirmed the framework effectiveness, demonstrating its capability to integrate individual device identification with other behavior-based cybersecurity solutions and enhance the security posture of the network. This work solved RQ4 and Objective 5, available in the thesis's fourth chapter (SpecForce: A Framework to Secure IoT Spectrum Sensors in the Internet of Battlefield Things (Article 4–IEEE_COMMAG)).

The next work done in the PhD Thesis, presented in the fifth chapter of this document (Adversarial attacks and defenses on ML- and hardware-based IoT device fingerprinting and identification (Article 5–FGCS)) and aligned with the fifth and sixth research questions (Objectives 6, 7, and 8), tackled the challenge of security threats. It focused on finding the best ML/DL technique for identification and then enhancing its resilience for reliable hardware behavior-based individual device identification. This involved a thorough analysis of potential adversarial attacks, emphasizing understanding the nature and impact of these sophisticated attacks. The presented threat model addresses context- and ML/DL-focused adversarial attacks, ensuring that the identification models remain reliable and secure even in the face of sophisticated threats. The impact of the attacks proposed in the literature was verified, demonstrating the solution vulnerability to adversarial attacks.

Then, countermeasures and mitigation strategies were formulated and implemented to fortify the identification models against adversarial manipulations. Adversarial training and knowledge distillation techniques [40] were incorporated to enhance model resilience against adversarial attacks.

The last work of this compendium worked on the seventh research question (RQ7) and Objective 9 (Single-board Device Individual Authentication based on Hardware Performance and Autoencoder Transformer Models (Article 6–COSE)). It dealt with the authentication problem instead of with identification. As explained in the Introduction, the main difference between these problems is that for authentication, only data from the device being authenticated can be employed in the training process and a higher granularity is necessary in the data processing. This fact makes much more difficult the authentication problem, as the distribution of hardware performance data on devices from the same model usually overlaps in a great proportion. The methodology followed to solve this problem involved time series processing to train Transformer-based autoencoder models [41], with each model tailored to a specific device. The data collection and preprocessing were similar to the one applied in the identification-focused works, only varying the size of the employed time window. Then, attention-based transformer models were designed to function as outlier detection mechanisms, identifying any deviations from the expected hardware behavior that would indicate an unauthorized device. This work placed a strong emphasis on the practical application of their methodology in real-world scenarios. Therefore, it conducted extensive testing and validation to ensure that the approach was not only theoretically sound but also effective in practice.

This thesis creates a holistic storyline, addressing the critical aspects of device behavior fingerprinting from foundational knowledge and practical identification techniques to data collection, integration with broader security solutions, and resilience against adversarial attacks. Finally, the authentication problem is also explored using modern transformer-based approaches. This comprehensive approach ensures a thorough understanding and robust application of device behavior fingerprinting in the realm of cybersecurity. This methodology allowed for meeting the objectives defined in the thesis, previously presented in Section II.

IV Results

The first publication of the PhD Thesis, presented in (Article 1–IEEE_COMST), offered a comprehensive study on device behavior fingerprinting, providing an extensive analysis and synthesis of solutions applied in the realm of cybersecurity, along with valuable insights and findings. The results highlighted a broad spectrum of data sources, techniques, application scenarios, and datasets prevalent in the current literature, underlining the multifaceted nature of device behavior fingerprinting. One of the key findings revolves around the diversity of data sources used for device behavior fingerprinting. The results indicated that a wide variety of data types are employed, ranging from network traffic and system logs to hardware-specific attributes. This diversity underscores the versatility of device fingerprinting solutions, demonstrating their applicability across different layers of the system and network architecture.

In terms of fingerprinting techniques, the study revealed a rich landscape of methodologies, each with its unique strengths and capabilities. The results show that there is no one-size-fits-all solution, with different techniques catering to specific requirements and scenarios. This variety ensures that practitioners and researchers have a plethora of options to choose from, enabling them to tailor their device fingerprinting solutions to meet the

specific needs of their application. When it comes to application scenarios, the document outlined a wide range of contexts in which device behavior fingerprinting is employed. From enhancing network security and detecting unauthorized devices, to facilitating device authentication and integrity verification, the applications are vast and varied. This highlighted the crucial role that device behavior fingerprinting plays in bolstering cybersecurity measures, providing an additional layer of security and trust in digital environments. The examination of datasets revealed that while there are numerous datasets available for research and development purposes, there is still a need for more comprehensive and standardized datasets. The results point out that the existing datasets vary significantly in terms of size, quality, and relevance, indicating a gap that needs to be addressed to further advance the field. The availability of high-quality, standardized datasets is paramount for the validation and benchmarking of device fingerprinting solutions, ensuring their reliability and effectiveness in real-world scenarios.

In the last section of the document, a profound and insightful analysis was presented, encapsulating the lessons learned, identifying prevailing trends, and highlighting the challenges faced in the domain of device behavior fingerprinting within cybersecurity. This section serves as a critical reflection on the current state of the field, offering guidance for future research and practical implementations. One of the main challenges identified is associated with the lack of solutions dealing with individual device identification, compared to other topics such as behavior-based malware detection or network security. Moreover, there is a lack of datasets available to enable the design of individual identification solutions. Another important challenge is related to the attacks directly focused on disrupting the effectiveness of behavior-based security solutions. Besides, an obvious imbalance between solutions for identification and authentication of devices is detected, possibly because it is a more complex problem to solve.

Following the main challenges found in the literature, the second publication (Article 2–JNCA) provided a detailed exploration and analysis of the steps required to uniquely identify IoT devices based on their hardware behavior, with a particular emphasis on leveraging ML and DL techniques. The results gleaned from this investigation offer valuable insights into the intricacies of device fingerprinting and the practicalities of implementing such methodologies in real-world scenarios.

The research identified essential properties for single-board device identification, including uniqueness, stability, diversity, scalability, efficiency, robustness, and security. A novel methodology was then introduced, relying on behavioral fingerprinting to identify identical single-board devices while meeting the aforementioned properties. This methodology utilizes the different built-in components of the system, along with ML/DL techniques, to compare the internal behavior of devices and detect variations that occurred during the manufacturing processes. A key outcome of the study was the identification and validation of specific hardware attributes that can be used as reliable indicators for device fingerprinting. These attributes, when analyzed and processed correctly, have been shown to provide a unique signature for each device, facilitating accurate and efficient identification. The results underscore the importance of selecting the right combination of hardware attributes, as this choice significantly impacts the effectiveness of the fingerprinting process.

The document also highlighted the critical role of hardware isolation in the device fingerprinting workflow. The results demonstrated that careful handling and preparation of the hardware attributes are paramount, as this ensures the integrity of the fingerprinting process and enhances the accuracy of device identification. Besides, the application of various data preprocessing techniques, including normalization and dimensionality reduction, has been shown to contribute positively to the outcome of the fingerprinting process.

The integration of ML and DL techniques into the device fingerprinting process has also been shown to introduce an element of adaptability and learning, enabling the system to evolve and improve over time. Deep learning techniques, including various configurations of neural networks, were explored for their ability to learn and model the device fingerprints directly from raw hardware data. ML and DL classifiers were among the architectures tested, chosen for their prowess in handling sequential and time-series data, which is prevalent in hardware behavior information. Additionally, the document addressed the challenge of model overfitting, especially pertinent when employing complex models like deep neural networks. Strategies such as cross-validation and regularization were applied, ensuring that the models generalize well to unseen data and maintain high performance when deployed in real-world settings. The methodology is validated in a real environment, consisting of 15 identical Raspberry Pi 4 Model B and 10 Raspberry Pi 3 Model B+ devices whose CPU and GPU performance was analyzed. The results showcase a 91.9% average True Positive Rate (TPR) with an XGBoost model, achieving identification for all devices by setting a 50% threshold in the evaluation process. Furthermore, the proposed methodology was engaged in a critical discussion, comparing the proposed solution with related work, highlighting the fingerprint properties not met by other solutions, and providing valuable lessons learned and limitations of the presented methodology.

The main result in the third publication of the thesis (Article 3–IoT) was the development of a low-level hardware benchmarking application tailored for SBCs, addressing the need for lower-level benchmarking applications and datasets in the realm of IoT identification. The benchmark was named *LwHBench* [42] and focuses on measuring the performance of CPU, GPU, Memory, and Storage, while taking into account the component constraints inherent in SBCs. The application has been specifically implemented for Raspberry Pi devices. It was run for 100 days on a set of 45 devices to generate an extensive dataset containing 2386126 vectors in +4GB of data. This dataset paves the way for the application of AI techniques in scenarios where performance data can aid in the device management process. To showcase the inter-scenario capability of the dataset, the document also presented a series of AI-enabled use cases related to device identification and the impact of context on performance. In a practical setup, the benchmark application was adapted and applied to a scenario involving three RockPro64 devices, demonstrating its versatility and applicability in real-world settings.

In the fourth publication of the thesis (Article 4–IEEE_COMMAG), the research delves into the emerging and highly dynamic field of the Internet of Battlefield Things (IoBT). It focused particularly on the pivotal role of wireless communications within this sphere. In this intricate battlefield scenario, a myriad of devices, ranging from soldiers to diverse military equipment, interact in real time, exchanging information wirelessly and forming a complex network of interconnected entities. The proposed scenario delves into three primary use cases: IoT device identification, malware detection, and Spectrum Sensing Data Falsification (SSDF) attack detection.

To solve these use cases, an IoT behavior fingerprinting framework was introduced, namely *SpecForce*, which was meticulously designed to enhance the security of IoBT spectrum sensors, crucial components in monitoring the frequency spectrum, transmitting over unoccupied bands, intercepting enemy transmissions, and decoding valuable information. In this real-world scenario, individual device identification is essential to avoid possible attacks based on identity manipulations. *SpecForce* stands out as a robust solution, employing device behavioral fingerprinting alongside ML/DL techniques. The framework was adept at considering heterogeneous data sources, enhancing its capability to detect and mitigate a wide array of cyber threats effectively. The emphasis was placed on ensuring

the integrity and reliability of communications within the battlefield scenario, a critical aspect considering the spectrum scarcity and the burgeoning number of IoBT devices. The framework included an AI-based cybersecurity module that employs ML/DL classification algorithms for identifying different IoBT spectrum sensors based on RPi devices. The document provides a comprehensive analysis of various ML/DL models for classification. The results indicate that Random Forest and XGBoost are the best-performing models, achieving over 91% TPR. The document also discusses a use case demonstrating the capability of the system to uniquely identify 25 IoBT spectrum sensors, addressing identity-focused attacks and enhancing security.

As commented before, *SpecForce* was equipped with other cybersecurity approaches using kernel event and syscall behavior monitoring to detect higher-level cyberattacks. In the context of SSDF attack detection, the framework allows the syscall monitoring of IoBT spectrum sensors, aiming to detect various SSDF attacks. System calls are processed to generate feature vectors that model the spectrum sensing activities, with anomaly detection algorithms employed to distinguish between normal and malicious behaviors. The results showcase a high performance in recognizing normal behavior, with over 99% True Negative Rate (TNR), and a commendable TPR of over 92% for SSDF attack detection. Besides, regarding heterogeneous malware detection (botnets, backdoors, etc.), high performance was achieved by monitoring kernel events combined with ML/DL anomaly detection, with a 90% TPR and a 96% TNR.

In the context of adversarial attacks, the next publication of the PhD Thesis (Article 5–FGCS) shed light on the potential vulnerabilities and threats that can compromise the integrity of device fingerprinting and identification mechanisms. It discusses various attack vectors, illustrating how malicious entities could manipulate or bypass hardware-based identification mechanisms to achieve their nefarious goals. The paper underscores the need for comprehensive defense strategies capable of mitigating the risks associated with adversarial attacks, ensuring the robustness of device identification processes. The first main result of this work was the improvement in the identification results achieved in previous works. Using time series approaches combined with DL models, identification results were increased to +0.96 average TPR with a minimum 0.80 TPR in the 45 RPi devices used for validation by leveraging an LSTM+1D-CNN combined model. Regarding adversarial attacks, both context- and ML/DL-focused attacks are applied to evaluate the robustness of the device identification model. A specific mention is made of a temperature-based context attack, which, interestingly, was found to be ineffective in disrupting the device identification process, as the hardware isolation during data collection was already considering context impact mitigation. However, the document does acknowledge the success of certain state-of-the-art ML/DL evasion attacks, such as BIM, MIM, and JSMA.

On the defense side, the document provides an exhaustive exploration of various strategies and methodologies aimed at protecting IoT devices from adversarial threats. It delves into ML and context-based approaches, evaluating their effectiveness in enhancing the security and reliability of device fingerprinting and identification. Context-based attacks focused on temperature are ineffective due to the hardware stability and isolation measures taken during data collection. Regarding defenses on ML/DL evasion attacks, knowledge distillation and adversarial training are applied to reduce the impact of the attacks. The results highlight the critical role of these defense mechanisms in maintaining the integrity of IoT ecosystems, ensuring that devices are accurately identified and malicious entities are thwarted. In terms of performance, the success ratio of attacks was reduced from 0.88 to 0.17 in the worst-case scenario without causing a substantial degradation in performance. Finally, various security metrics are used to assess the resilience of neural networks

against adversarial perturbations and input variations. Metrics such as CLEVER score, Loss sensitivity, and Empirical robustness [40] are discussed, providing insights into how the robustness of ML models can be quantified and evaluated.

The last publication of the PhD thesis (Article 6–COSE) focused on the individual authentication problem. It proposed an authentication framework that utilized hardware performance data and transformer-based autoencoder models. The framework design is supported by a threat model that outlines the security challenges encountered in implementing hardware-based authentication in IoT contexts. As in previous works, key hardware components, such as CPU, GPU, RAM, and storage, were monitored for fingerprint data collection. These fingerprints were then utilized as time series data, applying time windows from 10 to 100 values. The generated time series are then used to train transformer models for outlier detection, tailored to each individual device, thereby aiming to represent and authenticate it accurately. The framework effectiveness was further demonstrated through its application in a spectrum crowdsensing system using Raspberry Pi devices. Transformer models were compared with LSTM and 1D-CNN approaches in terms of performance. Here, in a series of rigorous experiments involving 45 devices for validation, each device transformer model proved capable of accurately authenticating it. In contrast, other approaches were not capable of uniquely authenticating all the devices. The approach achieved an impressive average TPR of 0.74 ± 0.13 and maintained an average maximum FPR of 0.06 ± 0.09, underscoring its potential to significantly enhance authentication, security, and trustworthiness in crowdsensing applications. Moreover, the resource usage of the different approaches tested was also analyzed, confirming one of the main drawbacks of transformer models; this was the ML/DL technique using the most resources in terms of time and memory.

V Conclusions and future work

This thesis provides an in-depth and comprehensive examination of the field of device behavior fingerprinting, with a specific lens focused on its application within the realm of cybersecurity, and a nuanced emphasis on single-board and IoT device identification and authentication. The research begins with an extensive and systematic review of the current landscape, capturing the breadth and depth of device behavior fingerprinting solutions that have been explored and implemented within the cybersecurity domain. This initial phase of exploration serves to lay a robust foundation of knowledge, unraveling the complexities of various data sources, fingerprinting techniques, application scenarios, and the datasets that are prevalent within the academic and practical spheres of this field. From this exploration, a novel set of literature lessons and trends is derived, giving a holistic view of previous works. However, the main novelty from a research perspective is the list of challenges identified in the literature, which were not defined before and paved the way for future research.

As the storyline progresses, the spotlight shifts to the practical implementation of device identification, with the proposal of a clear and detailed methodology outlined for uniquely identifying IoT devices. This process is intricately tied to the capabilities afforded by ML and DL techniques. These advanced computational tools can be harnessed to decipher the subtle nuances of hardware behavior, ensuring a high level of authenticity and integrity for devices embedded within a network. The importance of this process cannot be overstated, as it plays a pivotal role in safeguarding the security and reliability of interconnected devices, forming a critical component of the broader cybersecurity infrastructure. To highlight this importance, the presented methodology is compared to other works in the

field, where no methodological set of steps was followed. Previous solutions were not able to perform full identification in the real device scenario used for validation. Therefore, the methodology becomes the state-of-the-art procedure for developing a functional individual identification solution.

Building upon the established identification methodology, the narrative delves deeper into the critical aspect of data collection, presenting a comprehensive approach for the systematic acquisition of hardware performance data. Generating the datasets required for later behavior fingerprinting is essential. These datasets help ensure that the identification models are trained and validated on data that is both extensive and accurate. The meticulous attention to detail in this process ensures the reliability of the data, setting a high standard for the quality of information used in device behavior fingerprinting. As a result of the proposed tool for data collection, an exhaustive dataset is generated to be applied in the following steps of the thesis. This dataset is published to be employed by the research community in the field, together with the benchmark application employed to generate the data. This is one of the first public datasets of hardware performance data that is focused on the identification problem.

With a solid foundation of data in place, the exploration then navigates toward the integration of individual device identification within the larger ecosystem of behavior-based cybersecurity solutions. This integration is paramount, as it ensures that the device identification procedures do not operate in isolation but are seamlessly linked with other general security frameworks. For this integration, the focus is strategically placed on the Internet of Battlefield Things. A real-world scenario is employed to integrate the hardware-based identification solution with higher-level behavior monitoring for the detection of malware and spectrum-sensing data falsification attacks. Specifically, the unified approach monitors hardware, kernel events, and system calls to provide a unified security solution based on behavior. This holistic approach enhances the environment resilience, fortifying its defenses against a myriad of cyber threats and vulnerabilities, and ensuring a robust and secure digital environment. The unified framework demonstrates experimentally its capabilities to detect different malware samples and spectrum data-focused attacks, as well as perform individual identification of the sensors deployed. This is, to the best of our knowledge, the first framework combining these cybersecurity capabilities together.

As the work reaches its culmination, the focus turns to the security of the device identification models themselves, specifically addressing the challenges posed by adversarial attacks. The landscape of adversarial attacks is analyzed, particularly focusing on context-aware and ML/DL-centric threats that pose significant risks to the integrity of device identification methodologies. Then, the focus delves into strategies and methodologies designed to enhance the resilience of hardware behavior-based identification models. A particular emphasis is placed on counteracting sophisticated adversarial threats, including context-based and ML/DL-focused attacks. Context-based attacks are ineffective against the solution, but evasion attacks targeting the identification models achieve high success rates. Therefore, defense techniques based on adversarial training and knowledge distillation are applied. This ensures that the device identification methodologies remain reliable, secure, and effective, even in the face of evolving and complex cyber threats. This is one of the first works demonstrating experimentally the effectiveness of adversarial attacks on ML models for IoT device identification, and also of state-of-the-art defense methods.

Finally, the research explores the more complex problem of individual device authentication, where only data from one device can be leveraged for model generation. The methodology applied for identification is tweaked for its application in authentication. The main difference is the change of the classifier model for an anomaly detection model

per device. However, more powerful ML/DL models are necessary to solve this problem as data distributions overlap between devices from the same model. To solve this issue, transformer models are employed. Using large time windows for data processing (100 values), the transformer-based approach improves the results achieved by previous state-of-the-art models such as LSTM, 1D-CNN, and their combination. In contrast, higher training time and memory are employed in the model generation process. This outcome confirms the effectiveness of the transformer architecture in a novel area. Unlike previous model architectures in existing literature, it successfully addresses the issue of individually authenticating IoT devices based on their hardware performance in the experimental scenario studied.

Looking ahead, there is a vast horizon of opportunities for future work building upon the foundations laid by this thesis in the domain of device behavior fingerprinting. The first iteration of future work could delve into the adversarial attack evaluation over the unsupervised transformer models employed to solve the individual identification problem, and the consequent application of defense techniques. These results will close the work on the authentication topic in a similar manner to the methodology applied for the individual identification problem.

Another promising avenue is expanding the scope of device behavior fingerprinting to encompass a broader array of devices and contexts. The current work has predominantly focused on single-board and IoT devices. However, the principles and methodologies developed could be adapted and applied to other types of devices and networks, such as industrial control systems, automotive systems, and smart home devices. Future research could explore the nuances and specific requirements of these different contexts, tailoring the fingerprinting techniques to suit the unique characteristics of each device category and usage scenario. One of the pivotal areas highlighted for future exploration is the continual quest for new and diverse data sources. The field stands to benefit significantly from broadening the scope of data collection, capturing a wider array of device behaviors, and ensuring a richer and more comprehensive dataset for analysis. This endeavor is not just limited to increasing the quantity of data but also emphasizes the importance of improving the quality and reliability of the data collected. Future work in this area could explore advanced data collection methodologies, innovative sensor technologies, and novel data preprocessing techniques, all aimed at ensuring that the data used for device behavior fingerprinting is of the highest caliber.

Building upon the theme of data, there is a clear call for the development and refinement of fingerprinting techniques. The future holds potential for the exploration of new algorithms, ML models, and DL architectures, each offering unique capabilities and advantages for device identification. Federated Learning (FL), and more concretely decentralized FL, is one of these promising areas worth exploring in the following years. The continuous evolution of computational power and ML/DL technologies opens up exciting possibilities for creating more sophisticated and accurate device behavior fingerprinting models, capable of discerning even the most subtle nuances in device behavior.

Addressing the challenge of adversarial attacks, there is a pressing need for the development of robust countermeasures and mitigation strategies. Future work in this area could explore innovative approaches to enhancing the resilience of device behavior fingerprinting models, with a specific focus on countering sophisticated adversarial threats, including context-based and ML/DL-focused attacks. This involves not only fortifying the identification models but also developing comprehensive threat detection and response mechanisms, ensuring the long-term reliability and security of device identification methodologies.

Another critical area for future work lies in enhancing the adaptability of device be-

havior fingerprinting models. With the rapid pace of technological advancement, devices are constantly evolving, and their behavior patterns may change over time due to software updates, hardware modifications, or changes in usage patterns. Future research could focus on developing fingerprinting models that are capable of adapting to these changes, ensuring that they remain accurate and reliable over the device lifecycle. This could involve the integration of online learning techniques, continual learning approaches, or transfer learning methodologies to enable the models to update and refine their fingerprinting profiles in response to observed changes in device behavior.

Finally, there is significant potential for future work in the integration of device behavior fingerprinting solutions with other cybersecurity tools and frameworks. This thesis has laid the groundwork for such integration, demonstrating the potential benefits of combining device behavior fingerprinting with other behavior-based security solutions. Future research could build upon this, exploring ways to further streamline the integration process, enhance interoperability, and maximize the synergies between different security tools. This could lead to the creation of more holistic and resilient cybersecurity frameworks, providing comprehensive protection against a diverse range of cyber threats.

Resumen

I Introducción y motivación

Para 2025, se espera que casi 64 mil millones de dispositivos de Internet de las Cosas (IoT) estén conectados en diversos entornos de vanguardia, como Ciudades Inteligentes, Industria 4.0 y crowdsensing [1]. El crecimiento del IoT promete no solo mejorar la calidad y accesibilidad de los servicios, sino también revolucionar la experiencia del usuario, ya que fomenta ecosistemas inteligentes que reducen significativamente la intervención humana, agilizando así los procesos, reduciendo costos y mitigando errores. Sin embargo, los objetivos únicos de estos entornos también aumentan la complejidad de optimizar el rendimiento de dispositivos y servicios.

Desde una perspectiva de escenario, la vasta gama de dispositivos IoT utilizados hoy en día incluye dispositivos de placa única (SBC por su nombre en inglés) como Raspberry Pi (RPi), que han ganado popularidad debido a su flexibilidad, asequibilidad, amplio soporte y periféricos disponibles [2]. Sin embargo, las limitaciones de conectividad y recursos de las SBC y los dispositivos IoT generan numerosas preocupaciones de ciberseguridad para diversas plataformas [3]. Un problema significativo es la presencia de dispositivos no autorizados con configuraciones de hardware y software idénticas a las de los nodos autorizados, lanzando ataques que impactan áreas de aplicación como Industria 4.0 [4], teléfonos inteligentes [5] o Internet de las Cosas del Campo de Batalla (IoBT) [6]. Estos dispositivos maliciosos pueden estar presentes como consecuencia de varias amenazas de ciberseguridad [7], incluyendo i) suplantación de dispositivos, donde un atacante reemplaza un dispositivo legítimo por uno malicioso usando la misma identidad; ii) despliegue no autorizado de dispositivos, que implica la instalación de un nuevo dispositivo con una identidad no registrada; y iii) ataque Sybil, donde un dispositivo malicioso utiliza múltiples identidades para simular numerosos dispositivos. En consecuencia, otras amenazas como la filtración de información sensible, envenenamiento de datos y escalada de privilegios y movimientos laterales pueden surgir de dispositivos suplantados.

Para resolver estos problemas emergentes de ciberseguridad, la ciencia de datos del comportamiento se ha expandido desde el estudio de comportamientos humanos [8] hacia la modelización de comportamientos de dispositivos [9], con un enfoque en la creación de patrones de comportamiento de dispositivos (*huellas digitales*) para optimizar el rendimiento y detectar problemas potenciales tempranamente [10]. Dos escenarios primarios de aplicación de IoT para la construcción de huellas digitales de dispositivos son i) identificación y autenticación de dispositivos en diferentes niveles de granularidad [11], y ii) detección de

ciberataques, malfuncionamiento o comportamiento indebido [12, 13]. Diversos estudios han aplicado el fingerprinting de dispositivos a ambos escenarios [14, 15]. En el escenario de identificación, la ciencia de datos del comportamiento ha mejorado significativamente las capacidades de identificación de dispositivos, superando las limitaciones asociadas con metodologías convencionales que predominantemente dependen del uso de nombres, identificadores, etiquetas o tags para el reconocimiento de dispositivos [16]. Una limitación crítica de las estrategias tradicionales es su susceptibilidad a alteraciones y duplicaciones. Estas técnicas convencionales a menudo carecen del dinamismo para adaptarse al paisaje rápidamente evolutivo de los escenarios IoT, donde muchos dispositivos se despliegan con alta movilidad. Estos problemas son especialmente relevantes en escenarios con un aumento exponencial en la cantidad de dispositivos, como en industrias inteligentes o la agricultura. Por lo tanto, la identificación de dispositivos basada en huellas digitales de comportamiento ha mejorado significativamente las soluciones tradicionales al enfocarse en niveles de granularidad de tipo [11], modelo [17], e individual [18]. Por otro lado, la detección de mal comportamiento o malfuncionamiento debido a problemas de ciberseguridad ha visto el surgimiento del fingerprinting de dispositivos como una solución prometedora. Numerosos trabajos crean huellas digitales de comportamiento "normal" para detectar cambios causados por problemas como ciberataques, ejecución de malware o malfuncionamiento de dispositivos [12, 19].

En el nivel de granularidad de identificación individual, el fingerprinting (generación de huellas) del comportamiento del hardware se destaca como una vía prometedora para identificar de manera única dispositivos idénticos, una necesidad en el paisaje tecnológico interconectado de hoy. A pesar de su potencial, este dominio todavía está en desarrollo, caracterizado por desafíos abiertos y una falta de investigación dedicada específicamente a la identificación de dispositivos de placa única idénticos [20]. La complejidad de esta tarea se amplifica por las similitudes inherentes entre dichos dispositivos, lo que requiere metodologías de identificación innovadoras y precisas. En el contexto más amplio, para dispositivos que no están limitados por componentes y recursos, la literatura existente aboga por la adopción del *fingerprinting del comportamiento del hardware* [21]. Esta técnica es fundamental para discernir pequeñas variaciones de rendimiento que son un subproducto de imperfecciones de fabricación [22]. Al analizar meticulosamente estas sutiles diferencias, el fingerprinting del comportamiento del hardware proporciona un nivel granular de caracterización del dispositivo, abriendo el camino para una identificación precisa del dispositivo. Sin embargo, es importante distinguir entre identificación y autenticación en este contexto. Mientras que la identificación implica reconocer un dispositivo de un conjunto basado en sus características únicas, la autenticación va un paso más allá verificando la legitimidad del dispositivo. La autenticación aborda la pregunta de si un dispositivo es quien dice ser, ofreciendo una capa de seguridad que la mera identificación no proporciona.

El análisis del comportamiento del hardware para la identificación de dispositivos se ha realizado parcialmente en la literatura pero sin seguir una metodología clara sobre los pasos a realizar para lograr una solución exitosa [20]. Por lo tanto, existe una necesidad de esta metodología específica, arraigada en la necesidad crítica de reforzar las medidas de ciberseguridad en infraestructuras de red que incorporan dispositivos de ordenador de placa única idénticos, como Raspberry Pi. El conjunto de estos dispositivos trabajando de manera coordinada es lo que permite ofrecer servicios de IoT basados en crowdsourcing o recolección/procesamiento de datos conjuntos. Estos dispositivos, a menudo desplegados extensamente y en entornos donde los recursos son escasos, son vulnerables a una miríada de amenazas de seguridad [23, 24] que afectan la fiabilidad de los servicios ofrecidos. Estos van desde intentos de suplantación de dispositivos maliciosos hasta la introducción de

dispositivos no autorizados en una red. Asegurar la capacidad de identificar y autenticar dispositivos de manera precisa se convierte en un requisito no negociable, fundamental para mantener la integridad y la resiliencia del servicio. Por lo tanto, un nivel de seguridad consistente es esencial para tener servicios confiables. La implementación del fingerprinting del comportamiento del hardware en estos contextos exige un nivel de precisión e innovación que trasciende los métodos convencionales de identificación. Los desafíos provienen de las sutiles diferencias entre dispositivos idénticos y las limitaciones impuestas por sus recursos limitados [25]. Abordar estos desafíos requiere un esfuerzo de investigación concertado, dirigido a desentrañar las complejidades del fingerprinting del comportamiento del hardware, seleccionar las características de rendimiento del hardware más prometedoras, desarrollar algoritmos de identificación robustos y establecer mejores prácticas para la aplicación práctica.

Además, tener una solución de identificación completa basada en hardware es crítico para la seguridad de IoT. Sin embargo, en escenarios del mundo real, las soluciones de identificación individual no funcionan aisladas de otras aplicaciones de ciberseguridad. Deben integrarse con otras soluciones de red y comportamiento que buscan cubrir problemas de ciberseguridad heterogéneos, como el malware. En este contexto, la integración del fingerprinting del comportamiento del hardware para dispositivos de placa única se convierte en un componente estratégico de un framework de seguridad más amplio. Identificar y verificar de manera única cada dispositivo permite una base sólida para la comunicación segura y la integridad de los datos. Sin embargo, aún se requieren otras herramientas para alcanzar un nivel de seguridad más completo. Este nivel de seguridad no se trata solo de proteger la información; se trata de garantizar la efectividad operativa y el éxito de los entornos IoT.

Para reforzar la efectividad del fingerprinting del comportamiento del hardware para dispositivos de placa única idénticos, es imperativo identificar las fuentes de datos más relevantes para resolver este problema. Luego, se requiere tener un punto de referencia integral y un conjunto de datos bien curado [26]. Estos recursos sirven como herramientas críticas en la verificación y validación de la metodología, asegurando su precisión, fiabilidad y aplicabilidad en escenarios del mundo real. Un punto de referencia preciso es necesario para evaluar sistemáticamente el rendimiento de los dispositivos, proporcionando una medida estandarizada que se puede usar para discernir las sutiles variaciones en el comportamiento del hardware [27]. Esto se vuelve particularmente crucial al tratar con dispositivos idénticos, donde las diferencias son mínimas pero significativas para una identificación precisa. Sin embargo, no hay aplicaciones de benchmarking disponibles en la literatura para el fingerprinting de comportamiento de hardware de bajo nivel [26]. Junto con un punto de referencia robusto, un conjunto de datos rico juega un papel vital en el proceso. Actúa como un repositorio de perfiles de comportamiento de dispositivos de hardware, capturando las complejidades y características únicas de cada dispositivo. Este conjunto de datos se convierte en un punto de referencia, ayudando en el entrenamiento de algoritmos y sirviendo como un punto de referencia para la validación. La combinación de un punto de referencia integral y un conjunto de datos detallado asegura un enfoque holístico para la identificación de dispositivos, fortaleciendo la metodología de fingerprinting del comportamiento del hardware. La motivación para establecer herramientas de verificación tan rigurosas surge de las necesidades cambiantes de los paradigmas de IoT y computación Edge, donde los dispositivos están cada vez más interconectados y la integridad del entorno es primordial. En estos escenarios, la capacidad de identificar y autenticar dispositivos con precisión no es solo una medida de seguridad; es una necesidad para garantizar el funcionamiento fluido de la red y la confiabilidad de los datos intercambiados. Al invertir en el desarrollo

de puntos de referencia precisos y conjuntos de datos integrales, la metodología de fingerprinting del comportamiento del hardware puede empoderarse, mejorando su precisión y fiabilidad. Esto, a su vez, allana el camino para un futuro donde dispositivos de placa única idénticos puedan integrarse sin problemas en redes complejas, operando de manera segura y eficiente, y contribuyendo a la solidez de la infraestructura digital.

Tras adquirir las huellas digitales, surge un dominio de investigación apasionante, centrado en emplear técnicas óptimas para procesarlas y evaluarlas. Mientras que los métodos estadísticos han mantenido una posición dominante en este campo durante décadas, la llegada de la Inteligencia Artificial (IA), específicamente el Machine Learning (ML) y el Deep Learning (DL), ha precipitado un cambio de paradigma, ganando mayor prominencia en las soluciones que se están desarrollando y desplegando hoy en día [28]. La complejidad y diversidad de los comportamientos de los dispositivos IoT requieren modelos de DL sofisticados capaces de capturar patrones y variaciones intrincadas. Técnicas como las Redes Neuronales Convolucionales (CNNs) se están adaptando para discernir relaciones espaciales en huellas digitales de hardware [29], mientras que las Redes Neuronales Recurrentes (RNNs), incluyendo su variante más sofisticada, las redes de Memoria a Corto y Largo Plazo (LSTM), se están desplegando para el análisis de datos temporales, crucial para comprender el comportamiento del dispositivo a lo largo del tiempo [17]. Los autoencoders también han creado un nicho en este sector, permitiendo la reducción de la dimensionalidad para datos de huellas digitales de gran volumen y la detección de anomalías mediante la reconstrucción del comportamiento normal del dispositivo y destacando desviaciones [30]. Además, los avances recientes en Redes Neuronales de Grafos (GNNs) permiten la incorporación de datos topológicos, facilitando la modelización de relaciones complejas dentro de redes de dispositivos IoT interconectados. Finalmente, los modelos basados en atención como los transformers están logrando un éxito extraordinario en herramientas centradas en el lenguaje, como ChatGPT [31], y también pueden aplicarse a otros datos de series temporales. Sin embargo, la aplicabilidad de todas estas técnicas modernas está condicionada por la falta de conjuntos de datos disponibles, ya que requieren la existencia de conjuntos de datos exhaustivos capaces de entrenar los modelos requeridos [32]. Por lo tanto, después de tener suficientes datos, el siguiente desafío es explorar los métodos de ML/DL disponibles para encontrar cuál podría proporcionar el mejor rendimiento en la identificación individual de dispositivos. Aquí, se pueden crear nuevas arquitecturas de red para modelar mejor el comportamiento del dispositivo y mejorar las capacidades de identificación y autenticación.

Además, una solución totalmente funcional y confiable debe considerar los posibles problemas de seguridad intrínsecos a la identificación individual de dispositivos basada en hardware [33]. Los ataques adversarios surgen como una de las principales amenazas dirigidas a soluciones de identificación en IoT. Estos ataques sofisticados presentan un desafío formidable, ya que están específicamente diseñados para manipular o evadir los mecanismos de seguridad existentes, lo que podría llevar a accesos no autorizados, violaciones de datos y compromiso de la integridad de la red [34, 35]. Los ataques adversarios en el contexto de la identificación de dispositivos de placa única explotan los mecanismos diseñados para garantizar la autenticidad del dispositivo y la seguridad de la red. Al alterar sutilmente el comportamiento del dispositivo o imitar las características de dispositivos legítimos, los adversarios pueden engañar a los sistemas de identificación, lo que resulta en la clasificación errónea o la aceptación falsa de dispositivos pícaros. Esto no solo socava la confiabilidad de la red, sino que también abre la puerta a actividades maliciosas adicionales, poniendo en peligro la confidencialidad, integridad y disponibilidad de los datos y servicios. Profundizando en la naturaleza de estos ataques adversarios, hay una tendencia

creciente en tácticas de evasión basadas en contexto y ML/DL. Los ataques basados en contexto manipulan inteligentemente las condiciones ambientales o el contexto operativo de los dispositivos, con el objetivo de distorsionar los datos utilizados para la identificación y, por lo tanto, engañar al sistema [36]. Estos ataques son particularmente insidiosos, ya que explotan la variabilidad natural en el comportamiento del dispositivo debido a cambios en factores externos, haciéndolos más difíciles de detectar y contrarrestar. Por otro lado, los ataques de evasión basados en ML/DL representan un asalto sofisticado y calculado a los sistemas de identificación de dispositivos de placa única. Los adversarios que emplean estas técnicas utilizan modelos avanzados de ML para aprender los patrones y características de dispositivos legítimos. Armados con este conocimiento, crean muestras adversarias que imitan de cerca a dispositivos auténticos, difuminando efectivamente las líneas entre dispositivos legítimos y pícaros [37]. Estas muestras se utilizan luego para sondear y engañar al sistema de identificación, llevando a clasificaciones erróneas y potencialmente otorgando acceso no autorizado a la red. Estos ataques explotan las complejidades inherentes y variabilidades en el comportamiento del dispositivo y las capacidades avanzadas de los modelos de ML/DL para engañar y comprometer la seguridad de la red. Abordar estos desafíos requiere una estrategia de seguridad integral y adaptable capaz de anticipar, detectar y mitigar la miríada de amenazas que representan los ataques adversarios.

Aparte de todo el trabajo por hacer en el contexto de identificación individual de dispositivos IoT, el problema de la autenticación es un desafío abierto por sí solo debido a sus requisitos más complejos [38]. Aunque el fingerprinting del comportamiento del hardware ofrece una base robusta para la identificación, extender este enfoque para incluir mecanismos de autenticación es vital para garantizar un grado más alto de seguridad en sistemas interconectados. Sin embargo, los dispositivos del mismo modelo pueden exhibir características de comportamiento muy similares o incluso superpuestas debido a procesos de producción estandarizados. Esta similitud en el comportamiento del hardware hace que sea desafiante distinguir entre dispositivos legítimos y no autorizados basándose únicamente en datos de comportamiento del hardware, ya que las distribuciones de datos se fusionan. Por lo tanto, la autenticación requiere una granularidad de análisis de datos mucho más fina. Aquí, los enfoques de procesamiento tradicionales y las técnicas de ML/DL no han logrado resultados notables. Sin embargo, nuevos algoritmos de ML/DL con capacidades avanzadas de reconocimiento de patrones, como los transformers basados en atención, podrían mejorar el rendimiento de la autenticación basada en el comportamiento del rendimiento del hardware.

Como resumen de los desafíos abiertos, la literatura sugiere que, aunque el fingerprinting del comportamiento del hardware presenta una solución prometedora, el campo aún es incipiente, con brechas de investigación considerables en metodología y práctica para dispositivos de placa única. La identificación individual de estos dispositivos requiere técnicas innovadoras para distinguir sutiles diferencias de fabricación, así como puntos de referencia precisos y conjuntos de datos extensos para validar nuevos algoritmos. La evolución de la IA, particularmente ML y DL, está cambiando el paradigma para procesar y evaluar huellas digitales de dispositivos, y se requieren nuevas técnicas en esta área. Además, sigue siendo una necesidad crítica la integración del fingerprinting del hardware con medidas de seguridad de red más amplias para abordar varias amenazas de ciberseguridad. A continuación, los ataques adversarios representan un riesgo formidable, aprovechando tácticas basadas en contexto o modelos de ML/DL para crear muestras que imitan dispositivos legítimos, lo que requiere una estrategia de seguridad robusta y adaptable que abarque la identificación y mitigación integral de tales amenazas. Finalmente, es vital explorar el problema de la autenticación de dispositivos aprovechando soluciones de ML/DL más complejas.

Las soluciones de autenticación tienen que generar patrones avanzados en la huella digital, generando modelos únicos para cada dispositivo basados en sus características intrínsecas.

Basado en las consideraciones anteriores, esta Tesis Doctoral explora la viabilidad de la identificación y autenticación de dispositivos individuales basada en el fingerprinting del comportamiento del rendimiento del hardware. Investiga la definición de la metodología, su aplicación en frameworks y herramientas reales, su integración en dispositivos y escenarios del mundo real, y el análisis y mitigación de amenazas de seguridad asociadas. En este sentido, varias preguntas de investigación surgieron de los desafíos anteriores, guiando el proceso de investigación de esta Tesis Doctoral, y se presentan de la siguiente manera:

- RQ1: ¿Cuál es el estado actual de las soluciones de fingerprinting del comportamiento de dispositivos aplicadas en la identificación individual de dispositivos y qué fuentes de datos, técnicas, escenarios de aplicación y conjuntos de datos están presentes en la literatura?
- RQ2: ¿Qué metodología se debe seguir para identificar de manera única dispositivos IoT de manera escalable mientras se aprovecha el comportamiento del hardware y las técnicas de ML/DL?
- RQ3: ¿Qué métricas de hardware se pueden recolectar extensivamente para medir el rendimiento del hardware y generar los conjuntos de datos requeridos para el fingerprinting del comportamiento posterior?
- RQ4: ¿Cómo se puede integrar la identificación individual de dispositivos con otras soluciones de ciberseguridad basadas en comportamiento desplegadas en escenarios del mundo real?
- RQ5: ¿Qué técnicas de ML/DL pueden lograr el mejor rendimiento para la identificación individual de dispositivos?
- RQ6: ¿Cómo se puede mejorar la resiliencia de los modelos de identificación individual de dispositivos basados en el comportamiento del hardware frente a ataques adversarios centrados en contexto y ML/DL?
- RQ7: ¿Pueden las técnicas de ML/DL basadas en atención, como los transformers, permitir la identificación individual de dispositivos siguiendo un enfoque de detección de anomalías? ¿Qué recursos se requieren?

II Objetivos

El objetivo principal de esta tesis doctoral es avanzar en el campo del fingerprinting del comportamiento de dispositivos para mejorar la identificación y seguridad de los dispositivos IoT, con un enfoque particular en computadoras de placa única y sistemas con recursos limitados. La investigación tiene como objetivo desarrollar metodologías y frameworks de trabajo novedosos para la identificación individual de dispositivos, aprovechando las técnicas de ML y DL. Al examinar varios escenarios de aplicación, incluyendo Ciudades Inteligentes, Industria 4.0 y el Internet de las Cosas del Campo de Batalla, la tesis tiene como objetivo abordar las crecientes amenazas de ciberseguridad, como el despliegue no autorizado de dispositivos, y otros problemas emergentes debido al crecimiento exponencial de dispositivos interconectados en el mundo computacional basado en redes.

La tesis explora diferentes aspectos del fingerprinting del comportamiento de dispositivos, incluyendo fuentes de datos, técnicas, escenarios de aplicación y conjuntos de

datos. La investigación se centra en resolver los problemas únicos enfrentados al definir metodologías para identificar computadoras de placa única idénticas basadas en el fingerprinting del comportamiento del hardware y ML. Algunos de estos problemas abiertos son la disponibilidad de datos y la integración de la solución en escenarios del mundo real. Por lo tanto, se exploran como parte de los objetivos de investigación. La investigación también investiga ataques adversarios y defensas en el fingerprinting de IoT, con el objetivo de desarrollar arquitecturas resilientes. Por último, el trabajo explora el problema de la autenticación, donde cada dispositivo debe ser reconocido sin considerar a otros y, por lo tanto, son necesarios modelos de ML más complejos. A partir del objetivo de cubrir los aspectos anteriores, se derivan varios objetivos específicos como se presenta a continuación, indicando las preguntas de investigación relacionadas con ellos:

O.1. Analizar el estado actual del arte respecto al fingerprinting del comportamiento de dispositivos para la identificación individual a través de una revisión completa, análisis y comparación de fuentes de datos, técnicas, aplicaciones y conjuntos de datos para guiar investigaciones y soluciones futuras (RQ1).

O.2. Identificar brechas existentes en la literatura en la identificación individual de dispositivos basada en el fingerprinting del comportamiento del hardware y los pasos requeridos para cubrirlas (RQ1).

O.3. Abordar los desafíos de identificación de dispositivos de placa única con una metodología novedosa aprovechando el fingerprinting del comportamiento del hardware, técnicas de ML/DL y propiedades esenciales para mejorar la precisión y robustez (RQ2).

O.4. Desarrollar una solución de benchmarking de hardware de bajo nivel para Computadoras de Placa Única para habilitar soluciones de IA basadas en Edge y gestión versátil de dispositivos a través de extensos conjuntos de datos de rendimiento (RQ3).

O.5. Integrar la solución de identificación de dispositivos de placa única en un framework de seguridad utilizando fingerprinting del comportamiento y técnicas de ML/DL para detectar con precisión diversos ciberataques (RQ4).

O.6. Encontrar las mejores técnicas basadas en ML/DL para la identificación individual, iterando sobre los conjuntos de datos disponibles para lograr el mejor rendimiento (RQ5).

O.7. Investigar el impacto de ataques contextuales y centrados en ML/DL contra soluciones de identificación de dispositivos basadas en el comportamiento del hardware aprovechando modelos de ML/DL (RQ6).

O.8. Implementar y evaluar mecanismos de defensa para fortalecer la resiliencia de la solución contra ataques adversarios mientras se mantiene su rendimiento en el contexto de la identificación de dispositivos basada en el comportamiento del hardware (RQ6).

O.9. Explorar el problema de la autenticación individual para verificar las capacidades de modelos de ML/DL basados en atención para la detección de anomalías en los datos de comportamiento del hardware (RQ7).

III Metodología

Esta Tesis Doctoral se llevó a cabo siguiendo un enfoque científico basado en el estudio continuo del estado del arte y el análisis de los resultados obtenidos durante las distintas

etapas de la investigación. Esta tesis se define como un conjunto de seis artículos publicados en revistas de alto impacto indexadas en los Journal Citation Reports (JCR).

El primer paso fue resolver la RQ1 y proporcionar una visión completa del fingerprinting del comportamiento de dispositivos. Se realizó una amplia exploración de la literatura existente para proporcionar una comprensión integral del estado actual de las soluciones de fingerprinting del comportamiento de dispositivos en ciberseguridad. Esta exploración abarcó una variedad de publicaciones académicas, que abarcan revistas y actas de conferencias, asegurando una amplia cobertura del tema. La literatura recopilada fue meticulosamente categorizada y analizada en función de criterios específicos, como las fuentes de datos utilizadas para el fingerprinting, las técnicas empleadas, los escenarios de aplicación abordados y los conjuntos de datos utilizados. Esta categorización granular facilitó la identificación de tendencias prevalentes, métodos comúnmente utilizados y posibles brechas dentro del cuerpo actual de conocimiento. El análisis proporcionó una comprensión matizada de cómo se aplica el fingerprinting del comportamiento de dispositivos en diferentes dominios de ciberseguridad, destacando su versatilidad y la variedad de formas en que se puede implementar. Esta revisión extensa culminó en una visión holística del campo, ofreciendo valiosas perspectivas y una base sólida para futuros esfuerzos de investigación en el fingerprinting del comportamiento de dispositivos. Todas estas consideraciones resultaron en la primera publicación de esta Tesis Doctoral, presentada en el primer capítulo (Article 1–IEEE_COMST), que cubrió los dos primeros Objetivos de la tesis.

Después de realizar el análisis del estado del arte e identificar las brechas actuales en las soluciones de identificación individual de IoT, el foco se trasladó a resolver la RQ2 y abordar el Objetivo 3, que trata el problema de identificación individual de dispositivos IoT. Para identificar de manera única dispositivos IoT basándose en su comportamiento de hardware mientras se aprovechan las técnicas de ML y DL, se empleó un enfoque estructurado. El proceso comenzó con la recopilación de un conjunto diverso y extenso de datos de comportamiento de hardware de una variedad de modelos de RPi. Estos datos sirvieron como base para el análisis posterior. Tras la fase de recopilación de datos, se aplicaron técnicas de preprocesamiento para refinar y preparar los datos para los modelos de ML y DL. Se seleccionaron y entrenaron varios modelos de ML/DL en los datos preprocesados, lo que resultó en el desarrollo de algoritmos de identificación. El rendimiento de estos algoritmos fue evaluado rigurosamente utilizando métricas comunes de clasificación de ML/DL y dispositivos reales de SBC, asegurando su precisión y efectividad en escenarios del mundo real. Posteriormente, se comparó la metodología con otros enfoques existentes en la literatura, aunque no presentaron un conjunto estructurado de pasos para desarrollar una solución de identificación. En esta comparación, se contrastó prácticamente la mejora del rendimiento de la metodología. Este enfoque integral facilitó una metodología clara y práctica para la identificación única de dispositivos IoT basada en su comportamiento de hardware, aprovechando los últimos avances en ML y DL. La propuesta y validación de la metodología de identificación de dispositivos individuales resultó en el segundo capítulo (Article 2–JNCA).

En este punto, surgió una limitación en la línea de investigación con respecto a la disponibilidad de datos y cómo recopilarlos. Como parte de esta Tesis Doctoral, se desarrolló una herramienta de benchmarking especializada y un conjunto de datos para abordar el desafío de la recopilación extensiva de datos para medir el rendimiento del hardware y generar los conjuntos de datos requeridos para el fingerprinting del comportamiento posterior. Esta herramienta fue meticulosamente diseñada para medir y registrar el rendimiento de varios componentes de hardware en una variedad de dispositivos IoT, asegurando un proceso de recopilación de datos estandarizado y completo. La herramienta se ejecutó luego

en algunos dispositivos SBC para extraer un conjunto de datos completo y realista. Los datos recopilados fueron sometidos a rigurosos procesos de validación y control de calidad, verificando su precisión y fiabilidad. Además de la recopilación de datos, se proporcionaron pautas claras sobre cómo estructurar y almacenar los datos, facilitando su integración en conjuntos de datos integrales. Este enfoque meticuloso aseguró que los datos recopilados no solo fueran extensos, sino también de alta calidad, proporcionando una base sólida para aplicaciones y análisis de fingerprinting del comportamiento posteriores. La aplicación de benchmarking, junto con el conjunto de datos recopilados, están disponibles públicamente para otros investigadores en el área. Este trabajo resultó en el tercer capítulo de esta tesis (Article 3–IoT), respondiendo a la RQ3 y completando el Objetivo 4.

Una vez verificada la viabilidad de la identificación individual con el conjunto de datos exhaustivos recopilados mediante la aplicación de benchmark, se desarrolló un framework holístico e interoperable. Este framework buscó integrar la identificación individual de dispositivos con otras soluciones de ciberseguridad basadas en comportamiento. Este framework se basó en un análisis extenso del fingerprinting del comportamiento de dispositivos y soluciones de ciberseguridad existentes, asegurando una comprensión integral de los componentes y procesos necesarios. El framework diseñado facilitó una integración sin problemas, permitiendo compartir y utilizar fácilmente las huellas digitales y los perfiles de comportamiento de los dispositivos en diferentes soluciones de seguridad. Para lograr esto, el framework fue diseñado para ser altamente modular y escalable, permitiendo una fácil integración con una variedad de soluciones de ciberseguridad basadas en el comportamiento. Ya fueran sistemas de detección de intrusiones, soluciones de gestión de información y eventos de seguridad, o herramientas avanzadas de protección contra amenazas, el framework era capaz de interactuar con ellas, mejorando sus capacidades y proporcionando una capa adicional de seguridad. Esta interoperabilidad fue crucial para mejorar la seguridad general de la red y asegurar la efectividad de las soluciones integradas. Para validar el rendimiento del framework, se realizaron extensas simulaciones y pruebas en el mundo real, evaluando su funcionalidad en diversos escenarios y contra varios modelos de amenazas. La validación en el mundo real se realizó considerando un escenario de Internet of Battlefield Things (IoBT) basado en la plataforma ElectroSense [39]. Luego, se integró una solución de monitoreo de eventos de kernel y llamadas al sistema junto con la solución de identificación individual para proporcionar un enfoque de seguridad completo para la plataforma. Los resultados de estas evaluaciones confirmaron la efectividad del framework, demostrando su capacidad para integrar la identificación individual de dispositivos con otras soluciones de ciberseguridad basadas en el comportamiento y mejorar la postura de seguridad de la red. Este trabajo resolvió RQ4 y el Objetivo 5, disponible en el cuarto capítulo de la tesis (Article 4–IEEE_COMMAG).

El siguiente trabajo realizado en la Tesis Doctoral, presentado en el quinto capítulo de este documento (Article 5–FGCS) y alineado con las quintas y sextas preguntas de investigación (Objetivos 6, 7 y 8), abordó el desafío de las amenazas de seguridad. Se centró en encontrar la mejor técnica de ML/DL para la identificación y luego mejorar su resiliencia para una identificación fiable de dispositivos individuales basada en el comportamiento del hardware. Esto involucró un análisis exhaustivo de los posibles ataques adversarios, enfatizando la comprensión de la naturaleza y el impacto de estos sofisticados ataques. El modelo de amenazas presentado aborda los ataques adversarios centrados en contexto y ML/DL, asegurando que los modelos de identificación sigan siendo confiables y seguros incluso frente a amenazas sofisticadas. Se verificó el impacto de los ataques propuestos en la literatura, demostrando la vulnerabilidad de la solución a los ataques adversarios. Luego, se formularon e implementaron contramedidas y estrategias de mitigación para fortalecer

los modelos de identificación contra manipulaciones adversarias. Se incorporaron técnicas de entrenamiento adversario y destilación de conocimiento [40] para mejorar la resiliencia del modelo contra ataques adversarios.

El último trabajo de este compendio se enfocó en la séptima pregunta de investigación (RQ7) y el Objetivo 9 (Article 6–COSE). Se ocupó del problema de autenticación en lugar de la identificación. Como se explicó en la Introducción, la principal diferencia entre estos problemas es que para la autenticación, solo se pueden emplear datos del dispositivo que se está autenticando en el proceso de entrenamiento y se necesita una mayor granularidad en el procesamiento de datos. Este hecho hace que el problema de autenticación sea mucho más difícil, ya que la distribución de datos de rendimiento del hardware en dispositivos del mismo modelo generalmente se superpone en gran proporción. La metodología seguida para resolver este problema implicó el procesamiento de series temporales para entrenar modelos autoencoder basados en transformers [41], con cada modelo adaptado a un dispositivo específico. La recolección y preprocesamiento de datos fueron similares a los aplicados en los trabajos centrados en identificación, variando solo el tamaño de la ventana de tiempo empleada. Luego, se diseñaron modelos transformers basados en atención para funcionar como mecanismos de detección de anomalías, identificando cualquier desviación del comportamiento del hardware esperado que indicaría un dispositivo no autorizado. Este trabajo puso un fuerte énfasis en la aplicación práctica de su metodología en escenarios del mundo real. Por lo tanto, realizó pruebas y validaciones extensas para asegurarse de que el enfoque no solo fuera teóricamente sólido sino también efectivo en la práctica.

Esta tesis crea una narrativa holística, abordando los aspectos críticos del fingerprinting del comportamiento de dispositivos desde el conocimiento fundamental y técnicas prácticas de identificación hasta la recolección de datos, integración con soluciones de seguridad más amplias y resiliencia frente a ataques adversarios. Finalmente, también se explora el problema de autenticación utilizando enfoques modernos basados en transformers. Este enfoque integral asegura una comprensión profunda y una aplicación robusta del fingerprinting del comportamiento de dispositivos en el ámbito de la ciberseguridad. Esta metodología permitió cumplir con los objetivos definidos en la tesis, previamente presentados en la Sección II.

IV Resultados

La primera publicación de la Tesis Doctoral, presentada en (Article 1–IEEE_COMST), ofreció un estudio exhaustivo sobre el modelado del comportamiento de dispositivos, proporcionando un análisis y síntesis amplios de soluciones aplicadas en el ámbito de la ciberseguridad, junto con valiosos conocimientos y hallazgos. Los resultados destacaron un amplio espectro de fuentes de datos, técnicas, escenarios de aplicación y conjuntos de datos prevalentes en la literatura actual, subrayando la naturaleza multifacética del modelado del comportamiento de dispositivos. Uno de los hallazgos clave gira en torno a la diversidad de fuentes de datos utilizadas para el modelado del comportamiento de dispositivos. Los resultados indicaron que se emplea una amplia variedad de tipos de datos, que van desde el tráfico de red y registros del sistema hasta atributos específicos del hardware. Esta diversidad subraya la versatilidad de las soluciones de modelado de dispositivos, demostrando su aplicabilidad en diferentes capas de la arquitectura del sistema y de la red.

En términos de técnicas de modelado, el estudio reveló un rico panorama de metodologías, cada una con sus fortalezas y capacidades únicas. Los resultados muestran que no hay una solución única para todos, con diferentes técnicas que atienden a requisitos y escenarios

específicos. Esta variedad asegura que los profesionales e investigadores tengan una plétora de opciones para elegir, permitiéndoles adaptar sus soluciones de modelado de dispositivos para satisfacer las necesidades específicas de su aplicación. En cuanto a los escenarios de aplicación, el documento esbozó una amplia gama de contextos en los que se emplea el modelado del comportamiento de dispositivos. Desde mejorar la seguridad de la red y detectar dispositivos no autorizados, hasta facilitar la autenticación de dispositivos y la verificación de integridad, las aplicaciones son vastas y variadas. Esto destacó el papel crucial que juega el modelado del comportamiento de dispositivos en el refuerzo de las medidas de ciberseguridad, proporcionando una capa adicional de seguridad y confianza en entornos digitales. El examen de los conjuntos de datos reveló que, aunque hay numerosos conjuntos de datos disponibles para fines de investigación y desarrollo, todavía se necesita más conjuntos de datos completos y estandarizados. Los resultados señalan que los conjuntos de datos existentes varían significativamente en términos de tamaño, calidad y relevancia, indicando una brecha que necesita ser abordada para avanzar aún más en el campo. La disponibilidad de conjuntos de datos estandarizados de alta calidad es fundamental para la validación y evaluación comparativa de soluciones de modelado de dispositivos, asegurando su fiabilidad y efectividad en escenarios del mundo real.

En la última sección del documento, se presentó un análisis profundo y perspicaz, encapsulando las lecciones aprendidas, identificando tendencias predominantes y destacando los desafíos enfrentados en el dominio del modelado del comportamiento de dispositivos dentro de la ciberseguridad. Esta sección sirve como una reflexión crítica sobre el estado actual del campo, ofreciendo orientación para futuras investigaciones e implementaciones prácticas, incluyendo los próximos trabajos en esta tesis doctoral.

La segunda publicación (Article 2–JNCA) proporcionó una exploración y análisis detallados de los pasos necesarios para identificar de manera única dispositivos IoT basados en su comportamiento de hardware, con un énfasis particular en el aprovechamiento de técnicas de ML y DL. Los resultados obtenidos de esta investigación ofrecen valiosas perspectivas sobre las complejidades del modelado de dispositivos y las prácticas de implementar tales metodologías en escenarios del mundo real.

La investigación identificó propiedades esenciales para la identificación de dispositivos de placa única, incluyendo la unicidad, estabilidad, diversidad, escalabilidad, eficiencia, robustez y seguridad. Luego se introdujo una metodología novedosa, basada en el modelado de comportamiento para identificar dispositivos de placa única idénticos mientras se cumplen las propiedades mencionadas. Esta metodología utiliza los diferentes componentes integrados del sistema, junto con técnicas de ML/DL, para comparar el comportamiento interno de los dispositivos y detectar variaciones que ocurrieron durante los procesos de fabricación. Un resultado clave del estudio fue la identificación y validación de atributos de hardware específicos que pueden usarse como indicadores confiables para el modelado de dispositivos. Estos atributos, cuando se analizan y procesan correctamente, han demostrado proporcionar una firma única para cada dispositivo, facilitando una identificación precisa y eficiente. Los resultados subrayan la importancia de seleccionar la combinación adecuada de atributos de hardware, ya que esta elección impacta significativamente en la efectividad del proceso de modelado.

El documento también destacó el papel crítico del aislamiento de hardware en el flujo de trabajo del modelado de dispositivos. Los resultados demostraron que el manejo y la preparación cuidadosos de los atributos de hardware son fundamentales, ya que esto asegura la integridad del proceso de modelado y mejora la precisión de la identificación de dispositivos. Además, se ha demostrado que la aplicación de varias técnicas de preprocesamiento de datos, incluyendo la normalización y reducción de dimensionalidad, contribuye

positivamente al resultado del proceso de modelado.

La integración de técnicas de ML y DL en el proceso de modelado de dispositivos también ha demostrado introducir un elemento de adaptabilidad y aprendizaje, permitiendo que el sistema evolucione y mejore con el tiempo. Se exploraron técnicas de Deep Learning, incluyendo varias configuraciones de redes neuronales, por su capacidad para aprender y modelar las huellas de dispositivos directamente a partir de datos brutos de hardware. Clasificadores de ML y DL estuvieron entre las arquitecturas probadas, elegidos por su habilidad en manejar datos secuenciales y de series temporales, que son prevalentes en la información de comportamiento del hardware. Además, el documento abordó el desafío del sobreajuste del modelo, especialmente pertinente al emplear modelos complejos como redes neuronales profundas. Se aplicaron estrategias como la validación cruzada y la regularización, asegurando que los modelos generalicen bien a datos no vistos y mantengan un alto rendimiento cuando se desplieguen en entornos del mundo real. La metodología se validó en un entorno real, consistiendo en 15 dispositivos Raspberry Pi 4 Model B y 10 Raspberry Pi 3 Model B+, cuyo rendimiento de CPU y GPU fue analizado. Los resultados muestran una Tasa de Positivos Verdaderos (TPR) promedio del 91.9% con un modelo XGBoost, logrando la identificación de todos los dispositivos estableciendo un umbral del 50% en el proceso de evaluación. Además, se entabló una discusión crítica sobre la metodología propuesta, comparando la solución propuesta con trabajos relacionados, destacando las propiedades de modelado no cumplidas por otras soluciones y proporcionando lecciones valiosas aprendidas y limitaciones de la metodología presentada.

El principal resultado en la tercera publicación de la tesis (Article 3–`IoT`) fue el desarrollo de una aplicación de evaluación comparativa de hardware de bajo nivel adaptada para SBCs, abordando la necesidad de aplicaciones y conjuntos de datos de evaluación comparativa de bajo nivel en el ámbito de IoT. La evaluación comparativa se denominó *LwHBench* y se centra en medir el rendimiento de CPU, GPU, Memoria y Almacenamiento, teniendo en cuenta las limitaciones de componentes inherentes en los SBCs. La aplicación se implementó específicamente para dispositivos Raspberry Pi. Se ejecutó durante 100 días en un conjunto de 45 dispositivos para generar un extenso conjunto de datos que contiene 2386126 vectores en más de 4GB de datos. Este conjunto de datos allana el camino para la aplicación de técnicas de AI en escenarios donde los datos de rendimiento pueden ayudar en el proceso de gestión de dispositivos. Para mostrar la capacidad inter-escenario del conjunto de datos, el documento también presentó una serie de casos de uso habilitados por AI relacionados con la identificación de dispositivos y el impacto del contexto en el rendimiento. En una configuración práctica, la aplicación de evaluación comparativa se adaptó y aplicó a un escenario que involucra tres dispositivos RockPro64, demostrando su versatilidad y aplicabilidad en entornos del mundo real.

En la cuarta publicación de la tesis (Article 4–`IEEE_COMMAG`), la investigación se adentra en el campo emergente y altamente dinámico del Internet of Battlefield Things (IoBT). Se centró particularmente en el papel crucial de las comunicaciones inalámbricas dentro de este ámbito. En este intrincado escenario de batalla, una miríada de dispositivos, que van desde soldados hasta diversos equipos militares, interactúan en tiempo real, intercambiando información de manera inalámbrica y formando una red compleja de entidades interconectadas. El escenario propuesto profundiza en tres casos de uso principales: identificación de dispositivos IoT, detección de malware y detección de ataques de Falsificación de Datos de Detección de Espectro (SSDF, por sus siglas en inglés).

Para resolver estos casos de uso, se introdujo un framework de modelado de comportamiento IoT, denominado *SpecForce*, que fue meticulosamente diseñado para mejorar la seguridad de los sensores de espectro IoBT, componentes cruciales en el monitoreo del

espectro de frecuencias, la transmisión sobre bandas desocupadas, la intercepción de transmisiones enemigas y la decodificación de información valiosa. En este escenario del mundo real, la identificación individual de dispositivos es esencial para evitar posibles ataques basados en manipulaciones de identidad. *SpecForce* se destaca como una solución robusta, empleando modelado de comportamiento de dispositivos junto con técnicas de ML/DL. El framework fue hábil al considerar fuentes de datos heterogéneas, mejorando su capacidad para detectar y mitigar eficazmente una amplia gama de amenazas cibernéticas. Se hizo hincapié en garantizar la integridad y fiabilidad de las comunicaciones dentro del escenario de batalla, un aspecto crítico considerando la escasez de espectro y el creciente número de dispositivos IoBT. El framework incluyó un módulo de ciberseguridad basado en AI que emplea algoritmos de clasificación de ML/DL para identificar diferentes sensores de espectro IoBT basados en dispositivos RPi. El documento proporciona un análisis exhaustivo de varios modelos de ML/DL para la clasificación. Los resultados indican que Random Forest y XGBoost son los modelos con mejor rendimiento, logrando más del 91% de TPR. El documento también discute un caso de uso que demuestra la capacidad del sistema para identificar de manera única 25 sensores de espectro IoBT, abordando ataques enfocados en la identidad y mejorando la seguridad.

Como se comentó anteriormente, *SpecForce* estaba equipado con otros enfoques de ciberseguridad utilizando monitoreo de eventos de kernel y comportamiento de llamadas al sistema para detectar ataques cibernéticos de nivel superior. En el contexto de la detección de ataques SSDF, el framework permite el monitoreo de llamadas al sistema de sensores de espectro IoBT, con el objetivo de detectar varios ataques SSDF. Las llamadas al sistema se procesan para generar vectores de características que modelan las actividades de detección de espectro, utilizando algoritmos de detección de anomalías para distinguir entre comportamientos normales y maliciosos. Los resultados muestran un alto rendimiento en el reconocimiento de comportamientos normales, con más del 99% de Tasa de Negativos Verdaderos (TNR) y un TPR loable de más del 92% para la detección de ataques SSDF. Además, en lo que respecta a la detección de malware heterogéneo (botnets, puertas traseras, etc.), se logró un alto rendimiento al monitorear eventos de kernel combinados con detección de anomalías de ML/DL, con un TPR del 90% y un TNR del 96%.

En el contexto de ataques adversariales, la siguiente publicación de la Tesis Doctoral (Article 5–FGCS) arrojó luz sobre las posibles vulnerabilidades y amenazas que pueden comprometer la integridad de los mecanismos de modelado e identificación de dispositivos. Discute varios vectores de ataque, ilustrando cómo las entidades maliciosas podrían manipular o eludir mecanismos de identificación basados en hardware para alcanzar sus nefastos objetivos. El artículo subraya la necesidad de estrategias de defensa integrales capaces de mitigar los riesgos asociados con ataques adversariales, asegurando la robustez de los procesos de identificación de dispositivos. El primer resultado principal de este trabajo fue la mejora en los resultados de identificación alcanzados en trabajos anteriores. Utilizando enfoques de series temporales combinados con modelos de DL, los resultados de identificación se incrementaron a un TPR promedio de +0.96 con un TPR mínimo de 0.80 en los 45 dispositivos RPi utilizados para la validación, aprovechando un modelo combinado de LSTM+1D-CNN. Con respecto a ataques adversariales, tanto ataques centrados en el contexto como en ML/DL se aplican para evaluar la robustez del modelo de identificación de dispositivos. Se hace una mención específica de un ataque de contexto basado en la temperatura, que, curiosamente, se encontró ineficaz para interrumpir el proceso de identificación de dispositivos, ya que el aislamiento de hardware durante la recolección de datos ya estaba considerando la mitigación del impacto del contexto. Sin embargo, el documento reconoce el éxito de ciertos ataques de evasión de ML/DL de última generación, como BIM,

MIM y JSMA.

En el lado de la defensa, el documento proporciona una exploración exhaustiva de varias estrategias y metodologías destinadas a proteger los dispositivos IoT de amenazas adversariales. Profundiza en enfoques basados en ML y contexto, evaluando su efectividad para mejorar la seguridad y fiabilidad del modelado e identificación de dispositivos. Los ataques basados en contexto centrados en la temperatura son ineficaces debido a las medidas de estabilidad y aislamiento de hardware tomadas durante la recolección de datos. En cuanto a las defensas contra ataques de evasión de ML/DL, se aplican la destilación de conocimientos y el entrenamiento adversarial para reducir el impacto de los ataques. Los resultados destacan el papel crítico de estos mecanismos de defensa para mantener la integridad de los ecosistemas IoT, asegurando que los dispositivos sean identificados con precisión y que las entidades maliciosas sean frustradas. En términos de rendimiento, la tasa de éxito de los ataques se redujo de 0.88 a 0.17 en el peor de los casos sin causar una degradación sustancial en el rendimiento. Finalmente, se utilizan varias métricas de seguridad para evaluar la resiliencia de las redes neuronales contra perturbaciones y variaciones adversarias en la entrada. Se discuten métricas como el puntaje CLEVER, la sensibilidad a la pérdida y la robustez empírica [40], proporcionando ideas sobre cómo se puede cuantificar y evaluar la robustez de los modelos de ML.

La última publicación de la tesis doctoral (Article 6–COSE) se centró en el problema de la autenticación individual. Propuso un framework de autenticación que utilizaba datos de rendimiento del hardware y modelos autoencoder basados en transformers. El diseño del framework está respaldado por un modelo de amenazas que describe los desafíos de seguridad encontrados al implementar la autenticación basada en hardware en contextos IoT. Como en trabajos anteriores, se monitorearon componentes clave del hardware, como la CPU, GPU, RAM y almacenamiento, para la recolección de datos de modelado. Estas huellas fueron utilizadas como datos de series temporales, aplicando ventanas de tiempo de 10 a 100 valores. Las series temporales generadas se utilizan entonces para entrenar modelos transformers para la detección de anomalías, adaptados a cada dispositivo individual, con el objetivo de representarlo y autenticarlo con precisión. La efectividad del framework se demostró además a través de su aplicación en un sistema de crowdsensing de espectro utilizando dispositivos Raspberry Pi. Los modelos transformers se compararon con enfoques LSTM y 1D-CNN en términos de rendimiento. Aquí, en una serie de experimentos rigurosos que involucraron 45 dispositivos para validación, cada modelo transformador del dispositivo demostró ser capaz de autenticarlo con precisión. En contraste, otros enfoques no fueron capaces de autenticar de manera única todos los dispositivos. El enfoque logró una impresionante TPR promedio de 0.74 ± 0.13 y mantuvo un FPR máximo promedio de 0.06 ± 0.09, subrayando su potencial para mejorar significativamente la autenticación, la seguridad y la confiabilidad en aplicaciones de crowdsensing. Además, también se analizó el uso de recursos de los diferentes enfoques probados, confirmando uno de los principales inconvenientes de los modelos transformers; esta fue la técnica de ML/DL que utilizó más recursos en términos de tiempo y memoria.

V Conclusiones y trabajo futuro

Esta tesis proporciona un examen exhaustivo y completo del campo del fingerprinting del comportamiento de dispositivos, con un enfoque específico en su aplicación dentro del ámbito de la ciberseguridad, y un énfasis matizado en la identificación y autenticación de dispositivos de placa única y IoT. La investigación comienza con una revisión extensa y sistemática del panorama actual, capturando la amplitud y profundidad de las soluciones

de fingerprinting del comportamiento de dispositivos que se han explorado e implementado dentro del dominio de la ciberseguridad. Esta fase inicial de exploración sirve para sentar una base sólida de conocimiento, desentrañando las complejidades de varias fuentes de datos, técnicas de fingerprinting, escenarios de aplicación y los conjuntos de datos que prevalecen dentro de las esferas académicas y prácticas de este campo. A partir de esta exploración, se deriva un nuevo conjunto de lecciones y tendencias de la literatura, dando una visión holística de trabajos anteriores. Sin embargo, la principal novedad desde una perspectiva de investigación es la lista de desafíos identificados en la literatura, que no estaban definidos antes y allanaron el camino para futuras investigaciones.

A medida que la historia avanza, el foco se desplaza a la implementación práctica de la identificación de dispositivos, con la propuesta de una metodología clara y detallada para identificar de manera única dispositivos IoT. Este proceso está intrínsecamente ligado a las capacidades proporcionadas por las técnicas de ML y DL. Estas herramientas computacionales avanzadas pueden ser aprovechadas para descifrar las sutiles matices del comportamiento del hardware, asegurando un alto nivel de autenticidad e integridad para dispositivos integrados dentro de una red. La importancia de este proceso no puede ser exagerada, ya que juega un papel fundamental en la salvaguarda de la seguridad y confiabilidad de dispositivos interconectados, formando un componente crítico de la infraestructura de ciberseguridad más amplia. Para resaltar esta importancia, la metodología presentada se compara con otros trabajos en el campo, donde no se siguió un conjunto metodológico de pasos. Las soluciones anteriores no fueron capaces de realizar una identificación completa en el escenario de dispositivo real utilizado para la validación. Por lo tanto, la metodología se convierte en el procedimiento de vanguardia para desarrollar una solución funcional de identificación individual.

Construyendo sobre la metodología de identificación establecida, la narrativa profundiza en el aspecto crítico de la recolección de datos, presentando un enfoque integral para la adquisición sistemática de datos de rendimiento del hardware. Generar los conjuntos de datos requeridos para el fingerprinting de comportamiento posterior es esencial. Estos conjuntos de datos ayudan a asegurar que los modelos de identificación se entrenen y validen con datos que son tanto extensos como precisos. La meticulosa atención al detalle en este proceso asegura la fiabilidad de los datos, estableciendo un alto estándar para la calidad de la información utilizada en el fingerprinting del comportamiento de dispositivos. Como resultado de la herramienta propuesta para la recolección de datos, se genera un conjunto de datos exhaustivo para ser aplicado en los siguientes pasos de la tesis. Este conjunto de datos se publica para ser utilizado por la comunidad de investigación en el campo, junto con la aplicación de benchmark empleada para generar los datos. Este es uno de los primeros conjuntos de datos públicos de datos de rendimiento del hardware que se centra en el problema de identificación.

Con una base sólida de datos en su lugar, la exploración luego navega hacia la integración de la identificación individual de dispositivos dentro del ecosistema más amplio de soluciones de ciberseguridad basadas en comportamiento. Esta integración es fundamental, ya que asegura que los procedimientos de identificación de dispositivos no operen de manera aislada, sino que estén vinculados sin problemas con otros frameworks de seguridad generales. Para esta integración, el enfoque se coloca estratégicamente en el Internet de las Cosas del Campo de Batalla. Se emplea un escenario del mundo real para integrar la solución de identificación basada en hardware con un monitoreo de comportamiento de nivel superior para la detección de malware y ataques de falsificación de datos de espectro. Específicamente, el enfoque unificado monitorea hardware, eventos de kernel y llamadas al sistema para proporcionar una solución de seguridad unificada basada en comportamiento.

Este enfoque holístico mejora la resiliencia del entorno, fortaleciendo sus defensas contra una miríada de amenazas y vulnerabilidades cibernéticas, y asegurando un entorno digital robusto y seguro. El framework unificado demuestra experimentalmente sus capacidades para detectar diferentes muestras de malware y ataques centrados en datos de espectro, así como realizar la identificación individual de los sensores desplegados. Hasta donde sabemos, este es el primer framework que combina estas capacidades de ciberseguridad juntas.

A medida que el trabajo alcanza su culminación, el enfoque se dirige a la seguridad de los modelos de identificación de dispositivos en sí mismos, abordando específicamente los desafíos planteados por los ataques adversarios. Se analiza el panorama de los ataques adversarios, centrándose particularmente en las amenazas conscientes del contexto y centradas en ML/DL que representan riesgos significativos para la integridad de las metodologías de identificación de dispositivos. Luego, el enfoque se adentra en estrategias y metodologías diseñadas para mejorar la resiliencia de los modelos de identificación basados en el comportamiento del hardware. Se pone un énfasis particular en contrarrestar amenazas adversarias sofisticadas, incluyendo ataques basados en contexto y centrados en ML/DL. Los ataques basados en contexto son ineficaces contra la solución, pero los ataques de evasión dirigidos a los modelos de identificación logran altas tasas de éxito. Por lo tanto, se aplican técnicas de defensa basadas en entrenamiento adversario y destilación de conocimiento. Esto asegura que las metodologías de identificación de dispositivos sigan siendo confiables, seguras y efectivas, incluso frente a amenazas cibernéticas en evolución y complejas. Este es uno de los primeros trabajos que demuestra experimentalmente la efectividad de los ataques adversarios en modelos de ML para la identificación de dispositivos IoT, y también de métodos de defensa de vanguardia.

Finalmente, la investigación explora el problema más complejo de la autenticación de dispositivos individuales, donde solo se pueden aprovechar los datos de un dispositivo para la generación del modelo. La metodología aplicada para la identificación se ajusta para su aplicación en la autenticación. La principal diferencia es el cambio del modelo clasificador por un modelo de detección de anomalías por dispositivo. Sin embargo, se necesitan modelos de ML/DL más potentes para resolver este problema, ya que las distribuciones de datos se superponen entre dispositivos del mismo modelo. Para resolver este problema, se emplean modelos transformers. Utilizando ventanas de tiempo grandes para el procesamiento de datos (100 valores), el enfoque basado en transformers mejora los resultados logrados por modelos de vanguardia anteriores, como LSTM, 1D-CNN y su combinación. En contraste, se emplea más tiempo de entrenamiento y memoria en el proceso de generación del modelo. Este resultado confirma la efectividad de la arquitectura transformadora en un área novedosa. A diferencia de las arquitecturas de modelos existentes en la literatura, aborda con éxito el problema de autenticar individualmente dispositivos IoT basados en su rendimiento de hardware en el escenario experimental estudiado.

Mirando hacia adelante, hay un vasto horizonte de oportunidades para trabajos futuros basados en los cimientos establecidos por esta tesis en el dominio del fingerprinting del comportamiento de dispositivos. La primera iteración de trabajos futuros podría profundizar en la evaluación de ataques adversarios sobre los modelos transformers no supervisados empleados para resolver el problema de identificación individual, y la consecuente aplicación de técnicas de defensa. Estos resultados cerrarán el trabajo sobre el tema de autenticación de manera similar a la metodología aplicada para el problema de identificación individual.

Otra avenida prometedora es expandir el alcance del fingerprinting del comportamiento de dispositivos para abarcar una gama más amplia de dispositivos y contextos. El trabajo actual se ha centrado predominantemente en dispositivos de placa única e IoT. Sin em-

bargo, los principios y metodologías desarrollados podrían adaptarse y aplicarse a otros tipos de dispositivos y redes, como sistemas de control industrial, sistemas automotrices y dispositivos de hogares inteligentes. Investigaciones futuras podrían explorar las sutilezas y requisitos específicos de estos diferentes contextos, adaptando las técnicas de fingerprinting para ajustarse a las características únicas de cada categoría de dispositivo y escenario de uso. Uno de los ámbitos cruciales destacados para futuras exploraciones es la búsqueda continua de nuevas y diversas fuentes de datos. El campo se beneficiaría significativamente de ampliar el alcance de la recolección de datos, capturando una gama más amplia de comportamientos de dispositivos y asegurando un conjunto de datos más rico y completo para análisis. Este esfuerzo no solo se limita a aumentar la cantidad de datos sino también enfatiza la importancia de mejorar la calidad y fiabilidad de los datos recopilados. Trabajos futuros en esta área podrían explorar metodologías avanzadas de recolección de datos, tecnologías innovadoras de sensores y técnicas novedosas de preprocesamiento de datos, todo con el objetivo de asegurar que los datos utilizados para el fingerprinting del comportamiento de dispositivos sean de la más alta calidad.

Construyendo sobre el tema de datos, hay un claro llamado para el desarrollo y refinamiento de técnicas de fingerprinting. El futuro tiene potencial para la exploración de nuevos algoritmos, modelos de ML y arquitecturas de DL, cada uno ofreciendo capacidades y ventajas únicas para la identificación de dispositivos. El Aprendizaje Federado (FL), y más concretamente el FL descentralizado, es una de estas áreas prometedoras que vale la pena explorar en los próximos años. La continua evolución del poder computacional y las tecnologías de ML/DL abre posibilidades emocionantes para crear modelos de fingerprinting del comportamiento de dispositivos más sofisticados y precisos, capaces de discernir incluso las sutilezas más sutiles en el comportamiento de dispositivos.

Abordando el desafío de los ataques adversarios, hay una necesidad apremiante de desarrollar contramedidas robustas y estrategias de mitigación. Trabajos futuros en esta área podrían explorar enfoques innovadores para mejorar la resiliencia de los modelos de fingerprinting del comportamiento de dispositivos, con un enfoque específico en contrarrestar amenazas adversarias sofisticadas, incluyendo ataques basados en contexto y centrados en ML/DL. Esto implica no solo fortalecer los modelos de identificación sino también desarrollar mecanismos integrales de detección y respuesta a amenazas, asegurando la fiabilidad y seguridad a largo plazo de las metodologías de identificación de dispositivos.

Otra área crítica para trabajos futuros radica en mejorar la adaptabilidad de los modelos de fingerprinting del comportamiento de dispositivos. Con el rápido ritmo de avance tecnológico, los dispositivos están en constante evolución, y sus patrones de comportamiento pueden cambiar con el tiempo debido a actualizaciones de software, modificaciones de hardware o cambios en los patrones de uso. Investigaciones futuras podrían centrarse en desarrollar modelos de fingerprinting capaces de adaptarse a estos cambios, asegurando que sigan siendo precisos y confiables a lo largo del ciclo de vida del dispositivo. Esto podría implicar la integración de técnicas de aprendizaje en línea, enfoques de aprendizaje continuo o metodologías de aprendizaje por transferencia para permitir que los modelos actualicen y refinan sus perfiles de fingerprinting en respuesta a los cambios observados en el comportamiento de los dispositivos.

Finalmente, hay un potencial significativo para trabajos futuros en la integración de soluciones de fingerprinting del comportamiento de dispositivos con otras herramientas y frameworks de ciberseguridad. Esta tesis ha sentado las bases para dicha integración, demostrando los beneficios potenciales de combinar el fingerprinting del comportamiento de dispositivos con otras soluciones de seguridad basadas en comportamiento. Investigaciones futuras podrían basarse en esto, explorando formas de optimizar aún más el proceso de

integración, mejorar la interoperabilidad y maximizar las sinergias entre diferentes herramientas de seguridad. Esto podría llevar a la creación de frameworks de ciberseguridad más holísticos y resilientes, proporcionando protección integral contra una amplia gama de amenazas cibernéticas.

Bibliography

[1] K. Riad, T. Huang, and L. Ke, "A dynamic and hierarchical access control for IoT in multi-authority cloud storage," *Journal of Network and Computer Applications*, vol. 160, p. 102633, 2020.

[2] R. Fayos-Jordan, S. Felici-Castell, J. Segura-Garcia, J. Lopez-Ballester, and M. Cobos, "Performance comparison of container orchestration platforms with low cost devices in the fog, assisting internet of things applications," *Journal of Network and Computer Applications*, vol. 169, p. 102788, 2020.

[3] A. L. Perales Gómez, L. Fernández Maimó, A. Huertas Celdran, F. J. García Clemente, C. Cadenas Sarmiento, C. J. Del Canto Masa, and R. Méndez Nistal, "On the generation of anomaly detection datasets in industrial control systems," *IEEE Access*, vol. 7, pp. 177 460–177 473, 2019.

[4] S. Jagdale, "The Role of Hardware Root of Trust in Edge Devices," https://www.eetimes.eu/the-role-of-hardware-root-of-trust-in-edge-devices/, 2022, [Online; accessed 21-June-2022].

[5] E. Montalbano, "Bluetooth Spoofing Bug Affects Billions of IoT Devices," https://threatpost.com/bluetooth-spoofing-bug-iot-devices/159291/, 2020, [Online; accessed 21-June-2022].

[6] R. Francese, M. Frasca, and M. Risi, "Are iobt services accessible to everyone?" *Pattern Recognition Letters*, vol. 147, pp. 71–77, 2021.

[7] Y. Liu, J. Wang, J. Li, H. Song, T. Yang, S. Niu, and Z. Ming, "Zero-bias deep learning for accurate identification of internet-of-things (iot) devices," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2627–2634, 2020.

[8] A. Fuentes, "Human niche, human behaviour, human nature," *Interface Focus*, vol. 7, no. 5, p. 20160136, 2017.

[9] N. Shone, Q. Shi, M. Merabti, and K. Kifayat, "Misbehaviour monitoring on system-of-systems components," in *2013 International Conference on Risks and Security of Internet and Systems*, Oct. 2013, pp. 1–6.

[10] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2019. DOI: 10.1109/TMC.2018.2866249

[11] S. Marchal, M. Miettinen, T. D. Nguyen, A. Sadeghi, and N. Asokan, "AuDI: Toward autonomous IoT device-type identification using periodic communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019. DOI: 10.1109/JSAC.2019.2904364

[12] K. Haefner and I. Ray, "ComplexIoT: Behavior-based trust for IoT networks," in *1st IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications*, Dec. 2019, pp. 56–65. DOI: 10.1109/TPS-ISA48467.2019.00016

[13] G. Manco, E. Ritacco, P. Rullo, L. Gallucci, W. Astill, D. Kimber, and M. Antonelli, "Fault detection and explanation through big data analysis on sensor streams," *Expert Systems with Applications*, vol. 87, pp. 141–156, 2017.

[14] M. Miettinen, S. Marchal, I. Hafeez, T. Frassetto, N. Asokan, A. Sadeghi, and S. Tarkoma, "IoT Sentinel: Automated device-type identification for security enforcement in IoT," in *37th IEEE International Conference on Distributed Computing Systems*, June 2017, pp. 2511–2514.

[15] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A. Sadeghi, "DÏot: A federated self-learning anomaly detection system for IoT," in *39th IEEE International Conference on Distributed Computing Systems*, July 2019. ISSN 1063-6927 pp. 756–767. DOI: 10.1109/ICDCS.2019.00080

[16] X. Liu, B. Xiao, S. Zhang, and K. Bu, "Unknown tag identification in large RFID systems: An efficient and complete solution," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1775–1788, 2014.

[17] J. Ortiz, C. Crawford, and F. Le, "DeviceMien: Network device behavior modeling for identifying unknown IoT devices," in *International Conference on Internet of Things Design and Implementation*, Apr. 2019. ISBN 9781450362832 p. 106–117. [Online]. Available: https://doi.org/10.1145/3302505.3310073. DOI: 10.1145/3302505.3310073

[18] H. Jafari, O. Omotere, D. Adesina, H. Wu, and L. Qian, "IoT devices fingerprinting using deep learning," in *2018 IEEE Military Communications Conference*, Oct. 2018. ISSN 2155-7578 pp. 1–9. DOI: 10.1109/MILCOM.2018.8599826

[19] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S. K. Ng, "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," in *28th International Conference on Artificial Neural Networks*, Sept. 2019, pp. 703–716.

[20] T. Sabhanayagam, "A comparative analysis to obtain unique device fingerprinting," in *Proceedings of International Conference on Deep Learning, Computing and Intelligence*. Springer, 2022, pp. 349–354.

[21] C. M. Ahmed and A. P. Mathur, "Hardware identification via sensor fingerprinting in a cyber physical system," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion*, July 2017, pp. 517–524.

[22] A. Al-Omary, A. Othman, H. M. AlSabbagh, and H. Al-Rizzo, "Survey of hardware-based security support for iot/cps systems," *KnE Engineering*, pp. 52–70, 2018.

[23] D. Marabissi, L. Mucchi, and A. Stomaci, "Iot nodes authentication and id spoofing detection based on joint use of physical layer security and machine learning," *Future Internet*, vol. 14, no. 2, p. 61, 2022.

[24] Z. Chen, J. Liu, Y. Shen, M. Simsek, B. Kantarci, H. T. Mouftah, and P. Djukic, "Machine learning-enabled iot security: Open issues and challenges under advanced persistent threats," *ACM Computing Surveys (CSUR)*, 2022.

[25] I. Sanchez-Rola, I. Santos, and D. Balzarotti, "Clock around the clock: Time-based device fingerprinting," in *2018 ACM SIGSAC Conference on Computer and Communications Security*, Jan. 2018, pp. 1502–1514.

[26] P. M. Sánchez Sánchez, J. M. Jorquera Valero, A. Huertas Celdrán, G. Bovet, M. Gil Pérez, and G. Martínez Pérez, "A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1048–1077, 2021. DOI: 10.1109/COMST.2021.3064259

[27] B. Varghese, N. Wang, D. Bermbach, C.-H. Hong, E. D. Lara, W. Shi, and C. Stewart, "A survey on edge performance benchmarking," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–33, 2021.

[28] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral fingerprinting of iot devices," in *Proceedings of the 2018 workshop on attacks and solutions in hardware security*, 2018, pp. 41–50.

[29] S. Thouti, N. Venu, D. R. Rinku, A. Arora, and N. Rajeswaran, "Investigation on identify the multiple issues in iot devices using convolutional neural network," *Measurement: Sensors*, vol. 24, p. 100509, 2022.

[30] S. Zhang, Z. Wang, J. Yang, D. Bai, F. Li, Z. Li, J. Wu, and X. Liu, "Unsupervised iot fingerprinting method via variational auto-encoder and k-means," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[31] J. Kocoń, I. Cichecki, O. Kaszyca, M. Kochanek, D. Szydło, J. Baran, J. Bielaniewicz, M. Gruza, A. Janz, K. Kanclerz *et al.*, "Chatgpt: Jack of all trades, master of none," *Information Fusion*, p. 101861, 2023.

[32] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[33] O. Ibitoye, R. Abou-Khamis, A. Matrawy, and M. O. Shafiq, "The threat of adversarial attacks on machine learning in network security–a survey," *arXiv preprint arXiv:1911.02621*, 2019.

[34] Z. Bao, Y. Lin, S. Zhang, Z. Li, and S. Mao, "Threat of adversarial attacks on dl-based iot device identification," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 9012–9024, 2021.

[35] A. Namvar, C. Thapa, S. S. Kanhere, and S. Camtepe, "Evaluating the security of machine learning based iot device identification systems against adversarial examples,"

in *International Conference on Service-Oriented Computing*. Springer, 2021, pp. 800–810.

[36] T. Laor, N. Mehanna, A. Durey, V. Dyadyuk, P. Laperdrix, C. Maurice, Y. Oren, R. Rouvoy, W. Rudametkin, and Y. Yarom, "Drawnapart: A device identification technique based on remote gpu fingerprinting," *arXiv preprint arXiv:2201.09956*, 2022.

[37] K. Sadeghi, A. Banerjee, and S. K. Gupta, "A system-driven taxonomy of attacks and defenses in adversarial machine learning," *IEEE transactions on emerging topics in computational intelligence*, vol. 4, no. 4, pp. 450–467, 2020.

[38] I. Ali, S. Sabir, and Z. Ullah, "Internet of things security, device authentication and access control: a review," *arXiv preprint arXiv:1901.07309*, 2019.

[39] S. Rajendran, R. Calvo-Palomino, M. Fuchs, B. V. den Bergh, H. Cordobés, D. Giustiniano, S. Pollin, and V. Lenders, "Electrosense: Open and Big Spectrum Data," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 210–217, January 2018.

[40] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Adversarial machine learning attacks and defense methods in the cyber security domain," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–36, 2021.

[41] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," *arXiv preprint arXiv:2202.07125*, 2022.

[42] P. M. S. Sánchez, J. M. J. Valero, A. H. Celdrán, G. Bovet, M. G. Pérez, and G. M. Pérez, "Lwhbench: A low-level hardware component benchmark and dataset for single board computers," *Internet of Things*, vol. 22, p. 100764, 2023.

Other publications/works

In addition to the main publications composing the PhD Thesis, several works have been published due to the research activities carried toward the PhD completion. Apart from the *six JCR articles* composing the present thesis, *sixteen other JCR journal articles*, *thirteen conference articles*, *four book chapters*, and *one 5G PPP technical report* have been co-authored. Besides, *three tutorial* sessions have been given at different conferences. All these publications can be framed in different collaborations and side-projects developed during the last years:

- Conference tutorial sessions directly based on the PhD thesis content:
  - (*Tutorial*) Alberto Huertas, **Pedro M. Sánchez Sánchez**, Muriel Franco, Bruno Rodrigues, Gérôme Bovet, Gregorio Martínez, Burkhard Stiller. (2021). Intelligent Behavioral Fingerprinting - From Theory to Practice. 17th IEEE International Conference on Network and Service Management (CNSM 2021).
  - (*Tutorial*) Alberto Huertas, **Pedro M. Sánchez Sánchez**, Muriel Franco, Gérôme Bovet, Gregorio Martínez, Burkhard Stiller. (2022). Theoretical and Practical Intelligent Behavioral Fingerprinting. IEEE/IFIP Network Operations and Management Symposium (NOMS 2022).
- Collaboration with the University of Zurich and armasuisse S&T in AI Trustworthiness, Federated Learning, and behavior fingerprinting for cyberattack detection:
  - (*Journal*) Rey, V., **Sánchez Sánchez, P. M.**, Huertas Celdrán, A., & Bovet, G. (2022). Federated learning for malware detection in iot devices. Computer Networks, 204, 108693.
  - (*Journal*) Huertas Celdrán, A., **Sánchez Sánchez, P. M.**, Castillo, M. A., Bovet, G., Martínez Pérez, G., & Stiller, B. (2022). Intelligent and behavioral-based detection of malware in IoT spectrum sensors. International Journal of Information Security, 1-21.
  - (*Journal*) **Sánchez Sánchez, P. M.**, Huertas Celdrán, A., Schenk, T., Iten, A. L. B., Bovet, G., Martínez Pérez, G., & Stiller, B. (2022). Studying the Robustness of Anti-Adversarial Federated Learning Models Detecting Cyberattacks in IoT Spectrum Sensors. IEEE Transactions on Dependable and Secure Computing. In press.

- (*Journal*) Huertas Celdrán, A., **Sánchez Sánchez, P. M**., Feng, C., Bovet, G., Martínez Pérez, G., & Stiller, B. (2023). Privacy-preserving and Syscall-based Intrusion Detection System for IoT Spectrum Sensors Affected by Data Falsification Attacks. IEEE Internet of Things Journal, 10 (10), 8408-8415.
- (*Journal*) **Sánchez Sánchez, P. M**., Huertas Celdrán, A., Buendía Rubio, J. R., Bovet, G., & Martínez Pérez, G. (2023). Robust Federated Learning for execution time-based device model identification under label-flipping attack. Cluster Computing, 1-12.
- (*Journal*) Huertas Celdrán, A., **Sánchez Sánchez, P. M**., Bovet, G., Martínez Pérez, G., & Stiller, B. (2023). CyberSpec: Behavioral Fingerprinting for Intelligent Attacks Detection on Crowdsensing Spectrum Sensors. IEEE Transactions on Dependable and Secure Computing. In press.
- (*Journal*) Huertas Celdrán, A., **Sánchez Sánchez, P. M.**, von der Assen, J., Shushack, D., Gómez, Á. L. P., Bovet, G., ... & Stiller, B. (2023). Behavioral Fingerprinting to Detect Ransomware in Resource-constrained Devices. Computers & Security, 103510.
- (*Journal*) **Sánchez Sánchez, P. M.**, Huertas Celdrán, A., Xie, N., Bovet, G., Martínez Pérez, G., & Stiller, B. (2024). FederatedTrust: A solution for trustworthy federated learning. Future Generation Computer Systems, 152, 83-98.
- (*Conference*) **Sánchez Sánchez, P. M**., Huertas Celdran, A., Bovet, G., Martínez Pérez, G., & Stiller, B. (2021). Secure Crowdsensing Platforms Through Device Behavior Fingerprinting. Jornadas Nacionales en Investigación en Ciberseguridad (JNIC).
- (*Conference*) Huertas Celdrán, A., **Sánchez Sánchez, P. M.**, Scheid, E. J., Besken, T., Bovet, G., Martínez Pérez, G., & Stiller, B. (2022, April). Policy-based and Behavioral Framework to Detect Ransomware Affecting Resource-constrained Sensors. In NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium (pp. 1-7). IEEE.
- (*Conference*) Huertas Celdrán, A., **Sánchez Sánchez, P. M**., Bovet, G., Martínez Pérez, G., & Stiller, B. (2022, May). Intelligent Fingerprinting to Detect Data Leakage Attacks on Spectrum Sensors. In ICC 2022-IEEE International Conference on Communications (pp. 4080-4085). IEEE.
- (*Conference*) Huertas Celdrán, A., Bauer, J., Demirci, M., Leupp, J., Franco, M. F., **Sánchez Sánchez, P. M**., ... & Stiller, B. (2022, October). RITUAL: a Platform Quantifying the Trustworthiness of Supervised Machine Learning. In 2022 18th International Conference on Network and Service Management (CNSM) (pp. 364-366). IEEE.
- (*Conference*) Huertas Celdrán, A., Kreischer, J., Demirci, M., Leupp, J., **Sanchez, P. M**., Franco, M. F., ... & Stiller, B. (2023). A Framework Quantifying Trustworthiness of Supervised Machine and Deep Learning Models. In SafeAI2023: The AAAI's Workshop on Artificial Intelligence Safety (pp. 2938-2948).
- (*Conference*) Huertas Celdrán, A., von der Assen, J., Moser, K., **Sánchez Sánchez, P. M.**, Bovet, G., Martínez Pérez, G., & Stiller, B. (2023, May). Early Detection of Cryptojacker Malicious Behaviors on IoT Crowdsensing Devices. In NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium (pp. 1-8). IEEE.

  - (*Conference*) **Sánchez Sánchez, P. M.**, Huertas Celdrán, A., Bovet, G., Martínez Pérez, G., & Stiller, B. (2023, June). A Trustworthy Federated Learning Framework for Individual Device Identification. In 2023 JNIC Cybersecurity Conference (JNIC) (pp. 1-8). IEEE.
  - (*Conference*) J. von der Assen, A. H. Celdrán, **P. M. S. Sánchez**, J. Cedeño, G. Bovet, G. M. Pérez, and B. Stiller, (2023) A Lightweight Moving Target Defense Framework for Multi-purpose Malware Affecting IoT Devices, in IEEE International Conference on Communications, ICC 2023.
  - (*Conference*) **Sánchez Sánchez, P. M.**, Huertas Celdrán, A., Beltrán, E. T. M., Wassink, R., Bovet, G., Martínez Pérez, G., & Stiller, B. (2023) Stealth Spectrum Sensing Data Falsification Attacks Affecting IoT Spectrum Monitors on the Battlefield, in 2023 IEEE Military Communications Conference (MILCOM 2023) (pp. 1-6). IEEE.
  - (*Conference*) J. von der Assen, A. H. Celdrán, J. Luechinger, **P. M. S. Sánchez**, G. Bovet, G. M. Pérez, and B. Stiller, (2023) RansomAI: AI-powered Ransomware for Stealthy Encryption, in IEEE Global Communications Conference (GLOBECOM 2023) (pp. 1-6). IEEE.
  - (*Chapter*) Huertas Celdrán, A., **Sánchez Sánchez, P. M**., Sisi, F., Bovet, G., Martínez Pérez, G., & Stiller, B. (2022). Creation of a Dataset Modeling the Behavior of Malware Affecting the Confidentiality of Data Managed by IoT Devices. In Robotics and AI for Cybersecurity and Critical Infrastructure in Smart Cities (pp. 193-225). Cham: Springer International Publishing.

- Collaboration in 5GZORRO (Zero-tOuch secuRity and tRust for ubiquitous cOmputing and connectivity in 5G networks) H2020 project and José María's PhD work:

  - (*Journal*) Jorquera Valero, J. M., **Sánchez Sánchez, P. M**., Lekidis, A., Hidalgo, J. F., Gil Pérez, M., Siddiqui, M. S., ... & Martínez Pérez, G. (2022). Design of a Security and Trust Framework for 5G Multi-domain Scenarios. J. Netw. Syst. Manag., 30(1), 7.
  - (*Journal*) Jorquera Valero, J. M., **Sánchez Sánchez, P. M**., Gil Pérez, M., Huertas Celdrán, A., & Martínez Pérez, G. (2022). Toward pre-standardization of reputation-based trust models beyond 5G. Computer Standards & Interfaces, 81, 103596.
  - (*Journal*) Jorquera Valero, J. M., **Sánchez Sánchez, P. M.**, Gil Pérez, M., Huertas Celdrán, A., & Martinez Perez, G. (2023). Cutting-Edge Assets for Trust in 5G and Beyond: Requirements, State of the Art, Trends, and Challenges. ACM Computing Surveys, 55(11), 1-36.
  - (*Journal*) Jorquera Valero, J. M., **Sánchez Sánchez, P. M.**, Gil Pérez, M., Huertas Celdrán, A., & Martínez Pérez, G. (2023). Trust-as-a-Service: A reputation-enabled trust framework for 5G network resource provisioning. Computer Communications, 211, 229-238.
  - (*Chapter*) Jorquera Valero, J. M., **Sánchez Sánchez, P. M**., Lekidis, A., Martins, P., Diogo, P., Gil Pérez, M., ... & Martínez Pérez, G. (2022). Trusted Execution Environment-enabled platform for 5G security and privacy enhancement. Security and Privacy Preserving for IoT and 5G Networks: Techniques, Challenges, and New Directions, 203-223.

  - (*Technical Report*) 5G PPP Technology Board. (2021). AI and ML–Enablers for beyond 5G Networks. `https://zenodo.org/record/4299895`

- Collaboration in EU-GUARDIAN (European framework and proofs-of-concept for the intelliGent aUtomAtion of cybeR Defence Incident mAnagemeNt) European Defence Fund (EDF) project:
  - (*Conference*) Cid, M. I. G., Gil Pérez, M., Jorquera Valero, J. M., Martínez, A. L., Vidal, J. M., Martínez Pérez, G., ..., **Sánchez Sánchez, P. M.**, & Monge, M. A. S. (2023, June). European framework and proofs-of-concept for the intelliGent aUtomAtion of cybeR Defence Incident mAnagemeNt. In 2023 JNIC Cybersecurity Conference (JNIC) (pp. 1-2). IEEE.

- Work in Decentralized Federated Learning and Enrique Martínez's PhD supervision:
  - (*Journal*) Beltrán, E. T. M., Pérez, M. Q., **Sánchez Sánchez, P. M.**, Bernal, S. L., Bovet, G., Gil Pérez, M., Martínez Pérez, G., & Huertas Celdrán, A. (2023). Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. IEEE Communications Surveys & Tutorials, vol. 25, no. 4, pp. 2983-3013.
  - (*Journal*) Beltrán, E. T. M., Peráles Gómez, A. L., Feng, C., **Sánchez Sánchez, P. M.**, Bernal, S. L., Bovet, G., Gil Pérez, M., Martínez Pérez, G., & Huertas Celdrán, A. (2024) Fedstellar: A Platform for Decentralized Federated Learning. Expert Systems with Applications, 242, 122861.
  - (*Conference*) Beltrán, E. T. M., **Sánchez Sánchez, P. M.**, López, S., Bernal, G. B., Gil Pérez, M., Martínez Pérez, G., & Huertas Celdrán, A., (2023) Fedstellar: A platform for training models in a privacy-preserving and decentralized fashion, in International Joint Conference on Artificial Intelligence (IJCAI-23) Demo Track.
  - (*Tutorial*) Alberto Huertas Celdrán, Gregorio Martínez Pérez, Enrique Tomás Martínez Beltrán, **Pedro Miguel Sánchez Sánchez**, Gérôme Bovet, Burkhard Stiller. (2023). Decentralized Federated Learning: Enabling Collaborative AI With Enhanced Trust and Efficiency. 26th European Conference on Artificial Intelligence (ECAI 2023)

- Work in user behavior fingerprinting for authentication, an extension of my Master's thesis:
  - (*Journal*) **Sánchez Sánchez, P. M**., Jorquera Valero, J. M., Zago, M., Huertas Celdrán, A., Maimó, L. F., Bernal, E. L., ... & Martínez Pérez, G. (2020). BEHACOM-a dataset modelling users' behaviour in computers. Data in Brief, 31, 105767.
  - (*Journal*) **Sánchez Sánchez, P. M**., Fernández Maimó, L., Huertas Celdrán, A., & Martínez Pérez, G. (2021). AuthCODE: A privacy-preserving and multi-device continuous authentication architecture based on machine and deep learning. Computers & Security, 103, 102168.
  - (*Conference*) **Sánchez Sánchez, P. M**., Huertas Celdrán, A., Fernández Maimó, L., Martínez Pérez, G., & Wang, G. (2019). Securing smart offices through an intelligent and multi-device continuous authentication system. In Smart City and Informatization: 7th International Conference, iSCI 2019, Guangzhou, China, November 12–15, 2019, Proceedings 7 (pp. 73-85). Springer Singapore.

– (*Chapter*) **Sánchez Sánchez, P. M.**, Jorquera Valero, J. M., Huertas Celdrán, A., & Martínez Pérez, G. (2020). Intelligent User Profiling Based on Sensors and Location Data to Detect Intrusions on Mobile Devices. In Handbook of Research on Intrusion Detection Systems (pp. 1-25). IGI Global.

– (*Chapter*) Jorquera Valero, J. M., **Sánchez Sánchez, P. M.**, Huertas Celdrán, A., & Martínez Pérez, G. (2020). Machine Learning as an Enabler of Continuous and Adaptive Authentication in Multimedia Mobile Devices. In Handbook of Research on Multimedia Cyber Security (pp. 21-47). IGI Global.

Publications composing

the PhD Thesis

1

A Survey on Device Behavior Fingerprinting: Data Sources, Techniques, Application Scenarios, and Datasets

Title:	A Survey on Device Behavior Fingerprinting: Data Sources, Techniques, Application Scenarios, and Datasets
Authors:	Pedro Miguel Sánchez Sánchez, José María Jorquera Valero, Alberto Huertas Celdrán, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez
Journal:	IEEE Communications Surveys & Tutorials
JIF:	33.84 D1 (2021)
Publisher:	IEEE
Volume:	23
Number:	2
Pages:	1048-1077
Year:	2021
Month:	Mar
DOI:	10.1109/COMST.2021.3064259
Status:	Published

Abstract

In the current network-based computing world, where the number of interconnected devices grows exponentially, their diversity, malfunctions, and cybersecurity threats are increasing at the same rate. To guarantee the correct functioning and performance of novel environments such as Smart Cities, Industry 4.0, or crowdsensing, it is crucial to identify the capabilities of their devices (e.g., sensors, actuators) and detect potential misbehavior that may arise due to cyberattacks, system faults, or misconfigurations. With this goal in mind, a promising research field emerged focusing on creating and managing fingerprints that model the behavior of both the device actions and its components. The article at hand studies the recent growth of the device behavior fingerprinting field in terms of application scenarios, behavioral sources, and processing and evaluation techniques. First, it performs a comprehensive review of the device types, behavioral data, and processing and evaluation techniques used by the most recent and representative research works dealing with two major scenarios: device identification and device misbehavior detection. After that, each work is deeply analyzed and compared, emphasizing its characteristics, advantages, and limitations. This article also provides researchers with a review of the most relevant characteristics of existing datasets as most of the novel processing techniques are based

on Machine Learning and Deep Learning. Finally, it studies the evolution of these two scenarios in recent years, providing lessons learned, current trends, and future research challenges to guide new solutions in the area.

Keywords

Behavioral data · cyberattack detection · device behavior datasets · device behavior fingerprinting · device identification · processing and evaluation techniques

1048 IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 23, NO. 2, SECOND QUARTER 2021

A Survey on Device Behavior Fingerprinting: Data Sources, Techniques, Application Scenarios, and Datasets

Pedro Miguel Sánchez Sánchez, José María Jorquera Valero, Alberto Huertas Celdrán, Gérôme Bovet, Manuel Gil Pérez, and Gregorio Martínez Pérez, *Member, IEEE*

***Abstract*—In the current network-based computing world, where the number of interconnected devices grows exponentially, their diversity, malfunctions, and cybersecurity threats are increasing at the same rate. To guarantee the correct functioning and performance of novel environments such as Smart Cities, Industry 4.0, or crowdsensing, it is crucial to identify the capabilities of their devices (e.g., sensors, actuators) and detect potential misbehavior that may arise due to cyberattacks, system faults, or misconfigurations. With this goal in mind, a promising research field emerged focusing on creating and managing fingerprints that model the behavior of both the device actions and its components. The article at hand studies the recent growth of the device behavior fingerprinting field in terms of application scenarios, behavioral sources, and processing and evaluation techniques. First, it performs a comprehensive review of the device types, behavioral data, and processing and evaluation techniques used by the most recent and representative research works dealing with two major scenarios: device identification and device misbehavior detection. After that, each work is deeply analyzed and compared, emphasizing its characteristics, advantages, and limitations. This article also provides researchers with a review of the most relevant characteristics of existing datasets as most of the novel processing techniques are based on Machine Learning and Deep Learning. Finally, it studies the evolution of these two scenarios in recent years, providing lessons learned, current trends, and future research challenges to guide new solutions in the area.**

***Index Terms*—Device behavior fingerprinting, device identification, cyberattack detection, behavioral data, processing and evaluation techniques, device behavior datasets.**

I. INTRODUCTION

PREVISIONS for 2025 estimate nearly 64 billion IoT devices connected to each other into diverse cutting-edge environments such as Smart Cities, Industry 4.0, or crowdsensing (e.g., Flightradar24, OpenSky, ElectroSense), among others [1]. These environments have their own particularities in terms of devices, data, communications, and purposes, which increase the complexity of achieving one of their common challenges: to optimize the performance of devices and provide accurate services. To meet this challenge, the advancement of communication networks and computing paradigms has influenced that behavioral data science evolved from studying theoretical and empirical issues related to human behaviors [2] –its initial scope– to conquer the cyberworld and offer a promising alternative to model device behaviors [3]. Nowadays, a thriving research field within behavior data science focuses on creating device behavior patterns (*fingerprints*) able to optimize their performance and detect potential issues in the early stages [4], [5]. In this context, this article studies the recent growth of the device behavior research field in terms of application scenarios, behavioral sources, and processing and evaluation techniques. Fig. 1 shows an overview of the typical life cycle implemented by the literature, where different devices, techniques, and application scenarios are considered.

The first step to build a device fingerprint is to identify the application scenario where it will be needed. By keeping in mind the goal of optimizing devices and systems performance, the literature has recognized two critical application scenarios. The first one consists in identifying devices with different granularity levels –to differentiate them and fully exploit their capabilities [6]– while the second focuses on detecting cyberattacks [7], malfunction [8], or misbehavior [9] –to mitigate them. The nature of each scenario influences the selection of behavioral sources, data, and techniques employed to create fingerprints since the detection of misbehavior produced by a given cyberattack is different from identifying several IoT devices of the same family. Even in the same application scenario, the behavioral data might be different as well; this is the case of some cyberattacks affecting network communications [10], while others impact the CPU usage [11].

In both application scenarios, the literature contains an extensive number of works where device fingerprinting has been applied [3], [4], [12], [13], [14], [15], [16]. On the one hand and in terms of device identification, behavioral data science has dramatically improved the limitations of traditional solutions, mainly focused on using names, identifiers,

Manuscript received August 7, 2020; revised December 9, 2020 and January 26, 2021; accepted March 3, 2021. Date of publication March 11, 2021; date of current version May 21, 2021. This work was supported in part by the Swiss Federal Office for Defence Procurement (armasuisse) under Grant Aramis R-3210/047-31 and Grant CYD-C-2020003. *(Corresponding author: Pedro Miguel Sánchez Sánchez.)*

Pedro Miguel Sánchez Sánchez, José María Jorquera Valero, Manuel Gil Pérez, and Gregorio Martínez Pérez are with the Department of Information and Communications Engineering, University of Murcia, 30100 Murcia, Spain (e-mail: pedromiguel.sanchez@um.es; josemaria.jorquera@um.es; mgilperez@um.es; gregorio@um.es).

Alberto Huertas Celdrán is with the Communication Systems Group, Department of Informatics (IFI), University of Zurich, 8050 Zürich, Switzerland (e-mail: huertas@ifi.uzh.ch).

Gérôme Bovet is with the Cyber-Defence Campus, Armasuisse Science and Technology, 3602 Thun, Switzerland (e-mail: gerome.bovet@armasuisse.ch).

Digital Object Identifier 10.1109/COMST.2021.3064259

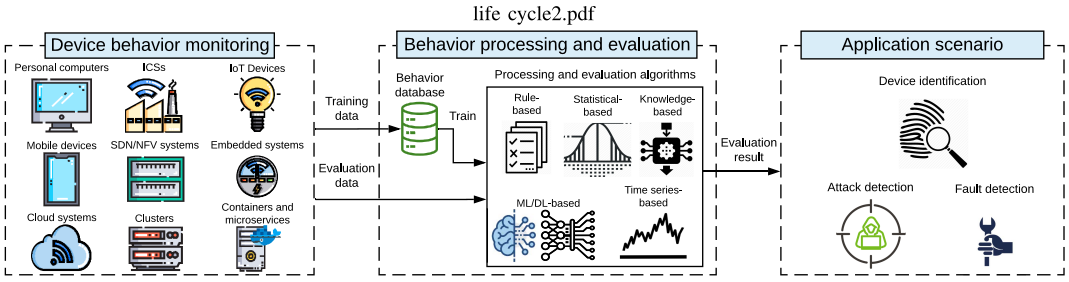


Fig. 1. Common life cycle implemented by device behavior fingerprinting solutions.

labels, or tags to identify devices [17]. The main limitation of these approaches is that they can be modified or even duplicated in an environment where the number of devices grows exponentially. Another relevant drawback appears when device identification is performed at different granularity levels, requiring multiple labels and increasing management complexity. Nowadays, the literature categorizes the following identification granularity levels: *type*, with the main goal of creating fingerprints able to detect different types of devices [6]; *model*, focused on identifying different models of devices based on common hardware and software [18]; and *individual*, probably the most challenging level because it tries to identify identical physical devices according to minor differences occurred during manufacturing processes [14].

On the other hand and with the goal of detecting misbehavior or malfunction caused by cybersecurity issues, novel and sophisticated cyberattacks are influencing the replacement of traditional cybersecurity techniques. Existing mechanisms based on signatures are no longer effective against unseen, encrypted, or large-scale cyberattacks, and device fingerprinting has been identified as one of the most promising solutions to tackle this challenge [19]. A relevant number of works found in the literature rely on creating "normal" behavioral fingerprints to spot changes caused by some previous issues [7], [15], [20]. In this case, fingerprint evaluation is usually tackled from an anomaly detection perspective [7], [21].

In this context, the article at hand performs a comprehensive analysis of the main characteristics –devices, behavioral sources, data, and techniques– considered by the most representative and recent works of device identification and malfunctioning detection scenarios. Besides, it studies how characteristics of device identification, and misbehavior and malfunction detection scenarios are evolving since last years.

Once having the fingerprints, there is another exciting research area focused on applying the most suitable techniques to process and evaluate them. Statistical approaches have been dominating the field for the last decades. However, the incursion of Artificial Intelligence (AI), and more concretely Machine and Deep Learning (ML and DL) as the dominating trend, shifted the field and generated an open discussion concerning the most suitable methods per scenario. This manuscript seeks to help readers understand the trend concerning behavior processing and evaluation techniques, as well as the most appropriate techniques for each application scenario.

Influenced by the rise of AI techniques, there is also a crescent necessity of exhaustive datasets with which algorithms can train models able to learn and infer valuable information aligned with the target scenarios. Datasets are also critical to have standard benchmarks enabling fair comparisons of existing techniques and solutions. In this direction, this article also pretends to support researchers working on the device behavior research field with a review of the most relevant characteristics of existing datasets.

II. Motivation and Contributions

Device behavior fingerprinting is an encouraging research field that has inspired the publication of several survey articles for the last years. In terms of device identification, in 2016, Xu *et al.* [22] reviewed unique device fingerprinting in wireless networks. Moreover, Baldini and Steri [23] published in 2017 a review on mobile phone identification based on its hardware components. Regarding the usage of device fingerprint for cybersecurity purposes, the surveys related to this study are mainly focused on Intrusion Detection Systems (IDS). In 2018, Elrawy *et al.* [25] published a study focused on IDS and IoT-based smart environments. Similarly, Khraisat *et al.* [26], in 2019, published another review on general IDS-related solutions and public datasets, mostly containing network data. In [19], Mishra *et al.* published a survey, in 2017, where IDS analysis is addressed with a focus on cloud environments. This work explicitly considers system behavior analysis, one of the main sources to ensure a cloud system. Finally, in 2018, Liu *et al.* [24] analyzed existing solutions and datasets covering attack detection based on system calls, with a special focus on embedded devices.

Despite the contributions of the previous works, as illustrated in Table I, none of them addresses device identification and misbehavior detection in the same study. Besides, no previous survey contemplates device behavior fingerprinting for component malfunctioning detection. In addition, there is no recent work reviewing from a broad and exhaustive perspective datasets designed both for device identification and for intrusion or malfunction detection. Moreover, other surveys in domains such as digital forensics [27], threat hunting,

TABLE I
COMPARISON OF SURVEY WORKS CONSIDERING DEVICE BEHAVIOR FINGERPRINTING

Work	Year	Device Types / Area	Device Identification	Intrusion Detection	Malfunction Detection	Dataset review	Focus and solution categorization
[22]	2015	Wireless devices	✓	✗	✗	✗	• Survey on device fingerprinting in wireless networks. • Authors differentiate between white list-based and unsupervised algorithms.
[23]	2017	Mobile phones	✓	✗	✗	✗	• Survey on mobile device identification based on physical components. • Fingerprinting techniques are classified in two different categories, emitted signal-based and electronic component-based.
[19]	2017	Cloud environments	✗	✓	✗	✗	• Survey on IDSs applications focused on cloud computing environments. • Intrusion detection techniques are divided into misuse detection (rule-based) and anomaly detection (behavior-based).
[24]	2018	Any, focus on embedded devices	✗	✓	✗	✓	• Survey on IDSs deployed in hosts and based on system calls. • IDSs solutions are divided into anomaly and detection-based and misuse detection-based.
[25]	2018	IoT Environments	✗	✓	✗	✗	• Survey on IDSs focused on IoT-based smart environments. • IDS types are divided into anomaly, specification and misuse-based.
[26]	2019	Any	✗	✓	✗	✓	• IDS survey, groups the solutions in signature-based and anomaly-based. • Data sources divided into network and system logs and audits.
This work	2020	Any, focus on IoT	✓	✓	✓	✓	• General survey on device behavior fingerprinting, its application scenarios, processing techniques and public datasets.

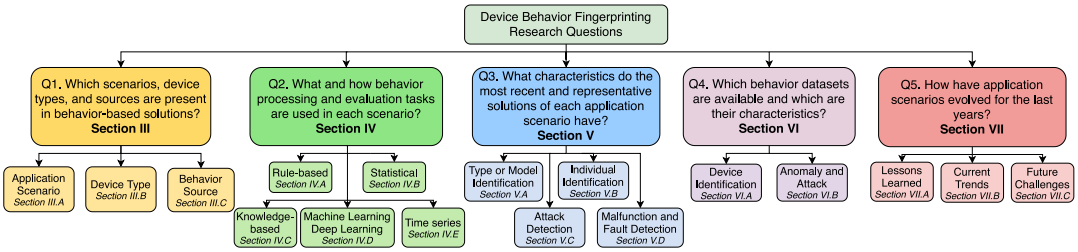


Fig. 2. Discussed questions per article section.

and threat intelligence [28], relying on device identification or attack and fault detection as a basis, also considered behavior fingerprinting as an issue or challenge to cover, motivating the importance of this work. In this context, the literature has some research questions that need to be solved. As the main relevant, we highlight:

- *Q1. Which scenarios, device types, and sources are present in behavior-based solutions?* Depending on the application scenario –device identification or malfunction detection– and the problem to be solved, the devices and behavioral sources vary. However, in the literature, there is no solution detailing these elements and how they are combined.
- *Q2. What and how behavior processing and evaluation tasks are used in each scenario?* Device behavior can be processed and evaluated following diverse approaches. However, the literature has not studied these approaches from a broad perspective to have a complete view in the area.
- *Q3. What characteristics do the most recent and representative solutions of each application scenario have?* It is required to analyze how device types and behavioral sources are utilized to solve the problems motivated by each application scenario. Furthermore, it is also needed to detect the limitations of solutions related to both scenarios.
- *Q4. Which behavior datasets are available and which are their characteristics?* There is no study detailing the public datasets aligned with device behavior from a broad perspective, analyzing their characteristics, and defining in which application scenarios they can be utilized.
- *Q5. How have application scenarios evolved for the last years?* To establish the guidelines for future research, it is critical to describe how device behavior analysis is evolving in the last years and which are the current trends and open challenges of the area.

These research questions are closely related to each other and draw a complete picture of the existing challenges in device behavior analysis for identification and attack and malfunctioning detection. *Q1* and *Q2* deal with devices, data sources, and techniques used for device fingerprinting. *Q3* and *Q4* refer to current publications and datasets of device behavior –the key aspects of this survey and core sections of the document. While *Q5* focuses on the consequences of the research done so far and its future. Fig. 2 shows where and how the previous questions are addressed in the article at hand, acting as table of contents.

To answer the previous questions and provide readers with an up-to-date vision of device behavior fingerprinting, the main contributions of this manuscript are:

- An analysis of the behavior data sources and device types utilized in the literature, paying attention to the application scenarios in which each source is contemplated (answering *Q1* in Section III).
- A description and comparison of the main techniques and algorithms utilized to model and evaluate device behavior based on the morphology of the available data (answering *Q2* in Section IV).

- A comprehensive review and comparison of the characteristics, advantages, and limitations of the most relevant proposals that consider device behavior to 1) identify device models or types, 2) identify individual devices, 3) detect cyberattacks, and 4) detect device/system functioning faults (answering *Q3* in Section V).
- A description of the principal public datasets containing device activity and behavior. This description is divided into datasets designed for device identification and for attack or behavior anomaly detection (answering *Q4* in Section VI).
- A set of lessons learned, current trends, and future challenges drawn from the device behavior works and datasets reviewed (answering *Q5* in Section VII).

The remainder of this article is organized as follows. Section III gives an analysis of device types, application scenarios, and behavior sources. Section IV reviews the main approaches and algorithms utilized to process behavioral data. Section V describes and compares the main solutions found in the state-of-the-art. Section VI examines the main public datasets containing device activities. Section VII draws a set of lessons learned, current trends, and future challenges in the research area. Finally, Section VIII provides an insight into the conclusions extracted from the present work.

III. Behavior Characterization Analysis

With the goal of answering *Q1* (*Which scenarios, device types, and sources are present in behavior-based solutions?*), this section studies the most used and promising scenarios where device behavior has been considered: device identification and misbehavior detection. After that, and aligned with these scenarios, it analyzes the main device types from which behavioral data is obtained, and the most common behavior dimensions and characteristics considered by device fingerprint solutions existing in the literature.

A. Application Scenario

According to the heterogeneous capabilities of device behavior fingerprinting, the literature has applied it in a wide variety of scenarios with different objectives. After reviewing the state-of-the-art, we highlight the following two categories as the most used and well-known: *Device identification* and *Misbehavior detection*.

1) Device Identification: It uses the behavior of devices to identify them and their characteristics. This task can be performed from the following two perspectives.

Device type or model identification. Device type identification [6], [12] aims to recognize the device category such as general computer, IoT sensor, or embedded device, among others. In contrast, device model identification [18], [29] aims to differentiate between devices of the same type but different hardware and software configurations.

Individual device identification [14], [30] distinguishes between devices with identical hardware and software capabilities. This approach requires the lower level data, usually related to hardware variations during fabrication. Although device activity can also be employed to model user behavior and perform user's identification and authentication [31], [32], [33], user inputs and activity monitoring fall out of the scope of this study, which is focused only on device behavior analysis, without human interaction.

2) Misbehavior Detection: It seeks to identify anomalous situations based on changes in normal device behaviors. The anomalous situations are very varied; therefore, the solutions trying to recognize these situations are also heterogeneous. The next two main families of behavior anomaly detection solutions can be found in the literature.

Attack detection [7], [20], [34], [35] intends to detect anomalies, created by cyber threats, according to the previously known normal device behavior. These solutions are commonly deployed as an IDS based on device behavior, being either Network-based (NIDS) or Host-based (HIDS). The cyberattacks detected using behavior are very diverse and depend on the monitored dimensions. These can range from impersonation and spoofing to malware execution.

Malfunction and fault detection [8], [16], [36] tries to identify devices that are not functioning correctly because some component or service is failing. The malfunctioning could be caused by faults such as damaged hardware, a service or hardware overload, or network issues. Solutions addressing this approach assume that the fault will somehow affect the general device behavior.

B. Device Type

Device activities, properties, and interactions can be monitored in an exhaustive range of heterogeneous devices and systems. Then, behavioral patterns can be built with diverse goals by almost any device. However, the data collection process is different depending on factors such as device hardware and software. At this point, it is important to describe the principal device and system categories used in the previous application scenarios.

Personal computers: This category includes computers commonly found in homes and workplaces [37]. We can differentiate two main kinds of personal computers, desktop devices and laptops, differentiated by power supply.

Mobile devices: Smartphones and tablets are grouped in this category. Mobile devices are mainly constrained by battery.

Embedded systems: These low-cost systems are designed and built to perform very specific tasks and their functionality is usually limited by processing and energy constraints [38].

Industrial Control Systems (ICS): This family groups devices and systems that supervise and control critical services of industrial processes [39], involving sensors and actuators. ICSs are usually deployed as supervisory control and data acquisition (SCADA) systems [40].

IoT devices: Any system with processing power and connected to the Internet can be considered as an IoT device. Typically, the IoT device concept is associated with embedded systems with connectivity capabilities such as sensors and smart-home objects. However, it covers a wider variety of devices [38], including drones, or wearable devices, among others.

Cloud systems: They provide the following three principal service models, in which resources can be accessed remotely and through network [41]: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). In the last years, Cloud paradigm has evolved towards Fog [42] and Edge Computing [43], where cloud systems are deployed closer to end-user devices, reducing latency and speeding up computations.

SDN/NFV systems: SDN and NFV are concepts that usually appear together, although they can also be utilized separately [44]. The Software Defined Networking (SDN) paradigm [45] is a network architecture where network control is decoupled from the data plane, having a centralized controller managing the traffic flows and enabling network programmability and abstraction. Network Function Virtualization (NFV) paradigm [46] is a network architecture where network devices are vitalized using software implementations.

Containers and microservices: Containers are software packages that include an application code and all its dependencies, allowing a lightweight deployment. Microservices [47] are applications with a single fixed function, commonly deployed as containers. Several microservices can be combined to build more complex applications distributedly.

Clusters: A cluster is a set of computers, typically Linux devices [48], connected closely to combine their resources and work as a single system. Then, the cluster behavior will be defined by the behavior of its components.

C. Behavior Source

Once the most representative application scenarios and devices have been explained, it is necessary to describe the behavior sources found in the literature, their pros and cons, and the solutions using each source. This description has been structured by following the next two main categories considered in the literature: *externally-collected behavior sources* and *in-device behavior sources*. Finally, the key aspects of the behavior data considered by each solution are compared.

1) Externally-Collected Behavior Sources: In this category, an external device is used to monitor the device behavior. Concretely, network communications and emitted electromagnetic signals are the main externally-collected sources used to model devices behavior. In the case of network-based data, data is usually collected by a proxy or a gateway, while electromagnetic signal-based data is collected by a sensor through an antenna.

Network communications: From the network communications perspective, a diverse set of behavioral features can be extracted by monitoring network packets. They depend on the granularity of the traffic inspection and the TCP/IP layers gathered. The main advantage of this dimension is its universality, as almost any device has network interfaces, and the possibility of monitoring many devices from a single gateway. As drawbacks, this dimension can suffer impersonation attacks and encryption makes data analysis more difficult. In this context, some solutions only focus on the amount of data sent/received and the IPs to which the device is connected [9], [49]. Other solutions also perform packet header and flow statistics analysis [12], [50]. And finally, other solutions also include data related to transport or application layer protocols or payload data [51], [52]. Generally, payload data is protected using encryption methods, so the majority of solutions utilize header and flow-based data. However, some works focus on encrypted communication analysis for fingerprinting [53], [54]. From the application usage point of view, this category is utilized for device model identification [4], [50], device type identification [6], [13], [55], attack detection [7], [15], [56] and fault detection [57].

Clock Skew: Based on crystal oscillator imperfections that occurred during the manufacturing process, internal clock counters of different devices have slight variations. In this sense, it is possible to utilize this characteristic to differentiate devices based on their hardware behavior. The main advantage of this source is that it can be collected from outside the device. As drawback, clock skew distribution concentrates around 0, so this source cannot be applied as a unique source in large device deployments [58]. Clock skew can be calculated by observing how internal device timestamps vary in time, mainly using TCP and ICMP timestamps [59] and Wi-Fi beacon timestamps [60], [61], so it can be seen as a special category of network-based data. From the application perspective, clock skew has been utilized for individual device identification [60], [61], [62], [63].

Electromagnetic signals: This category relies on the behavior of electromagnetic signals emitted by each device. Its main advantage is the difficulty of tampering it, as it depends on emitted signal properties. In terms of disadvantages, we highlight that the data gathering process must be physically close to the monitored device, since electromagnetic signals lose intensity as the distance to the transmitter increases. Radio signals are used in the literature to distinguish drone models [64], [65], [66], [67] and to identify physical devices [14], [68]. However, although radio signals have been utilized to detect anomalies in the radio spectrum [69], no solution specifically focused on device behavior anomaly detection using radio signals has been found. Following a similar approach, other solutions utilize the electromagnetic signals radiated from the device components to identify physical devices [70].

Table II compares the main characteristics of externally-collected data. As observed, features related to network communications are used both for device identification and misbehavior detection, as this source is very heterogeneous. In contrast, clock skew and electromagnetic-based features are only applied in device identification, as they are lower-level sources related to device component characteristics.

2) In-Device Behavior Sources: In this category, behavioral data monitoring is performed on the target devices. Thus, lower-level data related to the device internal functioning can be collected. This approach has the advantage of not requiring a connection to an external monitoring device. In contrast, as a drawback, if the device suffers an anomaly, such as an attack, the monitoring solution may suffer it as well.

Hardware Events: Hardware Performance Counters (HPC) are special registers built into modern microprocessors that store hardware-related event counters. The main advantage of

TABLE II
EXTERNALLY-COLLECTED BEHAVIOR CHARACTERISTICS. (DI: DEVICE IDENTIFICATION. MD: MISBEHAVIOR DETECTION)

Feature	Behavior Source	Device Type	Application Scenario	
			DI	MD
Packet headers statistics	Network Communications	Computers, IoT devices, ICS	[13] [12] [18] [29] [71] [53]	[5] [15] [56] [72] [9] [73]
Network flows statistics	Network Communications	Computers, IoT devices, ICS	[74] [6] [4] [75] [76]	[7] [10] [34] [77] [78] [57] [79] [80] [81] [82] [83] [84]
Packet payload data and statistics	Network Communications	Computers, IoT devices, SDN, ICS	[52] [50] [51] [85]	[86] [87] [88] [54] [89]
Clock drift in time	Clock Skew	Computers, mobile and IoT devices, ICSs	[60][61] [62][63]	✗
Raw IQ samples	Electromagnetic signals	Computers, mobile and IoT devices, ICSs	[64][65] [14] [68] [66] [67]	✗
Signal frequency	Electromagnetic signals	Computers, mobile and IoT devices, ICSs	[70]	✗

this category is the precision achieved to model the device operation from a low-level perspective. In contrast, the quantity and morphology of the HPCs depend on the device CPU model, which makes it difficult to build general solutions. In the literature, some solutions [90], [91], [92] utilize HPCs to model software behavior and detect abnormal operations. In addition, [91] also utilizes HPCs to identify and authenticate different devices.

System processors and oscillators: Some devices have hardware components that include a crystal oscillator. As in clock skew, the manufacturing imperfections of these components can be utilized to differentiate physical devices by comparing their counters drift in time. The main advantage of this source is its low-level, which enables to differentiate devices with the same software and hardware. However, the device should include hardware using oscillators, something unusual in resource-constrained devices. Moreover, manufacturing errors are usually small [58]. In the literature, two components used for this purpose are the Real Time Clock (RTC) and the Digital Signal Processor (DSP) [93]. In addition, the time it takes to execute a particular code or function can also be used to model system behavior. In this case, this data has been used to identify device models and the devices themselves [94].

Resource Usage: In this category, different device components usage and status are monitored. Commonly, the monitored components are CPU, memory, disk, and network. Various parameters can be extracted from each component, such as usage percentage or input/output statistics. In terms of advantages, this source is quite general and can be monitored in many devices and systems. As drawback, continuous resource usage monitoring consumes many resources. In the literature, this data is utilized to identify devices [30] and detect behavior anomalies caused by cyberattacks [21] or system malfunctioning [36], [49], [95].

Software and Processes: The software deployed in a device or system also has its particular behavior. Then, the conjunction of the isolated software behaviors can be utilized to model a global device behavior fingerprint. As advantage, software monitoring can accurately model normal device behavior. However, this source is affected by system updates and legitimate software modifications. Software can be modeled in several ways:

- *System calls and logs:* They serve to monitor the interactions between the programs running on a device and its operating system. These interactions encompass process, file, and communication management operations. From the application usage point of view, system call sequences and logs have been used to characterize device behavior and detect anomalies [35], [96], [97], [98], [99], [100], [101].
- *Process properties:* Device software behavior can be modeled by monitoring each process properties, such as name, status, or threads. This category also includes the resources utilized to execute a particular program or code. In the literature, this category is commonly monitored together with resource usage or system calls to detect anomalous behaviors [102].
- *Software signatures:* Software snapshots (signatures) are generated for the different device executable and their configuration files using hashing algorithms. Then, the snapshots are used to detect software modifications that cause behavior anomalies [16], [103].

Device Sensors and Actuators: The data collected in this dimension is very diverse and depends on the device and scenario typology. The main advantage of this source is that it can also detect environment failures or attacks. As drawback, environment knowledge is required to analyze and understand the data from this dimension, as each device may have different sensors and actuators. From the application usage point of view, sensor and actuator measurements are utilized to detect anomalies [8], [20], [89], [104], [105] and model device types [4], while sensor hardware information is used to physically identify the devices [106].

To conclude, on the one hand, Table III compares the main characteristics of data directly collected from the modeled device. It can be appreciated how HPCs, CPU percentage, system calls, software signatures, and sensor values are used both for device identification and misbehavior detection. Besides, low-level information related to the system processors and sensor hardware is only employed for device identification. Finally, features related to resource usage and process properties are only employed in misbehavior detection. On the other hand, Fig. 3 shows the behavior sources considered by each device type, and in which application scenario these sources are utilized. The numbers indicate the total number of connections each element has. It can be appreciated that the most extended sources, based on their generality, are network communications, hardware events, resource usage, and software and processes.

IV. Behavior Processing and Evaluation Techniques

Once reviewed the behavioral data monitored per type of device and application scenario, the data needs to be

TABLE III
IN-DEVICE BEHAVIOR CHARACTERISTICS. (DI: DEVICE IDENTIFICATION. MD: MISBEHAVIOR DETECTION.)

Feature	Behavior Source	Device Type	Application Scenario	
			DI	MD
HPC	Hardware Events	Embedded systems, IoT devices	[91]	[90] [91] [92]
RTC drift	System processors and oscillators	Computers	[93]	✗
DSP performance	System processors and oscillators	Computers	[93]	✗
Code execution time	System processors and oscillators	Computers	[94]	✗
CPU usage percentage	Resource Usage	Computers, embedded devices, microservices, cloud, NFV, and cluster systems	[30]	[21] [36] [107] [11] [95] [49] [108] [109] [16] [110]
CPU activity	Resource Usage	Microservices, NFV, cloud, and cluster systems	✗	[36] [107] [95] [109] [3]
System storage usage	Resource Usage	Microservices, NFV, cloud, and cluster systems	✗	[36] [107] [95] [108]
System memory usage	Resource Usage	Microservices, NFV, cloud, and cluster systems	✗	[36] [107] [95] [49] [108] [109] [16] [110] [3]
I/O throughput per network interface	Resource Usage	Microservices, NFV, cloud, and cluster systems	✗	[21] [36] [95] [49] [108] [109] [110]
System calls and logs	Software and Processes	Computers, resource-constrained devices, cloud and NFV systems	[35]	[35][96] [97] [101] [98][100] [99]
Process properties	Software and Processes	Computers	✗	[102] [111] [3]
Software signatures	Software and Processes	IoT devices	[103]	[16] [103]
Sensor measurements values	Device Sensors and Actuators	ICS, IoT devices	[4]	[20][8][89] [104][105]
Sensor hardware properties	Device Sensors and Actuators	ICS	[106]	✗

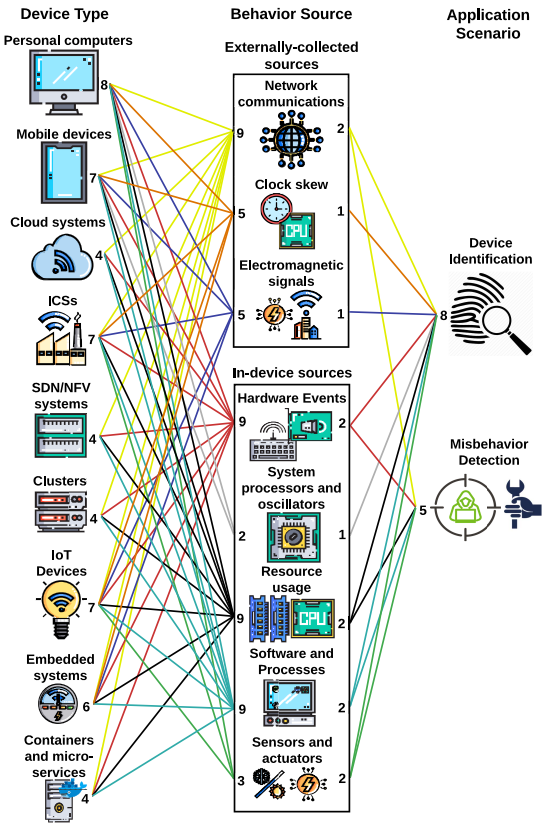


Fig. 3. Behavior sources available in each device type and application scenarios. (The numbers shown for each item indicate its total number of connections.)

processed to create a fingerprint. This section deals with *Q2 (What and how behavior processing and evaluation tasks are used in each scenario?)* by detailing the algorithms and techniques commonly used in the literature to create and evaluate fingerprinting profiles, highlighting their main advantages and drawbacks. The existing techniques are categorized in the following five groups: *rule-based, statistical, knowledge-based, Machine Learning and Deep Learning, and time series approaches*. The previous categories are not mutually exclusive and a particular solution can belong to several categories. Furthermore, the behavior processing can be centralized, in the own device or a server, or distributed using technologies such as blockchain [112], distributed [113] or federated learning [114], among others.

A. Rule-Based

This is the most straightforward approach to create behavioral profiles. It is useful for devices with a well-known behavior and a reduced set of actions. In this approach, a set of rules defines how the system should behave, that is, its behavioral fingerprint. Rules can be defined statically, based on pre-defined actions, or dynamically, based on the historical actions performed by the device. Any deviation from these rules is considered a fault or anomaly. The main advantages of this approach are its speed and simplicity. As drawbacks, it requires previous knowledge about the device behavior, and it is not suitable for changing and complex scenarios. Rule-based evaluation is utilized for device type or model definition and anomaly detection.

For device behavior evaluation, a recent approach is the usage of Manufacturer Usage Descriptions (MUDs) standard [115] files, which define the normal device functioning and are commonly issued by vendors. This method is mainly utilized for IoT behavior fingerprint generation and evaluation [10], [80]. Another rule-based approach is to explicitly define the software that the device can execute [103] or thresholds for resource usage [116].

B. Statistical

In this approach, relatively basic statistical data processing techniques are utilized to extract inferences (properties) from

data samples. This approach is usually considered in data pre-processing and anomaly detection. The main advantage of this approach is its simplicity and that these algorithms do not require large datasets. However, it does not handle well multi-dimensional data, and consistent evaluation decisions require previous knowledge in the area.

For pre-processing, it is common to infer features using statistical functions such as average, standard deviation, quartiles, maximum, or minimum, among others. Regarding evaluation, in some solutions [21], the interquartile range (IQR) is used as a statistical measure representing the presence of outliers and anomalies based on data variability (dispersion). In the same line, Euclidean Distance is used by some approaches [9], [57], [89] to determine anomaly values based on the distance between two data measurements. Finally, some works [8], [60] utilize *Expectation Maximization* algorithm for clustering and parameter estimation based on statistically-inferred latent variables.

C. Knowledge-Based

This approach aims to represent knowledge extracted from received data and build a reasoning system capable of inferring new knowledge. Commonly, the knowledge is built based on a set of ontologies, and the decision-making process is based on if-then derivation rules. The main advantages of this approach are the explainability of the inferred solutions and that it can solve problems involving incomplete data. As drawbacks, this approach takes longer time, and it has reduced scalability, as the system could become too complex if large amounts of data are utilized.

Knowledge-based approaches are utilized mainly for behavioral anomaly detection, being the main ones look-ahead algorithms and finite state machines. *Look-ahead algorithms* are commonly combined or used to make decisions in more complicated approaches, such as state machines. Furthermore, these algorithms are also directly used to detect anomalies [35]. *Finite state machines*, such as *Markov Models* [117] and *n-gram models* [118], describe the sequential logic followed by a certain entity and predict its future status based on the previous ones. In the literature, they are widely applied for behavior anomaly detection [10], [16], [35], [92].

D. Machine Learning and Deep Learning

In recent years, and based on the increase of processing power and available data, Machine Learning (ML) [119] and Deep Learning (DL) [120] algorithms have gained enormous relevance in almost every industrial or research area, becoming the dominating trend for data processing and evaluation. The main advantages of ML/DL based approaches are their capacity to detect complex data patterns, handle multi-dimensional and multi-variate data, and adapt themselves to dynamic and heterogeneous scenarios using massive data. As disadvantages, the model decisions are usually hardly explainable, based on the black-box nature of the generated models. Besides, these algorithms, especially in DL, require large amounts of data to be trained, and the algorithm training can take much time and resources. Also, most algorithms require parameter tuning, which implies repeating the training process several times. Since ML and DL techniques are very diverse, they have been widely used for device behavior fingerprint generation and evaluation, both for device identification [4], [13], [14], [50], [52], [68], [70], [74], [76] and misbehavior detection [5], [15], [56], [72], [77], [78], [88], [100].

According to the morphology of the data they receive and the type of predictions they make, ML/DL algorithms applied in behavior analysis are distinguished into two main categories: Supervised Learning and Unsupervised Learning.

The goal of *Supervised learning* is to infer a model capable of predicting the output of data vectors based on training labeled data [119]. Supervised algorithms are mainly divided into classification and regression techniques.

- *Classification* algorithms try, based on the training data, to predict the class to which unseen data vectors belong. Additionally, anomaly detection can be performed using classification algorithms by labeling the data as normal/anomaly. Common ML classification algorithms are *Decision Tree* (*DT*) [121], *Random Forest* (*RF*) [122], *Logistic Regression* (*LR*) [123], *Naive Bayes* (*NB*) [124] or *Support Vector Machine* (*SVM*) [125]. These algorithms are widely utilized for behavior evaluation in device identification [4], [6], [12], [13], [29], [50], [51], [52], [70], [71], [75] and behavioral anomaly recognition [72], [73], [77], [78], [79], [83], [88], [97], [110].
- Regarding *Regression* algorithms, the output is a continuous number and not a class. Usual ML regression algorithms are *Linear and Polynomial Regression* [126], which are applied in behavior analysis to evaluate device behavior and its fluctuation [72].

In *Unsupervised learning* [119], data vectors are not labeled, so feature vectors only contain input data. This kind of algorithm is used to extract patterns by modeling probability densities on the given data. The three main applications of Unsupervised learning are dimensionality reduction, clustering, and anomaly detection.

- *Dimensionality Reduction* algorithms aim to reduce the number of variables or features under consideration by obtaining a set of principal variables from the input data. In behavior-based solutions, *Principal Component Analysis* (*PCA*) [127] and *t-Distributed Stochastic Neighbor Embedding* (*t-SNE*) [128] are utilized to speed up computations and derive new features [5], [10], [75]. Moreover, dimensionality reduction is combined with statistical algorithms for anomaly evaluation [36], [57], [107], [109].
- *Clustering* algorithms have the objective of grouping the input vectors into a different set of objects based on their similarities. In device behavior fingerprinting, *k-means* [129] and *Density-based spatial clustering of applications with noise* (*DBSCAN*) [130] are usually applied to infer device classes or types [6], [55], [3], [51], [108].
- *Anomaly Detection* algorithms seek to identify rare items, events, or observations based on a set of unlabeled data points and the assumption that most of the training data

TABLE IV
BEHAVIORAL PROCESSING APPROACHES COMPARISON

Approach	Simplicity	Expert knowledge required	Fast computation / Low resource	Large datasets required	Large training time	Multi-dimensional data	Decision explainability	Adaptability	Complex feature correlations
Rule-based	✓	✓	✓	✗	✗	✗	✓	Dynamic approaches	✗
Statistical	✓	✓	✓	✗	✗	✗	✗	✗	✗
Knowledge-based	Partial	✗	✗	✗	✗	✗	✓	✗	Partial
ML/DL-based	✗	✗	✗	Mainly DL	Mainly DL	✓	Partial	✓	✓
Time series	✗	✗	✗	✓	✓	ML/DL-based	✗	ML/DL-based	ML/DL-based

is normal. From this approach, *One-Class SVM* (*OC-SVM*) [131] and *Isolation Forest* (*IF*) [132] are widely used in the literature [7], [34], [133].

From a DL perspective, *Artificial Neural Networks* (*ANN*) [120] are frequently used in the above approaches. However, a type of architecture cannot be related to a specific use due to neural networks flexibility, as layers, neurons, and their connections can be organized in many ways depending on the problem to be solved. The main types of networks applied in behavior processing are: *Multi-Layer Perceptrons* (*MLP*), utilized for device identification [29], [76] and anomaly type classification [79]; *Autoencoders*, applied for behavior anomaly detection [18] and dimensionality reduction purposes; *Recurrent Neural Networks* (*RNN*), such as *Long Short-Term Memory networks* (*LSTM*) and *Gated Recurrent Unit networks* (*GRU*), applied from a time series perspective for device identification [14], [18] and behavior anomaly recognition [15], [20], [81], [100], [105]; and *Convolutional Neural Networks* (*CNN*), utilized for physical device identification based on signal processing from a time series approach [14], [68].

The previous network topologies can be combined to perform more complex tasks. For example, some solutions [18] utilize LSTM layers to build an autoencoder, while other approaches [82] combine different neural networks to build *Generative Adversarial Networks* (*GAN*) [134].

E. Time Series

Time series analysis utilizes data measurements as a sequence of values where each measurement is related to the previous and the next ones. It includes a wide variety of algorithms and models, including the ones based on ML/DL or statistical algorithms. This approach is utilized both for device identification and anomaly detection, directly in the model generation or as data pre-processing. The main advantages of this approach are its improved performance over single-value processing approaches. However, it requires a large amount of data to detect the temporal patterns, and the processing is time-consuming.

Time series analysis methods are divided into two different types, *frequency-based* methods, which analyze data as a signal with a certain frequency, and *time-based* methods, which analyze data evolution with respect to time.

In terms of frequency-based methods, *Fourier Transform* (*FT*) [135], and derived functions, are applied as pre-processing to obtain the frequencies that form the value signal [6], [91]. From time-based methods, *AutoRegressive*

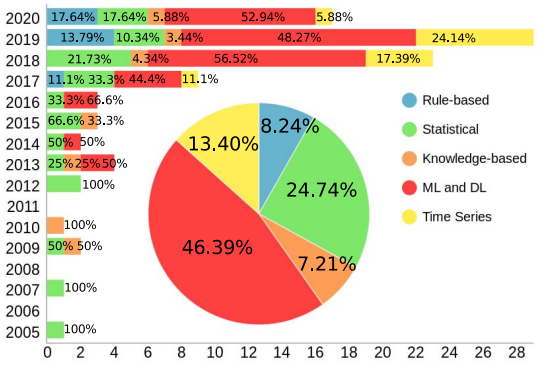


Fig. 4. Yearly and global distribution of processing techniques used by device behavior fingerprinting solutions.

Moving Average (*ARMA*) and derived algorithms are used in behavior prediction applications [9], [49]. In addition, *Dynamic Time Warping* algorithm is also utilized in device behavior evaluation [30], directly comparing the values of two time series.

Besides, as stated before, Deep Learning has been applied in behavioral data evaluation from a time series perspective utilizing RNNs [15], [18], [20], [81], [100], [105] and CNNs [14], [68].

Table IV compares the main properties of the five behavior processing approaches identified in the literature analysis. As general conclusion, when the behavior of the device is composed of a limited and known number of actions and there is not a large number of dimensions in the data, the appropriate approaches would be those based on rules and statistical algorithms, given their reduced complexity and resource consumption. However, when the data features maintain complex relationships between them, the most suitable solutions are those based on knowledge and ML/DL approaches. Finally, when there is a relationship between the different measurements based on their order, a time series approach may provide improved results. Depending on the amount of data, the available resources, and the complexity of the feature correlations, some particular algorithms are better than others. For example, a simple IoT device, like a bulb, with a limited and known set of actions, can be modeled with a rule-based approach, leveraging its limited resources. In contrast, a cloud service that executes different tasks would be hard to model using rules, instead, an ML/DL-based approach exploiting the correlations in the sources available would be more successful.

TABLE V
COMMON EVALUATION METRICS CONSIDERED BY DEVICE BEHAVIOR FINGERPRINTING SOLUTIONS

Metric name	Description	Equation
Accuracy	Total number of correct predictions over the total made	$\frac{TP+TN}{TP+FP+TN+FN}$
Precision	Ratio of actual positives over all the elements predicted as positives	$\frac{TP}{TP+FP}$
Recall, Sensitivity or True Positive Rate (TPR)	Proportion of actual positives correctly identified	$\frac{TP}{TP+FN}$
Specificity or True Negative Rate (TNR)	Proportion of actual negatives correctly identified	$\frac{TN}{FP+TN}$
False Positive Rate (FPR) or False Acceptance Rate (FAR)	Proportion of the elements wrongly determined as positive among the actual negatives	$\frac{FP}{FP+TN}$
False Negative Rate (FNR) or False Rejection Rate (FRR)	Proportion of the elements wrongly determined as negatives among the actual positives	$\frac{FN}{TP+FN}$
F1-Score	It is the harmonic mean of precision and recall. Also known as F-Score or F-measure	$\frac{2 \times precision \times recall}{precision + recall}$
Equal Error Rate (EER)	Threshold that equals the FAR and FRR.	$FAR = FRR$
Area Under Curve (AUC)	Area covered by the plot of TPR and FPR (ROC Curve) at different threshold values between 0 and 1	$\int ROC$
Mean Squared Error (MSE)	Average of the squares of the prediction errors. It is utilized in regression	$\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2$
Root Mean Squared Error (RMSE)	Root of the average of the squares of the prediction errors. It is utilized in regression	$\sqrt{(\frac{\sum_{i=1}^{n}(y_i - x_i)^2}{n})}$
Mean Absolute Error (MAE)	Absolute average of the prediction errors. It is utilized in regression	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - x_i\rvert$
Root Relative Squared Error (RRSE)	Error relative to a simple predictor that always returns the average of the actual values	$\sqrt{(\frac{\sum_{i=1}^{n}(y_i - x_i)^2}{\sum_{i=1}^{n}(x_i - X)^2})}$, $X = \frac{1}{n}\sum_{i=1}^{n} x_i$
Detection or modeling time	Period elapsed between an attack or anomaly starts and the monitoring system detects it, or the time elapsed to model the device behavior accurately [12], [15]	–
Processing overhead or resource consumption	Resource usage of behavior monitoring and processing, which is particularly relevant in resource-constrained devices [90], [92], [95]	–

TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative.

Overall, Fig. 4 shows the global and per year distribution of works using each technique, note that some works may utilize techniques belonging to more than one category. ML and DL rise as the leading group of processing techniques applied to device behavior fingerprinting, as it is already the main trend in the area and is still gaining even more prominence.

Additionally, to properly evaluate and compare the solutions performance, it is critical to define relevant metrics. Then, independently of the evaluation approach followed, there is a set of common metrics utilized in the majority of behavior-based solutions. Table V shows these common metrics. In the case of classification approaches, these metrics are based on the values present on a confusion matrix, while in the case of regression approaches, the metrics are based on prediction errors [11], [136]. Moreover, some solutions also consider factors such as detection time or resource usage.

V. BEHAVIOR-BASED SOLUTIONS AND APPLICATIONS

After analyzing the processing and evaluation techniques used in device fingerprinting (*Q2*), and the scenarios, devices, and data sources (previously, with *Q1*), we have the background needed to review and understand device behavior-based solutions. In this sense, this section performs an in-depth review of the most relevant works of the literature that deal with behavioral fingerprinting to answer *Q3* (*What characteristics do the most recent and representative solutions of each application scenario have?*). The analysis of each solution considers the application scenario, device type, behavior source, data monitored, processing and evaluation algorithms, and results criteria. We give particular importance to IoT devices because of their role in current real-world deployments. Still, it is important to note that other devices could be fingerprinted considering the same data sources. Below, the approach followed by each solution is detailed and grouped by application scenario and behavior source.

A. Device Type or Model Identification

In this application scenario, we review solutions whose objective is to identify device models or types. Devices belonging to the same model or type are treated as equals by the literature. The main characteristics, algorithms and performance of each solution are compared in Table VI.

1) Network-Based Identification: Many works in the area of device type or model behavior fingerprinting address the identification problem from a network analysis perspective, deriving statistical features for ML/DL technique application. Furthermore, they are mainly focused on IoT and ICS devices differentiation, as this section shows. In this context, the authors of [71], proposed two fingerprinting methods for ICS device models. The first was based on the response time between a TCP acknowledgment and the application layer response, once the data had been processed. The second method used physical operation times by measuring the time elapsed to apply some actions in an actuator. In [12], Miettinen *et al.* proposed IoT Sentinel, an IoT device type identification approach based on device setup network communications. The main goal of this work was to recognize potentially vulnerable device types and enhance their security based on rules. Packet headers were analyzed to derive features resilient to traffic encryption.

Bezawada *et al.* [52] also presented a network-based methodology to perform behavioral fingerprinting and device type identification inspired in SIP-based fingerprinting [137], [138]. A behavior model data was divided into static, based on the header protocols used by the IoT device, and dynamic, based on flow sequences and packet payloads. By following the same direction, Shahid *et al.* [75] identified different IoT device types using bidirectional flow characteristics. Four different device types were utilized: sensor, camera, bulb, and plug.

Also dealing with device type or model identification, the authors of [13], utilized ping operations to generate a fingerprint of different IoT devices to distinguish real embedded machines from virtual and emulated embedded systems.

TABLE VI
DEVICE TYPE OR MODEL IDENTIFICATION SOLUTIONS BASED ON DEVICE BEHAVIOR FINGERPRINTING

Work	Year	Device Type	Approach	Algorithms	Behavior Source	Features	Dataset	Classes	Results
[71]	2016	ICS	Classification	ANN, NB	Network	Response delay times	Private	Device Model	99% and 92% accuracy for response and operation time recognition, respectively.
[12]	2017	IoT Devices	Classification	RF	Network	Packet header-based	[12]	Device Type	81.5% average accuracy on 27 devices, over 95% for 17 of them.
[52]	2018	IoT Devices	Classification	Gradient boosting, k-NN, DT	Network	Header and payload statistics	Private	Device Type	99% average accuracy and 86-99% TPR
[75]	2018	IoT Devices	Classification	t-SNE, RF	Network	Flow statistics	Private	Device Type	99.9% accuracy differentiating sensor, camera, bulb, and plug devices.
[13]	2018	IoT Devices	Classification	RF	Network	Ping timestamps	Private	Real / Virtual Device	Detection rate of 99.5% using 25 pings and 99.9% using 200 pings.
[29]	2018	IoT Devices	Classification	RF, SVM, MLP	Network	Clock skew and timestamp features	Private	Device Model	97.03% precision, 94.64% recall and 99.76% accuracy identifying 51 models.
[51]	2018	IoT Devices	Classification	k-means, RF	Network	IoT protocol flows statistics	Private	Device Type	97% accuracy, +97% F1-Score (14/16 classes)
[6]	2019	IoT Devices	Classification	Clustering + k-NN	Network	Flow periods (DFT)	To be published	Device Type	F1-Score above 90% for 21/23 labels and 98.2% overall accuracy.
[4]	2019	IoT Devices	Classification	RF	Network	Flow and packet statistics	[4]	Device Model	99.88% accuracy 5.06% RRSE.
[53]	2019	IoT Devices	Classification	AdaBoost	Network	Encrypted flow statistics	[4]	Device Model	95.5% accuracy and F1-Score.
[50]	2019	IoT Devices	Classification	RF, k-NN, Gradient Boosting	Network	Data exchange statistics	[140]	Device Behavior Type	99.69% accuracy, 93.93% F1-Score and 96.82% TPR, and 11.96% UBMR.
[143]	2019	IoT Devices	Classification	Rules + RF	Network	Header and app-layer statistics	Private	Device Type	96% accuracy on a manually labeled subset.
[18]	2019	IoT Devices	Classification	LSTM-autenc., DBSCAN, OC-SVM	Network	Derived using LSTM-autoencoders	Private / [4]	Device Model	Seen devices: 99% accuracy. Unseen devices: 82% F1-Score and 70% accuracy.
[84]	2020	IoT Devices	Classification	DBSCAN, State machine	Network	Packet sequence statistics	Private	Device Activities	97.05-97.48% avg detection and 0.18-0.32% avg FPR in actions of 19 devices.
[76]	2020	IoT Devices	Classification	DNN	Network	Images generated from raw data	[4]	Device Type	99% accuracy identifying 10 network flow types (9 IoT and 1 non-IoT).
[55]	2020	IoT Devices	Classification	C-means and interpolation	Network	Flow and header statistics	[55]	Device Type / Anomalies	99% accuracy for device type identification and 98% TPR, 4% FPR and 98% F1-Score for attack detection.
[85]	2020	IoT Devices	Classification	Statistical based on term frequency	Network	DNS analysis	To be published	Device model	≈95% avg AUC and 0.01% max FPR on 53 models.
[64]	2019	IoT Devices	Classification	k-NN	Radio signals	IQ samples	Private	Drone model	98.13% accuracy identifying 15 UAV controllers.
[65]	2019	IoT Devices	Classification	DNN	Radio signals	IQ samples	[153]	Drone model	99.7% accuracy using 2 classes, 84.5% using 10 classes.
[66]	2020	IoT Devices	Classification	1D CNN	Radio signals	IQ samples	[153]	Drone model	94.6% accuracy for 10 drone classes.
[67]	2020	IoT Devices	Classification	CNN	Radio signals	IQ samples	Private	Drone model	≈99% accuracy for 10 drones and controllers when SNR is 0 dB.

Several devices were grouped in each category to make them diverse enough to model previously unseen devices. For each ping, time-based statistical features were calculated using ping requests separated by 0.2 seconds. Oser *et al.* [29] utilized TCP timestamps to measure the clock skew of different IoT device models and identify them. 562 devices of 51 different models were utilized for classification-based testing. Using only clock skew, the system could not identify most of the devices. Then, the authors decided to utilize 12 additional features derived from the timestamps gathered to calculate the clock skew. Thangavelu *et al.* [51] proposed DEFT, a distributed device fingerprint and identification system. In this approach, SDN network gateways performed device monitoring and classification locally, while a centralized control entity generated and distributed the classifiers. Statistical features were extracted based on packet headers and application layer protocols, and grouped in 15-minutes sessions. To identify new device types, clustering algorithms (k-means) were applied. Similarly, Perdisci *et al.* [85] analyzed DNS application protocol to derive IoT model fingerprints following a document retrieval-based approach.

Another relevant work in the scenario of IoT device model identification was proposed by Marchal *et al.* [6]. The authors presented AuDI (Autonomous IoT Device-Type Identification), a system designed to identify IoT device type by passively analyzing its periodic network communications, grouping them using clustering algorithms. To recognize periodic flows, Discrete Fourier Transform (DFT) was applied to candidate periods, transforming time domain to frequency domain. Then, 33 different features were calculated for each period. Similarly, Arunan Sivanathan *et al.* [4], [136] worked on device type classification. In this case, packet and flow-based statistical features were utilized to perform device classification and behavioral monitoring tasks. Using the same dataset, Msadek *et al.* [53] focused on encrypted traffic analysis to identify IoT device models. In this work, the authors derived statistical features from headers using a sliding window.

In the same direction, OConnor *et al.* [50] proposed HomeSnitch, a framework designed to classify home IoT devices communication by semantic behavior (e.g., firmware update/check, audio/video recording, data uploading). To build application-level models from packet headers, HomeSnitch used *adudump* [139] traffic analysis tool. After that, 13 different features were extracted to describe application data exchanges. The authors used YourThings dataset [140] for

solution testing. Similarly, Trimananda *et al.* [84] proposed Ping-Pong, a tool designed to extract packet-level signatures for events (e.g., light bulb turning ON/OFF) based on device model. This work covered traffic encryption and unknown proprietary protocols by applying a clustering-based approach over statistical packet analysis. Furthermore, Hafeez *et al.* [55] proposed IoT-KEEPER, a system for both identify device types and detect malicious activities using an unsupervised approach based on fuzzy C-means clustering and interpolation.

Applying more sophisticated DL-based solutions, Ortiz *et al.* [18] presented DeviceMien, a probabilistic framework for device identification which considered stacked LSTM-autoencoders to automatically learn features and classes from raw TCP packets. Then, the system modeled, using DBSCAN, each device as a distribution of the generated classes. For testing, the authors used two different datasets, one public, [4], and another private. Kotak and Elovici [76], as a novel approach, performed a pre-processing step that converted the TCP network traffic (pcap format) to grayscale images. Then, an MLP was utilized to classify different device flows based on the device type. The dataset utilized was from [4].

Another research line covering device identification is based on the analysis of deployment scenarios such as Smart Homes or agriculture networks [141], [142]. Kumar *et al.* [143] analyzed home networks in order to perform device type identification and security analysis. In total, 83M devices deployed in 16M households were collected, analyzing their distribution and known vulnerability issues. In a device subset, a 96% accuracy was achieved using expert rules and an ensemble of RF classifiers trained using data from different application layer protocols. Another smart home device analysis was performed by Huang *et al.* [144]. However, device categories were only manually standardized in this study, mentioning device type identification and anomaly detection as future work paths.

Digital forensics has also leveraged device identification when it helps in forensic investigations, as the increasing number of devices generates new challenges and motivates to work on more advanced identification methods [145], [146]. One example of these scenarios is Amazon Alexa ecosystem forensics [147], [148], where the behavior-based identification of the devices present in the scenario is a highly valuable asset. Moreover, the digital forensics field is also leveraging new technologies such as blockchain when dealing with large scenarios such as IoT environments [149].

2) Radio-Based Identification: Drone model identification is the main research area where radio behavior fingerprinting is employed for type or model identification. Although this problem has been traditionally addressed based on physical characteristics, such as images [150], RADAR and LIDAR [151], or sound [152] (out of the scope of this study), there is an emerging research line based on radio analysis and fingerprinting. A relevant work presented by Ezuma *et al.* [64] analyzed controller signals to classify unmanned aerial vehicles (UAV). In the same line, Al-Sa'd *et al.* [65] used DNNs to classify drone models based on their radio communications. Using the same dataset, Allahham *et al.* [66] improved the previous results using a 1D CNN. Similarly, Basak *et al.* [67] also applied CNNs for drone identification but using their own dataset, which will be published in the near future.

Table VI compares the solutions focused on device type and model identification. From the previous solution analysis, we can observe that the device type and model identification application scenario has been mainly covered from a network communication perspective. Moreover, it is noticed that most of the solutions in this area are focused on IoT, as the heterogeneous nature of IoT devices motivates the usefulness of solutions capable of distinguishing devices according to their type and model. Many solutions achieve classification results over 99% in accuracy and F1-Score metrics, which indicates that this area is relatively covered by approaches with good performance. Besides, drone identification is the main application of radio-based fingerprinting for model identification. Here, further research is still required to achieve similar performance to network-based identification.

B. Individual Device Identification

This section analyzes behavior-based solutions focused on identifying the device itself. It means that they differentiate devices with the same hardware/software. At this point, it is important to note that these approaches will also be able to distinguish different device types and models (the previous category), and this fact is also considered and evaluated in some of them. In these solutions, features usually have a lower level, related to hardware components, trying to differentiate fabrication variations on the device components. Table VII compares the main characteristics, algorithms applied and performance of solutions detailed in this subsection.

1) Processor-Based Identification: In this category, Salo's [93] proposed a fingerprinting software method capable of differentiating identical personal computers using quartz crystals characteristics. Concretely, the author utilized the CPU Time-Stamp Counter (TSC), the Real-Time Clock (RTC), and the Sound Card Digital Signal Processor (DSP). The solution aimed to verify how accurate the RTC and DSP were in terms of CPU cycles by measuring the one-second ticks of the RTC and the time needed by the DSP to process one second of audio. Then, statistical analysis was applied to distinguish computer pairs between them. Also exploiting processor differences, but based on execution time, Sanchez-Rola *et al.* [94] proposed CryptoFP, a novel approach to identify machines with the same software and hardware through the generation of a fingerprint using the time taken to execute a specific function. This fingerprint was generated by executing the same function many times, repeating different parameters to model its time variability. In the fingerprint comparison, the tool compared the most frequent (mode) time values for each call parameter over all iterations. The authors conducted several experiments to test long-term fingerprint stability, and CPU workload and temperature impact in the fingerprint generation. For future work, the authors considered solution scalability as fingerprints are compared one by one. Finally, Lorenz *et al.* [154] considered embedded circuits of IoT sensors for unique fingerprinting. To perform the fingerprinting, predefined voltage sequences were supplied

to the sensor, monitoring how its output varies. Fingerprints were evaluated directly comparing output sequences and using RMSE as error measure. Results in individual identification varied according to sensor model, meaning that some models have more fabrication variability that others.

2) Clock-Based Identification: Based on clock skew capabilities, Jana and Kasera [60] worked on uniquely differentiate wireless access points (AP) based on the clock skew calculated from their beacon frame timestamps. This work utilized the uw/sigcomm2004 dataset [155]. The results, using Expectation Maximization statistical algorithm to compare AP frames, indicated that clock skew seems to be an efficient and robust fingerprinting method capable of detecting different WLAN APs. Similar results to the previous ones were presented by Sharma *et al.* in [62]. In this case, the authors utilized TCP and ICMP timestamp headers to calculate the clock skew between two devices, validating the work of Kohno *et al.* [59]. They tested their approach with 210 different devices, some of them identical, finding that they were able to distinguish both different and identical devices. Besides, they also tested clock skew stability based on the measurement methodology and on several environmental factors, such as temperature or operating system. Based on these results, the authors concluded that this approach is suitable for moderate size networks.

Focused on wireless unique device identification, Lanze *et al.* [61] considered clock skew stability and uniqueness. To measure the clock skew, the authors took the timestamps from a wireless AP (sender) sent in wireless beacons and the timestamps from the measuring wireless client (receiver). To carry out their experiments, they gathered clock skews using five different laptops from 388 different APs. Through their experiments, they concluded that all clock skews were in a rather short range ($\pm$30 ppm) due to restrictions of the suppliers' quality specifications. Therefore, although the clock skew restricts the set of possible devices, it cannot serve as a unique fingerprint for a wireless access point and has to be enriched with other features to achieve uniqueness. In the same line, Radhakrishnan *et al.* [74] published GTID, a system for individual wireless device and device type fingerprinting based on clock skew. This approach utilized clock skew and communication patterns to generate device signatures from a DL-based time series approach. The system was tested using a previous dataset of the team [156], [157], collected from 37 different devices, including some repeated models. Similarly to [61] and [74], Polčák and Franková [58], [63] also discussed clock skew performance when uniquely identifying different devices. Here, the authors concluded that clock skew is not completely stable. Besides, based on the clock skew distribution of the evaluated devices, the authors claimed that clock skews are distributed close to 0 ppm. These factors prevent a quick fingerprint technique to be capable of uniquely differentiate devices in large scenarios. Finally, the authors also discussed and demonstrated the possibility of masquerading or falsifying the clock skew. The authors concluded that this technique might be suitable for small networks or in combination with additional data.

3) Resource Usage-Based Identification: Resource usage was exploited for individual identification in [30]. In this work, the authors developed a fingerprinting method based on the CPU usage graph when the device is executing a fixed task. For this purpose, a benchmark program that included several read/write operations and calculations was developed. In the evaluation process, the graph was compared to the previous ones of the same device using the Dynamic Time Warping algorithm. The percentage of stable fingerprints was calculated using the Shannon entropy and stability measurement, achieving a 93.43% of unique fingerprints.

4) Electromagnetic Signal-Based Identification: Other works solved the identical device identification problem using electromagnetic signals as data source. Using radio signals, Jafari *et al.* [14] used DL techniques to identify wireless devices and distinguish among identical wireless devices from the same manufacturer. The authors used ZigBee devices from which a historical radio frequency trace dataset was obtained. In total, six identical devices were employed in the tests, concluding that it was possible to identify devices based on their radio frequency traces, even if they were from the same model. A similar approach was addressed in [68], where Riyaz *et al.* utilized raw radio samples to build a unique device signature using Software Defined Radio (SDR) transmissions. This solution was tested on 5 identical devices. In addition, the authors analyzed how detection accuracy is impacted by measuring distance, concluding that classification performance starts to degrade at 34 feet. Finally, Cheng *et al.* proposed in [70] a method capable of identifying identical laptops and smartphone devices (also different models) based on the electromagnetic signals radiated from the CPU. As a drawback, this solution requires the use of an external sensor to measure the CPU radiated signals within a 16 mm range.

Table VII compares the solutions focused on individual device identification. As a general view of individual device identification solutions, it can be appreciated that solutions are focused on general computers and wireless devices. This ensures solution universality, but opens the door to future perspectives focused on more specific device types such as IoT or ICS. It is also noticed the lower-level nature of the behavior sources utilized, which in this case are mainly based on clock and processor properties, and electromagnetic signals. Many solutions achieved high individual identification performance. However, many of these approaches noticed scalability issues in large device deployments, as fabrication variations are limited within determined quality standards.

C. Attack Detection

The third main scenario where behavior fingerprinting is highly relevant is attack detection. Abnormal situations can have a wide range of forms, such as network attacks, malware, malicious firmware modifications, or unauthorized user interactions. Detection can be performed either modeling normal device behavior and detecting deviations, from an anomaly detection standpoint, or collecting normal and abnormal labeled data and performing classification tasks. Table VIII compares the main characteristics, algorithms applied and performance of solutions detailed in this subsection.

TABLE VII
INDIVIDUAL DEVICE IDENTIFICATION SOLUTIONS BASED ON DEVICE BEHAVIOR FINGERPRINTING (WORKS ARE GROUPED BY BEHAVIOR SOURCE, USING DOUBLE HORIZONTAL LINES TO SEPARATE THEM, AND SORTED BY YEAR)

Work	Year	Device Type	Approach	Algorithms	Behavior Source	Features	Dataset	Classes	Results
[93]	2007	General computers	Classification	Statistical	System processors and oscillators	RTC and DSP drift compared to the TSC	Private	Different physical devices	98.5% and 93.3% of differentiation by RTC and DSP in 38 PCs, respectively.
[94]	2018	General computers	Classification	Statistical (Mode)	System processors	Matrix of code execution times	Private	Different physical devices	100% host-based and +80% web-based device identification in two sets of 89 and 176 PCs.
[154]	2020	IoT Devices	Classification	Statistical	System circuits	Outputs based on voltage	Private	IoT sensors	0% to 94.3% FPR in individual sensor and 0% FPR in model identification.
[60]	2009	Wireless access points	Classification	Expectation Maximization	Clock skew	Wi-Fi beacons timestamps	[155]	Known APs	Clock skew is a robust method and can detect different WLAN APs.
[62]	2012	General computers	Classification	Statistical	Clock skew	TCP and ICMP timestamp	Private	Different physical devices	Both identical and different devices correctly identified.
[61]	2012	Wireless devices	Data analysis	Statistical	Clock skew	Wi-Fi beacons timestamps	Private	Different physical devices	Clock skew is not enough to uniquely identify a large set of devices.
[74]	2014	Wireless devices	Classification	ANN	Clock skew + Network	Communication skew and patterns	[156]	Individual devices and device type	From 99 to 95% accuracy and 74% recall on ID, and 86% accuracy and 68% recall on type classification.
[63]	2015	General computers	Data analysis	Statistical	Clock skew	TCP timestamps	Private	Different physical devices	Clock skew is only suitable for small networks or combined with other data.
[30]	2019	General computers	Classification	Dynamic Time Warping	Resource usage	CPU usage-based graph	Private	Physical devices	93.43% of uniqueness in the generated fingerprints of 10 identical devices.
[14]	2018	Wireless devices	Classification	MLP, CNN, LSTM	Electromagnetic signals	Radio frequency IQ samples	Private	Different physical devices	96.3% accuracy for MLP, 94.7% for CNN and 75% for LSTM when identifying 6 identical ZigBee devices.
[68]	2018	Wireless devices	Classification	CNN	Electromagnetic signals	Raw frequency IQ samples	Private	Different physical devices	98% accuracy is achieved when identifying 5 identical devices.
[70]	2019	Laptops and Smartphones	Classification	Extra-Trees	Electromagnetic signals	CPU radiated magnetic signals	Private	Different physical devices	99.1% average precision and recall for all devices (70), and >98.6% precision and recall for 30 identical devices.

1) Network-Based Attack Detection: The most exploited source in terms of behavior-based attack detection is network monitoring. Many solutions, mainly focused on IoT [5], [7], [9], [10], [15], [34], [56], [72], [77], [78], [87], [88], [158] but also on SDN/NFV [79], [80] and general computers [73], [81], [82], [83], have utilized this source for attack detection.

One of the leading research lines focuses on detecting attacks that deploy unauthorized devices in the environment. In [78], the authors worked on unauthorized IoT device detection using white lists and classification ML algorithms. TCP/IP flows were used to extract features capable of characterizing nine different types of devices (17 distinct IoT devices were used). This work also discussed the system resilience to cyberattacks. Similarly, in [77], the authors used packet headers and payload data to extract flow-based features capable of creating device type fingerprints. Then, unknown or suspicious devices with abnormal behavior could be identified, and their communication restricted for further monitoring. The dataset used for testing came from IoT Sentinel [12]. In the same line, Ferrando and Stacey [9] built a behavior profile of IoT devices based on entropy and dispersion of metrics related to IP directions, ports, bytes received/sent, and latency. Anomalies were detected based on the distance between the average values and the ones being evaluated.

In contrast, the majority of works in this area cover the detection of direct cyberattacks, both common ones such as flooding or port scans, and more sophisticated ones like DDoS, botnets or ransomware. Amouri *et al.* [72] proposed an IDS based on IoT device network behavior. This system had a distributed architecture composed of traffic sniffers in the local network and a central super node. Device behavior was built on packet counters determined by MAC and network layer data. The proposed architecture applied DT algorithm to classify network instances, and then Linear Regression to generate time-based device profiles relying on the measure of behavior fluctuation.

Also from an ML-based perspective, Sivanathan *et al.* [5] addressed behavioral changes and attack monitoring based on flow and packet network analysis and clustering. The authors tested both direct network attacks (ARP Spoofing, Ping of Death, TCP SYN flooding, and Fraggle) and reflection attacks (Smurf, SNMP, SSDP, and TCP SYN reflection). A similar approach was followed in [88], where the authors performed attack and anomaly classification using MQTT protocol traces gathered from DS2OS dataset [159]. In the same line, Filho *et al.* [73] presented an approach for detecting DoS/DDoS attacks using ML techniques. The authors built a customized attack dataset based on several public datasets (CIC-DoS, CIC-IDS2017, and CIC-IDS2018 [160]) to benchmark normal traffic and different DoS/DDoS classification. The solution presented in [81] also considered network traffic data extracted from the CIC-IDS 2017 [160] dataset, but in this case for an unsupervised anomaly detection approach. Here, traffic sequences were modeled in sliding windows that were fed to an LSTM network. Similarly, traffic-based anomaly detection is covered by a wide variety of other works using anomaly detection approaches [79], [87].

A different view was provided by Yin *et al.* [82], who applied DL for botnet behavior modeling and detection. This solution was based on a GAN that generates simulated data, augmenting the model trained with the original data. The authors utilized network flows derived from ISCX botnet

dataset [161] as benchmark. Also focused on botnet attacks, Blaise *et al.* [56] presented a bot detection technique based on host behavior. This solution was divided into three steps: characterizing the host behavior based on network signatures (aggregated attribute frequency distribution), inferring benign host behavior using clustering algorithms (DBSCAN), and classifying new hosts based on previously labeled instances. To validate the approach, the authors used the CTU-13 dataset [162]. On similar research paths, Maimó *et al.* [158] analyzed ransomware detection based on behavior analysis in Medical Cyber-Physical Systems. This work analyzed network flows extracting different statistical features. Then, anomaly detection and classification ML models were combined to evaluate the live generated vectors.

In another line, some authors have proposed the usage of Manufacturer Usage Descriptions (MUDs) to enhance IoT security. In Hamza *et al.* [10], flow counters were used to generate feature vectors, applying PCA and k-means for dimensionality reduction and clustering, respectively. Then, an approach based on boundary detection and Markov Chains was applied for MUD monitoring and anomaly detection, testing it on several network attacks such as ARP spoofing, TCP SYN, and UDP flooding or reflection attack. Another approach using MUD to improve IoT security was proposed by Afek *et al.* in [80]. From an NFV perspective, this proposal presented a hybrid approach where MUD compliance checking is a service implemented as a virtual network function (VNF), and traffic monitoring is implemented on the network gateway to ensure P2P communications. For devices with no MUD, the authors used the algorithm proposed in [163] for MUD generation.

Additionally, other works also apply trust-based approaches to their solutions, increasing the granularity of the evaluation. Haefner and Ray presented ComplexIoT in [7], a behavioral framework designed to evaluate each traffic flow in an IoT device and calculate a trust score for it. The authors collected traffic of 25 devices approximately (general computers, smartphones, IoT devices). Based on the Flow Trust Score of each connection, calculated using IF, different policies and rules are applied to mitigate possible attacks. This solution is deployed on an enforcement architecture as an SDN environment based on OpenFlow.

From a distributed perspective, the authors of [15] used federated learning to build DÃŔoT, an autonomous self-learning distributed system for detecting compromised IoT devices. The system created communication profiles for each device based on network packets and flows. Then, an anomaly detection-based approach was applied to detect changes in the device behavior caused by network attacks (Mirai botnet). The architecture was deployed using a network gateway (router) as the Anomaly Detection component. Besides, an IoT Security Service was in charge of maintaining a repository of GRU models. Another DL-based distributed solution was proposed in [34], in which Ali *et al.* submitted an IoT device behavior capturing system powered by blockchain and designed to enable trust-level confidence to outside networks. The authors deployed a Trusted Execution Environment (TEE) [164] to provide a secure execution environment for sensitive code and blockchain data. The data came from the N-BaIoT dataset [165] and contained network features related to benign and botnet attack flows. Also from a distributed perspective, in [83], the authors proposed a behavior anomaly detection system based on network traffic. Here, the data was stored using a Hadoop Distributed File System (HDFS), and the processing was based on distributedly training a Deep Belief Network (DBN) and a stacked layer SVM using Apache Spark. The system was tested using different datasets, (KDD99 [166], NSL-KDD [167], UNSW NB-15 [168], CIC-IDS 2017 [160]).

2) Sensor-Based Attack Detection: Regarding sensor measurements to detect attacks, the main solutions based on this approach are applied to IoT and ICS environments [20], [89], [104], [105]. Pacheco and Hariri [89] focused on IoT sensor behavior analysis to detect common attacks such as DoS, Flooding or Impersonation. This approach recognized previously known and unknown attacks by calculating Euclidean distance from normal sensor measurements. The authors of [20] performed anomaly detection in cyber-physical systems (CPS), using GANs and time series data. From this perspective, the authors built an unsupervised GAN framework based on LSTM networks , which was tested using SWaT dataset [169], WADI dataset [170], and KDD99 dataset [166].

Similarly, Neha *et al.* [105] proposed a behavioral-based IDS for ICSs, in this case for SCADA systems. This approach applied RNNs to detect cyber-physical attacks. The model received sensor measurements gathered from the SWaT dataset [169]. Zhanwei and Zenghui [104] also proposed an anomaly detection system for ICSs, but based on the behavior of the data sequences from the industrial control Modbus/TCP network traffic. The authors tested their system both in a simulated water tank scenario and in a real chemical mixing infrastructure, utilizing sensor measurements to generate a behavior model and predict future behavior.

3) System Calls, Logs, and Software Signature-Based Attack Detection: Other solutions rely on system calls, execution logs, and software signatures to model device activity and detect attack situations [35], [96], [97], [99], [103]. These solutions cover a wide range of device types, including resource-constrained devices, general computers, and cloud systems.

Based on system call collection and processing, Gideon Creech [96] developed an IDS based on system call patterns. The authors utilized a semantic approach over the system call traces to understand running programs and detect anomalies utilizing an Extreme Learning Machine (ELM). A Linux system was monitored under different types of vulnerability exploitation attacks, and the dataset was made publicly available as ADFA-LD [96]. Also covering cloud intrusion detection using system calls, in [98], the authors developed a HIDS for cloud environments that utilized system calls to build a normal behavior profile based on Term Frequency-Inverse Document Frequency (TF-IDF). Then, ML-based classifiers were employed to recognize the attacks. Following similar paths, Liu *et al.* [99] developed a general IDS based on system call TF-IDF statistical patterns derived from n-gram models. In [97], Deshpande *et al.* also faced cloud computing intrusion detection based on system calls using ML classifiers and call frequency vectors.

From a different perspective, Attia *et al.* proposed in [35] an adaptive host-based anomaly detection framework for resource-constrained devices. The designed use case targeted the detection of malicious updates on Android applications. It generated a normal behavioral model for each monitored application using n-gram language models. Additionally, He *et al.* [103] proposed BoSMoS, a distributed software status monitoring system for Industrial IoT (IIoT) enabled by blockchain. To accomplish its goal, the system stored a snapshot of the device software in the blockchain and then monitored its system file calls. This solution was executed in 300s intervals, so modified software did not run for more than these 300s. Finally, the authors also tested solution scalability, performance, and security.

4) Hardware Event-Based Attack Detection: Apart from the behavioral data considered by the previous solutions, other works such as [90], [91], [92] used Hardware Performance Counters (HPC) to model system behavior. These solutions focused on resource-constrained devices such as embedded systems and IoT devices. In [90], the authors presented ConFirm, a technique to identify device behavior and detect malicious modifications in the firmware of embedded systems using HPCs. Deviations, based on execution paths, were calculated to evaluate the system performance. The proposal was tested on ARM and PowerPC embedded processors, verifying that the solution was able to detect all the tested modifications with low resource overhead. In [92], Golomb *et al.* proposed CIoTA, a lightweight framework using blockchain to perform distributed and collaborative anomaly detection in resource-constrained devices. In this solution, an Extended Markov Model (EMM) captured an application control-flow asynchronously using HPCs. Attack informing blocks were submitted to the blockchain (validated by neighbor devices) to ensure that an attacker cannot exploit a large number of devices within a short period of time. The system was tested in an IoT platform composed of 48 Raspberry Pi simulating smart cameras and lights. An exploit was executed to simulate a bot behavior in some devices. The authors also mentioned some countermeasures, such as alerts, service restart, or poweroff.

Ott and Mahapatra [91] utilized HPCs and their occurrence frequency to enable continuous authentication of embedded software. For this purpose, the HPCs streams were processed using Short-Time Fourier Transforms (STFT) to extract frequency information. The authors discussed the usage of classifiers; however, they considered these models too heavy for embedded systems and chose to build their own authentication algorithm based on cyclic redundancy check (CRC) function and state machines.

5) Resource Usage-Based Attack Detection: An alternative approach to detect anomalies caused by attacks consists in resource usage monitoring [3], [11], [21], applied mainly in cloud and container systems. Shone *et al.* proposed in [3] a misbehavior monitoring solution for DoS detection in cluster-based systems. This solution utilized resource usage metrics together with process and file modification monitoring to model the system behavior. Anomaly detection was addressed based on thresholds, clustering, and statistical similarity calculation. Similarly, Barbhuiya *et al.* proposed in [21] a DDoS and cryptomining attack detection framework for cloud data centers. The solution, called RADS (Real-time Anomaly Detection System), monitored CPU and network utilization as a time series for anomaly detection. Then, different window-based approaches were applied to perform attack identification based on IQR Spike detection analysis. For testing, a real-world dataset was gathered from Bitbrains data center [171].

Additionally, some works have also covered attack countermeasure actions. In this line, the authors of [11] presented an anomaly detection mechanism based on resource behavior designed to identify when a cloud system should be autoscaled. To detect anomalies, an AutoRegressive (AR) model was trained using CPU usage statistics, and the prediction error on the test dataset was used as anomaly measurement. The system was only tested using two DoS and stress attacks, detecting both of them.

The main characteristics of the attack detection solutions are summarized in Table VIII. Based on the attack detection solution analysis, we can claim that attack detection is the most varied behavior application scenario. Although network is the most used source, others such as system calls or resource usage also have notable relevance. The same heterogeneous distribution can be observed regarding processing and evaluation approaches, having a balance between classification and anomaly detection. The concrete sources and techniques applied are related to the type of attacks addressed. Thus, although many solutions achieved successful results, the rapid evolution of attack techniques leads to the need for new future solutions in this area.

D. Malfunction and Fault Detection

The last behavior application scenario identified is malfunction and fault detection. In these solutions, the purpose is to detect faulty devices or malfunctioning components based on device behavior changes. This approach has been applied to several device types, such as IoT [57], [86], ICSs [8], NFV systems [49], [95], [100], [108], general computers [101], cloud systems [36], [107], and containers [16], [109], [110]. Table IX compares the solutions detailed in this subsection.

1) Network-Based Fault Detection: Choi *et al.* [86] addressed faulty IoT device identification based on behavior fingerprinting from sensor data and its correlation. This solution was named DICE, and it was installed in the network gateway to extract context from application-layer communications and generate statistical features for a vector distance-based evaluation. In the same line, Spanos *et al.* [57] proposed a security solution based on the generation of behavioral templates using the IoT device network communications. PCA dimensionality reduction and DBSCAN clustering were applied to the network data to detect abnormal devices. Based on Euclidean distance, devices located far from a cluster center generated an alert and triggered some mitigation actions. This proposal was validated under simulated physical damage and mechanical exhaustion anomalies.

2) Sensor-Based Fault Detection: Sensor data has also been applied in the literature for fault detection.

TABLE VIII
MAIN ATTACK DETECTION SOLUTIONS BASED ON DEVICE BEHAVIOR FINGERPRINTING (WORKS ARE GROUPED BY BEHAVIOR SOURCE, USING DOUBLE HORIZONTAL LINES TO SEPARATE THEM, AND SORTED BY YEAR)

Work	Year	Device Type	Approach	Algorithms	Behavior Source	Features	Dataset	Attack Type	Results
[78]	2017	IoT Devices	Classification	RF	Network	Flow-based statistics	Private	Untargeted / targeted attacks	99% accuracy in white-listed devices and 96% in not white-listed.
[9]	2017	IoT Devices	Classification	ARIMA, Euclidean distance	Network	Header statistics	Private	Unusual changes and attacks	Anomalies visualized based on behavioral distance, no performance metrics were given.
[72]	2018	IoT Devices	Classification	DT, Linear Regression	Network	Mac and network layer counters	Private	Traffic anomalies	100% detection (TPR) after 3000s (3 reports).
[82]	2018	General computers	Classification	GAN	Network	Traffic flow statistics	[161]	Botnet behavior	74.04% precision, 71.17% accuracy, 70.59% F1-Score, 15.59% TPR for botnet activity detection.
[83]	2018	General computers	Classification	(Spark) DBN and SVM	Network	Traffic flow statistics	[166], [167], [168], [160]	Network attacks	93-97% F1-Score in the tested datasets.
[79]	2018	SDN	Anomaly Detection	SVM, kNN, MLP	Network	Traffic statistics	Private	DDoS, port-scan and flash crowd	Attacks were detected and mitigated
[81]	2018	General networks	Anomaly Detection	LSTM	Network	Traffic flows	[160]	Common network attacks	87% AUC average, over 71% AUC in all attacks.
[87]	2019	IoT Devices	Anomaly Detection	RPNI + RANSAC	Network	Application-layer series	Private	IoT anomalies	The attacks are discovered with high accuracy.
[7]	2019	PCs, IoT Devices	Anomaly Detection	IF	Network	Flow statistics	Private	DDoS and botnets	Different device confidence based on behavior Flow Trust Score.
[10]	2019	IoT Devices	Anomaly Detection	PCA, k-means, Markov Chains	Network	Flow counters	[10]	Network attacks	94.9% accuracy, 89.7% TPR, and 5.1% FPR.
[80]	2019	NFV	Anomaly Detection	While-listing (MUD)	Network	Traffic flows	Private	Unauthorized connections	Unknown connections forbidden
[77]	2019	IoT Devices	Classification	RF	Network	Flow-based statistics	[12]	Attack prevention	90.3% accuracy using RF, outperforming other ML algorithms.
[88]	2019	IoT Devices	Classification	SVM, RF, ANN, LR	Network	MQTT-traces features	[159]	DoS, control, Scan	99% F1-Score classifying normal and attack traces.
[73]	2019	General computers	Classification	RF	Network	TCP/IP header statistics	[73]	DoS/DDoS	96.5% attack detection rate, 99.5% F1-Score, 0.2% FAR
[158]	2019	CPSs	Anomaly Detection / Classification	OC-SVM / NB	Network	Flow statistics	Private	Ransomware attacks	95.9% F1-Score, 4.6% FPR in anomaly detection, and +99% classification accuracy.
[15]	2019	IoT Devices	Anomaly Detection	(Fed. Learn.) GRU	Network	Header statistics	To be published	IoT attacks	95.6% attack detection rate and fast ($\approx$257 ms) attack detection.
[5]	2020	IoT Devices	Anomaly Detection	PCA, k-means	Network	Header statistics	[4], [10]	Network attacks	91.3%-84.3% average detection rate for direct network attacks, and 99.1%-58.8% for reflection attacks.
[34]	2020	IoT Devices	Anomaly Detection	(Blockchain) Neural Network	Network	Flow statistics	[165]	DDoS attacks	99.2% TPR and 175 $\pm$ 230ms to attack detection.
[56]	2020	IoT Devices	Classification	DBSCAN	Network	TCP, UDP, ICMP headers	[162]	Botnet detection (and attacks)	100% TPR, 0.9% FPR
[89]	2018	IoT Devices	Classification	Euclidean Distance	Sensors	Sensor measurements	Private	Common network attacks	98% accuracy for known attacks and up to 97.4% for unknown attacks.
[20]	2019	ICSs	Anomaly Detection	LSTM-based GAN	Sensors	Measurement value sequences	[169], [170], [166]	Cyber-physical attacks	99.99%-46.98% precision, 99.98%-96.33% recall and 94%-37% F1-Score, depending on the dataset.
[104]	2019	ICSs	Anomaly Detection	Linear model	Sensors	Sensor measurements	Private	Tampering and MitM	5.5-6.4% FPR and 11-17% FNR
[105]	2020	ICSs	Classification	RNN	Sensors	Sensor value sequences	[169]	Cyber-physical attacks	98.05% accuracy and 97% TPR when classifying normal and injected data.
[96]	2013	General computers	Anomaly Detection	ELM	System calls	Semantic features	[96]	Vulnerability exploitation	100% TPR and 0.6% FPR.
[35]	2015	Mobile devices	Anomaly Detection	Look-ahead, N-gram tree	System calls	n-gram sequences	Private	Malicious app updates	$\approx$70% detection rate and 0% FPR. 20-50% CPU and $<$8% RAM.
[97]	2018	Cloud systems	Classification	k-NN	System calls	System call traces (audit)	Private	Anomalous call sequences	90% accuracy, 96% TPR, 42.5% TNR
[98]	2020	Cloud systems	Classification	RF	System calls	Frequency statistics	[172]	Anomalous system calls	100-94% TPF, 6.2-0% FPR and 6-0% FNR.
[99]	2020	General computers	Classification	IF, LOF, OC-SVM, k-NN	System calls	n-gram sequence stats.	[96], [173], [102]	Anomalous system calls	73.7% overall best AUC. $<$ 110s for evaluation.
[103]	2020	IoT Devices	Anomaly Detection	Hash equality checking	Software signatures	File snapshots	Private (Simulated)	Software modification	Executable modification detection within 300 seconds.
[90]	2015	Embedded systems	Anomaly Detection	Execution path deviation	Hardware Events	HPCs	Private	Firmware modifications	The system is plactical with low overhead
[92]	2018	IoT Devices	Anomaly Detection	(Blockchain) EMM	Hardware Events	HPCs app control-flow	Private	Adversarial attacks	Exploit execution easily identified, enhancing network overall security.
[91]	2019	Embedded systems	Continuous Authenticat.	Own (Window + Fourier + CRC)	Hardware Events	HPCs	Private	Abnormal software	97% TPR, 1.5% FPR in the authentication of embedded software.
[3]	2013	Cluster systems	Anomaly Detection	Threshold+ k-means + statistical	Resource usage	Hardware, process and file info	Private	DoS attacks	0.11% FPR and 0% FNR detecting DoS attacks, consuming only 0.5% RAM and 14% of CPU.
[21]	2018	Cloud data centers	Anomaly Detection	IQR	Resource usage	CPU, network	[171]	DDoS, Cryptomining	90-95% F1-Score and FPR of 0-3%
[11]	2018	Cloud systems	Anomaly Detection	Autoregressive (AR) model	Resource usage	CPU	Private	DoS, service stress attack	Attacks are fully detected

In this line, Manco *et al.* [8] explored ICS fault detection based on sensor stream data analysis. The system performed window-based processing to obtain statistical features, and then clustering to build classes from unlabeled data. Finally, outlier detection was performed to distinguish failures using Expectation Maximization

TABLE IX
MAIN MALFUNCTION AND FAULT DETECTION SOLUTIONS THAT USE DEVICE BEHAVIOR FINGERPRINTING (WORKS ARE GROUPED BY BEHAVIOR SOURCE, USING DOUBLE HORIZONTAL LINES TO SEPARATE THEM, AND SORTED BY YEAR)

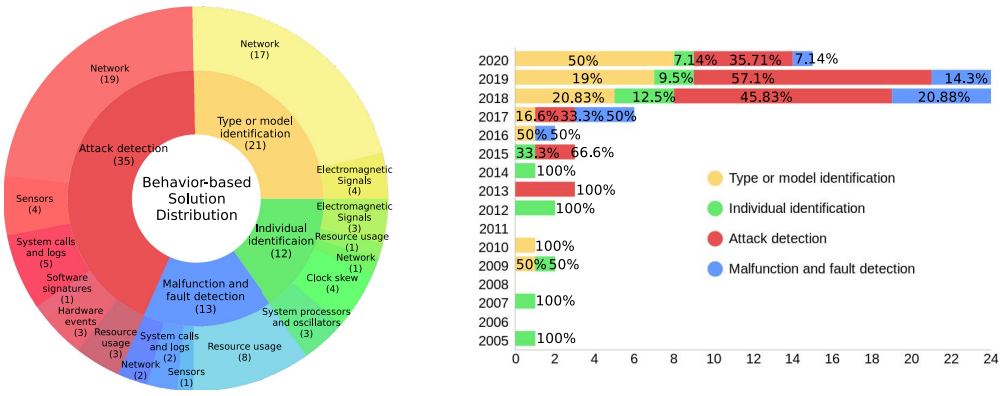
Work	Year	Device Type	Approach	Algorithms	Behavior Source	Features	Dataset	Anomaly	Results
[86]	2018	IoT Devices	Anomaly Detection	Vector distance	Network	Sensor values statistics	[176], [177]	Faulty IoT sensors	94.9% and 92.5% average precision and recall, respectively. 3 mins for detection.
[57]	2019	IoT Devices	Classification	PCA, DBSCAN, Euclidean distance	Network	Statistical features	Private	Physical and mechanical errors	Successful threat detection regarding physical damage and mechanical exhaustion.
[8]	2017	ICSs	Anomaly Detection	Expectation Maximization	Sensors	Sensor values statistics	Private	System Faults	89.5% AUC detection train door failures.
[100]	2019	NFV systems	Anomaly Detection	LSTM	System logs	Execution traces statistics	Private	Microservice anomalies	>90% accuracy using real-word cloud traces.
[101]	2019	General computers	Statistical Analysis	PANAL (time series)	System logs	Performance metrics	Private	Anomalous behavior	Study on metric correlations regarding performance, event, and process logs.
[95]	2016	NFV systems	Anomaly Detection	Clustering and Classification	Resource usage	CPU, memory, disk, network	Private	NFV anomalies	95% recognition of pre-defined anomalous scenarios.
[109]	2017	Cluster systems	Anomaly Detection	PCA	Resource usage	CPU, memory, disk, network	Private	Cluster anomalies	Anomalies correctly detected.
[107]	2017	Cloud systems	Anomaly Detection	Robust PCA	Resource usage	CPU, memory, disk	Private	Cloud faults	88.54% accuracy and 86% F1-Score
[49]	2018	NFV systems	Anomaly Detection	Online ARIMA	Resource usage	CPU, memory, disk, network	Private	NFV Resource anomalies	100% accuracy detecting controlled HDD, CPU and memory anomalies.
[36]	2018	Cloud systems	Anomaly Detection	PCA, eigenvector	Resource usage	CPU, memory, disk, network	Private	Cloud faults	The system detects injected test faults.
[108]	2018	NFV systems	Anomaly Detection	BIRCH	Resource usage	CPU, memory, disk, network	Private	System anomalies	All anomalies detected, except 83% detection for memory leak and CPU stress.
[110]	2018	Microservices Containers	Classification	SVM, RF, k-NN, NB	Resource usage	CPU, memory, network	Private	Container anomalies	97-93% F1-Score using k-NN as classifier.
[16]	2020	Container clusters	Anomaly Detection	HHMM	Resource usage	CPU, memory	Private	Resource exhaustion	95-90% F1-Score and 19-31% FAR.

algorithm. This approach was tested in train door failure detection.

3) System Log-Based Fault Detection: From the system log perspective, in [100], the authors applied a multimodal LSTM network approach to perform anomaly detection in NFV microservices based on distributed execution traces. Kubacki and Sosnowski [101] explored abnormal behavior detection based on system logs related to performance metrics. The authors performed a pulse-oriented time series analysis to characterize periodical behaviors and detect anomalies using a self-developed algorithm called PANAL. The correlation between metrics was evaluated on real logs, finding a high correlation during anomalous situations such as truncated cyberattacks or data backups.

4) Resource Usage-Based Fault Detection: In the malfunction and fault detection scenario, the most common data source is resource usage, especially for fault finding in cloud and container systems. In this context, Gulenko *et al.* [95] proposed an anomaly detection architecture for large-scale NFV systems. In this proposal, different resource usage metrics were collected from each host in short time intervals. To process the data, the architecture used techniques based on online unsupervised clustering and classification algorithms. The authors claimed that the preliminary evaluation showed a high degree of reliable recognition of pre-defined failure scenarios. In addition, Sorkunlu *et al.* [109] published a method to track the behavior of a cluster system based on its resource usage. Data was organized into three-dimensional tensors (compute nodes, usage metrics, and time). To measure behavior changes, data was grouped in ten-minute time windows and dimensionality reduction algorithms were applied. Finally, the reconstruction error was measured. In [49], by the same team as [95], the authors proposed an unsupervised detection approach using the Online ARIMA forecasting algorithm [174]. This model was based on predicting the next expected values and comparing them with the actual ones. The authors introduced controlled anomalies, such as disk pollution, or HDD, CPU, and memory stress and leak, being able to recognize all of them. This team also addressed black-box service modeling [108] based on clustering to detect functioning anomalies like in the previous work. The used clustering algorithm was BIRCH [175].

Following a similar approach, Wang *et al.* [36] proposed a self-adaptive monitoring architecture for online anomaly detection in cloud computing. The system gathered performance metrics from different sources such as CPU, Network, Memory, and Disk. To calculate anomalies, the PCA-based eigenvector of the metrics was compared to the standard eigenvector. The adaptability could be achieved by adjusting a sliding window based on the estimated anomaly degree. A similar line to this work was covered by Agrawal *et al.* [107], where similar features were collected and PCA was used as dimensionality reduction algorithm. Besides, Du *et al.* [110] proposed a framework to monitor and classify anomalous behaviors in microservices and containers. Different anomalies, such as high CPU consumption or memory leak, were injected, and the generated data was labeled for using ML classifiers. Finally, Samir and Pahl [16] utilized hierarchical hidden Markov models (HHMM) to detect anomalies in container clusters. HHMM model was compared with Dynamic Bayesian Network and Hierarchical Temporal Memory to detect resource exhaustion and workload contention, achieving the best results in three different generated datasets.



(a) Scenario and source distribution scheme. (Internal ring: Application Scenario. External ring: Behavior Source.)

(b) Yearly solution distribution.

Fig. 5. Distribution graphs of device behavior fingerprinting solutions.

Table IX compares the main characteristics and results of the solutions focused on fault and malfunction detection. From the description of the previous solutions, we can observe that resource usage and system logs are the most used behavior source for fault detection, especially in NFV, cloud, containers, and microservice systems. In contrast, IoT devices and ICSs faults have been solved based on a network and sensor-based perspective. Moreover, most of the solutions are focused on anomaly detection-based evaluation, instead of using labeled data. Finally, Fig. 5 shows the distribution of the analyzed solutions regarding their application scenario and behavior source, and their publication year.

VI. PUBLIC DATASETS

To address *Q4* (*Which behavior datasets are available and which are their characteristics?*), this section reviews the main public datasets containing device behavior activities and characteristics found in the literature. Specifically, it analyzes datasets contemplating the scenarios, devices and sources discussed in *Q1*, allowing validating most of the techniques presented in *Q2*, and studying the scope of the solutions analyzed in *Q3*. Each dataset is described by taking into account the devices and sources monitored, and data morphology. Below, the analysis is organized according to the two main application scenarios stated in Section III, which are Device identification and Misbehavior detection–attack and anomaly detection.

A. Device Identification Datasets

Several datasets published in recent years and collecting device behavior are conceived to perform device model, type, or individual identification. In 2006, Maya Rodrig *et al.* published the uw/sigcomm2004 dataset [155]. The main purpose of this dataset is to analyze how Wi-Fi networks work and how they can be improved. This dataset contains 70 GB of both wired and wireless traces. The wireless traces were collected for five days using three computers in monitor mode near access points. Selcuk Uluagac published in [156] the dataset associated with his research work on network-based individual device identification [74], [157]. This dataset contains the inter-arrival time of network traffic packets collected from 30 wireless devices. 1.5 GB of data was collected both actively, directly communicating with the devices, and passively, sniffing the communications. This dataset can be used to generate network-based fingerprints and derive parameters such as approximated clock skew.

With a similar goal, but focused on IoT, Miettinen *et al.* published the IoT Sentinel dataset [12]. This dataset contains the traffic generated during the setup of 31 IoT devices of 27 different types (4 types have 2 devices). To avoid anomalies and have data variety, the device setup process was collected at least 20 times for each device, generating a total of 64 MB of data. Another dataset dealing with IoT devices is the Yourthings dataset [140], which contains raw network traffic from 45 different smart-home IoT devices. The data was collected for 10 days in March and April of 2018. Each day data contains from 10 to 13 GB. Following the same approach, in [4], Sivanathan *et al.* published a dataset collected for IoT device classification under IoT Traffic Traces name. The data was collected in 2016 for 20 days from 28 different IoT devices, including cameras, lights, plugs, sensors, appliances, and health-monitors. In addition, this dataset also includes captures from non-IoT devices such as laptops and smartphones. In total, ≈9.5 GB of raw pcap files are available. As additional content, post-processing tools to obtain IP, NTP, and DNS flows are also enclosed. Regarding radio frequency, Allahham *et al.* [153] published DroneRF in 2019, a dataset containing 3.8 GB of radio data collected from 3 different drones during functioning. This dataset has been designed for drone detection, identification and tracking. More recently, Hagelskjær *et al.* published in 2020 a dataset designed for IoT device identification based on radio

TABLE X
MOST RELEVANT DEVICE IDENTIFICATION DATASETS THAT USE DEVICE BEHAVIOR FINGERPRINTING

Dataset	Year	Device Type	Data Source	Data	Size	Details
The uw/sigcomm2004 dataset [155]	2006	Wireless and wired devices	Network	Raw traces	70 GB	This dataset includes the traces collected by wireless and wired monitoring using tcpdump.
The gatech/fingerprinting dataset [156]	2014	Wireless devices	Network	Inter-arrival time information	1.5 GB	Inter-arrival time information collected from 30 wireless devices to generate unique fingerprints.
IoT Sentinel [12]	2017	IoT devices	Network	Raw traces and processed features	64 MB	Network communications dataset collected during the setup process of 31 devices.
Yourthings [140]	2018	IoT devices	Network	Raw traces and processed features	+110 GB	10 days of network traffic collected from 45 different smart-home IoT devices. Flows utilized to evaluate security.
IoT Trace Dataset [4]	2018	IoT devices	Network	Raw traces and processed features	≈9.5 GB	Network flows collected during 20 days from 28 different IoT Devices. The source includes tools to derive flow statistics.
DroneRF [153]	2019	IoT devices (Drones)	Radio spectrum	Radio segments	3.8 GB	227 segments collected from 3 different drones during functioning.
Device spectrum identification [178]	2020	IoT devices	Radio Spectrum	Raw spectrum and processed features	+50 GB	863-870 MHz radio spectrum measurements collected in diverse scenarios, like in the same room and different rooms.

spectrum monitoring [178]. The dataset contains +50 GB of 863-870 MHz band raw spectrum measurements with a sampling frequency of 10 MSPS collected in November 2018. The published dataset contains both raw spectrum captures and pre-processed features extracted with PCA.

Table X summarizes the public datasets previously described, paying attention in their publication year, monitored devices, and data sources collected. Most of the datasets (5 of 7) contain network traces or network-based features. It could be due to the facility to monitor from outside the device behavior without modifying its software. Furthermore, this source is quite generic as almost every device has at least one network interface. Additionally, the two datasets not based in network communications contain spectrum measurements, another externally-collected source. In this context, there is a missing spot for device identification datasets containing sources such as clock skew, system logs or events, and resource usage metrics.

B. Anomalous Behavior and Attack Datasets

The second dataset category is based on public datasets containing anomalous device behavior, either based on attacks or other exceptional situations. Note that most of these datasets also contain normal or benign device behavior, which can be utilized to model normal device behavior and identify it, like in the previous subsection. Next, the main datasets found in the literature will be detailed.

The family of datasets that considers network communications to create device behavior fingerprints is extensive. One of the most representative is the CTU-13 dataset [162], a botnet traffic activity dataset collected in 2011. 13 different botnet samples were captured during different attack conditions such as Command and Control (C&C) connection and the launching of diverse attacks –DDoS, or port scanning, among others. Additionally, the dataset also contains normal and background network traffic. In total, this dataset contains +140 hours of network traffic with a total size of ≈700 GB. Besides, the dataset has been updated in the last years to include IoT malware captures. A set of relevant datasets, IDS 2017 and 2018 datasets [160], was created by the Canadian Institute of Cybersecurity (CIC). They contain raw network traces and derived features obtained during different network attacks. Concretely, the monitored attacks were FTP and SSH Brute Force, DoS, Heartbleed, Web Attacks, Infiltration, Botnet, and DDoS. In addition, these datasets also contain benign traffic. The 2017 dataset was collected from 25 users and contains 51.1 GB of data, while the 2018 dataset contains 220 GB of traffic from 500 different devices. The previous datasets were collected and processed by Filho *et al.* [73] to extract ≈40 MB of vectors with 73 features relative to IP headers of the traffic flows. Then, the dataset was published together with a research article. Also from CIC, the ISCX botnet dataset [161] contains raw network captures of 16 different botnet malware. This dataset is generated by combining previous CIC datasets containing botnet activity. In total, the dataset contains 5.3 GB of training traces and 8.5 GB for testing. Aligned with the previous datasets, in [179], the authors provided a novel network dataset, published in September 2019, which contains several types of attacks in an IoT environment. The dataset is composed of ≈ 1.5 GB of real and simulated attacks, such as port scanning, flooding, brute force, or ARP spoofing, among others. In the case of real attacks, the network packets were obtained from Mirai botnet. To identify the network behavior of the devices infected, packets were captured while simulating attacks through tools such as NMAP.

Anomalous behavior or attacks affecting IoT devices is another cutting edge field where several datasets have been created and published. In this sense, the N-BaIoT dataset [165] contains more than 7 million vectors, with 115 features each, giving around 20 GB, obtained by processing the network communications of 9 different IoT commercial devices under attack. Vectors contain 11 labels, 10 for different botnet attacks, produced by Mirai and BASHLITE, and 1 for benign traffic. Similarly, the DS2OS dataset [159] contains 61 MB of features obtained from application layer traces collected from simulated IoT devices such as light controllers, thermometers, movement sensors, washing machines, batteries, thermostats, smart doors, and smartphones. This dataset is designed for anomaly detection in IoT node communications. In the same line, the USNW IoT Benign and Attack Traces Dataset [10] monitored network communications of 27 devices for 30 days, being 10 of these devices victims of network attacks such as ARP spoofing, TCP/UDP flooding, and packet reflection. In total, more than 64 GB of data is available. This dataset also provides the source code to derive vectors with 238 features using packet counters and traffic flows. Another relevant dataset is the NGIDS-DS dataset [102], which consists of

6.7 GB of labeled network and device operating system logs collected on a simulated critical infrastructure. The dataset is designed for host-based intrusion detection and contains normal and attack scenarios. The authors used the *IXIA Perfect Storm* tool to generate a wide variety of network attacks. The data was obtained from a machine running *Ubuntu 14.04* and different common services such as *Apache*. The OS logs contain the date, process id, system call, event id, and the network data consist of raw traffic.

A similar approach was followed to generate the UNSW-NB15 dataset [168]. This dataset contains 100 GB of raw traffic flows and derived features from several attacks launched using *IXIA Perfect Storm*. This attack set includes the same type of attacks as NGIDS-DS dataset. The Aposemat IoT-23 dataset [180], published in January 2020 by the same team as for CTU-13 [162], is another labeled dataset containing 23 captures of malicious and benign IoT network traffic. Concretely, 20 captures include malware activity, while 3 include normal network activity of 3 IoT device types. The dataset includes 11.3 GB of pcap files and 8.7 GB of network log files. The authors utilized known malware, such as Mirai, Okiru, or Torii botnets, port scanning, DDoS, C&C connections. In the same direction, IoT-KEEPER dataset [55] was published in 2020. This dataset contains 11.8GB of pcap files collected from several IoT devices affected by common attacks such as port scanning, botnet execution, DoS, or malware injection. Besides, it also contains network activity from real computers, replicating a real edge network environment. Finally, LITNET-2020 dataset [181] contains feature vectors generated during 12 attacks on general computers deployed on an academic network. In total, this dataset contains 26.9 GB of vectors with 85 processed flow features extracted using Netflow.

Focused on application layer communications of general computers, ECML-PKDD 2007 [182] and HTTP CSIC 2010 [183] datasets are available. ECML-PKDD 2007 [182] contains 80 MB of application layer requests in XML format. There are 25000 valid and 15000 attack requests, the attack requests include SQL Injection, LDAP Injection, cross-site scripting (XSS), and command execution, among others. The data includes Web requests and also context information such as server operating system, services, etc. The HTTP CSIC 2010 dataset [183] includes 56 MB of normal and abnormal HTTP requests. It was published by the Spanish Research National Council (CSIC) to test Web application attack protection systems. The dataset is divided into 36000 normal and 25000 anomalous requests. The anomalous requests are divided into three types of attacks: static, dynamic, and unintentional illegal requests. Concretely, static attacks try to gather hidden resources, while dynamic attacks are SQL injections, XSS, etc. This dataset is usually used as benchmark for HTTP anomalous behavior detection solutions.

From the system calls and execution traces perspective, it is worth commenting the ADFA Intrusion Detection Datasets for Linux [96] and Windows [173]. These datasets contain 9 MB of Linux system call identifiers and 13.6 GB of Windows XML system call traces of DLL libraries. Both datasets include normal and attack system calls. Attacks include HydraFTP, HydraSSH, Meterpreter, Webshell, and a poisoned executable. Currently, these are widely used for benchmarking solutions based on system call traces [184], [185]. The Firefox-SD dataset [186] is also based on system calls, but in this case made by *Firefox* browser in Linux. The dataset contains +1 TB of normal activity traces, collected while executing seven browser testing frameworks, and attack-based traces, generated under attacks using known exploits such as memory consumption, integer overflow, or null pointer exploit.

Dealing with ICSs and anomaly detection, one of the reference datasets is the Secure Water Treatment (SWaT) dataset [169]. This dataset was collected in 2016 from a real water treatment testbed managed by a SCADA system. It contains 11 days of continuous operation, 7 of them normal and 4 under attack by 36 different data injections. This dataset contains ≈16 GB of traffic logs and 361 MB of measurements obtained from 51 sensors and actuators. Additionally, SWaT dataset was updated in December 2019 with 45 GB of raw traffic and 6 MB of measurement logs, collected during 3 hours of normal traffic and 1 hour in which 6 attacks were launched. Similarly, the Water Distribution (WADI) dataset [170] contains 575 MB of labeled sensor and actuator logs collected in the same water treatment plant. In this case, the dataset contains data from 123 sensors and actuators collected during 16 days of operation, having 14 days of normal traffic and 2 days with 15 data injection attacks launched in total. Also in the ICS field, in [133], Perales *et al.* developed a dataset called Electra, based on a railway electric traction substation. The monitored network protocols were Modbus TCP and S7Comm, common in SCADA systems. This dataset contains 1.7 GB of derived features originating from raw captures.

Regarding resource usage monitoring, the GWA-T-12 Bitbrains dataset [171] contains performance metrics collected from 1750 virtual machines located in Bitbrains data center. Resource usage metrics are collected in five-minute samples, the monitored resources are the CPU usage, memory usage, disk read/write throughput, and network received/transmitted throughput. In total, 2.7 GB of traces are available, divided into two sets of machines (1250 VMs used for fast storage and 500 with lower performance). Although BEHACOM [187] dataset is focused on user activity monitoring (keyboard and mouse interactions), it also contains resource usage metrics regarding active applications, CPU, and memory. This data was collected from the computers of 12 users over 55 days. In total, this dataset contains 6.1 GB of features derived from user activity. Also dealing with resource usage monitoring but from the mobile devices prism, CIC has released two different datasets on dynamic smartphone behavior and its relationship with malware. The first one is CIC-AAGM (CIC Android Adware and General Malware) [188], which contains +20 GB of traffic flows generated when installing 1900 different applications, being 250 adware, 150 malware, and 1500 benign. The second is InvesAndMal2019 [189] dataset, which includes device status, traffic flows, permissions, API calls, and logs generated by 426 malware and 5065 benign Android applications. In total +275 MB of logs and features are available.

Other existing datasets are more than 20 years old, which makes them outdated with regard to current scenarios. This is

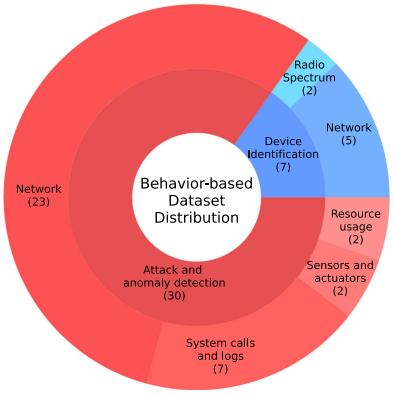
TABLE XI
MOST RELEVANT ANOMALOUS BEHAVIOR AND ATTACK DATASETS THAT USE DEVICE BEHAVIOR FINGERPRINTING

Dataset	Year	Device Type	Data Source	Data	Size	Details
DARPA [190], [191]	1998-1999	General computers	Network and system logs	Raw network packets and logs (bsm)	≈ 10 GB	Attack and normal network and system activity. One of the most used IDS datasets, but it is outdated.
KDD99 [166]	1999	General computers	Network	Connection record features	1.2 GB	Derived features based on DARPA 1998/1999 network traffic.
UNM dataset [172]	1999	General computers	System calls	System calls and process IDs	≈500 KB	System call identifiers collected during normal behavior and under some attacks.
ECML-PKDD 2007 [182]	2007	Web systems	Network	Requests and contextual information	80 MB	25000 valid and 15000 attack XML web queries, including context information such as server OS.
NSL-KDD [167]	2009	General computers	Network	Connection record features	60 MB	Based on KDD99 data, but with additional processing like filtering duplicated data.
HTTP CSIC 2010 [183]	2010	Web systems	Network	HTTP requests	56 MB	61000 normal and anomalous HTTP requests. It includes diverse attacks and also unintentional illegal requests.
CTU-13 [162]	2011	General computers	Network	Raw captures and flows	≈700 GB	13 different scenarios were botnet activity is combined with normal traffic.
Firefox-SD [186]	2013	Application (Firefox)	System calls	Raw system calls	+1 TB	Firefox browser system calls while normal activity and under different attacks.
ADFA-LD [96]	2013	General computers	System calls	Linux system logs	9 MB	System calls collected on 60 different attack sets.
ADFA-WD [173]	2014	General computers	System calls	XML Windows DLL traces	13.6 GB	System call dataset composed by virtual kernel calls done by DLL libraries.
CIC-ISCX [161]	2014	General computers	Network	Raw captures	13.8 GB	Botnet activity dataset collected from 16 real botnet malware.
GWA-T-12 Bitbrains [171]	2015	Distributed data centers (Cloud)	Resource usage	CPU, Memory, Disk and Network statistics	2.7 GB	Performance metrics (CPU, memory, disk and network) collected from 1750 VMs each 5 mins.
SWaT [169]	2016	ICSs	Network, and sensors/actuators	Network and sensor/actuator logs	≈16.3 GB	7 days of normal activity and 4 days of data injection attacks in a real water treatment testbed.
WADI [170]	2016	ICSs	Sensors / actuators	Sensor/actuator logs	575 MB	16 days of logs of 123 industrial sensors and actuators. 15 attacks launched over 2 days.
NGIDS-DS [102]	2017	Critical infrastructure	Network and system logs	Raw network packets and audit logs	6.7 GB	Critical infrastructure attacks simulated on an Ubuntu 14.04 machine using IXIA PerfectStorm tool.
UNSW-NB15 [168]	2017	General computers	Network	Raw captures and processed features	100 GB	IDS dataset, attacks generated using IXIA PerfectStorm tool.
CIC-IDS 2017[160]	2017	General computers	Network	Raw captures and processed features	51.1 GB	IDS dataset based on 25 users activity, it contains common network attacks.
CIC-AAGM [188]	2017	Mobile devices	Network	Raw captures and processed features	+20 GB	Flows generated by 1900 different applications (250 adware, 150 malware, 1500 benign).
DS2OS [159]	2018	IoT devices	Network	Application traces	61 MB	IoT smart home devices normal and abnormal activity.
N-BaIoT [165]	2018	IoT devices	Network	Processed features	≈20 GB	Botnet (Mirai and BASHLITE) activity collected from 9 IoT devices.
CIC-IDS 2018 [160]	2018	General computers	Network	Raw captures and processed features	220 GB	IDS dataset collected in 500 devices which contain common network attacks.
Smart-Detection [73]	2019	General computers	Network	Processed features	≈40 MB	DoS detection based on previous datasets (CIC-DoS, CIC-IDS 2017 and CIC-IDS 2018).
ELECTRA [133]	2019	ICSs	Network	Processed features	1.7 GB	Data collected from attacks to an electric traction system.
USNW IoT Benign and Attack Traces [10]	2019	IoT devices	Network	Raw captures and processed features	+64 GB	IoT benign and attack network traces. Attacks include ARP spoofing, TCP/UDP flooding and packet reflection.
IoT network intrusion dataset [179]	2019	IoT devices	Network	Raw captures	≈1.5 GB	Network captures of real and simulated attacks to IoT and non-IoT devices.
InvesAndMal2019 [189]	2019	Mobile devices	System logs and Network	Processed logs and features	+275 MB	Device status, traffic flows, API calls and logs generated from +5500 apps (426 malware and 5065 benign).
BEHACOM [187]	2020	General computers	Resource usage	CPU and memory statistics	6.1 GB	Active application, CPU and memory statistics collected from 12 users over 55 days.
IoT-23 [180]	2020	IoT devices	Network	Raw captures	20 GB	By the same team that CTU-13. 20 attack and 3 benign traces. Attacks simulated using infected Raspberry Pis.
IoT-KEEPER dataset [55]	2020	IoT devices	Network	Raw captures	11.8 GB	Network captures collected from IoT devices under common attacks.
LITNET-2020 [181]	2020	General Computers	Network	Processed features	26.9 GB	Dataset collected on an academic network under 12 different attacks.

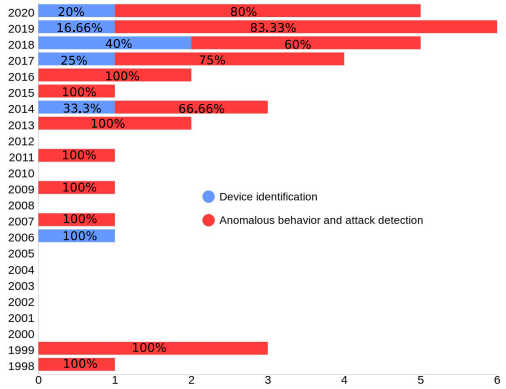
the case of DARPA 1998/1999 [190], [191], KDD99 [166], and NSL-KDD [167] datasets. The original datasets, DARPA 1998 and 1999, are composed of ≈ 10 GB of network traffic and system logs collected by MIT Lincoln Laboratory. The aim of these datasets was to build a generic evaluation dataset for intrusion detection. 56 different attacks were recorded, including different DoS, buffer overflow, and reconnaissance attacks, among others. The network traces were stored in tcpdump format and the system logs as BSM/NT audit data. Afterward, KDD99 dataset was derived from DARPA traffic by extracting 1.2 GB of features from the traffic flows. Besides, NSL-KDD is a refinement of KDD99 were duplicated entries are deleted and classes are more balanced, reducing the dataset to around 60 MB. These datasets have become some of the most popular datasets for intrusion detection evaluation. However, as commented before, they are outdated compared to current networks and attacks.

The same issue occurs with the system call dataset of the University of New Mexico (UNM) [172]. This dataset was collected in 1999 and contains ≈500 kB of system call and process identifiers. The collected system calls contain normal activity and different attacks such as buffer overflows and trojans. This dataset has been widely used as benchmark for system call anomalies-based attack detectors [111]. However, the system call arguments are not available and it is outdated regarding modern attacks.

Table XI gives an overview of the public datasets with focus on behavior anomaly and attack detection. It can be



(a) Scenario and source dataset distribution. (Internal ring: Application scenario. External ring: Behavior Source.)



(b) Yearly dataset distribution.

Fig. 6. Distribution graphs of device behavior fingerprinting dataset.

appreciated how most of the datasets are focused on network, followed by system calls and logs. The datasets monitoring the previous sources are varied and cover several device types such as IoT, ICSs, mobile devices, or general computers. However, other sources such as resource usage or HPCs are under-exploited regarding public datasets for anomaly detection. Datasets monitoring IoT device communications during attack and malware execution have gained importance for the last years and nowadays are the dominant type of anomalous behavior datasets. The most common IoT malware families, such as Mirai botnet, have been largely monitored from a network-based perspective in datasets such as CTU, USNW IoT Traces or IoT-23. However, there is no IoT-based dataset containing in-device behavior sources, something highly useful for modeling how malware works and what changes occur within the device functioning itself.

Fig. 6 shows the dataset distribution regarding main application scenarios and behavior source collected, and their publication year. Note that some datasets can contain several sources at the same time, for example, network communications and system logs. As final section thoughts, we notice that when it comes to developing a behavior evaluation solution, a key aspect is data availability, as the underlying solutions depend on it. Many works utilize self-collected private datasets to validate their approaches. However, to have a proper performance comparison, it is worth having public datasets allowing to cross-verify the proposed solutions. Furthermore, some teams do not have enough resources to collect enough data but have good processing and evaluation ideas. Therefore, having public datasets is essential to make diverse and well-performing behavior-based proposals possible.

VII. Lessons Learned, Trends and Challenges

Based on the different aspects of behavior fingerprinting analyzed through questions *Q1-Q4*, this section responds *Q5* (*How have application scenarios evolved for the last years?*). To this end, it summarizes the main lessons learned, trends, and open challenges extracted from the present study of device behavior fingerprinting.

A. Lessons Learned

After reviewing and analyzing the state-of-the-art, we were able to identify the following main lessons:

Network communications are the most exploited source. As Fig. 5 shows, it is utilized in 85% of works focused on device models or type identification, and in 54.28% of attack detection solutions. However, this source is less exploited in individual device identification (8.33% of the solutions) and malfunction detection (15.38%). This is because the data obtained from the network communication perspective is not sensitive enough as required for these scenarios, e.g., two devices of the same model deployed with the same purpose will have almost identical network communications.

Clustering is widely applied for inferring classes. As Table VI shows, in device type or model identification approaches, many solutions combine unlabeled data with clustering to group data samples and derive device classes, and then apply ML/DL classification approaches. Besides, some attack and malfunction detection techniques also rely on this approach (see Table VIII and Table IX). This fact shows the viability of clustering techniques for deriving classes from unlabeled behavioral data.

ML and DL are the favorite approaches for both classification and anomaly detection. Table VI, VII, VIII, and IX show that ML and DL are the main solutions applied for data processing, no matter if the objective is classification (of device types/models or attacks) or anomaly detection, either to detect attacks or faults. This fact shows the enormous flexibility and capabilities of these techniques inferring complex data patterns, outperforming traditional processing methods.

Individual device identification is one of the most complex application scenarios. Only some lower-level features, such as system clocks, code execution time, clock skew, or electromagnetic signals are sensitive enough to detect minimum physical differences that occurred during the device manufacturing processes. Thus, these are the ones required for individual identification. However, the monitoring of these sources is usually complex.

There is no consensus in misbehavior detection solutions. As Table VIII and IX show, attack and malfunction detection is addressed from heterogeneous perspectives. The selection of data sources and processing techniques depends on the type of anomalies that will be detected. Although network is the most used source, many solutions use system calls and logs, hardware events, or resource usage.

Public datasets are mainly focused on network, system calls, and logs. Fig. 6 shows that there are 35 datasets containing these sources (note that some datasets contain both sources at the same time, so they are counted both as network and calls/logs source). Moreover, Table X and XI show that in most cases the datasets contain raw data instead of processed information or features.

B. Current Trends

The main approaches expected in future works, based on the evolution of the proposals published in recent years, are:

ML and DL algorithms are gaining prominence. As Fig. 4 shows, ML and DL are the most usual techniques, with 46.39% of importance (note that many solutions utilize different techniques). In addition, DL-based techniques are gaining more importance, especially for time series processing, due to their performance handling raw data without pre-calculated features. Table VI, VII, VIII, and IX show that in both behavior fingerprinting scenarios (identification and misbehavior detection), ML and DL approaches are gaining importance in the last years. Overall, ML and DL algorithms are applied in the 71.87% of identification and in the 62.5% of misbehavior detection solutions.

Statistical and knowledge-based algorithms relegation. As Fig. 4 shows, processing and evaluation techniques based on statistical and knowledge-based algorithms are losing importance as evaluation approaches, in favor of ML and DL trend.

IoT and ML/DL convergence. In modern IoT scenarios where devices are massively deployed, behavior fingerprinting is critical management solution, grouping similar devices and detecting faults. ML and DL techniques are the best alternative when it comes to leverage the vast amount of data generated with the required performance and adaptability. This fact can be observed in the solution comparisons located in Table VI, VII, VIII, and IX.

Dataset publication. As it can be appreciated in Fig. 6 and in Table X and Table XI, a good number of datasets have been published for the last years. In the last five years (2016-2020), 23 public datasets were released, while in the previous five years (2011-2015) were only 7. This trend is influenced by the AI explosion, as ML and DL are powered by datasets.

Attack detection and model identification are the prominent application scenarios. Fig. 5 shows how attack detection and type or model identification solutions have been gaining prominence in the last years, increasing from 50% in 2017 to 85.71% in 2020. This trend is a direct consequence of the explosion in IoT deployments, as new requirements rise associated with the heterogeneous variety of devices and the new security issues generated by them.

C. Future Challenges

Based on the current state-of-the-art, the following points represent the main challenges that future behavior fingerprinting solutions might consider to enhance current solutions.

Usage of public datasets for behavior-based solution performance comparison. Many solutions are based on private datasets, which makes it difficult, if not impossible, to compare performance between different solutions. Among the solutions analyzed, only 45% of device model/type identification used public datasets. The same goes for the 16.66% about individual device identification, 42.85% tackling attacks, and 7.69% concerning malfunction detection, by using public datasets. Thus, a right direction for future approaches is to evaluate and compare their performance through public datasets.

Diverse and quality behavior dataset publication. Regarding device identification, the main publicly available datasets are focused on the network communications source. However, there is a lack of modern and variate datasets based on other sources. Then, it would be interesting for novel proposals addressing behavioral fingerprinting to publish the collected datasets, if any. Besides, datasets should have enough quality to ensure that research results are not influenced or damaged by low-quality data.

Solution scalability regarding the number of monitored devices and deployment architecture. Scalability is an issue that affects various aspects of behavior monitoring solutions. Many solutions covering individual device identification have detected that the number of devices is a challenge [61], [63], [74]. The more devices in the scenario, the worse classification results. Furthermore, centralized deployments may suffer if too many devices send behavioral data, or blockchain-based solutions may suffer block validation issues. Finally, during data evaluation, solutions based on statistical approaches that require one to one evaluation [94] may not scale at all when the number of devices increases.

Define anomaly countermeasures to apply when an attack or fault is detected. Many solutions solve the misbehavior detection problem, both when it is caused by a cyberattack or a system fault. However, most solutions do not propose any countermeasure [192] to mitigate the detected misbehavior. Only a few works propose some remedies for misbehavior, such as [7], [92].

Secure the behavior monitoring and analysis process against attacks. The fingerprinting solutions can suffer attacks or modifications performed by malicious entities. This fact can jeopardize the entire fingerprinting mechanism, and in the case of centralized processing solutions, even affect other device

behavior evaluation. However, few works took behavior monitoring security into account [90]. To solve this issue, additional security mechanisms, such as encryption, should be added to current solutions. Besides, there is an emerging area on adversarial attacks to ML/DL models that should also be considered in future solutions [193]. Finally, trust frameworks [194] can be included in behavior monitoring deployments to guarantee system safety.

Private device model and type to guarantee security. In some circumstances, like when there are known vulnerabilities, the model and type of devices should be private to avoid targeted attacks [195]. It has been demonstrated how privacy leakage attacks can be used to identify device model and type [196], [197] and further countermeasures, such as dummy traffic generation are required. In this context, there is a growing research area on device privacy enhancement [198] working in different solutions such as blocking traffic, concealing DNS, tunneling traffic, and shaping and injecting traffic.

User's privacy impact and awareness. Device behavior analysis can be leveraged to perform users' activity tracking and behavior monitoring [199]. The inference of users' activity has been demonstrated possible by behavior analysis [200] in health care and smart home IoT environments, even with encrypted traffic [201]. As an example, the TV channels watched by a given subject have been inferred in [202]. Therefore, manufacturers and service providers should include solutions to improve users privacy and defend them against activity inference attacks. These solutions are aligned with the ones commented in the previous challenge, as they can cover both user and device privacy at the same time.

Guarantee behavioral data and model privacy. As in user behavior, data and model privacy is a crucial aspect to consider when performing data analysis. From an ethical perspective, behavior analysis solutions should be employed to fingerprint devices in a non-intrusive way. However, privacy laws, such as GDPR [203] in Europe, are mainly focused on user perspective, leaving some device behavior fingerprinting methods out of their scope. To solve this problem, privacy-preserving solutions, such as federated learning [114] combined with differential privacy [204], allow training ML/DL models that ensure data privacy.

Apply novel ML/DL approaches for behavior processing and evaluation. As ML and DL are fast-evolving fields, some recent techniques have not been applied yet. For example, UMAP [205] for dimensionality reduction, or XGBoost [206] for classification, could improve current solution performance. Besides, DL architectures may combine convolutional and recurrent neuron layers for DL-based time series processing [207], [208]. Finally, any of the analyzed solutions addressed an approach based on *Reinforcement learning* [209], which has gained notable relevance in communications and networking areas [210], and human behavior analysis [211].

Consider ML/DL models behavior in the device analysis. Nowadays, devices usually include embedded ML and DL models that perform specific tasks with the data the device manipulates. However, the ML and DL models deployed on the devices have their own behavior [212], which influences the general device behavior. Then, understanding AI-powered applications and services is critical to identify the device behavior and its anomalies.

VIII. CONCLUSION

Device behavior fingerprinting has been determined in recent years as a promising solution to identify devices with different granularity levels, as well as to detect misbehavior originated by cyberattacks or faulty components. The article at hand studies the evolution of the device behavior research field, performing a comprehensive review of the devices, behavioral sources, datasets, and techniques used in both application scenarios. In this context, the present work has been performed with the goal of answering the following research questions.

Q1. Which scenarios, device types, and sources are present in behavior-based solutions? Section III reviews how these three aspects are used in the most recent and representative works of the literature. The performed analysis shows a relevant heterogeneity of device types and behavioral sources in the existing solutions, and highlights the usage of network communications in the majority of the solutions.

Q2. What and how behavior processing and evaluation tasks are used in each scenario? Section IV analyzes the main techniques and algorithms –rule-based, statistical, knowledge-based, ML and DL, and time-series approaches– used by works dealing with device and misbehavior identification. The analysis results show how ML and DL-based approaches are gaining importanc due to their versatility and excellent performance when enough training data is available, and to the detriment of statistical and knowledge-based solutions.

Q3. What characteristics do the most recent and representative solutions of each application scenario have? In the core section of this article, Section V, the reviewed solutions are described, analyzed, and compared according to their application scenario, device types, sources, techniques, and results. Regarding sources, this section shows that in device type or model identification solutions, network source is the dominant approach. In individual device identification, clock skew and electromagnetic signals are the main data sources. Attack detection is also mainly tackled using network communications. In contrast, for fault detection, the main approach is to utilize resource usage data. In terms of processing and evaluation techniques, ML and DL techniques are dominant in all the considered scenarios.

Q4. Which behavior datasets are available and which are their characteristics? In Section VI, the main public datasets containing device behavioral data are analyzed according to their application scenario. It also details the characteristics of the data they contain and how they were collected. This section shows the prominence of network source in the current public datasets, and the lack of other sources such as resource usage or hardware events.

Q5. How have application scenarios evolved for the last years? Lessons learned, current trends, and future challenges have been drawn in Section VII, which details how

network source and ML/DL algorithms are gaining prominence. Furthermore, it is also remarkable that novel ML/DL approaches, such as recurrent and convolutional neuron layer combination or Reinforcement learning, have not yet been applied in the area, which opens up pathways for future research. It also depicts how dataset publication is gaining importance during the last years; however, more relevant datasets are still required for sources and devices that are not covered in recent ones, e.g., resource usage or system logs in IoT devices or ICSs.

Aligned with the current trend and challenges drawn in this work, we will focus our next efforts on designing and implementing scalable behavior-based solutions to identify individual devices and detect cyberattacks affecting IoT devices. In both scenarios, we plan to utilize privacy-preserving ML and DL techniques, such as distributed and federated learning, to protect behavioral data while guaranteeing performance capabilities. Finally, we plan to build datasets for both scenarios, which will be publicly accessible to improve current dataset diversity and quality.

References

[1] K. Riad, T. Huang, and L. Ke, "A dynamic and hierarchical access control for IoT in multi-authority cloud storage," *J. Netw. Comput. Appl.*, vol. 160, Jun. 2020, Art. no. 102633.

[2] A. Fuentes, "Human niche, human behaviour, human nature," *Interface Focus*, vol. 7, no. 5, 2017, Art. no. 20160136.

[3] N. Shone, Q. Shi, M. Merabti, and K. Kifayat, "Misbehaviour monitoring on system-of-systems components," in *Proc. Int. Conf. Risks Security Internet Syst.*, La Rochelle, France, Oct. 2013, pp. 1–6.

[4] A. Sivanathan *et al.*, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1745–1759, Aug. 2019.

[5] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Detecting behavioral change of IoT devices using clustering-based network traffic modeling," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7295–7309, Aug. 2020.

[6] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "AuDI: Toward autonomous IoT device-type identification using periodic communication," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1402–1412, Jun. 2019.

[7] K. Haefner and I. Ray, "ComplexIoT: Behavior-based trust for IoT networks," in *Proc. 1st IEEE Int. Conf. Trust Privacy Security Intell. Syst. Appl.*, Los Angeles, CA, USA, Dec. 2019, pp. 56–65.

[8] G. Manco *et al.*, "Fault detection and explanation through big data analysis on sensor streams," *Expert Syst. Appl.*, vol. 87, pp. 141–156, Nov. 2017.

[9] R. Ferrando and P. Stacey, "Classification of device behaviour in Internet of Things infrastructures: Towards distinguishing the abnormal from security threats," in *Proc. 1st Int. Conf. Internet Things Mach. Learn.*, Oct. 2017, pp. 1–7.

[10] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on lot devices via SDN-based monitoring of mud activity," in *Proc. ACM Symp. SDN Res.*, Apr. 2019, pp. 36–48.

[11] R. Ravichandiran, H. Bannazadeh, and A. Leon-Garcia, "Anomaly detection using resource behaviour analysis for autoscaling systems," in *Proc. 4th IEEE Conf. Netw. Softw. Workshops*, Montreal, QC, Canada, Jun. 2018, pp. 192–196.

[12] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated device-type identification for security enforcement in IoT," in *Proc. 37th IEEE Int. Conf. Distrib. Comput. Syst.*, Atlanta, GA, USA, Jun. 2017, pp. 2511–2514.

[13] V. Selis and A. Marshall, "A classification-based algorithm to detect forged embedded machines in IoT environments," *IEEE Syst. J.*, vol. 13, no. 1, pp. 389–399, Mar. 2019.

[14] H. Jafari, O. Omotere, D. Adesina, H.-H. Wu, and L. Qian, "IoT devices fingerprinting using deep learning," in *Proc. IEEE Mil. Commun. Conf.*, Los Angeles, CA, USA, Oct. 2018, pp. 1–9.

[15] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," in *Proc. 39th IEEE Int. Conf. Distrib. Comput. Syst.*, Dallas, TX, USA, Jul. 2019, pp. 756–767.

[16] A. Samir and C. Pahl, "Detecting and localizing anomalies in container clusters using Markov models," *Electronics*, vol. 9, no. 1, p. 64, 2020.

[17] X. Liu, B. Xiao, S. Zhang, and K. Bu, "Unknown tag identification in large RFID systems: An efficient and complete solution," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1775–1788, Jun. 2015.

[18] J. Ortiz, C. Crawford, and F. Le, "DeviceMien: Network device behavior modeling for identifying unknown IoT devices," in *Proc. Int. Conf. Internet Things Design Implement.*, Apr. 2019, pp. 106–117.

[19] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, "Intrusion detection techniques in cloud environment: A survey," *J. Netw. Comput. Appl.*, vol. 77, pp. 18–47, Jan. 2017.

[20] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," in *Proc. 28th Int. Conf. Artif. Neural Netw.*, Sep. 2019, pp. 703–716.

[21] S. Barbhuiya, Z. Papazachos, P. Kilpatrick, and D. S. Nikolopoulos, "RADS: Real-time anomaly detection system for cloud data centres," 2018. [Online]. Available: arXiv:1811.04481.

[22] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 94–104, 1st Quart., 2016.

[23] G. Baldini and G. Steri, "A survey of techniques for the identification of mobile phones using the physical fingerprints of the built-in components," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1761–1789, 3rd Quart., 2017.

[24] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, "Host-based intrusion detection system with system calls: Review and future trends," *ACM Comput. Surveys*, vol. 51, no. 5, pp. 1–36, 2018.

[25] M. F. Elrawy, A. I. Awad, and H. F. A. Hamed, "Intrusion detection systems for IoT-based smart environments: A survey," *J. Cloud Comput.*, vol. 7, no. 1, p. 21, 2018.

[26] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019.

[27] J. Hou, Y. Li, J. Yu, and W. Shi, "A survey on digital forensics in Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 1–15, Jan. 2020.

[28] W. Tounsi and H. Rais, "A survey on technical threat intelligence in the age of sophisticated cyber attacks," *Comput. Security*, vol. 72, pp. 212–233, Jan. 2018.

[29] P. Oser, F. Kargl, and S. Lüders, "Identifying devices of the Internet of Things using machine learning on clock characteristics," in *Proc. 11th Int. Conf. Satellite Workshops Security Privacy Anonymity Comput. Commun. Storage*, Dec. 2018, pp. 417–427.

[30] S. Dong, F. Farha, S. Cui, J. Ma, and H. Ning, "CPG-FS: A CPU performance graph based device fingerprint scheme for devices identification and authentication," in *Proc. IEEE Int. Conf. Depend. Auton. Secure Comput. Int. Conf. Pervasive Intell. Comput. Int. Conf. Cloud Big Data Comput. Int. Conf. Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Fukuoka, Japan, Aug. 2019, pp. 266–270.

[31] S. Arabia and A. Alruban, "A systematic literature review of behavioural profiling for smartphone security: Challenges and open problems," *Int. J. Inf. Security Res.*, vol. 7, pp. 734–743, Jun. 2017.

[32] J. M. J. Valero *et al.*, "Improving the security and QoE in mobile devices through an intelligent and adaptive continuous authentication system," *Sensors*, vol. 18, no. 11, p. 3769, 2018.

[33] P. M. S. Sánchez, A. H. Celdrán, L. F. Maimó, and G. M. Pérez, "AuthCODE: A privacy-preserving and multi-device continuous authentication architecture based on machine and deep learning," 2020. [Online]. Available: arXiv:2004.07877.

[34] J. Ali, A. S. Khalid, E. Yafi, S. Musa, and W. Ahmed, "Towards a secure behavior modeling for IoT networks using blockchain," 2020. [Online]. Available: arXiv:2001.01841.

[35] M. B. Attia, C. Talhi, A. Hamou-Lhadj, B. Khosravifar, V. Turpaud, and M. Couture, "On-device anomaly detection for resource-limited systems," in *Proc. 30th Annu. ACM Symp. Appl. Comput.*, Apr. 2015, pp. 548–554.

[36] T. Wang, J. Xu, W. Zhang, Z. Gu, and H. Zhong, "Self-adaptive cloud monitoring with online anomaly detection," *Future Gener. Comput. Syst.*, vol. 80, pp. 89–101, Mar. 2018.

[37] J. P. Robinson and M. Kestnbaum, "The personal computer, culture, and other uses of free time," *Soc. Sci. Comput. Rev.*, vol. 17, no. 2, pp. 209–216, 1999.

[38] Q. Jabeen, F. Khan, M. N. Hayat, H. Khan, S. R. Jan, and F. Ullah, "A survey: Embedded systems supporting by different operating systems," 2016. [Online]. Available: arXiv:1610.07899.

[39] H. Holm, M. Karresand, A. Vidström, and E. Westring, "A survey of industrial control system testbeds," in *Proc. 20th Nordic Conf. Secure IT Syst.*, Oct. 2015, pp. 11–26.

[40] J. A. G. Gomez, "Survey of SCADA systems and visualization of a real life process," Dept. Elect. Eng., Linköping University, Linköping, Sweden, 2002.

[41] J. W. Rittinghouse and J. F. Ransome, *Cloud Computing: Implementation, Management, and Security*, Boca Raton, FL, USA: CRC Press, 2016.

[42] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K.-K. R. Choo, and M. Dlodlo, "From cloud to fog computing: A review and a conceptual live VM migration framework," *IEEE Access*, vol. 5, pp. 8284–8300, 2017.

[43] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[44] M. S. Bonfim, K. L. Dias, and S. F. L. Fernandes, "Integrated NFV/SDN architectures: A systematic literature review," *ACM Comput. Surveys*, vol. 51, no. 6, pp. 1–39, 2019.

[45] M.-K. Shin, K.-H. Nam, and H.-J. Kim, "Software-defined networking (SDN): A reference architecture and open APIs," in *Proc. Int. Conf. ICT Converg.*, Jeju, South Korea, Oct. 2012, pp. 360–361.

[46] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[47] C. Pahl and P. Jamshidi, "Microservices: A systematic mapping study," in *Proc. 6th Int. Conf. Cloud Comput. Serv. Sci.*, Apr. 2016, pp. 137–146.

[48] A. Vrenios, *Linux Cluster Architecture*. Indianapolis, IN, USA: SAMS, 2002.

[49] F. Schmidt, F. Suri-Payer, A. Gulenko, M. Wallschläger, A. Acker, and O. Kao, "Unsupervised anomaly event detection for cloud monitoring using online Arima," in *Proc. IEEE/ACM Int. Conf. Utility Cloud Comput. Companion*, Zürich, Switzerland, Dec. 2018, pp. 71–76.

[50] T. J. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.-R. Sadeghi, "HomeSnitch: Behavior transparency and control for smart home IoT devices," in *Proc. 12th Conf. Security Privacy Wireless Mobile Netw.*, May 2019, pp. 128–138.

[51] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy, "DEFT: A distributed IoT fingerprinting technique," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 940–952, Feb. 2019.

[52] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "IoTSense: Behavioral fingerprinting of IoT devices," 2018. [Online]. Available: arXiv:1804.03852.

[53] N. Msadek, R. Soua, and T. Engel, "IoT device fingerprinting: Machine learning based encrypted traffic analysis," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Marrakesh, Morocco, 2019, pp. 1–8.

[54] R. Dai, C. Gao, B. Lang, L. Yang, H. Liu, and S. Chen, "SSL malicious traffic detection based on multi-view features," in *Proc. 9th Int. Conf. Commun. Netw. Security*, 2019, pp. 40–46.

[55] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, "IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 45–59, Mar. 2020.

[56] A. Blaise, M. Bouet, V. Conan, and S. Secci, "BotFP: Fingerprints clustering for bot detection," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp.*, Budapest, Hungary, Apr. 2020, pp. 1–7.

[57] G. Spanos, K. M. Giannoutakis, K. Votis, and D. Tzovaras, "Combining statistical and machine learning techniques in IoT anomaly detection for smart homes," in *Proc. 24th IEEE Int. Workshop Comput.-Aided Model. Design Commun. Links Netw.*, pp. 1–6, Limassol, Cyprus, Sep. 2019, pp. 1–6.

[58] L. Polčák and B. Franková, "On reliability of clock-skew-based remote computer identification," in *Proc. 11th Int. Conf. Security Cryptogr.*, Vienna, Austria, Aug. 2014, pp. 1–8.

[59] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Depend. Secure Comput.*, vol. 2, no. 2, pp. 93–108, Apr.–Jun. 2005.

[60] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," *IEEE Trans. Mobile Comput.*, vol. 9, no. 3, pp. 449–462, Mar. 2010.

[61] F. Lanze, A. Panchenko, B. Braatz, and A. Zinnen, "Clock skew based remote device fingerprinting demystified," in *Proc. IEEE Global Commun. Conf.*, Anaheim, CA, USA, Dec. 2012, pp. 813–819.

[62] S. Sharma, A. Hussain, and H. Saran, "Experience with heterogenous clock-skew based device fingerprinting," in *Proc. Workshop Learn. Authoritative Security Exp. Results*, Jul. 2012, pp. 9–18.

[63] L. Polcák and B. Franková, "Clock-skew-based computer identification: Traps and pitfalls," *J. Univ. Comput. Sci.*, vol. 21, no. 9, pp. 1210–1233, 2015.

[64] M. Ezuma, F. Erden, C. K. Anjinappa, O. Ozdemir, and I. Guvenc, "Detection and classification of UAVs using RF fingerprints in the presence of Wi-Fi and bluetooth interference," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 60–76, 2019.

[65] M. F. Al-Sa'd, A. Al-Ali, A. Mohamed, T. Khattab, and A. Erbad, "RF-based drone detection and identification using deep learning approaches: An initiative towards a large open source drone database," *Future Gener. Comput. Syst.*, vol. 100, pp. 86–97, Nov. 2019.

[66] M. S. Allahham, T. Khattab, and A. Mohamed, "Deep learning for RF-based drone detection and identification: A multi-channel 1-D convolutional neural networks approach," in *Proc. IEEE Int. Conf. Informat. IoT Enabling Technol. (ICIoT)*, Doha, Qatar, 2020, pp. 112–117.

[67] S. Basak, S. Rajendran, S. Pollin, and B. Scheers, "Drone classification from RF fingerprints using deep residual nets," in *Proc. Int. Conf. COMmun. Syst. NETw. (COMSNETS)*, Bangalore, India, 2020, pp. 548–555.

[68] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 146–152, Sep. 2018.

[69] S. Rajendran, W. Meert, V. Lenders, and S. Pollin, "Unsupervised wireless spectrum anomaly detection with interpretable features," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 3, pp. 637–647, Sep. 2019.

[70] Y. Cheng, X. Ji, J. Zhang, W. Xu, and Y.-C. Chen, "DeMiCPU: Device fingerprinting with magnetic signals radiated by CPU," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, Nov. 2019, pp. 1149–1170.

[71] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. A. Beyah, "Who's in control of your control system? device fingerprinting for cyber-physical systems," in *Proc. Netw. Distrib. Syst. Security Symp.*, Feb. 2016, pp. 1–15.

[72] A. Amouri, V. T. Alaparthy, and S. D. Morgera, "Cross layer-based intrusion detection based on network behavior for IoT," in *Proc. 19th IEEE Wireless Microw. Technol. Conf.*, Sand Key, FL, USA, Apr. 2018, pp. 1–4.

[73] F. S. D. L. Filho, F. A. F. Silveira, A. D. M. Brito Jr., G. Vargas-Solar, and L. F. Silveira, "Smart detection: An online approach for DoS/DDoS attack detection using machine learning," *Security Commun. Netw.*, vol. 2019, Oct. 2019, Art. no. 1574749.

[74] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, "GTID: A technique for physical device and device type fingerprinting," *IEEE Trans. Depend. Secure Comput.*, vol. 12, no. 5, pp. 519–532, Sep./Oct. 2014.

[75] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "IoT devices recognition through network traffic analysis," in *Proc. IEEE Int. Conf. Big Data*, Seattle, WA, USA, Dec. 2018, pp. 5187–5192.

[76] J. Kotak and Y. Elovici, "IoT device identification using deep learning," 2020. [Online]. Available: arXiv:2002.11686.

[77] S. A. Hamad, W. E. Zhang, Q. Z. Sheng, and S. Nepal, "IoT device identification via network-flow based fingerprinting and learning," in *Proc. 18th IEEE Int. Conf. Trust Security Privacy Comput. Commun 13th IEEE Int. Conf. Big Data Sci. Eng.*, Rotorua, New Zealand, Aug. 2019, pp. 103–111.

[78] Y. Meidan *et al.*, "Detection of unauthorized IoT devices using machine learning techniques," 2017. [Online]. Available: arXiv:1709.04647.

[79] L. F. Carvalho, T. Abrão, L. D. S. Mendes, and M. L. Proença Jr., "An ecosystem for anomaly detection and mitigation in software-defined networking," *Expert Syst. Appl.*, vol. 104, pp. 121–133, Aug. 2018.

[80] Y. Afek *et al.*, "NFV-based IoT security for home networks using MUD," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp.*, Budapest, Hungary, Apr. 2020, pp. 1–9.

[81] B. J. Radford, B. D. Richardson, and S. E. Davis, "Sequence aggregation rules for anomaly detection in computer network traffic," 2018. [Online]. Available: arXiv:1805.03735.

[82] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "An enhancing framework for botnet detection using generative adversarial networks," in *Proc. Int. Conf. Artif. Intell. Big Data*, Chengdu, China, May 2018, pp. 228–234.

[83] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, "Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using spark," *IEEE Access*, vol. 6, pp. 59657–59671, 2018.

[84] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "Packet-level signatures for smart home devices," Signature, vol. 10, no. 13, p. 54, 2020. San Diego, CA, USA, Feb. 2020.

[85] R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis, "IoTFinder: Efficient large-scale identification of IoT devices via passive DNS traffic analysis," in *Proc. IEEE Eur. Symp. Security Privacy (EuroS&P)*, Genoa, Italy, 2020, pp. 474–489.

[86] J. Choi *et al.*, "Detecting and identifying faulty IoT devices in smart home with context extraction," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, Luxembourg City, Luxembourg, Jun. 2018, pp. 610–621.

[87] T. Yu, Y. Sun, S. Nanda, V. Sekar, and S. Seshan, "RADAR: A robust behavioral anomaly detection for IoT devices in enterprise networks," Carnegie Mellon University CyLab., Carnegie Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-CyLab-19-003, 2019.

[88] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. M. A. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," *Internet Things*, vol. 7, Sep. 2019, Art. no. 100059.

[89] J. Pacheco and S. Hariri, "Anomaly behavior analysis for IoT sensors," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 4, p. e3188, 2018.

[90] X. Wang, C. Konstantinou, M. Maniatakos, and R. Karri, "ConFirm: Detecting firmware modifications in embedded systems using hardware performance counters," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2015, pp. 544–551.

[91] K. Ott and R. Mahapatra, "Continuous authentication of embedded software," in *Proc. 18th IEEE Int. Conf. Trust Security Privacy Comput. Commun. 13th IEEE Int. Conf. Big Data Sci. Eng.*, Rotorua, New Zealand, Aug. 2019, pp. 128–135.

[92] T. Golomb, Y. Mirsky, and Y. Elovici, "CIoTA: Collaborative IoT anomaly detection via blockchain," 2018. [Online]. Available: arXiv:1803.03807.

[93] T. J. Salo, "Multi-factor fingerprints for personal computer hardware," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Orlando, FL, USA, Oct. 2007, pp. 1–7.

[94] I. Sanchez-Rola, I. Santos, and D. Balzarotti, "Clock around the clock: Time-based device fingerprinting," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, Jan. 2018, pp. 1502–1514.

[95] A. Gulenko, M. Wallschläger, F. Schmidt, O. Kao, and F. Liu, "A system architecture for real-time anomaly detection in large-scale NFV systems," *Procedia Comput. Sci.*, vol. 94, pp. 491–496, Jan. 2016.

[96] G. Creech and J. Hu, "A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns," *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 807–819, Apr. 2014.

[97] P. Deshpande, S. C. Sharma, S. K. Peddoju, and S. Junaid, "HIDS: A host based intrusion detection system for cloud computing environment," *Int. J. Syst. Assurance Eng. Manag.*, vol. 9, no. 3, pp. 567–576, 2018.

[98] P. Mishra, V. Varadharajan, E. S. Pilli, and U. Tupakula, "VMGuard: A VMI-based security architecture for intrusion detection in cloud environment," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 957–971, Jul.–Sep. 2020.

[99] Z. Liu, N. Japkowicz, R. Wang, Y. Cai, D. Tang, and X. Cai, "A statistical pattern based feature extraction method on system call traces for anomaly detection," *Inf. Softw. Technol.*, vol. 26, Oct. 2020, Art. no. 106348.

[100] S. Nedelkoski, J. Cardoso, and O. Kao, "Anomaly detection from system tracing data using multimodal deep learning," in *Proc. 12th IEEE Int. Conf. Cloud Comput.*, Milan, Italy, Jul. 2019, pp. 179–186.

[101] M. Kubacki and J. Sosnowski, "Exploring operational profiles and anomalies in computer performance logs," *Microprocess. Microsyst.*, vol. 69, pp. 1–15, Sep. 2019.

[102] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling," *J. Netw. Comput. Appl.*, vol. 87, pp. 185–192, Jun. 2017.

[103] S. He, W. Ren, T. Zhu, and K.-K. R. Choo, "BoSMoS: A blockchain-based status monitoring system for defending against unauthorized software updating in industrial Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 948–959, Feb. 2020.

[104] S. Zhanwei and L. Zenghui, "Abnormal detection method of industrial control system based on behavior model," *Comput. Security*, vol. 84, pp. 166–178, Jul. 2019.

[105] N. Neha, S. Priyanga, S. Seshan, R. Senthilnathan, and V. S. S. Sriram, "SCO-RNN: A behavioral-based intrusion detection approach for cyber physical attacks in SCADA systems," in *Inventive Communication and Computational Technologies*. Singapore: Springer, 2020, pp. 911–919.

[106] C. M. Ahmed and A. P. Mathur, "Hardware identification via sensor fingerprinting in a cyber physical system," in *Proc. IEEE Int. Conf. Softw. Qual. Rel. Security Companion*, Prague, Czech Republic, Jul. 2017, pp. 517–524.

[107] B. Agrawal, T. Wiktorski, and C. Rong, "Adaptive real-time anomaly detection in cloud infrastructures," *Concurrency Comput. Pract. Exp.*, vol. 29, no. 24, p. e4193, 2017.

[108] A. Gulenko, F. Schmidt, A. Acker, M. Wallschläger, O. Kao, and F. Liu, "Detecting anomalous behavior of black-box services modeled with distance-based online clustering," in *Proc. 11th IEEE Int. Conf. Cloud Comput.*, San Francisco, CA, USA, Jul. 2018, pp. 912–915.

[109] N. Sorkunlu, V. Chandola, and A. Patra, "Tracking system behavior from resource usage data," in *Proc. IEEE Int. Conf. Clust. Comput.*, Honolulu, HI, USA, Sep. 2017, pp. 410–418.

[110] Q. Du, T. Xie, and Y. He, "Anomaly detection and diagnosis for container-based microservices with performance monitoring," in *Proc. 18th Int. Conf. Algorithms Archit. Parallel Process.*, Nov. 2018, pp. 560–572.

[111] X. D. Hoang, J. Hu, and P. Bertok, "A program-based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference," *J. Netw. Comput. Appl.*, vol. 32, no. 6, pp. 1219–1228, 2009.

[112] Z. Zheng, S. Xie, H. N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, pp. 352–375, 2018.

[113] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," 2019. [Online]. Available: arXiv:1912.09789.

[114] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[115] E. Lear, R. Droms, and D. Romascanu, "Manufacturer usage description specification," IETF, RFC 8520, 2019.

[116] R. Alcarria, B. Bordel, D. Martín, and D. S. De Rivera, "Rule-based monitoring and coordination of resource consumption in smart communities," *IEEE Trans. Consum. Electron.*, vol. 63, no. 2, pp. 191–199, May 2017.

[117] P. A. Gagniuc, *Markov Chains : From Theory to Implementation and Experimentation*. Hoboken, NJ, USA: Wiley, 2017.

[118] C. Wressnegger, G. Schwenk, D. Arp, and K. Rieck, "A close look on n-grams in intrusion detection: Anomaly detection vs. classification," in *Proc. ACM Workshop Artif. Intell. Security*, Nov. 2013, pp. 67–76.

[119] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, MA, USA: MIT Press, 2020.

[120] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[121] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 3, pp. 660–674, May/Jun. 1991.

[122] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[123] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic Regression*. New York, NY, USA: Springer, 2002.

[124] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Mach. Learn.*, vol. 29, nos. 2–3, pp. 131–163, 1997.

[125] I. Steinwart and A. Christmann, *Support Vector Machines*. New York, NY, USA: Springer, 2008.

[126] S. Weisberg, *Applied Linear Regression*, vol. 528. Hoboken, NJ, USA: Wiley, 2005.

[127] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

[128] L. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[129] K. Krishna and M. N. Murty, "Genetic K-means algorithm," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 3, pp. 433–439, Jun. 1999.

[130] T. N. Tran, K. Drab, and M. Daszykowski, "Revised DBSCAN algorithm to cluster data with dense adjacent clusters," *Chemometrics Intell. Lab. Syst.*, vol. 120, pp. 92–96, Jan. 2013.

[131] K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu, "Improving one-class SVM for anomaly detection," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Xi'an, China, Nov. 2003, pp. 3077–3081.

[132] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Min.*, Pisa, Italy, Dec. 2008, pp. 413–422.

[133] Á. L. P. Gómez *et al.*, "On the generation of anomaly detection datasets in industrial control systems," *IEEE Access*, vol. 7, pp. 177460–177473, 2019.

[134] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," 2016. [Online]. Available: arXiv:1611.02163.

[135] R. N. Bracewell and R. N. Bracewell, *The Fourier Transform and Its Applications*, vol. 31999. New York, NY, USA: McGraw-Hill, 1986.

[136] A. Sivanathan *et al.*, "Characterizing and classifying IoT traffic in smart cities and campuses," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Atlanta, GA, USA, May 2017, pp. 559–564.

[137] J. François, H. Abdelnur, R. State, and O. Festor, "Automated behavioral fingerprinting," in *Proc. 12th Int. Symp. Recent Adv. Intrusion Detection*, Sep. 2009, pp. 182–201.

[138] J. François, H. Abdelnur, R. State, and O. Festor, "Machine learning techniques for passive network inventory," *IEEE Trans. Netw. Service Manag.*, vol. 7, no. 4, pp. 244–257, Dec. 2010.

[139] J. Terrell, K. Jeffay, F. D. Smith, J. Gogan, and J. Keller, "Passive, streaming inference of TCP connection structure for network server management," in *Proc. 1st Int. Workshop Traffic Monitor. Anal.*, May 2009, pp. 42–53.

[140] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "SoK: Security evaluation of home-based IoT deployments," in *Proc. IEEE Symp. Security Privacy*, San Francisco, CA, USA, May 2019, pp. 1362–1380.

[141] J. Guth *et al.*, "A detailed analysis of IoT platform architectures: Concepts, similarities, and differences," in *Internet of Everything*. Singapore: Springer, 2018, pp. 81–101.

[142] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial Internet of Things (IIoT): An analysis framework," *Comput. Ind.*, vol. 101, pp. 1–12, Oct. 2018.

[143] D. Kumar *et al.*, "All things considered: An analysis of IoT devices on home networks," in *Proc. 28th USENIX Security Symp. (USENIX Security)*, 2019, pp. 1169–1185.

[144] D. Y. Huang, N. Apthorpe, F. Li, F. Acar, and N. Feamster, "IoT inspector: Crowdsourcing labeled network traffic from smart home devices at scale," *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, vol. 4, no. 2, pp. 1–21, 2020.

[145] A. MacDermott, T. Baker, and Q. Shi, "IoT forensics: Challenges for the ioa era," in *Proc. IEEE 9th IFIP Int. Conf. New Technol. Mobility Security (NTMS)*, Paris, France, 2018, pp. 1–5.

[146] I. Yaqoob, I. A. T. Hashem, A. Ahmed, S. M. A. Kazmi, and C. S. Hong, "Internet of Things forensics: Recent advances, taxonomy, requirements, and open challenges," *Future Gener. Comput. Syst.*, vol. 92, pp. 265–275, Mar. 2019.

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

[148] S. Li, K.-K. R. Choo, Q. Sun, W. J. Buchanan, and J. Cao, "IoT forensics: Amazon echo as a use case," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6487–6497, Aug. 2019.

[149] J. H. Ryu, P. K. Sharma, J. H. Jo, and J. H. Park, "A blockchain-based decentralized efficient investigation framework for IoT digital forensics," *J. Supercomput.*, vol. 75, no. 8, pp. 4372–4387, 2019.

[150] C. Ruiz *et al.*, "IDrone: Robust drone identification through motion actuation feedback," *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, vol. 2, no. 2, pp. 1–22, 2018.

[151] A. Coluccia, G. Parisi, and A. Fascista, "Detection and classification of multirotor drones in radar sensor networks: A review," *Sensors*, vol. 20, no. 15, p. 4172, 2020.

[152] A. Bernardini, F. Mangiatordi, E. Pallotti, and L. Capodiferro, "Drone detection by acoustic signature identification," *Electron. Imag.*, vol. 2017, no. 10, pp. 60–64, 2017.

[153] M. H. D. S. Allahham, M. F. Al-Sa'd, A. Al-Ali, A. Mohamed, T. Khattab, and A. Erbad, "DroneRF dataset: A dataset of drones for RF-based detection, classification and identification," *Data Brief*, vol. 26, Oct. 2019, Art. no. 104313.

[154] F. Lorenz *et al.*, "Fingerprinting analog IoT sensors for secret-free authentication," 2020. [Online]. Available: arXiv:2006.06296.

[155] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, J. Zahorjan, and E. Lazowska. (2006). *CRAWDAD Dataset UW/SIGCOMM2004 (V. 2006-10-17)*. Accessed: Jul. 31, 2020. [Online]. Available: https://crawdad.org/uw/sigcomm2004/20061017

[156] A. S. Uluagac. (2014). *CRAWDAD Dataset Gatech/Fingerprinting (v. 2014-06-09)*. Accessed: Jul. 31, 2020. [Online]. Available: https://crawdad.org/gatech/fingerprinting/20140609

[157] A. S. Uluagac, S. V. Radhakrishnan, C. Corbett, A. Baca, and R. Beyah, "A passive technique for fingerprinting wireless devices with wired-side observations," in *Proc. IEEE Conf. Commun. Netw. Security*, National Harbor, MD, USA, Oct. 2013, pp. 305–313.

[158] L. F. Maimó, A. H. Celdrán, Á. L. P. Gómez, F. J. G. Clemente, J. Weimer, and I. Lee, "Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments," *Sensors*, vol. 19, no. 5, p. 1114, 2019.

[159] M.-O. Pahl and F.-X. Aubet, "All eyes on you: Distributed multi-dimensional IoT microservice anomaly detection," in *Proc. 14th Int. Conf. Netw. Serv. Manag.*, Rome, Italy, Nov. 2018, pp. 72–80.

[160] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Security Privacy*, Jan. 2018, pp. 108–116.

[161] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Proc. IEEE Conf. Commun. Netw. Security*, San Francisco, CA, USA, Oct. 2014, pp. 247–255.

[162] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Security*, vol. 45, pp. 100–123, Sep. 2014.

[163] A. Hamza, D. Ranathunga, H. H. Gharakheili, M. Roughan, and V. Sivaraman, "Clear as MUD: Generating, validating and applying IoT behavioral profiles," in *Proc. Workshop IoT Security Privacy*, Aug. 2018, pp. 8–14.

[164] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *Proc. IEEE Int. Conf. Trust Security Privacy Comput. Commun.*, vol. 1. Helsinki, Finland, Aug. 2015, pp. 57–64.

[165] Y. Meidan *et al.*, "N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul.–Sep. 2018.

[166] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the JAM project," in *Proc. DARPA Inf. Survivability Conf. Exposit.*, vol. 2. Hilton Head, SC, USA, Jan. 2000, pp. 130–144.

[167] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, Ottawa, ON, Canada, Jul. 2009, pp. 1–6.

[168] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf.*, Canberra, ACT, Australia, Nov. 2015, pp. 1–6.

[169] A. P. Mathur and N. O. Tippenhauer, "SWaT: A water treatment testbed for research and training on ICS security," in *Proc. Int. Workshop Cyber Phys. Syst. Smart Water Netw.*, Vienna, Austria, Apr. 2016, pp. 31–36.

[170] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *Proc. 11th Int. Conf. Critical Inf. Infrastruct. Security*, Oct. 2016, pp. 88–99.

[171] S. Shen, V. V. Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *Proc. 15th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput.*, Shenzhen, China, May 2015, pp. 465–474.

[172] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Proc. IEEE Symp. Security Privacy*, Oakland, CA, USA, May 1999, pp. 133–145.

[173] G. Creech, "Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks," Ph.D. dissertation, Dept. Eng. Inf. Technol., Univ. New South Wales, Canberra, ACT, Australia, 2014.

[174] C. Liu, S. C. H. Hoi, P. Zhao, and J. Sun, "Online ARIMA algorithms for time series prediction," in *Proc. 30th AAAI Conf. Artif. Intell.*, Feb. 2016, pp. 1867–1873.

[175] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM Sigmod Rec.*, vol. 25, no. 2, pp. 103–114, 1996.

[176] *WSU CASAS Datasets*, Center Adv. Stud. Adapt. Syst., Pullman, WA, USA, 2020. Accessed: Jul. 31, 2020. [Online]. Available: http://casas.wsu.edu/datasets/

[177] T. V. Kasteren. (2010). *Datasets for Activity Recognition*. Accessed: Jul. 31, 2020. [Online]. Available: https://sites.google.com/site/tim0306/datasets/

[178] A. K. Hagelskjær, B. H. Grevenkop-Castenskiold, M. H. Jespersen, T. Arildsen, E. Carvalho, and P. Popovski. (2020). *IoT Device Identification Dataset*. Accessed: Jul. 31, 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3638165

[179] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim. (2019). *IoT Network Intrusion Dataset*. Accessed: Jul. 31, 2020. [Online]. Available: https://doi.org/10.21227/q70p-q449

[180] A. Parmisano, S. Garcia, and M. J. Erquiaga. (2020). *Aposemat IoT-23: A Labeled Dataset with Malicious and Benign IoT Network Traffic*. Accessed: Jul. 31, 2020. [Online]. Available: https://www.stratosphereips.org/datasets-iot23

[181] R. Damasevicius *et al.*, "LITNET-2020: An annotated real-world network flow dataset for network intrusion detection," *Electronics*, vol. 9, no. 5, p. 800, 2020.

[182] *ECML-PKDD Discovery Challenge*, Eur. Conf. Mach. Learn. Knowl. Discov., Warsaw, Poland, 2007. Accessed: Jul. 31, 2020. [Online]. Available: http://www.lirmm.fr/pkdd2007-challenge/

[183] C. T. Giménez, A. P. Villegas, and G. Á. Marañón. *HTTP Dataset CSIC 2010*. Accessed: Jul. 31, 2020. [Online]. Available: https://www.isi.csic.es/dataset

[184] E. Aghaei and G. Serpen, "Host-based anomaly detection using Eigentraces feature extraction and one-class classification on system call trace data," 2019. [Online]. Available: arXiv:1911.11284.

[185] B. S. Khater, A. W. B. Wahab, M. Y. I. B. Idris, M. A. Hussain, and A. A. Ibrahim, "A lightweight perceptron-based intrusion detection system for fog computing," *Appl. Sci.*, vol. 9, no. 1, p. 178, 2019.

[186] S. S. Murtaza, W. Khreich, A. Hamou-Lhadj, and M. Couture, "A host-based anomaly detection approach by representing system calls as states of kernel modules," in *Proc. 24th IEEE Int. Symp. Softw. Rel. Eng.*, Pasadena, CA, USA, Nov. 2013, pp. 431–440.

[187] P. M. S. Sánchez *et al.*, "BEHACOM—A dataset modelling users' behaviour in computers," *Data Brief*, vol. 31, Aug. 2020, Art. no. 105767.

[188] A. H. Lashkari, A. F. A. Kadir, H. Gonzalez, K. F. Mbah, and A. A. Ghorbani, "Towards a network-based framework for Android malware detection and characterization," in *Proc. 15th Annu. Conf. Privacy Security Trust*, Calgary, AB, Canada, Aug. 2017, pp. 233–23309.

[189] L. Taheri, A. F. A. Kadir, and A. H. Lashkari, "Extensible Android malware detection and family classification using network-flows and API-calls," in *Proc. Int. Carnahan Conf. Security Technol.*, Chennai, India, Oct. 2019, pp. 1–8.

[190] *DARPA Intrusion Detection Evaluation Dataset*, MIT Lincoln Lab., Lexington, MA, USA, 1998. Accessed: Jul. 31, 2020. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset

[191] *DARPA Intrusion Detection Evaluation Dataset*. MIT Lincoln Lab., Lexington, MA, USA, 1999. Accessed: Jul. 31, 2020. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset

[192] P. Nespoli, D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1361–1396, 2nd Quart., 2018.

[193] X. Wang, J. Li, X. Kuang, Y.-A. Tan, and J. Li, "The security of machine learning in an adversarial setting: A survey," *J. Parallel Distrib. Comput.*, vol. 130, pp. 12–23, Aug. 2019.

[194] D. D. S. Braga, M. Niemann, B. Hellingrath, and F. B. L. Neto, "Survey on computational trust and reputation models," *ACM Comput. Surveys*, vol. 51, no. 5, pp. 1–40, 2018.

[195] D. Bastos, M. Shackleton, and F. El-Moussa, "Internet of Things: A survey of technologies and security risks in smart home and city environments," in *Proc. Living Internet Things Cybersecurity IoT* , 2018, p. 7.

[196] A. Acar *et al.*, "Peek-a-boo: I see your smart home activities, even encrypted!" in *Proc. 13th ACM Conf. Security Privacy Wireless Mobile Netw.*, 2020, pp. 207–218.

[197] J. Bugeja, A. Jacobsson, and P. Davidsson, "On privacy and security challenges in smart connected homes," in *Proc. Eur. Intell. Security Informat. Conf. (EISIC)*, Uppsala, Sweden, 2016, pp. 172–175.

[198] N. Apthorpe, D. Reisman, and N. Feamster, "Closing the blinds: Four strategies for protecting smart home privacy from network observers," 2017. [Online]. Available: arXiv:1705.06809.

[199] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information exposure from consumer IoT devices: A multidimensional, network-informed measurement approach," in *Proc. Internet Meas. Conf.*, 2019, pp. 267–279.

[200] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, "Keeping the smart home private with smart (er) IoT traffic shaping," *Proc. Privacy Enhancing Technol.*, vol. 2019, no. 3, pp. 128–148, 2019.

[201] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster, "Spying on the smart home: Privacy attacks and defenses on encrypted IoT traffic," 2017. [Online]. Available: arXiv:1708.05044.

[202] Y. Xu, J.-M. Frahm, and F. Monrose, "Watching the watchers: Automatically inferring TV content from outdoor light effusions," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2014, pp. 418–428.

[203] A. H. Celdrán *et al.*, "PROTECTOR: Towards the protection of sensitive data in Europe and the US," *Comput. Netw.*, vol. 181, Nov. 2020, Art. no. 107448.

[204] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.

[205] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2018. [Online]. Available: arXiv:1802.03426.

[206] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, Aug. 2016, pp. 785–794.

[207] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jun. 2018, pp. 95–104.

[208] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2017.

[209] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Feb. 2018, pp. 3207–3214.

[210] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[211] P. L. Lockwood and M. Klein-Flügge, "Computational modelling of social cognition and behaviour-a reinforcement learning primer," *Soc. Cogn. Affect. Neurosci.*, to be published.

[212] I. Rahwan *et al.*, "Machine behaviour," *Nature*, vol. 568, no. 7753, pp. 477–486, 2019.

2

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

Title:	A methodology to identify identical single-board computers based on hardware behavior fingerprinting.
Authors:	Pedro Miguel Sánchez Sánchez, José María Jorquera Valero, Alberto Huertas Celdrán, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez
Journal:	Journal of Network and Computer Applications
JIF:	8.7 D1 (2022)
Publisher:	Elsevier
Volume:	212
Number:	
Pages:	103579
Year:	2023
Month:	Mar
DOI:	10.1016/j.jnca.2022.103579
Status:	Published

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.

Keywords

Device behavior fingerprinting · Device identification · Cyberattack detection · Behavioral data · Hardware fingerprinting

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*

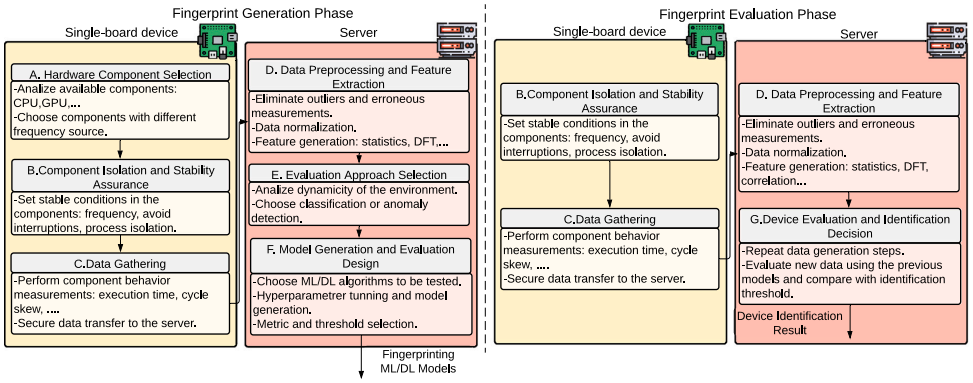


Fig. 1. Graphical representation of the proposed methodology for device fingerprinting and identification.

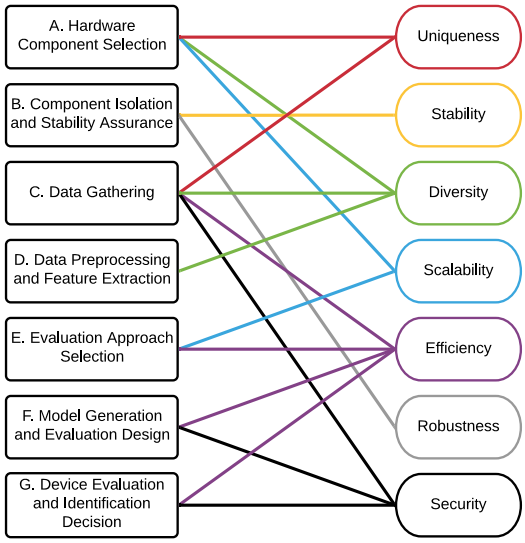


Fig. 2. Association between methodology steps and fingerprint properties.

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).

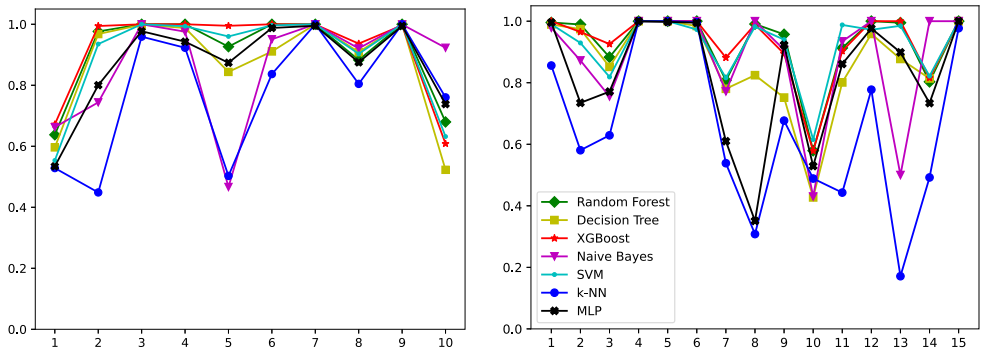
4.7. Device evaluation and identification decision

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%,

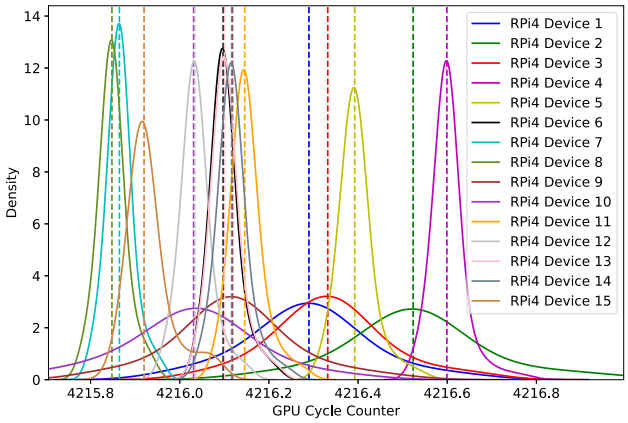


Fig. 4. Density plot for the GPU cycle counter in RPi4.

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)	–

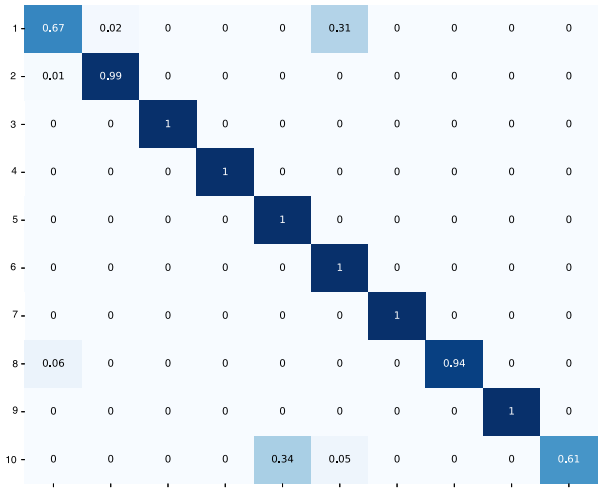
until they are rebooted. Finally, two different approaches were tested inspired by Sanchez-Rola et al. (2018). Both share the fact that the data collected was based on short functions executed in the CPU without considering stability measurements. They differ in the evaluation approach, one using anomaly 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.

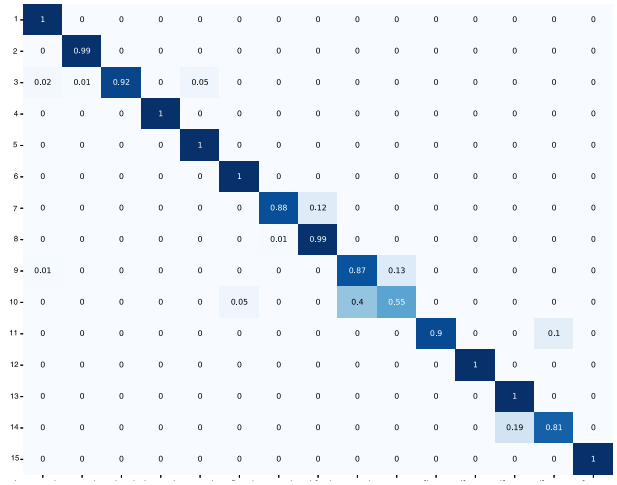
6.2. Lessons learned and limitations

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.

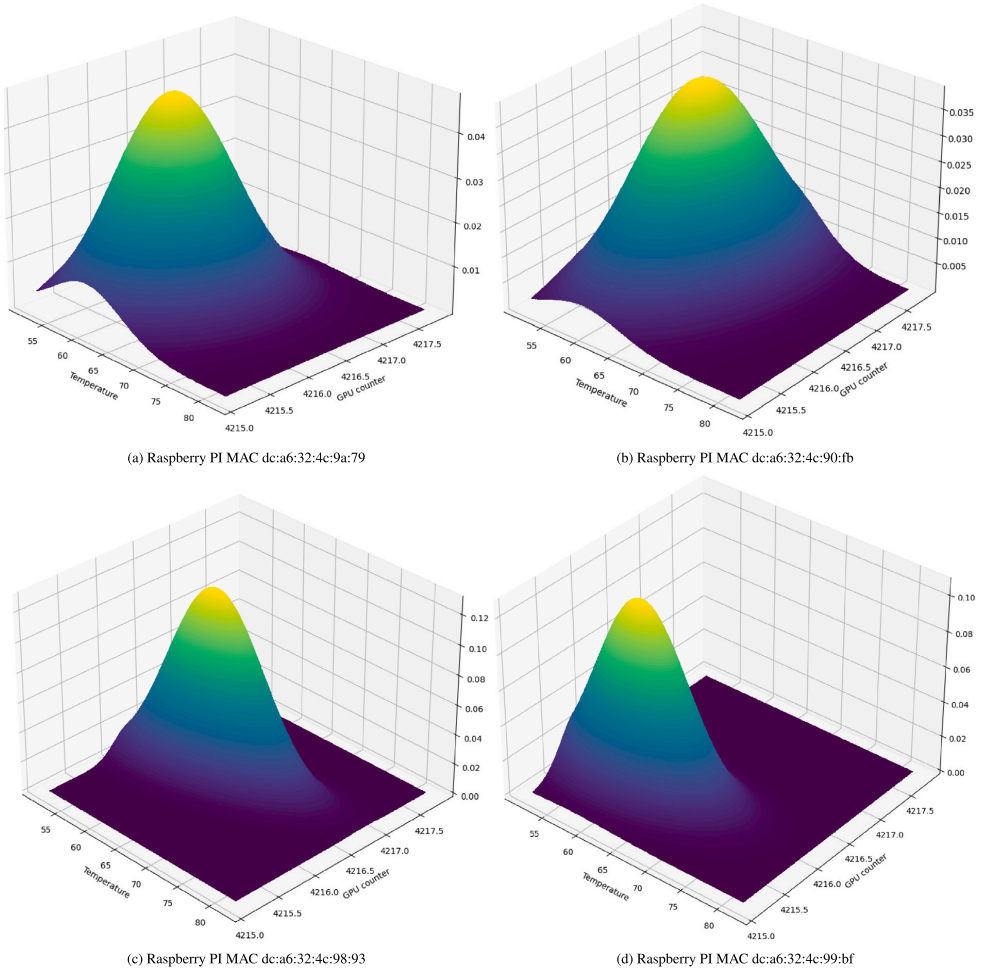
ensure the *stability* of the fingerprinting process. In this sense, in cases where the conditions of the components were not stable, it was not possible to reliably identify them after device rebooting.

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.

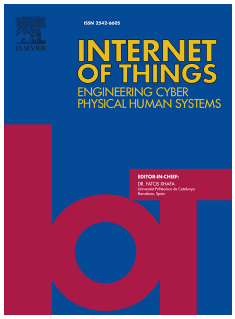
Gérôme Bovet is the head of data science for the Swiss DoD, where he leads a research team and a portfolio of about 30 projects. His work focuses on machine/deep learning approaches applied to cyber-defence use cases, with emphasis on anomaly detection, adversarial and collaborative learning. He received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

3

LwHBench: A low-level hardware component benchmark and dataset for Single Board Computers



Title:	LwHBench: A low-level hardware component benchmark and dataset for Single Board Computers
Authors:	Pedro Miguel Sánchez Sánchez, José María Jorquera Valero, Alberto Huertas Celdrán, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez
Journal:	Internet of Things
JIF:	5.9 Q1 (2022)
Publisher:	Elsevier
Volume:	22
Number:	
Pages:	100764
Year:	2023
Month:	Jul
DOI:	10.1016/j.iot.2023.100764
Status:	Published

Abstract

In today's computing environment, where Artificial Intelligence (AI) and data processing are moving toward the Internet of Things (IoT) and Edge computing paradigms, benchmarking resource-constrained devices is a critical task to evaluate their suitability and performance. Between the employed devices, Single-Board Computers arise as multi-purpose and affordable systems. The literature has explored Single-Board Computers performance when running high-level benchmarks specialized in particular application scenarios, such as AI or medical applications. However, lower-level benchmarking applications and datasets are needed to enable new Edge-based AI solutions for network, system and service management based on device and component performance, such as individual device identification. Thus, this paper presents LwHBench, a low-level hardware benchmarking application for Single-Board Computers that measures the performance of CPU, GPU, Memory and Storage taking into account the component constraints in these types of devices. LwHBench has been implemented for Raspberry Pi devices and run for 100 days on a set of 45 devices to generate an extensive dataset that allows the usage of AI techniques in scenarios where performance data can help in the device management process. Besides, to demonstrate the inter-scenario capability of the dataset, a series of AI-enabled use cases about

device identification and context impact on performance are presented as exploration of the published data. Finally, the benchmark application has been adapted and applied to an agriculture-focused scenario where three RockPro64 devices are present.

Keywords

Hardware benchmarking · System performance · Dataset · IoT device · Identification

Internet of Things 22 (2023) 100764

Contents lists available at ScienceDirect

Internet of Things

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

Research article

LwHBench: A low-level hardware component benchmark and dataset for Single Board Computers

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 within armasuisse Science & Technology, CH—3602 Thun, Switzerland

ARTICLE INFO

Keywords:
Hardware benchmarking
System performance
Dataset
IoT device
Identification

ABSTRACT

In today's computing environment, where Artificial Intelligence (AI) and data processing are moving toward the Internet of Things (IoT) and Edge computing paradigms, benchmarking resource-constrained devices is a critical task to evaluate their suitability and performance. Between the employed devices, Single-Board Computers arise as multi-purpose and affordable systems. The literature has explored Single-Board Computers performance when running high-level benchmarks specialized in particular application scenarios, such as AI or medical applications. However, lower-level benchmarking applications and datasets are needed to enable new Edge-based AI solutions for network, system and service management based on device and component performance, such as individual device identification. Thus, this paper presents LwHBench, a low-level hardware benchmarking application for Single-Board Computers that measures the performance of CPU, GPU, Memory and Storage taking into account the component constraints in these types of devices. LwHBench has been implemented for Raspberry Pi devices and run for 100 days on a set of 45 devices to generate an extensive dataset that allows the usage of AI techniques in scenarios where performance data can help in the device management process. Besides, to demonstrate the inter-scenario capability of the dataset, a series of AI-enabled use cases about device identification and context impact on performance are presented as exploration of the published data. Finally, the benchmark application has been adapted and applied to an agriculture-focused scenario where three RockPro64 devices are present.

1. Introduction

Performance benchmarking has been an issue explored since the early days of computer science [1]. Knowing the capabilities of a device is critical to create applications optimized for it [2]. In this sense, benchmarking has become a priority due to the magnification of the number of online devices provoked by new technologies such as 5G, IoT or cloud. In addition, the explosion of techniques such as Machine Learning (ML) and Deep Learning (DL), which usually require high computational power, has been another key factor increasing the need for processing measurement applications. In this context, *device performance benchmarking* is a research avenue that acquired a large momentum in the last years [3], especially in IoT and Edge computing paradigms.

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

https://doi.org/10.1016/j.iot.2023.100764
Received 29 September 2022; Received in revised form 20 February 2023; Accepted 20 March 2023
Available online 3 April 2023

Benchmarking can be defined as an intentional stress introduced in a system to measure how the device behaves regarding a determined set of metrics [4]. The main purposes of benchmarking are to (i) verify that the performance of a device is the promised one and suitable for a certain activity such as running AI tasks, and (ii) model the internal behavior of a certain device to identify it or verify that it is running properly [5]. Therefore, device benchmarking can be seen from two perspectives. First, as a high-level system benchmarking that seeks to measure how a certain application works on the device in terms of performance or Quality-of-Service (QoS), for example, in execution time or energy consumption [6]. Second, as a low-level hardware benchmarking, which goal is to characterize the device components in a more precise way [7], so it is possible to differentiate devices or detect imperfections and errors in the chips, among other options. "Low-level" indicates that the focus of the measurements is close to the physical level of the components, measuring values such as frequencies and cycles instead of QoS (time and energy). An example of this type of benchmarking would be to measure how many cycles it takes a processor to execute a certain simple task, measuring whether it meets its specification and the necessary stability requirements.

Moreover, due to the explosion undergone by AI, and more specifically ML and DL [8], these techniques have also landed in the IoT field, being applied in areas such as industry, network and service management or cybersecurity. Thus, the field of IoT benchmarking also has the task of evaluating the performance of training and deploying ML/DL in these devices hardware. Numerous solutions have explored the capabilities of IoT devices, mainly Single-Board Computers (SBC) such as Raspberry Pi, when running different ML/DL algorithms and libraries [9], so this is an area with enough pedigree and many recent works [10]. However, low-level hardware benchmarking and the application of ML/DL in the data generated from these benchmarks remain mostly unexplored in the IoT field. This is an important task as critical functionalities are moving to the IoT and having the components of these devices properly analyzed is essential.

Then, although many benchmarks have been proposed for SBCs in recent years, as [3] shows, some challenges are present in the area, such as *(i)* many benchmarks are proposed but no exhaustive execution datasets are provided; *(ii)* all recent benchmarks focus on high-level applications and none of them measures the performance of the hardware from a low-level perspective; *(iii)* there is no work measuring the components performance from a different component of the same device, which is important to avoid inconsistent values coming from the measured component (e.g. measure an execution time using the CPU as its own source of timestamps), as it cannot notice its own inaccuracy; *(iv)* only a few solutions consider storage and memory in the benchmark, most of them are focused on CPU QoS (execution time, computation and communication latency); and *(v)* none of the previous SBC benchmark solutions consider GPU low-level performance for non-graphics processing.

In order to improve the previous limitations and fill the literature gap, the main contributions of the present work are:

- A low-level hardware component benchmarking application, namely LwHBench, which measures CPU, GPU, memory and storage device performance from the device reference point. The benchmark is implemented as a proof of concept for Raspberry Pi devices [11], taking into account the particularities of their hardware components.
- A comprehensive dataset acquired as a result of running the previous benchmark on a set of 45 Raspberry Pi of various models for 100 days [12]. For data collection, a series of measures regarding the stability of the device performance have been taken, setting the frequency of the components to fixed values and trying to reduce as much as possible the possible noise introduced in the measurements by other processes running on the devices. This dataset contains a total of 4 GB of data (2 386 126 vectors), more than any other benchmark dataset in the literature, ready to be used in ML/DL-based applications by other researchers in a wide variety of use cases.
- A set of potential use cases described as possible application scenarios for the benchmark and the published dataset. These use cases are partially solved as a preliminary exploration of the dataset, so that other researchers know how to apply their ML/DL algorithms. The code used in these use cases is also publicly available at [11]. Besides, these use cases are also demonstrated in a real world IoT agriculture deployment using 3 RockPro64 devices, another SBC model.

The remainder of this article is structured as follows. Section 2 provides a review of the literature status regarding SBC performance benchmarking. Section 3 describes the methodology and approach followed for the benchmark implementation and generation of the dataset. Section 4 describes the functions executed in each one of the components considered, while Section 5 explores the collected data through a set of use cases. Section 6 depicts a real-world adaptation and deployment of the benchmark. Section 7 draws the main strong points of the proposed method together with the drawbacks identified during the work development. Finally, Section 8 draws the main conclusions of the present work and future research lines.

2. Related work

This section analyzes related work dealing with performance benchmarking, with a special focus on Edge computing and IoT, and existing datasets regarding IoT device performance monitoring.

Regarding performance benchmarking, Varghese et al. [3] surveyed the evolution of this field from the early 90s to 2020, with special consideration of Edge benchmarking since the 2010s. This survey shows that many performance benchmarking applications have been published in recent years, most of them centered on SBCs, such as Raspberry Pi. The vast majority of these applications focus on CPU and memory benchmarking [15], while only a few test additional resources [24], such as storage, network or accelerators (GPU/TPU). Despite the usefulness to test devices from different brands and models, none of the existing Edge performance benchmarks revised in [3] is focused on the extraction of low-level information capable of detecting hardware imperfections or malfunctioning. In contrast, these existing benchmarks are based on the execution of complex or advanced applications [17], such as AI libraries [9] or orchestrators, and not on fast execution code for low-level fingerprinting. Some

Table 1
Comparison of IoT Benchmarking applications and datasets.

Solution	Device type	Monitored components	Metrics	Public data	Open source
[13]	SBC	GPU	Processing time, resource usage (%)	✗	✗
[14]	SBC	CPU	GFLOPS, energy	✗	✗
[9]	SBC/PC	CPU, GPU, memory	Processing time, mem speed, energy	✗	✓
[15]	Medical IoT	CPU, memory	Hardware Performance Counters	✗	✓
[16]	Cloud	CPU, memory, network	Processing time, network/mem speed	✗	✗
[17]	SBC	CPU, memory, network	Processing time, network/mem speed	✗	✓
[18]	Edge servers	CPU	N Instructions	✓(2.5 MB)	✓
[19]	SBC	CPU	GFLOPS	✗	✓
[20]	IoT	CPU	Packet inter-arrival time	✓(1.5 GB)	✗
[21]	IoT	Radio transmitters	Raw transmission data	✓(+50 GB)	✗
[22]	SBC	CPU	GFLOPS, energy	✗	✓
[23]	Any	SRAM and Flash memories	Initial bit status	✓(To be published)	✓
This work	**SBC**	**CPU, GPU, Memory, Storage**	**Cycles counters and processing time**	**✓(4 GB)**	✓

benchmarks [16] also measure the performance of cloud platforms oriented to the IoT, such as AWS Greengrass or Azure IoT Edge. Regarding measurement metrics, most of the benchmarks use time-based metrics such as GFLOPS (Giga Floating Point Operations per Second) for CPU or MB/s for memory and network, with only a few of them using more complex and low-level ones such as Hardware Performance Counters [15]. Besides, although most benchmarking applications provide datasets with them (12 of 14 analyzed applications), only 5 out of 14 benchmarking applications use full open-source software, while 10 out of 15 use commercial-grade software. Pincheira et al. [25] benchmarked six IoT hardware platforms in terms of support to blockchain-based agriculture applications. Finally, [3] also shows that most high-level benchmarks use commercial or proprietary software in their implementations.

In the area of low-level benchmarking, [13] evaluated the performance of the GPUs embedded in ARM SBCs, noticing great improvements comparing the GPU performance to the CPU when doing mathematical operations, but with higher energy consumption. Furthermore, [14] built a Raspberry Pi cluster and performed CPU and energy consumption testing to find the best energy/price/performance model. Similarly, three clusters, each consisting of 16 different SBC models, were built in [22] to benchmark the SBC performance in terms of computing and energy consumption. [19] followed a similar cluster-oriented benchmarking but focusing on cryptography libraries. However, these benchmarks perform fairly simple metrics about performance and do not publish their data, which do not enable the application of the generated data to new domains such as fingerprinting. From a different domain, [26] explored recently the low-level benchmarking of quantum computers, an area gaining importance in recent years that supports the need for lower-level hardware benchmarking.

Dealing with datasets about hardware performance, there are just a few examples available in the literature. Many benchmarking applications include simple data samples [3]. One example is [18], which contains 2.5 MB of traces of different high-level benchmarking applications executed in Edge servers. However, these are not exhaustive datasets collected during long execution periods and are not suitable for ML/DL approaches due to their size constraints. Regarding datasets directly focused on low-level performance fingerprinting, [23] contains fingerprints from different SRAM (Static RAM) chips, which were used in [27] to perform individual identification. However, most SBC models do not include SRAM chips due to their higher cost. From a different perspective, [21] contains radio spectrum measurements from different IoT devices, which can be employed to fingerprint their transmission performance and properties. Similarly, [28] also contains raw IQ signals from 9 IoT devices that can be used for fingerprinting tasks. Moreover, [20] presents inter-arrival time information from different wireless routers and IoT devices, and it is aimed at individual and device type fingerprinting. In contrast, to the best of our knowledge, there is not any comprehensive dataset regarding low-level performance fingerprinting or benchmarking of hardware components.

Table 1 shows a comparison between the different benchmarking applications and datasets found in the literature and the present one. From the analysis made in this section, it is noticed that there is a gap regarding solutions focused on low-level benchmarking. Most of the recent solutions focus on high-level application benchmarking, and the ones focusing on low-level performance only use simple performance metrics and do not provide extensive datasets to enable ML/DL-based use cases or new applications. Moreover, the datasets found in the literature are focused on other areas such as device identification, and not on hardware component benchmarking.

3. Benchmark and dataset generation methodology

This section describes the methodology followed in order to implement the benchmark application and collect the samples available in the dataset: providing the details of the scenario used for data collection; describing the components monitored and how their performance is measured; detailing the libraries used to collect each metric; and finally, explaining the configuration options and measures taken to ensure stability and avoid noise in the samples published.

Table 2
Number, model and most relevant characteristics of the devices used for validation.

N	RPi model	Revision	SoC	CPU	GPU	Cache	RAM	SD
15	Raspberry Pi 4 Model B	1.1/1.4	BCM2711	1.5 GHz quad-core 64 bit ARM A72	500 MHz Broadcom VideoCore VI	2-way set associative 32 kB and 48 kB level one instruction and data caches, respectively, and a 1 MB unified level two cache	4 GB LPDDR4	16 GB A1 Type 10 *SandDisk Ultra*
10	Raspberry Pi 3 Model B+	1.3	BCM2837	1.4 GHz quad-core 64 bit ARM A53	400 MHz Broadcom VideoCore IV	2-way set associative 16 kB level one instruction and data caches, and 512 kB unified level two cache	1 GB LPDDR2	16 GB A1 Type 10 *SandDisk Ultra*
10	Raspberry Pi Model B+	1.2	BCM2835	700 MHz single-core 32 bit ARM 1176JZF-S	400 MHz Broadcom VideoCore IV	2-way set associative 16 kB level one instruction and data caches, and 128 kB unified level two cache	500 MB LPDDR2	16 GB A1 Type 10 *SandDisk Ultra*
10	Raspberry Pi Zero	1.3	BCM2835	1 GHz single-core 32 bit ARM 1176JZF-S	400 MHz Broadcom VideoCore IV	2-way set associative 16 kB level one instruction and data caches, and 128 kB unified level two cache	500 MB LPDDR2	16 GB A1 Type 10 *SandDisk Ultra*

3.1. Deployment and configuration

For the LwHBench benchmark implementation and testing, a wide number of devices is required. In this sense, the benchmark is executed, to collect the dataset, in a set of 45 physical devices composed of several models of Raspberry Pi (RPi) devices. Table 2 shows a summary of the devices employed for validation, their distribution and main characteristics.

All devices of the setup have identical software images, using *Raspbian 10 (buster) 32 bits* as OS and *Linux kernel 5.4.83*. The only variation in the kernel version is related to the core architecture of each device model, *ARMv6* for RPi1/Zero, *ARMv7* for RPi3, and *ARMv8* for RPi4.

Besides, to reduce physical context as much as possible, all devices are located in the same lab room with identical cases and aluminum heat sinks.

3.2. Monitored components

For reliable time/performance measurements of hardware components, the ideal setup is to use as reference a physical oscillator independent of the component being measured (the frequency of the component is not dependent on the reference oscillator). RPi devices include all the main processing components present in a normal computer, including oscillators. However, the number of physical crystal oscillators is reduced to save costs. RPi4 only includes a SoC base oscillator running at 54 MHz and an oscillator for the USB/Ethernet controllers running at 25 MHz. In contrast, RPi3/1/Zero only include one SoC base oscillator running at 19.2 MHz. Then, each component runs at a different frequency using Phase-Locked Loops (PLLs) for base frequency multiplication [29].

This condition implies that some auxiliary components which could be used as reference points, such as the Real-Time Clock (RTC), are simulated. This fact makes it hard to accurately measure the performance and skew of the system from the device itself, as a skew in the CPU timing will affect the time measurements that it is doing of itself.

However, each component can still show performance differences based on the multiplication factor applied to the base crystal oscillator frequency in the associated PLLs. For this reason, each component used to measure the performance of the device is monitored from another component of the device, measuring, in turn, the possible imperfections and deviations between the components. More clearly, for example, the performance of code execution on the CPU is measured in terms of GPU cycles and vice versa.

Following the previous approach, the components whose performance are monitored and stored in the dataset are:

- *CPU*. The execution time of code run in the CPU is measured by monitoring how many GPU cycles have elapsed during that period of time. In this way, the skew between CPU and GPU can be accurately measured. Therefore, the formula to measure CPU performance based on GPU cycle variation is:

$$CPU_{perf.} = \Delta GPU_{cycle\_counter} \vdash t_{cpu\_code\_exec} \tag{1}$$

- *GPU*. In the GPU, the performance of code executed in this component is measured using CPU-based time, just the opposite way to the previous case.

$$GPU_{perf.} = \Delta CPU_{cycle\_counter} \vdash t_{gpu\_code\_exec} \tag{2}$$

- *Memory*. Here, memory read/write operations are also monitored in terms of CPU-based timing, as the RAM chip in RPi has its own functioning frequency different to both the CPU and GPU.

$$Memory_{perf.} = \Delta CPU_{cycle\_counter} \vdash t_{mem\_ops} \tag{3}$$

- *Storage*. Storage performance measurement is done by input/output operations in the SD card attached to the device with the software image.

$$Storage_{perf.} = \Delta CPU_{cycle\_counter} \vdash t_{io\_ops} \tag{4}$$

3.3. Benchmark implementation

The LwHBench benchmark code is available in [11]. The data collection program has been implemented using Python 3, so it can be executed as a portable script on any device with the required libraries installed. Besides, this selection has also been influenced by the set of libraries available for CPU and GPU low-level interaction, as it is explained below.

For the GPU low-level interaction and register monitoring, Idein py-videocore [30] and py-videocore6 [31] are employed for VideoCore IV and VideoCore VI GPU-based devices, respectively. To measure GPU cycles, different registers should be considered depending on the GPU version. In particular, the register monitored in VideoCore VI is *CORE_PCTR_CYCLE_COUNT*, while in VideoCore IV the monitored registers are the *performance counters 13-19* [32].

When accessing the CPU cycle counter, there are two different possibilities. The first is to generate a kernel module to enable reading the cycle counter from *userspace*. The second consists of using the interfaces provided by performance monitoring tools such as *perf*. In the code, both approaches are tested.

Regarding the kernel module, the CPU cycle counter is stored in ARMv7 and ARMv8 AArch32 processors using *c15 Cycle Counter Register (CCNT)* and, by default, it can only be read in kernel space. Therefore, a custom kernel module is implemented to allow access to this register (*enable_ccr* folder in [11]). Once compiled, the kernel module should be loaded using *insmod* command with *root* privileges. Finally, the register can be read using the assembly operation *MRC p15, 0, <Rd>, c15, c12, 1*, where *<Rd>* represents the variable where to store the register value. To access *perf* time counting, the easiest method is to use the *time* built-in Python library, which includes *perf_counter_ns()* function to retrieve time in nanoseconds using the previous counter. After some experimentation, it was decided to follow the *perf*-based approach due to its simplicity compared to using a kernel module and assembly code, and its similar consistency in the performance measurements.

Moreover, to automatize the data collection process, a system service has been implemented (*data_collection.service* in the code folder). This service is in charge of the automated data collection script launching and periodic system rebooting in order to reduce the noise introduced by possible factors related to the system running time. Concretely, each device is rebooted after 800 samples are collected.

3.4. Device setup for component stability and isolation

One of the most critical aspects of collecting reliable samples is to ensure that the conditions in the device are as constant as possible, reducing potential sources of noise in the samples. To this end, a number of measures are taken to counteract the impact of other processes running on the device. Specifically, the measures implemented to ensure stability are:

- **Fixed CPU/GPU/RAM frequency**. By default, the kernel dynamically manages the frequency of the device components to save energy when no high task load is present. However, this dynamicity affects to the stability of the performance measurements. Therefore, a fixed frequency is required in the components to measure. In RPi, the frequency of the components can be set to be constant at the maximum using *turbo_mode=1* boot option. Besides, if only a fixed CPU frequency is wanted, *performance* can be used as *scaling_governor* option.
- **Kernel level priority**. Enabling a high priority for the data collection process minimizes the interruptions caused by other programs, removing noise and inaccurate measurements. The best option here is to set the process with the highest scheduling priority. If root privilege is available, using the command `chrt -rr XX` when launching the program enables the "real-time scheduling" of the process, just like a kernel process. If it is not possible to use kernel priority, another option is to use *nice -n -20* to set the maximum *user level* priority.
- **Disable Memory Address Space Layout Randomization (ASLR)**. Memory random address organization can affect the stability of memory-related measurements; therefore, this characteristic was disabled during data collection. It can be done using `sysctl kernel.randomize_va_space=0` command, but note that this should be only enabled during memory-related data acquisition, as having ASLR disabled increases the facility to perform memory-based attacks such as buffer overflows.
- **Profiled Guided Optimization (PGO)**. PGO is a compiler option intended to improve runtime performance based on static program analysis of code. Python interpreter can be compiled to use PGO by using *–enable-optimizations* option. Furthermore, *Python garbage collector* is also disabled to avoid unintended tasks during the execution.
- **Fixed hash seed**. As hash-based CPU performance measurements are generated, using a fixed seed improves the deterministic characteristics of the function. This is set using *PYTHONHASHSEED=0* (or any other number) as environment variable when running the data collection script. This option should only be used for benchmark, as it can lead to an attacker causing a Denial-of-Service (DoS) by using worst-case performance inputs to the function, which have $O(n^2)$ complexity.
- **Core isolation**. For the multi-core CPU devices, e.g. RPi3 and RPi4, one core is isolated from the rest to execute the benchmark on it using *cpu affinity*. This setup avoids the (kernel) interruptions caused by other processes running in the same CPU core while the data is being generated. Concretely, the kernel options employed were: *i) isolcpus*, to avoid the kernel to schedule any process in that core; *ii) nohz_full*, to tell the kernel to remove as much kernel noise as possible, such as tick interrupts; and *iii) rcu_nocbs*, to offload Read-Copy-Update (RCU) threads and callbacks.

Table 3
Features gathered during data collection.

Component	Function	Monitored feature
–	Timestamp	Unix timestamp
	Temperature	Device core temperature
CPU	1 s sleep	GPU cycles elapsed during 1 s CPU sleep
	2 s sleep	GPU cycles elapsed during 2 s CPU sleep
	5 s sleep	GPU cycles elapsed during 5 s CPU sleep
	10 s sleep	GPU cycles elapsed during 10 s CPU sleep
	120 s sleep	GPU cycles elapsed during 120 s CPU sleep
	String hash	GPU cycles elapsed during a fixed string hash calculation
	Pseudo random	GPU cycles elapsed while generating a software pseudo-random number
	Urandom	GPU cycles elapsed while generating 100 MB using */dev/urandom* interface
	Fib	GPU cycles elapsed while calculating Fibonacci number for 20 using the CPU
GPU	Matrix mul	CPU time taken to execute a GPU-based matrix multiplication
	Matrix sum	CPU time taken to execute a GPU-based matrix summation
	Scopy	CPU time taken to execute a GPU-based graph shadow processing
Memory	List creation	CPU time taken to generate a list with 1000 elements
	mem reserve	CPU time taken to fill 100 MB in memory
	csv read	CPU time taken to read a 500 kB *csv* file
Storage	read ×100	100 CPU time measurements for 100 kB storage read operations
	write ×100	100 CPU time measurements for 100 kB storage write operations

4. LwHBench benchmark and dataset

This section details the operations executed by the benchmark and the events collected as features for the generation of the dataset associated with this paper. Note that this list can be updated by any other researcher just changing the code of the benchmarking script [11].

As detailed in Section 3, the components leveraged for benchmarking are: CPU, GPU, Memory and Storage. They have been selected because they are the most common hardware elements in any SBC (and generic computer). Table 3 shows the list of features collected for each device component. As it can be appreciated, any of the operations measures typical processing power metrics such as GFLOPS. This is because they are not well suited for low-level component characterization. Besides, they have already been gathered in several previous studies, as shown in Section 2. Concretely, the operations implemented as proof of concept are:

- *CPU*. Different sleep times (from 1 to 120 s) are monitored, trying to measure the accuracy for time keeping in the component. Additionally, some quick-execution functions are monitored: hash calculation of a string, pseudorandom number generation, random number generation using */dev/urandom* interface, and Fibonacci number calculation.
- *GPU*. Three simple operations are monitored in terms of CPU time: a matrix multiplication, a matrix summation and the processing of a graphic shadow.
- *Memory*. The operations executed are the generation of a list object with 1000 integers, the reserve of 100 MB of data and the time to read a 500 kB *csv* file. These operations are measured also in terms of CPU time and represent 3 features in the generated data vector.
- *Storage*. 100 read and write operations of 100 kB of data are monitored in the device SD card, generating 200 features in total.

In the dataset [12], each value of Table 3 represents one feature in the vectors (each dataset entry). Besides, each vector ends with the MAC address of the device, which can be used as the label in supervised ML/DL tasks. The data of each device is stored in a *csv* file, whose name is the MAC address. Additionally, a text file named *MAC-Model.txt* contains the association between the MAC and the model of each device in the testbed.

For data collection purposes, an additional Linux service has been developed (*data_collection.service)*. It is in charge of launching LwHBench when the device is booted and it takes care of rebooting the device once 800 samples have been collected. This reboot is done to minimize the possible impact of running time in the collected data (e.g. memory usage of persistent processes). Fig. 1 shows the flow diagram of the data collection performed in each device.

A total of 2 386 126 vectors are available in the dataset, making 4 GB of data. Fig. 2 shows the number of samples per hour from each device model during the data collection period. It can also be seen how some devices went offline during the data collection, corresponding to each of the downward jumps shown in the RPi4 and RPi3 graphs. The dataset contains per device model: 505 584 samples of RPi 1B+, 784 095 samples of RPi4, 547 800 samples of RPi3 and 548 647 samples of RPiZero. With more than two million vectors, the present dataset is the one with the highest number of samples among those found in the IoT benchmarking literature. Besides, Fig. 3 shows the number of samples per device contained in the dataset. The number varies according to the device model, as more powerful ones generate more data in the same time. Besides, some devices suffered power outages during data collection or they were added lately to the set of available devices. Still, on average, more than 50 000 vectors per device are present. It can also be seen that 5 devices have less data due to interruptions during the data collection process.

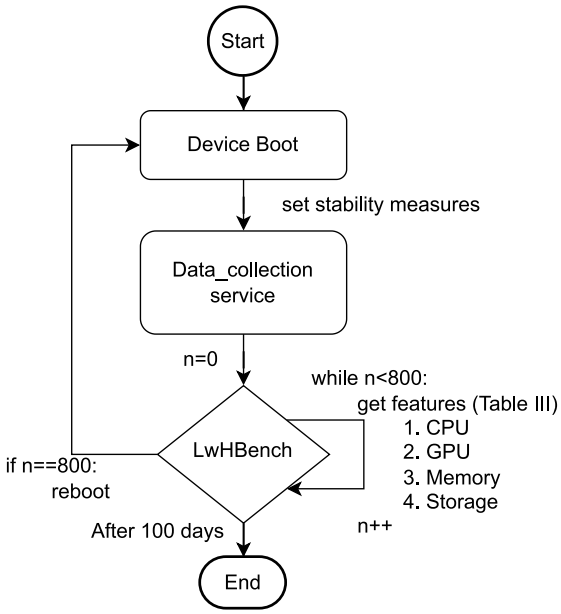


Fig. 1. LwHBench data collection flow diagram.

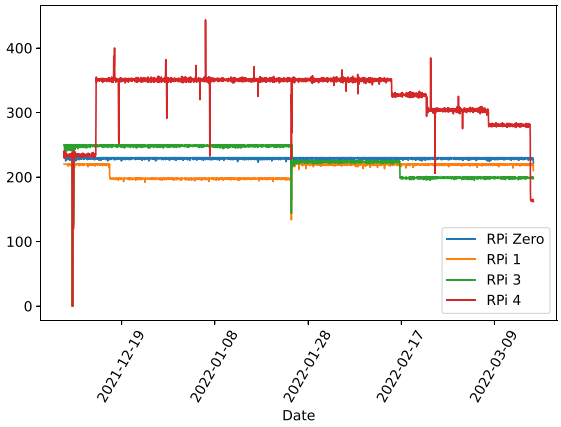


Fig. 2. Samples per hour generated from each device model.

5. Data exploration and use cases

This section explores the dataset described in the previous section. For that purpose, a set of ML/DL-based use cases for network and device management are presented using the data available. Note that the purpose of this section is to show the usefulness of the benchmarking application and the data collected with it, not to find the best solution to the problems proposed as illustrative examples.

5.1. Model/individual identification

The first use case where the dataset (and LwHBench benchmark to generate new data) can be applied is in device identification based on the performance of its device components. This can be a critical task in environments where the devices can be

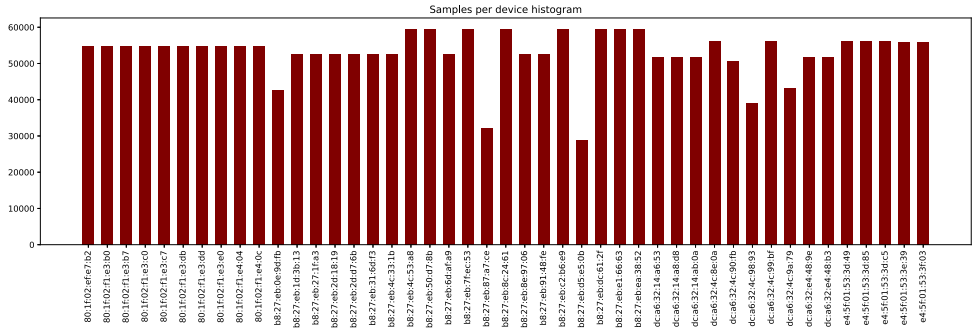


Fig. 3. Samples per device contained in the dataset.

impersonated by malicious ones with new hardware or software configurations [33]. In this sense, this use case can be seen from two different perspectives: *identification of the device model*, a relatively straightforward task from the data measuring the performance of each hardware component, as different hardware will have different performance values; and *individual identification of each device*, a more complex task since to uniquely identify devices with identical software and hardware it is necessary to analyze the differences and imperfections in the chips of each device [34].

5.1.1. SBC model identification

For this first perspective, a ML/DL-based dimensionality reduction and clustering approach [35] is followed to group together the data from each device model. This approach is selected due to its proven efficacy for automated class inference and identification in unlabeled data applied in many research areas, such as network-based IoT device type inference [36], intrusion detection [37] or even biology [38].

As a proof of concept, it is decided to apply clustering to all CPU, GPU, memory and storage features, discarding timestamps and temperature. As clustering algorithms, several options are tested, concretely *PCA*, *t-SNE* and *umap*, reducing the number of dimensions to two. So, the resultant data can be easily plotted for result explainability. Regarding clustering, *k-means* and *DBSCAN (Density-Based Spatial Clustering of Applications with Noise)* algorithms are applied over the data once the dimensions have been reduced to two. As the number of device models is previously known, the number of clusters in the algorithm configuration is set to four.

Fig. 4 shows the results when *the combination of PCA with k-means clustering* is applied as data processing approach. Different configurations of dimensionality reduction and clustering algorithms gave similar results. It can be seen how four different groups in the data emerge clearly separated, and how the clustering algorithm is able to associate the points successfully. To verify that the clustering is correct, the number of instances assigned to each of them is compared with the number of samples in the dataset belonging to each device model, as also portrayed in Fig. 2. Thus, the results are:

- *Cluster 0*: 784 095 samples, which coincides with the samples of RPi4.
- *Cluster 1*: 505 584 samples, which coincides with the samples of RPi1.
- *Cluster 2*: 547 800 samples, which coincides with the samples of RPi3.
- *Cluster 3*: 548 647 samples, which coincides with the samples of RPiZero.

From the above results, it can be concluded that LwHBench benchmark and dataset can be applied to solve the model identification problem. This approach has performed perfectly, since the number of samples in each cluster matches one of the RPi models deployed in the testbed. Moreover, it can be seen how Cluster 2 (green color) has its values much more concentrated than Clusters 0 and 3 (blue and red), which indicates higher stability in the values of this type of device.

5.1.2. Individual identification

In this case, the objective is to uniquely identify each one of the 45 devices used for dataset generation. Here, an ML/DL-based classification approach is followed due to its demonstrated performance in IoT device identification tasks [39,40].

For this task, all the features available in the dataset regarding components are used, and in this case, it is also included the temperature as a feature, since the correlation between this and the performance of each component can be one of the patterns that the ML/DL algorithm could detect when identifying each device individually. Besides, the storage-related features, 100 measurements for read time and 100 for write, are preprocessed, calculating the average, median, minimum and maximum for each feature group. As the number of devices to be identified is fixed, the 45 devices used to generate the dataset, ML/DL classification techniques [41] are used to identify each SBC. Therefore, the following techniques are compared: Decision Tree (DT), Random Forest (RF), XGBoost, k-Nearest Neighbors (k-NN), Naive Bayes (NB), Support Vector Machine (SVM) and Multi-Layer Perceptron (MLP). For k-NN, NB, SVM and MLP, normalization is applied using min–max : $x = \frac{x - x_{min}}{x_{max} - x_{min}}$. The dataset is split into

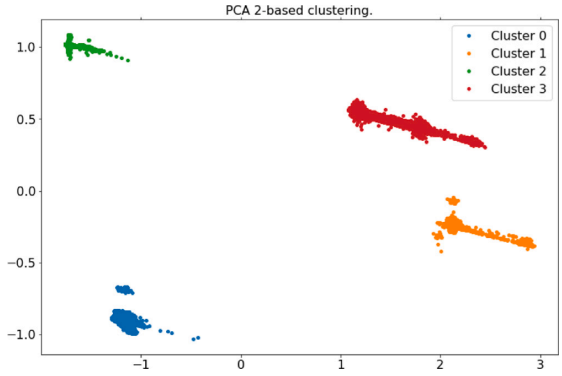


Fig. 4. PCA-based dimensionality reduction with k-means clustering.

Table 4
Device identification results.

Algorithm	Hyperparameters	Precision	Recall	F1-Score
DT	$max\_depth = None$	0.88	0.88	0.88
RF	$n\_estimators = 100$	0.93	0.93	0.93
XGBoost	$lr = 0.1, gamma = 0.01$ $max\_depth = 20$	0.97	0.97	0.97
k-NN	$k = 7$	0.32	0.32	0.31
NB	–	0.20	0.17	0.12
SVM	$kernel = linear$, $gamma = 0.01$	0.50	0.51	0.50
MLP	*1 relu hidden layer, 50 neurons*	0.58	0.57	0.56

80% of the data for training and cross-validation, and 20% for testing. Table 4 shows the Precision, Recall and F1-Score [42] per algorithm. As it can be seen, XGBoost is the algorithm providing the best results, with a 0.97 in Precision, Recall and F1-Score.

Moreover, Fig. 5 shows the confusion matrix for the 45 devices involved in the identification use case. Note that the RPi model has been added at the beginning of each label to have the devices ordered by model in the image. The results in this use case are satisfactory, as the minimum accuracy in all devices is 0.88, having more than 0.93 in most of them. Therefore, it can be concluded that the collected features are suitable for individual device identification.

Furthermore, more complex approaches could be applied to perform individual device identification depending on the scenario requirements. For example, applying time series approaches or advanced DL models such as LSTM (Long Short-Term Memory) or Transformer networks. One example of these solutions can be found in [43], where similar features to the ones collected about CPU performance were employed to differentiate 25 Raspberry Pi devices. In this case, a sliding window approach is used for vector preprocessing, incorporating new statistical information as features for the ML/DL classifier.

5.2. Performance analysis

The second use case where the benchmark and dataset can be applied is in the comparison of the performance of each device hardware component. This comparison can be seen from two different areas: intra-device comparison, where contextual circumstances such as temperature are analyzed to evaluate their impact on the device performance; and inter-device comparison, where components from different devices, but from the same model and in similar context conditions, are compared to find performance variations based on manufacturing variations.

5.2.1. Intra-device performance analysis

One intra-device use case where the collected dataset can be applied is in the analysis of the impact of temperature variations on the performance of the different device models and their components. This is another research area with a large interest in recent years [44] due to the deployment of SBC in a wide variety of critical scenarios.

For this use case, one of the devices available in the dataset is randomly selected and the impact of temperature on the other hardware-related metrics is analyzed. As the impact of temperature may vary according to the device model, one device per model is selected. Besides, only the first feature regarding storage read and write performance (storage_read_1 and storage_write_1) are

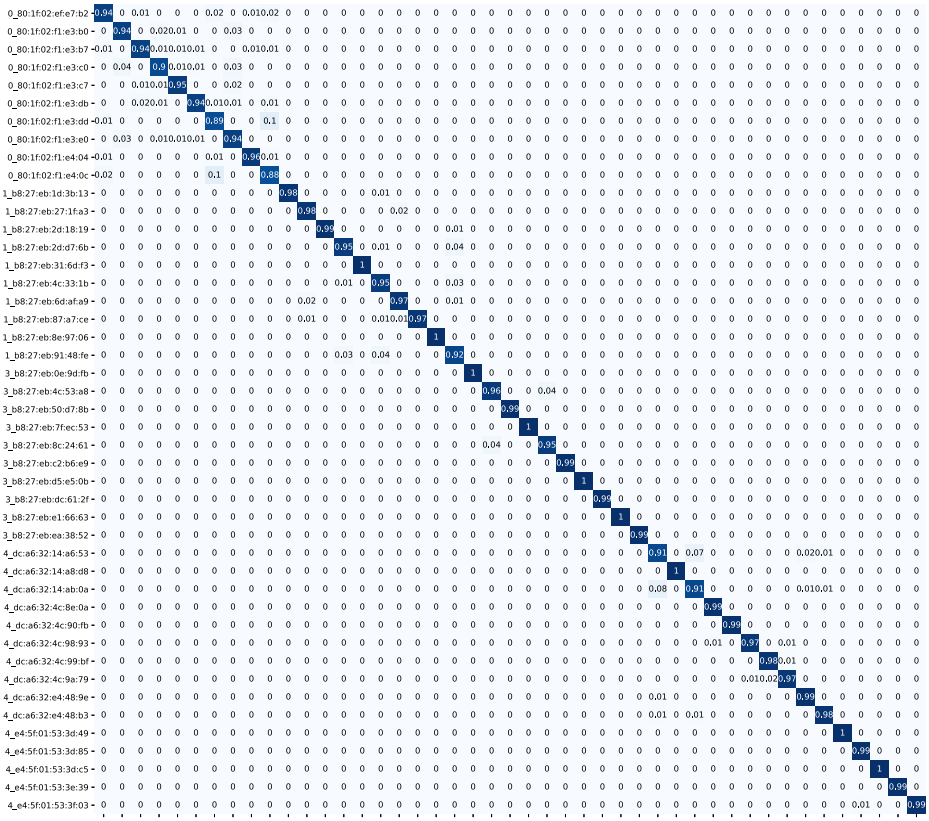


Fig. 5. Individual device identification confusion matrix.

analyzed. In order to test the impact of temperature value on the other features, the correlation between the temperature feature and the other features is studied. The correlation has values between 1 and −1 depending on whether the features increase their value linearly and positively, or inversely and proportionally.

Fig. 6 shows the correlation values for four different devices, one per available model. It can be appreciated how the RPi3 shows a high sensitivity to temperature in almost all features, only the ones based on sleep function execution seems to have a stable performance. This analysis has been repeated with the rest of the RPi3 devices to ensure that it was not a failure in one of them, with all the devices of this model showing very similar correlation graphs. In contrast, the rest of the models show much lower sensitivity to temperature changes, with values close to zero that only vary around ±0.05 in certain cases and devices.

Thus, from this use case it can be concluded that the RPi3 devices used for the generation of the dataset are the model with the highest sensitivity to temperature changes, as the correlation of this feature with the rest is high in some cases. This fact is interesting for deployments where the physical environment conditions are changing, but the performance is expected to remain stable over time.

Another interesting use case of intra-device analysis is the analysis of how the performance of a device can drop over time due to component wear and tear. For this use case, data needs to be collected over a long period of time. In the case of the available dataset, a total of 100 days (from 6th December 2021 to 17th March 2022) of data have been collected, so although it is not an extensive period, it could give clues about component and device aging.

5.2.2. Inter-device performance analysis

The last use case to be explored is the analysis of the performance variations of different hardware components within devices of the same model. This use case is closely related to individual identification, since it is the performance variations between devices of the same model that can be used to characterize each device separately.

To analyze this use case, the distribution densities of different features for devices of the same model are plotted. As a proof of concept, the selected model is the RPi4 and one feature from each component is shown: *cpu_sleep_120s* for the CPU, *gpu_matrixmul*

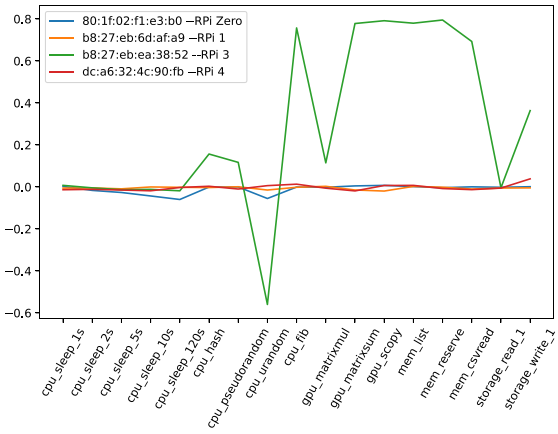


Fig. 6. Temperature correlation plot.

for the GPU, *storage_read_1* for the memory/storage, and *storage_write_1* for the storage. Although identical experiments could be performed for the other models, these are omitted for document space reasons.

Fig. 7 shows the density plots for each feature. Some interesting patterns can be appreciated in these plots. First, it can be seen how for the features *gpu_matrixmul* (Fig. 7(b)) and *storage_read_1* (Fig. 7(c)) the distributions are quite stable between the devices and all of them show similar shapes. However, *cpu_sleep_120s* (Fig. 7(a)) exhibits how each device has a Gaussian distribution centered in a different value, demonstrating that some performance variations are present within the device model. A similar situation is found with *storage_write_1* (Fig. 7(d)) feature, but in this case, the distribution plots tend to have two distinctive center values, and each device sticks to one of them (except for the dark green one with MAC *dc:a6:32:14:a8:d8* that has a higher value, although the distribution shape is similar to the rest).

From this use case, it can be concluded that performance variations are present within each device model depending on the exact device. So, it has been shown that although hardware specifications may be identical, chips contain variations that can be leveraged to perform fingerprinting or identification tasks.

6. Real-world deployment in an agriculture scenario

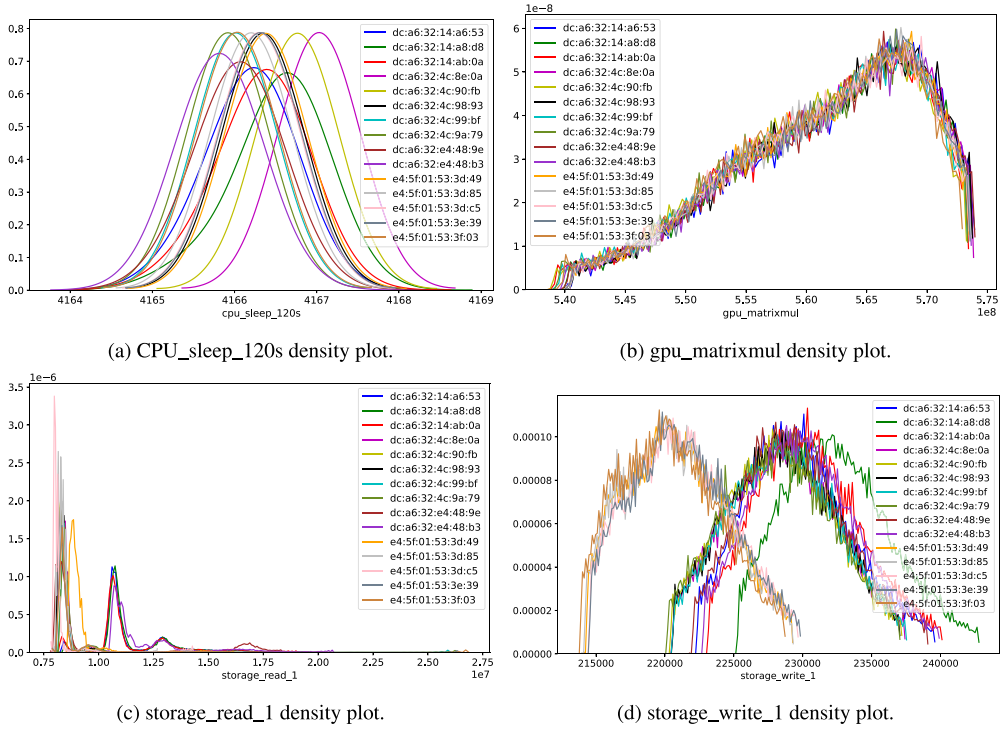
This section shows the benchmark adaptation and deployment into a real-world IoT environment based on SBC devices. Here, the aim is to show the use case applicability of the proposed solution as well as its adaptability process to other SBC models.

In this sense, an IoT sensor network for agriculture has been considered, being the sensors controlled by three PINE64 RockPro64 devices. As these devices have relatively powerful processing power, they perform the critical task of collecting the sensor data and processing it to infer information about the environment status. Then, they can activate different controllers in order to perform certain tasks, such as watering the plants or rise humidity/temperature, among others.

As the RockPro64 devices maintain control of the farming environment, they are a clear target for a potential attacker looking to extract critical process information or interfere with industrial production. One possible attack could be to replace one of the legitimate devices with an identical one but with modified software to interfere with the normal activity of the environment, for example by rising the temperature and killing the production plants. Therefore, the administrator wants to keep control of the environment, monitoring the performance of the devices during the run time while keeping them identified based on their manufacturing variations. In this sense, this approach can be seen as a real-world application of the individual device identification use case described in Section 5.1.

Each RockPro64 device includes a 6-core CPU: 4 × ARM Cortex A53 cores @ 1.4 GHz + 2 × ARM Cortex A72 cores @ 1.8 GHz; as GPU it features ARM Mali T860; and 2 GB LPDDR4 RAM. As operating system, they use 64 bit armbian (Debian-based Linux for ARM). As the GPU is different from the ones included in RPi devices, the code needed to be adapted to gather the counters from the ARM Mali T860 GPU. Concretely, the *GPU_ACTIVE* counter was selected from the ones available [45]. For cycle counter collection, *ARM HWCPipe* library [46] has been employed. For the CPU-based time gathering, *perf* time is gathered in the same way that for RPi devices, using the *perf_counter_ns()* function. The functions executed are the same as the ones depicted in Table 3, adapted in the case of the GPU to the new hardware using the *ARM Compute Library* [47]. The code is also available in [11].

For experimentation purposes, the code was deployed on the three identical devices for one week. ≈35.9 MB of data were collected, with a total of ≈12 800 vector samples (around 4000 per device). After the data were gathered, the approaches shown in Section 5.1 were implemented to monitor the performance differences between devices and perform individual identification. Fig. 8 shows the distribution plot for the CPU_sleep_120s feature. As it can be seen, the three device distributions are clearly differentiated



(a) CPU_sleep_120s density plot.

(b) gpu_matrixmul density plot.

(c) storage_read_1 density plot.

(d) storage_write_1 density plot.

Fig. 7. Raspberry Pi 4 feature density plots.

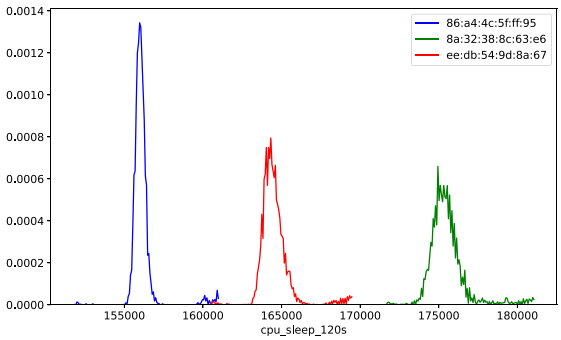


Fig. 8. RockPro64 SBC CPU_sleep_120s density plot.

between each other. This performance variation allows the individual recognition of the devices, as their performance distribution does not overlap. Therefore, the classification results, using the same ML classification algorithms as the ones depicted in Table 4, are perfect, giving 100% F1-Score and accuracy.

With this configuration, the benchmarking application is capable of verifying that the devices deployed in the environment are the legitimate ones. This real-world deployment demonstrates how the benchmarking application can be adapted and deployed in new SBC models with a relatively low effort, enabling the performance analysis of the devices where the application is deployed. This performance analysis enables different use cases, such as individual identification, to be applied in the scenario. However, some challenges were noticed during this real-world deployment. One of the main ones is the necessity of drivers to be able to interact with the CPU/GPU components in order to extract the cycle counters. This is a drawback due to the large amount of hardware

component versions available and the lack of proper documentation in some cases. The positive and negative outcomes of the complete benchmark implementation and data collection are described in the following section, Section 7.

7. Discussion

This section seeks to analyze the main advantages and weaknesses of the proposed benchmarking application and the associated dataset, highlighting the main lessons learned during the work development. From the advantages point of view, it is worth noting the next aspects:

- **An important literature gap has been covered.** As Section 2 shows, there is no low-level benchmarking application for SBC devices, enabling precise hardware analysis, nor any dataset regarding low-level SBC performance. The present work has partially covered these issues with the implementation of the benchmarking application and the release of a large dataset collected for 100 days.
- **Demonstrated utility for ML/DL-based use cases.** The collected dataset has been validated in a set of realistic use cases related to AI-based service management, mainly regarding device identification. Thus, it has been shown the utility of the benchmark when it comes to maintain under control an IoT environment where device fingerprinting is critical.
- **The benchmark is easy to deploy and extend in other RPi-based environments**. The LwHBench benchmark code is completely open-source [11], so other researchers and system administrators can execute it in their own scenarios just by installing the required dependencies. This fact also allows for the extension of the benchmark with new metrics according to the requirements of other scenarios or the available hardware.

Besides, from the drawback perspective, the next points have importance, mainly in future related research:

- **Hardware-based implementation.** LwHBench leverages CPU and GPU cycle counters, which accessed to monitor the performance of other components. Therefore, the implementation for new SBC models may require deep hardware study and understanding. For example, to access to the GPU cycle counter in a new GPU model, it would be needed to read the documentation and try the specific hardware drivers. Besides, due to the large variety in the hardware component versions, some specific components might not have open source drivers or libraries to interact with the performance counters.
- **The collected metrics are not directly applicable for traditional benchmarking.** CPU performance is measured in terms of GPU cycles, an unusual metric that is not very representative of the actual hardware performance when executing high-level tasks. Moreover, the differences between devices from the same model can come both from the measured or the measuring device, so it is difficult to measure where the chip imperfections are actually located.
- **Component isolation might reduce device performance**. The dataset has been collected while executing LwHBench in an isolated core, when possible (RPi3 and RPi4), and reducing the kernel interruptions from other processes to the maximum. Although the other cores can be used without performance limitations, isolating one core can downgrade the performance of critical tasks running in the SBC at the same time. Therefore, further experimentation with weaker isolation measurements can be interesting to know the impact of other processes on the collected values.

This work proposes the only low-level benchmark available in recent literature, and its usefulness has been validated through a series of ML/DL-enabled use cases, which are exciting topics for future IoT-based network and service management solutions. However, as this section details, the proposed solution has room for improvement regarding adaptation to other SBC models.

8. Conclusions and future work

This work has presented a low-level hardware component benchmarking application for SBC, namely LwHBench. It measures the performance of the CPU, GPU, Memory and Storage of the devices using other self-contained components. This approach ensures that the metrics collected are reliable and non-dependent on the imperfections of the component being measured. The benchmark has been implemented for Raspberry Pi devices, for all the models currently available in the market, from RPiZero to RPi4. In order to ensure optimal measurement stability, every possible action that can help to reduce the noise introduced by other programs running on the device has also been considered, such as isolating the core where the benchmark is running or blocking kernel interrupts.

In addition to the application implementation, an exhaustive dataset has been collected from running LwHBench benchmark on a set of 45 devices over 100 days, which contains more than 2 million vectors and 4 GB of data. Subsequently, to explore the available data and show possible areas of use, a series of use cases have been described and partially solved, reflecting the real needs of an environment whose management integrates modern AI-based solutions. These use cases have been divided into two groups: one on device identification, where it has been shown that using LwHBench it is possible to identify both the model and each device individually; and another on performance analysis, where the possible impact of temperature on hardware performance as well as the variations between devices of the same model have been studied.

As conclusions wrap up, the previous contributions intend to advance state of the art regarding system and device management, as it enables new solutions that, based on low-level hardware benchmarking and ML/DL techniques, improve the control over the devices deployed in modern networking environments, such as 5G-based industries.

As future work, it is planned to adapt the benchmarking application to other SBC models in order to collect data from them and perform more in-depth experiments regarding single-device identification and performance impact of temperature and device

aging. Moreover, the benchmarking application will continue being executed in the current devices, generating more data in order to analyze new use cases such as device and component aging. Finally, as further research line, it is planned to integrate the benchmarking application with federated learning techniques, so the data is processed directly in the device according to the use case, without requiring to take the data outside the SBC device.

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

Links to the data and code included in the manuscript

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, Switzerland.

References

[1] Bjørn Andersen, P.-G. Pettersen, Benchmarking handbook, Springer Science & Business Media, 1995.
[2] Roger W. Hockney, The Science of Computer Benchmarking, SIAM, 1996.
[3] Blesson Varghese, Nan Wang, David Bermbach, Cheol-Ho Hong, Eyal De Lara, Weisong Shi, Christopher Stewart, A survey on edge performance benchmarking, ACM Comput. Surv. 54 (3) (2021) 1–33.
[4] Lizy Kurian John, Lieven Eeckhout, Performance Evaluation and Benchmarking, CRC Press, 2018.
[5] P.M. Sánchez Sánchez, J.M. Jorquera Valero, A. Huertas Celdrán, G. Bovet, M. Gil Pérez, G. Martínez Pérez, A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1048–1077.
[6] Mahmoud M. El-Halwagi, Sustainable Design Through Process Integration: Fundamentals and Applications To Industrial Pollution Prevention, Resource Conservation, and Profitability Enhancement, Butterworth-Heinemann, 2017.
[7] Christopher Michael Wyant, Christopher Robert Cullinan, Timothy Richard Frattesi, Computing performance benchmarks among CPU, GPU, and FPGA, Computing (2012).
[8] Shi Dong, Ping Wang, Khushnood Abbas, A survey on deep learning and its applications, Comp. Sci. Rev. 40 (2021) 100379.
[9] Xingzhou Zhang, Yifan Wang, Weisong Shi, pCAMP: Performance comparison of machine learning packages on the edges, in: USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18), 2018.
[10] Stephan Patrick Baller, Anshul Jindal, Mohak Chadha, Michael Gerndt, DeepEdgeBench: Benchmarking deep neural networks on edge devices, in: 2021 IEEE International Conference on Cloud Engineering (IC2E), 2021, pp. 20–30.
[11] P.M. Sánchez Sánchez, LwHBench benchmarking application for raspberry pi, 2021, [Online; accessed 8-March-2022] https://github.com/sxz0/LwHBench.
[12] Pedro Miguel Sánchez Sánchez, José María Jorquera Valero, Alberto Huertas Celdrán, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, LwHBench dataset, 2022, IEEE DataPort.
[13] Ivan Grasso, Petar Radojkovic, Nikola Rajovic, Isaac Gelado, Alex Ramirez, Energy efficient HPC on embedded SoCs: Optimization techniques for Mali GPU, in: 2014 IEEE 28th International Parallel and Distributed Processing Symposium, 2014, pp. 123–132.
[14] Michael F. Cloutier, Chad Paradis, Vincent M. Weaver, A Raspberry Pi cluster instrumented for fine-grained power measurement, Electronics 5 (4) (2016) 61.
[15] Ankur Limaye, Tosiron Adegbija, HERMIT: A benchmark suite for the internet of medical things, IEEE Internet Things J. 5 (5) (2018) 4212–4222.
[16] Anirban Das, Stacy Patterson, Mike Wittie, EdgeBench: Benchmarking edge computing platforms, in: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion, 2018, pp. 175–180.
[17] Jonathan McChesney, Nan Wang, Ashish Tanwer, Eyal de Lara, Blesson Varghese, DeFog: Fog computing benchmarks, in: 4th ACM/IEEE Symposium on Edge Computing, 2019, pp. 47–58.
[18] Klervie Toczé, Norbert Schmitt, Ulf Kargén, Atakan Aral, Ivona Brandić, Edge workload trace gathering and analysis for benchmarking, in: 2022 IEEE 6th International Conference on Fog and Edge Computing, ICFEC, 2022, pp. 34–41.
[19] Daniel Hawthorne, Michael Kapralos, Raymond W Blaine, Suzanne J Matthews, Evaluating cryptographic performance of Raspberry Pi clusters, in: 2020 IEEE High Performance Extreme Computing Conference, HPEC, 2020, pp. 1–9.
[20] A.S. Uluagac, CRAWDAD dataset gatech/fingerprinting (v. 2014-06-09), 2014, [Online; accessed 31-March-2022] https://crawdad.org/gatech/fingerprinting/20140609.
[21] A.K. Hagelskjær, B.H. Grevenkop-Castenskiold, M.H. Jespersen, T. Arildsen, E. Carvalho, P. Popovski, IoT device identification dataset, 2020, [Online; accessed 31-March-2022] https://doi.org/10.5281/zenodo.3638165.
[22] Philip J. Basford, Steven J. Johnston, Colin S. Perkins, Tony Garnock-Jones, Fung Po Tso, Dimitrios Pezaros, Robert D. Mullins, Eiko Yoneki, Jeremy Singer, Simon J. Cox, Performance analysis of single board computer clusters, Future Gener. Comput. Syst. 102 (2020) 278–291.
[23] Yang Su, Yansong Gao, Surya Nepal, Damith C. Ranasinghe, NoisFre: Noise-tolerant memory fingerprints from commodity devices for security functions, 2021, IEEE DataPort.
[24] Flávio Ramalho, Augusto Neto, Virtualization at the network edge: A performance comparison, in: 2016 IEEE 17th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2016, pp. 1–6.
[25] Miguel Pincheira, Massimo Vecchio, Raffaele Giaffreda, Benchmarking constrained IoT devices in blockchain-based agri-food traceability applications, in: Blockchain and Applications: 3rd International Congress, Springer, 2022, pp. 212–221.
[26] K. Wright, K.M. Beck, S. Debnath, J.M. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N.C. Pisenti, M. Chmielewski, C. Collins, K.M. Hudek, J. Mizrahi, J.D. Wong-Campos, S. Allen, J. Apisdorf, P. Solomon, M. Williams, A.M. Ducore, A. Blinov, S.M. Kreikemeier, V. Chaplin, M. Keesan, C. Monroe, J. Kim, Benchmarking an 11-qubit quantum computer, Nature Commun. 10 (1) (2019) 1–6.
[27] Yansong Gao, Yang Su, Surya Nepal, Damith C. Ranasinghe, NoisFre: Noise-tolerant memory fingerprints from commodity devices for security functions, 2021, arXiv preprint arXiv:2109.02942.

[28] Gianmarco Baldini, IoT transient radio frequency signals, 2020, IEEE DataPort.
[29] Embedded Linux Wiki, The undocumented Pi, 2021, [Online; accessed 8-March-2022] https://elinux.org/The_Undocumented_Pi.
[30] Idein, Py-videocore. Python library for GPGPU on Raspberry Pi, 2021, [Online; accessed 8-March-2022] https://github.com/nineties/py-videocore/.
[31] Idein, Py-videocore6. Python library for GPU programming on Raspberry Pi 4, 2021, [Online; accessed 8-March-2022] https://github.com/Idein/py-videocore6/.
[32] Broadcom, VideoCore IV 3D architecture reference guide, 2013, [Online; accessed 30-March-2022] https://docs.broadcom.com/doc/12358545/.
[33] Leonardo Babun, Hidayet Aksu, A. Selcuk Uluagac, CPS device-class identification via behavioral fingerprinting: From theory to practice, IEEE Trans. Inf. Forensics Secur. 16 (2021) 2413–2428.
[34] Dror G. Feitelson, Workload Modeling for Computer Systems Performance Evaluation, Cambridge University Press, 2015.
[35] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, Arthur Zimek, A general framework for increasing the robustness of PCA-based correlation clustering algorithms, in: International Conference on Scientific and Statistical Database Management, 2008, pp. 418–435.
[36] Arunan Sivanathan, Hassan Habibi Gharakheili, Vijay Sivaraman, Inferring IoT device types from network behavior using unsupervised clustering, in: 2019 IEEE 44th Conference on Local Computer Networks, LCN, 2019, pp. 230–233.
[37] Liqun Liu, Bing Xu, Xiaoping Zhang, Xianjun Wu, An intrusion detection method for internet of things based on suppressed fuzzy clustering, EURASIP J. Wireless Commun. Networking 2018 (1) (2018) 1–7.
[38] Daniel Granato, Jânio S Santos, Graziela B Escher, Bruno L Ferreira, Rubén M Maggio, Use of principal component analysis (PCA) and hierarchical cluster analysis (HCA) for multivariate association between bioactive compounds and functional properties in foods: A critical perspective, Trends Food Sci. Technol. 72 (2018) 83–90.
[39] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, Vijay Sivaraman, Classifying IoT devices in smart environments using network traffic characteristics, IEEE Trans. Mob. Comput. 18 (8) (2018) 1745–1759.
[40] Ivan Cvitić, Dragan Peraković, Marko Periša, Brij Gupta, Ensemble machine learning approach for classification of IoT devices in smart home, Int. J. Mach. Learn. Cybern. 12 (11) (2021) 3179–3202.
[41] Aized Amin Soofi, Arshad Awan, Classification techniques in machine learning: applications and issues, J. Basic Appl. Sci. 13 (2017) 459–465.
[42] Margherita Grandini, Enrico Bagli, Giorgio Visani, Metrics for multi-class classification: an overview, 2020, arXiv preprint arXiv:2008.05756.
[43] Pedro Miguel Sánchez Sánchez, José María Jorquera Valero, Alberto Huertas Celdrán, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, Can evil IoT twins be identified? Now yes, a hardware behavioral fingerprinting methodology, 2021, arXiv preprint arXiv:2106.08209.
[44] Dimitris Gizopoulos, George Papadimitriou, Athanasios Chatzidimitriou, Vijay Janapa Reddi, Behzad Salami, Osman S Unsal, Adrian Cristal Kestelman, Jingwen Leng, Modern hardware margins: CPUs, GPUs, FPGAs recent system-level studies, in: 2019 IEEE 25th International Symposium on on-Line Testing and Robust System Design, IOLTS, 2019, pp. 129–134.
[45] Peter Harris, Mali Midgard family performance counters, 2016, [Online; accessed 30-July-2022] https://community.arm.com/arm-community-blogs/b/graphics-gaming-and-vr-blog/posts/mali-midgard-family-performance-counters/.
[46] A.R.M. Developers, HWCPipe, 2021, [Online; accessed 12-September-2022] https://github.com/ARM-software/HWCPipe.
[47] A.R.M. Developers, ARM Compute Library, 2021, [Online; accessed 12-September-2022] https://arm-software.github.io/ComputeLibrary/latest/index.xhtml.

4

SpecForce: A Framework to Secure IoT Spectrum Sensors in the Internet of Battlefield Things

Title:	SpecForce: A Framework to Secure IoT Spectrum Sensors in the Internet of Battlefield Things
Authors:	Pedro Miguel Sánchez Sánchez, Alberto Huertas Celdrán, Gérôme Bovet, Gregorio Martínez Pérez, Burkhard Stiller
JIF:	11.2 D1 (2022)
Publisher:	IEEE
Volume:	61
Issue:	5
Pages:	174 - 180
Year:	2023
Month:	May
DOI:	10.1109/MCOM.001.2200349
Status:	Published

Abstract

The battlefield has evolved into a mobile and dynamic scenario where soldiers and heterogeneous military equipment exchange information in real-time and wirelessly. This fact brings to reality the Internet of Battlefield Things (IoBT). Wireless communications are key enablers for the IoBT, and their management is critical due to the spectrum scarcity and the increasing number of IoBT devices. In this sense, IoBT spectrum sensors are deployed on the battlefield to monitor the frequency spectrum, transmit over unoccupied bands, intercept enemy transmissions, or decode valuable information. However, IoBT spectrum sensors are vulnerable to heterogeneous cyber-attacks, and their accurate detection is an open challenge in the literature. Thus, this paper presents SpecForce, a security framework for IoBT spectrum sensors based on device behavioral fingerprinting and ML/DL techniques. SpecForce considers heterogeneous data sources to detect the most dangerous and recent cyber-attacks affecting IoBT spectrum sensors, such as impersonation, malware, and spectrum sensing data falsification attacks. To evaluate the SpecForce detection performance, it has been deployed on 25 real spectrum sensors, and results show almost perfect detection for the three cyber-attack families previously mentioned.

Keywords

IoT · Battlefield · Spectrum Monitoring · Fingerprinting · Cybersecurity · Identification

MILITARY COMMUNICATIONS AND NETWORKS

SpecForce: A Framework to Secure IoT Spectrum Sensors in the Internet of Battlefield Things

Pedro Miguel Sánchez Sánchez, Alberto Huertas Celdrán, Gérôme Bovet, Gregorio Martínez Pérez, and Burkhard Stiller

The authors present SpecForce, a security framework for IoBT spectrum sensors based on device behavioral fingerprinting and ML/DL techniques.

Abstract

The battlefield has evolved into a mobile and dynamic scenario where soldiers and heterogeneous military equipment exchange information in real-time and wirelessly. This fact brings to reality the Internet of Battlefield Things (IoBT). Wireless communications are key enablers for the IoBT, and their management is critical due to the spectrum scarcity and the increasing number of IoBT devices. In this sense, IoBT spectrum sensors are deployed on the battlefield to monitor the frequency spectrum, transmit over unoccupied bands, intercept enemy transmissions, or decode valuable information. However, IoBT spectrum sensors are vulnerable to heterogeneous cyber-attacks, and their accurate detection is an open challenge in the literature. Thus, this paper presents SpecForce, a security framework for IoBT spectrum sensors based on device behavioral fingerprinting and ML/DL techniques. SpecForce considers heterogeneous data sources to detect the most dangerous and recent cyber-attacks affecting IoBT spectrum sensors, such as impersonation, malware, and spectrum sensing data falsification attacks. To evaluate the SpecForce detection performance, it has been deployed on 25 real spectrum sensors, and results show almost perfect detection for the three cyber-attack families previously mentioned.

Introduction

Today's battlefield and military operations are highly dependent on wireless communication technologies. Aircraft, warships, vehicles, weapons, and soldiers are equipped with connectivity capabilities to send and receive confidential information enabling successful offensive and defensive tactics. These deployments make up the so-called Internet of Battlefield Things (IoBT) [1], which combines the Internet of Things (IoT) characteristics with the requirements of military scenarios where properties such as security, privacy, and availability are even more critical than in civil scenarios. The dynamism of the IoBT, where troops, vehicles, and military equipment are constantly moving, requires wireless communications [2]. Here, Cognitive Radio Networks (CRN) [3] play a key role, endowing communications with programmability and high mobility in terms of used frequencies. Therefore, CRN should manage the radio frequency (RF) spectrum securely and adequately to select unoccupied frequency bands, establish secure transmissions, intercept enemy messages, and decode valuable information. In the IoBT, one of the most common approaches to enforce the previous tasks is to deploy resource-constrained spectrum sensors able to monitor and decode transmissions in different radio bands [4]. These sensors have numerous advantages, such as portability, accuracy, simplicity, and reduced cost, but they are vulnerable to cyber-attacks.

In the modern battlefield, cyberwar and cyber-attacks are common hostile acts aiming to penetrate strategic targets such as enemy communications, area defense, or critical infrastructures [5]. In this context, IoBT spectrum sensors are perfect targets due to their computational and storage constraints to maintain updated software and deploy cybersecurity mechanisms. Looking at cyber-attacks affecting IoT spectrum sensors, they can be categorized into three main families:

- *Identity-focused attacks*, whose goal is to impersonate legitimate IoBT spectrum sensors by deploying malicious ones with the same hardware and software configuration to extract sensitive military information and perform malicious activities;
- *Vulnerability-based attacks*, where typical threats such as malware are encompassed to disrupt military services, steal battlefield information, or initiate attacks to other military targets;
- *Spectrum Sensing Data Falsification (SSDF) attacks*, aiming to modify spectrum data reported by sensors to hide illegal transmissions, provoke interference and collisions, or create fictitious transmissions persuading enemies communication.

In the IoBT, the detection of the previous cyber-attack families has been tackled separately by the literature. Most works analyze software operations to detect malware and exploit vulnerabilities in generic IoT devices deployed in military scenarios [6]. However, only a few deal with SSDF attacks detection in IoBT cognitive radio

Pedro Miguel Sánchez Sánchez and Gregorio Martínez Pérez are with the University of Murcia, Spain; Alberto Huertas Celdrán and Burkhard Stiller are with the University of Zurich UZH, Switzerland; Gérôme Bovet is with the Cyber-Defence Campus within armasuisse Science & Technology, Switzerland.

Digital Object Identifier: 10.1109/MCOM.001.2200349

networks [7], and identification of IoBT devices [8]. Besides, outside the battlefield scenario, the previous three cyber-attack families have been covered in a wider manner [9] since they also affect other critical scenarios such as Industrial IoT or network management. In summary, and as can be seen in Table 1, the main limitation in the IoBT is that solutions detecting malware are not able to detect SSDF and spoofing attacks, and solutions detecting SSDF are useless for malware detection and identical device identification. Therefore, the main focus of this work is to explore novel approaches to detect all the previous attack families when they occur in IoBT spectrum sensors. One of the most promising and recent approaches to improve this situation is to combine device behavioral fingerprinting with Machine and Deep Learning (ML/DL) techniques [10]. In this context, different data sources, such as system calls, logs, hardware events, or clock skew, can be leveraged to characterize the *normal* behavior of spectrum sensors and detect anomalies or classify the three main cyber-attack families detailed before.

However, despite the achievements of existing work dealing with cybersecurity in the IoBT, and more specifically in IoBT spectrum sensors, there are still several open challenges that require further research efforts. Among the main ones, the following ones are highlighted:

- There is no definition of the threat model that IoBT spectrum sensors face, as previous solutions have analyzed their threats in a separated manner;
- Data sources and events accurately detecting normal and heterogeneous under-attack behaviors of IoBT spectrum sensors have not been investigated;
- Table 1 shows, there is no global solution detecting both system- and data-oriented cyber-attacks while evaluating the resource consumption in IoBT spectrum sensors.

In order to improve the previous challenges, the main contributions of this work include:

- The creation of a scenario where 25 real IoBT spectrum sensors are employed for radio transmission monitoring and decoding. In such a scenario, the threat model faced by these sensors is defined, and 18 heterogeneous cyber-attacks related to the thread model are considered to infect the IoBT spectrum sensors.
- The design and implementation of SpecForce, a security framework for IoBT spectrum sensors that combines device behavioral fingerprinting and ML/DL techniques. The implementation of SpecForce includes the analysis of the most suitable behavioral data sources and the ML/DL techniques for the defined threat model.
- The validation of SpecForce while detecting the cyber-attacks considered in the proposed scenario for
  - –Identity-based attacks, achieving an average 91.92 percent True Positive Rate (TPR)
  - –Heterogeneous malware detection, achieving ≈90 percent TPR and 96 percent True Negative Rate (TNR)
  - –SSDF attack detection, achieving 96–99 percent TNR and 92–100 percent TPR, depending on the attack.

Work	Scenario	Device type	Attack	Approach
[7]	IoBT	Spectrum Sensors	SSDF	Blockchain
[8]	IoBT	Generic	Generic security and trust	Blockchain
[9]	Generic	Computers	Identity	Hardware fingerprinting
This work	IoBT	Spectrum Sensors	Identity, Malware, SSDF	Behavior analysis + ML/DL

TABLE 1. Related work comparison.

Cybersecurity Threats of IoBT Spectrum Sensors

This work presents a scenario composed of 25 IoBT spectrum sensors based on Raspberry Pi (RPi) devices belonging to the ElectroSense platform [11] and deployed in different locations between Switzerland and Spain. The sensors are randomly deployed in the field since their location does not affect the framework performance (spectrum data is not leveraged for cyber-attack detection). Ten of these sensors are Raspberry Pi 3 Model B+ and 15 are Raspberry Pi 4 Model B. Each sensor is equipped with an RTL-SDR (RealTek Low cost — Software Defined Radio) USB kit and proper software to scan the RF spectrum (from 20 MHz to 1.6 GHz). Such functionality allows these sensors to monitor and decode heterogeneous wireless communications occurring between military equipment such as base stations, convoys, aircraft, helicopters, or satellites.

Despite the benefits of IoBT devices, they present some cybersecurity issues and vulnerabilities that have been already identified in [12]. In addition, some other cybersecurity issues related to spectrum sensing and hardware/software aspects of IoBT spectrum sensors need to be analyzed more in detail. In this sense, Table 2 summarizes the main threats identified after analyzing the vulnerabilities of the sensors considered in the proposed scenario. This table also provides a description and attack classification per threat.

Once the threats affecting IoBT spectrum sensors are identified, several representative and recent attack vectors per family are selected to infect the sensors. Details regarding each attack vector behavior are provided below.

- *Identity-focused*. This type of cyber-attack impersonates legit IoT spectrum sensors to steal data or execute malicious actions. For that, it utilizes identical hardware and software configurations to legit IoBT spectrum sensors [13].
- *Malware*. This type of malicious software causes harm to IoBT spectrum sensors by performing diverse malicious actions. From each malware type, different vector samples are executed in each sensor.
  - –*Rootkit*. Allow a malicious entity to gain remote control over IoBT spectrum sensors while providing self-hiding capabilities. The samples selected for testing are Beurk, Diamorphine, and Bdvl.
  - –*Botnet*. Generate a network of infected IoBT spectrum sensors to perform malicious activities, such as denial of service, in a coordinated manner. The samples selected are Bashlite and Mirai.
  - –*Backdoor*. Provide malicious actors with

Threat	Description	Identity	Malware	SSDF
Data Disclosure	Publish or access sensitive information sensed or maintained by IoBT spectrum sensors.	×	✓	✓
Spoofing	Replace legitimate spectrum sensors with malicious devices using the same identity. Usually, it is the starting point for further cyber-attacks like data injection.	✓	×	×
Sybil	Send a lot of fake data with many different IoBT sensors identities to alter the decisions generated by the IoBT platform.	✓	×	×
Jamming	Generate fake or repeated wireless signals to interrupt ongoing communications between legitimate sensors and the IoBT platform or disturb the collected data.	×	×	✓
Denial of Service (DoS)	Exhaust or degrade resources of the IoBT platform or spectrum sensors. It can affect at network level or directly at application level. Many devices can be coordinated to increase the impact of the attack, resulting in a Distributed DoS (DDoS).	✓	×	✓
Advanced Persistent Threat (APT)	Launch sophisticated, continuous, and targeted attacks over the IoBT platform or its spectrum sensors for a large time period.	✓	✓	×
Data Poisoning	Modify spectrum data monitored by IoBT sensors. It leads to wrong decisions while optimizing spectrum occupancy or decoding transmissions. Two variants are differentiated: Availability Attack and Targeted Attack.	×	×	✓
Smart Attacks	Use of machine learning techniques and security analysis devices to gather insights about the IoBT platform defense countermeasures and attack it.	✓	✓	×

TABLE 2. IoBT spectrum sensor threat model.

unintended IoBT spectrum sensor access and control. The samples selected for the present work are HttpBackdoor, Python Backdoor, and TheTick.

–*Ransomware*. Encrypt sensitive files and asks for economical ransoms for data recovery. Ransomware_PoC is the sample selected for this malware family.

- *SSDF*. This type of cyber-attack tampers the data scanned by IoBT spectrum sensors to disrupt the spectrum optimization, monitoring, and decoding services. The following SSDF attacks are executed after manipulating the ElectroSense source code [11]. The implementation details of each attack can be found in [14].

  –*Noise*. Add random noise to the spectrum data.

  –*Spoof*. Copy the spectrum data of one RF band into another band and add random noise.

  –*Repeat*. Replicate the same spectrum data in all affected RF bands.

  –*Confusion*. Swap the spectrum data between affected RF bands.

  –*Mimic*. Copy the spectrum data of one RF band into another one.

  –*Delay*. Sense different outdated spectrum data of affected RF bands.

  –*Freeze*. Sense the same outdated spectrum data in affected RF bands.

  –*Hop*. Add noise to random parts of affected segments.

Analyzing the different threats and cyber-attacks affecting the IoBT spectrum sensors proposed in the scenario, there is a clear need for solutions proving cybersecurity in a unified and homogeneous fashion.

SpecForce Framework

The SpecForce framework covers the previous limitations by combining device behavioral fingerprinting with ML/DL to detect heterogeneous cyber-attacks affecting IoBT spectrum sensors. In particular, the main objectives of SpecForce are to

- Identify malicious spectrum sensors,
- Detect heterogeneous malware
- Detect SSDF attacks manipulating spectrum data

To achieve these goals, SpecForce can be deployed in a hybrid way, where IoBT sensors host the behavior monitoring functionality and the server focuses on ML/DL-based detection. Additionally, all framework components can be deployed on the IoBT sensors.

Figure 1 shows the four main modules making up SpecForce. From an up-down prism, the *Data Gathering* module hosts three components able to periodically monitor the sensor behavior from different perspectives. These perspectives have been selected with the goal of covering the internal behavior of the IoBT in a broad fashion in terms of device components, events granularity level, and complexity. In particular, the *Kernel Software Events* component monitors activity from resources such as CPU, memory, network interfaces, or file system, among others. The *System Calls* component gathers the system calls performed by the processes of the sensor scanning the spectrum. Finally, the *Hardware Cycle Counters* component focuses on hardware manufacturing variations by monitoring the cycle counters of different hardware components. Figure 2 shows the data sources collected by each component to detect the cyber-attacks indicated earlier.

The Data Gathering module periodically sends the collected raw data to the *Data Processing* module, which is in charge of extracting valuable information and creating feature vectors with them. This module contains three different components with suitable feature extraction techniques for each data source type. As an example of their functionality,

- Highly correlated features are filtered for kernel software events,
- Different sequence and frequency (n-gram) features are calculated from raw system calls,

- Window-based statistical features are extracted from hardware cycle counters.

After that, the *Dataset Generation module* compiles all feature vectors generated by the previous module, generating datasets with the sensor behavioral data. Finally, the *AI-based Cybersecurity module* trains and evaluates supervised and unsupervised ML-based models identifying devices and cyber-attacks. For that, *Offline* and *Online* processes are considered. First, the *Offline process* selects suitable ML/DL algorithms and trains them with the created datasets. It generates the ML/DL models used by the framework. Secondly, the *Online* one evaluates the current device behavior the trained ML/DL models to detect cyber-attacks.

SpecForce Implementation and Results

To analyze the performance of SpecForce, it has been deployed on the 25 IoBT spectrum sensors of the scenario described earlier. Then, the following three use cases have been analyzed: one per cyberattack type:

- Device identification to avoid spoofing attacks
- Detection of heterogeneous malware
- Detection of SSDF attacks

Identity-Focused Attack

This use case focuses on chip imperfections affecting the hardware performance of IoBT spectrum sensors, which allow the generation of unique fingerprints per sensor to detect device spoofing attacks. Ideally, different physical oscillators should be used to analyze these imperfections, but Raspberries Pi acting as sensors only contain one oscillator used by all hardware as base frequency. Therefore, this work leverages the imperfections in circuits employed to multiply the base oscillator frequency for the device CPU and GPU separately.

After analyzing the hardware components, the *Hardware Cycle Counter* component of SpecForce monitors the data sources indicated in Fig. 2 for device identification. Then, it measures the skew between the CPU and the GPU cycle counters. To obtain stable fingerprints, different functions are executed on the CPU while the GPU cycle counter is monitored. Concretely, the functions selected are:

- Sleep during 120 seconds
- Hash calculation of a string
- Random number generation

Then, 400 output values of the previous functions are generated as raw data. To have stability in these values, process isolation measures are taken in the sensor to avoid kernel interruptions from other processes while the functions are running. Table 3 contains the details about the data gathering, data processing and evaluation steps of this use case. In total, a dataset with 10 fingerprints per device is generated (8 for training and 2 for testing). To identify the different IoBT spectrum sensors, the *AI-Based Cybersecurity* module employs ML/DL classification algorithms since the number of sensors in the scenario is constant. Once trained and evaluated, the average TPR of each model is 71.40 percent for k-NN, 89.65 percent for SVM, 91.92 percent for XGBoost, 86.47 percent for DT, 91.64 percent for RF, and 85.32 percent for MLP. As can be appreciated, RF and XGBoost are the best performing models, with

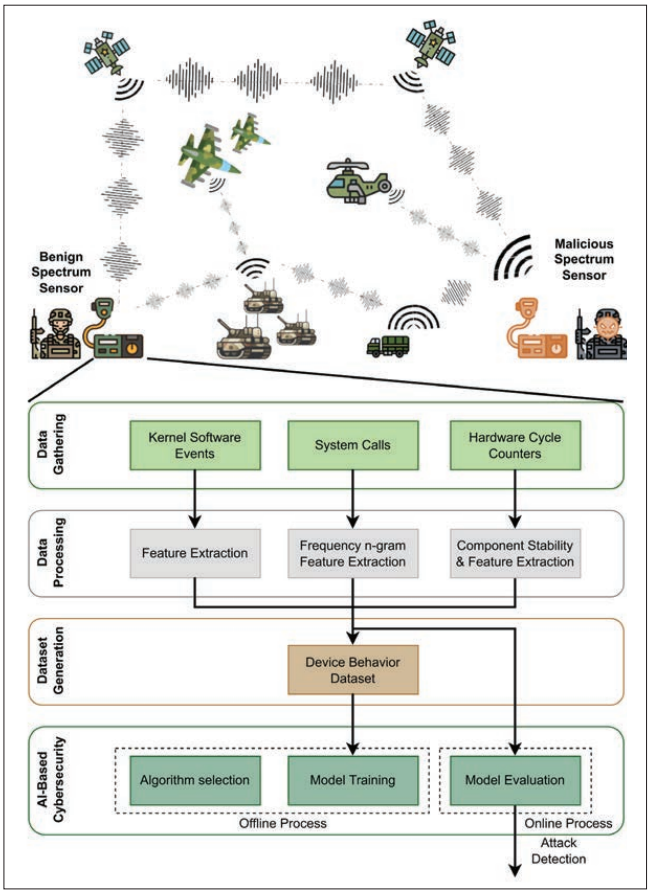


FIGURE 1. SpecForce Architectural Design and IoBT Scenario.

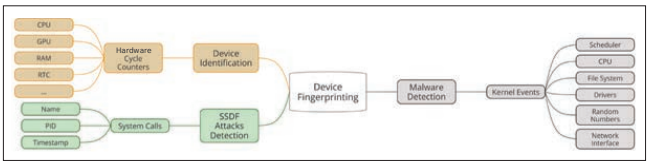


FIGURE 2. Collected data sources for device behavior fingerprinting.

+91 percent TPR. To identify those IoBT spectrum sensors having more and less similarity, Fig. 3 shows the XGBoost confusion matrix for the fingerprints used during testing. The results show how using a 50 percent TPR threshold, all IoBT spectrum sensors can be perfectly identified. Besides, XGBoost shows that the most important features to perform the identification are the median and average values of the 120 second sleep function.

This use case has demonstrated that SpecForce is able to uniquely identify 25 IoBT spectrum sensors by leveraging hardware manufacturing imperfections and ML classification techniques. In other words, SpecForce solves the issue of identity-focused attacks, as new or duplicated devices would be recognized before they can cause further harm.

	Data Gathering			Data Processing		Dataset Generation	AI-Based Cybersecurity	
Use Case	**Source**	**Freq.**	**Resources**	**Technique**	**Feature Vector**	**Dataset**	**Approach: Algorithms**	**Results**
Identity-focused Attack	CPU/GPU Cycle Counters	≈120 s	1 CPU Core	Sliding window (100 values)	18 features: Average time, standard deviation, minimum, maximum, or mode of selected functions	10 fingerprints per device, 300 vectors per fingerprint	Classification: k-NN, SVM, XGBoost, DT, RF, MLP	91.92% avg. F1-Score
Malware Detection	Kernel Events	5 s	1–4% CPU, 6.14MB RAM	Nothing	≈80 features: CPU, RAM, file system, drivers, and network events	6 hours per behavior, ≈2160 vectors per behavior	Anomaly Detection: Autoencoder, IF, COPOD, LOF, OC-SVM	+97% TNR, +97% TPR
SSDF Attack Detection	System Calls	60 s	20.2–30% CPU, 6.5MB RAM	Bag-of-Words 1-gram	17 features: number of repetitions per minute of each syscall	6 hours per behavior, 360 samples per behavior	Anomaly Detection: Autoencoder, IF, COPOD, LOF, OC-SVM	+99% TNR, +92% TPR

k-NN: k-Nearest Neighbors, SVM: Support Vector Machine, XGBoost, DT: Decision Tree, RF: Random Forest, and MLP: Multi-Layer Perceptron
IF: Isolation Forest, COPOD: Copula-Based Outlier Detection, LOF: Local Outlier Factor, OC-SVM: One-Class Support Vector Machine

TABLE 3. Technical details of the use cases implementation.

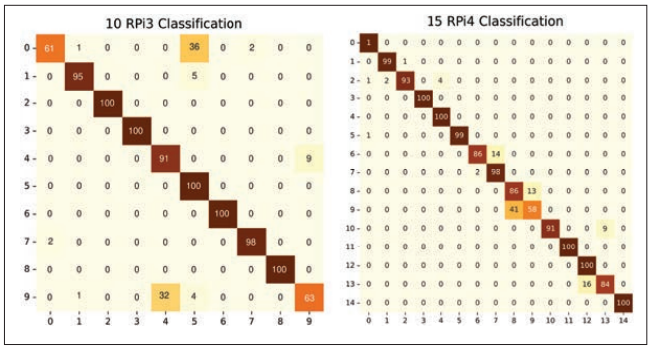


FIGURE 3. Confusion Matrices (in percentages) for Device Identification using XGBoost

Malware Detection

This second use case deals with the detection of heterogeneous malware. As starting point, a literature review is performed to study the data sources available in Raspberry Pis and the behavior of well-known malware. Due to the activity of running malware, the internal behavior of a device changes. This behavior can be reflected from several perspectives, such as syscalls, running processes or kernel events. In this sense, some known malware samples include evasion techniques that hide the malicious processes and syscalls. However, lower-level sources such as kernel events are harder to modify [10].

As output of such review, the SpecForce *Data Gathering* module is implemented to monitor in a periodic manner about 80 kernel events belonging to the usage of resources, hardware, and software activities produced in the IoBT sensors. Figure 2 shows the event families selected for malware detection. After that, one dataset with "normal" behavior of each IoBT spectrum sensor is collected for six hours. Then, the rootkits (Beurk, Diamorphine, and Bdvl), botnets (Mirai and Bashlite), backdoors (HttpBackdoor, Python Backdoor, and TheTick), and ransomware (Ransomware_PoC) mentioned earlier are executed in each IoBT spectrum sensor. Later, all malware samples are monitored for six hours while running in a passive way (without harmful actions being made) and some of them (backdoors and ransomware) performing command execution or data leakage. Anomaly detection is performed because usually attack behaviors are unknown. Therefore, deviations in the normal device behavior can allow

Figure 4 shows the detection performance of OC-SVM, the best model. Normal sensor behavior should be evaluated as "Normal" (in the X axis), and the rest of the attack behaviors should be evaluated as "Abnormal." Therefore, the higher the values in that case, the better the framework works. More than 95 percent of samples belonging to the different normal behaviors are correctly detected as "Normal." Looking at the rootkits, the passive and innocuous behavior of Diamorphine is not detected, but when it establishes an SSH connection every five seconds (Diamorphine5S), it is identified as malicious. In the case of passive behavior of Bdvl, it is detected only half of the time. In terms of Backdoors, the samples belonging to Data Leak behavior executed by TheTick are detected correctly. Similarly, it is important to highlight that the rest of the malicious behaviors are detected in an almost perfect fashion. This use case has demonstrated the capabilities of SpecForce to detect malicious activities performed by different malware affecting resource-constrained IoBT spectrum sensors. Using software kernel events and anomaly detection techniques, it is possible to characterize the behavior of the IoBT spectrum sensors and detect heterogeneous malware when they are in active and harmful mode.

SSDF Attacks Detection

The last use case focuses on detecting attacks affecting spectrum data. As in the previous ones, a literature review is conducted to identify and select behavioral data sources and events of IoBT spectrum sensors characterizing the activity of the spectrum scanning processes affected by SSDF attacks [15]. As these attacks are based on the modification of a legitimate process, the system calls generated by the software are features able to reflect the variations in normal activities.

The result of this step highlighted the suitability of *system calls* to perform such task in a precise way. Therefore, the SpecForce *Data Gathering* module uses *perf* to collect the system calls generated by a given set of processes scanning the spectrum (Fig. 2 for SSDF attacks detection). Once the data source is selected, the different normal and SSDF attack behaviors are monitored for ≈6 hours. The system calls are then processed to generate a feature vector modeling the activities of the IoBT spectrum sensing process. Then, the *AI-Based Cybersecurity* module selects, trains, and evaluates anomaly detection algorithms. Table 3 gives the technical details of this use case experimentation.

Figure 4 shows the Autoencoder True Negative Rate (TNR) for normal behavior and the TPR of the different SSDF attacks when modifying 20 MHz of the 1.6 GHz collected spectrum band. It can be seen how the normal behavior is recognized with high performance, showing +99 percent TNR. Besides, all SSDF attacks are detected with a +92 percent TPR.

This use case has demonstrated that SpecForce is able to successfully detect the different SSDF attacks executed in IoBT spectrum sensors with a low resource consumption. In particular, system calls have shown a precise characterization of the spectrum scanning process. Furthermore, when they are combined with ML/DL-based anomaly detection techniques, it is possible to detect heterogeneous SSDF attacks.

Conclusions

This work presents SpecForce, a framework combining behavior fingerprinting and ML/DL techniques to detect heterogeneous cyber-attacks affecting IoBT spectrum sensors. SpecForce has been deployed in a realistic battlefield scenario composed of 25 IoBT spectrum sensors based on Raspberry Pi. In such a scenario, first, the cybersecurity threats affecting the IoBT spectrum sensors have been analyzed to later choose identity attacks, malware, and SSDF attacks exploiting these threats.

The detection results obtained by SpecForce for each attack family affecting IoBT spectrum sensors demonstrate the suitability of the framework in a battlefield scenario. More in detail, for spoofing attacks, 25 IoBT spectrum sensors (10 identical RPi3 and 15 identical RPi4) have been individually identified based on their hardware chip variations. Regarding malware attacks, software kernel events and ML-based anomaly detection techniques have detected rootkits, botnets, backdoors and ransomware. Finally, eight different SSDF attacks have been detected by combining the system calls generated by the spectrum scanning process with anomaly detection techniques.

As future work, it is planned to deploy and validate the SpecForce framework in other scenarios, not only on spectrum sensors. Additionally, further objectives and research questions arise associated with the privacy management of the collected data, seeking to apply Federated Learning for distributed model generation without data sharing between sensors.

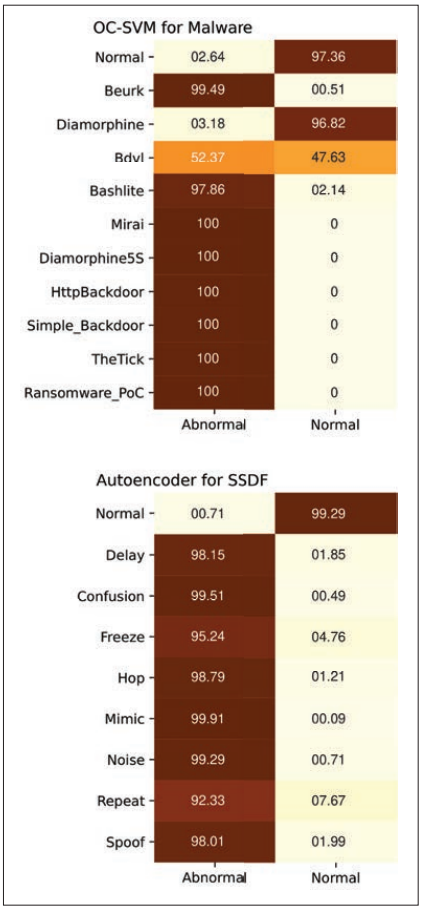


FIGURE 4. Confusion Matrices (in percentages) for Malware and SSDF Attacks.

As future work, it is planned to deploy and validate the SpecForce framework in other scenarios, not only on spectrum sensors.

Acknowledgments

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

References

[1] J. M. Stocchero *et al.*, "Secure Command and Control for Internet of Battle Things Using Novel Network Paradigms," *IEEE Commun. Mag.*, 2022.
[2] D. Papakostas *et al.*, "Backbones for Internet of Battlefield Things," *2021 16th Annual Conf. Wireless On-demand Network Systems and Services Conf. (WONS)*, 2021, pp. 1–8.
[3] A. S. Hamood and S. B. Sadkhan, "Cognitive Radio Network Security Status and Challenges," *2017 Annual Conf. New Trends in Information & Communications Technology Applications (NTICT)*, 2017, pp. 1–6.
[4] M. Suchansk *et al.*, "Electronic Warfare Systems Supporting the Database of the Radio Environment Maps," *XII Conf. Reconnaissance and Electronic Warfare Systems*, vol. 11055, SPIE, 2019, pp. 180–89.
[5] E. Izycki and E. W. Vianna, "Critical Infrastructure: A Battlefield for Cyber Warfare?," *ICCWS 2021 16th Int'l. Conf. Cyber Warfare and Security*, Academic Conferences Limited, 2021, p. 454.
[6] P. Théron and A. Kott, "When Autonomous Intelligent Goodware Will Fight Autonomous Intelligent Malware: A Possible Future of Cyber Defense," *MILCOM 2019 – 2019 IEEE Military Commun. Conf. (MILCOM)*, 2019, pp. 1–7.
[7] M. Patnaik *et al.*, "Probless: A Proactive Blockchain Based Spectrum Sharing Protocol Against SSDF Attacks in Cognitive Radio IoBT Networks," *IEEE Networking Letters*, vol. 2,

no. 2, 2020, pp. 67–70.
[8] D. K. Tosh *et al.*, "Blockchain-Empowered Secure Internet-of-Battlefield Things (IoBT) Architecture," *MILCOM 2018 – 2018 IEEE Military Commun. Conf. (MILCOM)*, 2018, pp. 593–98.
[9] I. Sanchez-Rola, I. Santos, and D. Balzarotti, "Clock Around the Clock: Time-based Device Fingerprinting," *Proc. 2018 ACM SIGSAC Conf. Computer and Commun. Security*, 2018, pp. 1502–14.
[10] P. M. S. Sanchez *et al.*, "A Survey on Device Behavior Fingerprinting: Data Sources, Techniques, Application Scenarios, and Datasets," *IEEE Commun. Surveys & Tutorials*, vol. 23, no. 2, 2021, p. 1048–77.
[11] S. Rajendran *et al.*, "Electrosense: Open and Big Spectrum Data," *IEEE Commun. Mag.*, vol. 56, no. 1, Jan. 2018, pp. 210–17.
[12] I. Agadakos *et al.*, "Security for Resilient IoBT Systems: Emerging Research Directions," *IEEE INFOCOM 2019 – IEEE Conf. Computer Commun. Wksps. (INFOCOM WKSHPS)*, 2019, pp. 1–6.
[13] P. M. S. Sánchez *et al.*, "A Methodology to Identify Identical Single-Board Computers Based on Hardware Behavior Fingerprinting," arXiv preprint arXiv:2106.08209, 2021.
[14] A. Huertas Celdrán *et al.*, "Cyberspec: Intelligent behavioral fingerprinting to detect attacks on crowdsensing spectrum sensors," arXiv e-prints, pp. arXiv–2201, 2022.
[15] H. HaddadPajouh *et al.*, "A Survey on Internet of Things Security: Requirements, Challenges, and Solutions," *Internet of Things*, vol. 14, 2021, p. 100129.

Biographies

Pedro M. Sánchez Sánchez (pedromiguel.sanchez@um.es) is pursuing his Ph.D. in computer science at University of Murcia. He received the M.Sc. degree in Computer Science from the University of Murcia, Spain. His research interests focus on continuous authentication, networks, 5G, cybersecurity, and machine learning and deep learning.

Alberto Huertas Celdrán (huertas@ifi.uzh.ch) is senior researcher at the Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH. He received the MSc and PhD degrees in Computer Science from the University of Murcia, Spain. His scientific interests include cybersecurity, machine and deep learning, continuous authentication, and computer networks.

Gérôme Bovet (gerome.bovet@armasuisse.ch) is the head of data science for the Swiss Department of Defense. He received his Ph.D. in networks and computer systems from Telecom ParisTech, France. His work focuses on Machine and Deep Learning, with an emphasis on anomaly detection, adversarial and collaborative learning in IoT sensors.

Gregorio Martínez Pérez (gregorio@um.es) 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, where he has published 160+ papers.

Burkhard Stiller (stiller@ifi.uzh.ch) chairs the Communication Systems Group CSG, Department of Informatics IfI, University of Zürich UZH, as a Full Professor. He received the M.Sc. and Ph.D. degrees, respectively, from the University of Karlsruhe, Germany. His main research interests include fully decentralized systems, including Blockchains, network and service management, IoT, and telecommunication economics.

5

Adversarial attacks and defenses on ML- and hardware-based IoT device fingerprinting and identification

Title:	Adversarial attacks and defenses on ML- and hardware-based IoT device fingerprinting and identification
Authors:	Pedro Miguel Sánchez Sánchez, Alberto Huertas Celdrán, Gérôme Bovet, Gregorio Martínez Pérez
Journal:	Future Generation Computer Systems
JIF:	7.5 D1 (2022)
Publisher:	Elsevier
Volume:	152
Number:	
Pages:	30-42
Year:	2024
Month:	March
DOI:	10.1016/j.future.2023.10.011
Status:	Published

Abstract

In the last years, the number of IoT devices deployed has suffered an undoubted explosion, reaching the scale of billions. However, some new cybersecurity issues have appeared together with this development. Some of these issues are the deployment of unauthorized devices, malicious code modification, malware deployment, or vulnerability exploitation. This fact has motivated the requirement for new device identification mechanisms based on behavior monitoring. Besides, these solutions have recently leveraged Machine and Deep Learning (ML/DL) techniques due to the advances in this field and the increase in processing capabilities. In contrast, attackers do not stay stalled and have developed adversarial attacks focused on context modification and ML/DL evaluation evasion applied to IoT device identification solutions. However, literature has not yet analyzed in detail the impact of these attacks on individual identification solutions and their countermeasures. This work explores the performance of hardware behavior-based individual device identification, how it is affected by possible context- and ML/DL-focused attacks, and how its resilience can be improved using defense techniques. In this sense, it proposes an LSTM-CNN architecture based on hardware performance behavior for individual device identification. Then, the most usual ML/DL classification techniques have been compared with the proposed architecture using a hardware performance dataset collected from 45

Raspberry Pi devices running identical software. The LSTM-CNN improves previous solutions achieving a +0.96 average F1-Score and 0.8 minimum TPR for all devices. Afterward, context- and ML/DL-focused adversarial attacks were applied against the previous model to test its robustness. A temperature-based context attack was not able to disrupt the identification, but some ML/DL state-of-the-art evasion attacks were successful. Finally, adversarial training and model distillation defense techniques are selected to improve the model resilience to evasion attacks, improving its robustness from up to 0.88 attack success ratio to 0.17 in the worst attack case, without degrading its performance in an impactful manner.

Keywords

Adversarial attacks · Device Identification · Internet of Things (IoT) Security · Context Attack · Machine Learning

Future Generation Computer Systems 152 (2024) 30–42

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Adversarial attacks and defenses on ML- and hardware-based IoT device fingerprinting and identification

Pedro Miguel Sánchez Sánchez [a,*], Alberto Huertas Celdrán [b], Gérôme Bovet [c], 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:
Adversarial attacks
Device identification
Internet of things (IoT) security
Context attack
Machine learning

ABSTRACT

In the last years, the number of IoT devices deployed has suffered an undoubted explosion, reaching the scale of billions. However, some new cybersecurity issues have appeared together with this development. Some of these issues are the deployment of unauthorized devices, malicious code modification, malware deployment, or vulnerability exploitation. This fact has motivated the requirement for new device identification mechanisms based on behavior monitoring. Besides, these solutions have recently leveraged Machine and Deep Learning (ML/DL) techniques due to the advances in this field and the increase in processing capabilities. In contrast, attackers do not stay stalled and have developed adversarial attacks focused on context modification and ML/DL evaluation evasion applied to IoT device identification solutions. However, literature has not yet analyzed in detail the impact of these attacks on individual identification solutions and their countermeasures. This work explores the performance of hardware behavior-based individual device identification, how it is affected by possible context- and ML/DL-focused attacks, and how its resilience can be improved using defense techniques. In this sense, it proposes an LSTM-CNN architecture based on hardware performance behavior for individual device identification. Then, the most usual ML/DL classification techniques have been compared with the proposed architecture using a hardware performance dataset collected from 45 Raspberry Pi devices running identical software. The LSTM-CNN improves previous solutions achieving a +0.96 average F1-Score and 0.8 minimum TPR for all devices. Afterward, context- and ML/DL-focused adversarial attacks were applied against the previous model to test its robustness. A temperature-based context attack was not able to disrupt the identification, but some ML/DL state-of-the-art evasion attacks were successful. Finally, adversarial training and model distillation defense techniques are selected to improve the model resilience to evasion attacks, improving its robustness from up to 0.88 attack success ratio to 0.17 in the worst attack case, without degrading its performance in an impactful manner.

1. Introduction

The advances in processing and communication technologies achieved in recent years, enabled by more powerful chips and enhanced connectivity, have motivated an explosion in the deployment of Internet-of-Things (IoT) devices [1]. These devices have generated various use cases and scenarios [2], such as Industry 4.0, Smart Cities and Homes, or Healthcare. Therefore, the typology of IoT devices is also very heterogeneous depending on the required capabilities of each scenario. In this context, Single-Board Computers (SBC), such as Raspberry Pi, have gained prominence due to their flexibility, relatively high processing power and reduced cost.

However, this increase in processing power not only comes with advantages. Cybersecurity issues have a greater impact when more powerful IoT devices are compromised, as they can perform stronger attacks such as more petitions in a Distributed Denial of Service (DDoS) or calculations for cryptojacking [3]. Therefore, the securitization of the IoT scenario leveraging SBCs is a key factor in guaranteeing its correct functioning. One of the most important aspects is the correct identification of each device deployed, avoiding the presence of unauthorized devices. Static identifiers such as credentials or certificates were traditionally assigned to the devices, but attackers can clone or modify these to introduce illegitimate entities [4]. To solve this issue,

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

https://doi.org/10.1016/j.future.2023.10.011
Received 19 July 2023; Received in revised form 5 September 2023; Accepted 20 October 2023

the literature has widely explored the usage of device behavior and hardware to identify the devices deployed [5].

Behavior-based IoT device identification tasks can be tackled from different granularity levels depending on the environment requirements [5]. There exist two main approaches: *device model or type identification* (e.g., camera, light bulb, etc.), based on characteristics such as network activities or running processes [6], and *individual device identification*, where devices from the same model are differentiated based on hardware manufacturing variations using low-level component analysis or radio frequency fingerprinting [7]. Individual device identification is the one offering the best security guarantees. However, it requires lower-level behavior monitoring, as chip manufacturing variations must be analyzed to differentiate devices with the same hardware and software [8]. This work focuses on individual device identification and the attacks that can be used to spoil the identification: attacks on the data (through manipulation of the device or environment/context) and attacks on the identification techniques (poisoning). In this sense, hardware performance analysis is one of the most exploited techniques for identification, monitoring how components such as CPU, GPU, or RAM behave or perform when executing a certain task [7]. However, in these solutions, an attacker might exploit the component usage or device context to modify the values generating the fingerprint for device identification and interrupt the identification process.

Following the trend in other fields, the application of Machine Learning (ML) and Deep Learning (DL) techniques has gained prominence in IoT security during the last years, including the device identification task [9]. But with the deployment of these techniques, adversarial attacks against ML/DL models have appeared [10], trying to affect the training process or fool the model predictions [11]. These attacks can affect various steps of the ML/DL pipeline. Usually, they target: (i) the training data to poison the model or include backdoors in it [12]; (ii) the testing data to find vulnerabilities in the trained model and change its predictions [13]; or (iii) privacy to infer data from the model and its gradients [14].

ML/DL-based IoT identification solutions have recently been an objective for adversarial attacks [15], demonstrating that these solutions are also affected by context modifications [16] or when malicious adversarial samples are employed in the identification process [17,18]. However, several questions remain when joining hardware behavior-based individual device identification, ML/DL techniques, and adversarial attacks. Some of these research questions are: (i) which is the best-performing ML/DL technique for device identification based on hardware behavior?; (ii) which is the threat model faced by these solutions?; (iii) how device context affects the identification performance?; (iv) how ML/DL-focused adversarial attacks affect the identification process?; (v) how defense techniques improve the resilience to context- and ML/DL-focused adversarial attacks?

To tackle the previous challenges, the main contributions of the present work are:

- A LSTM-CNN neural network architecture for individual device identification based on hardware performance behavior of device CPU, GPU, memory and storage. Hardware monitoring and data collection are performed from the device itself, leveraging the different components as internal reference points to avoid the usage of external sensors. Then, the proposed model considers performance measurements as data points in a time series for data processing and pattern extraction. The architecture performance is compared with different ML/DL classification approaches using the LwHBench dataset [19]. This dataset contains hardware performance and behavior data from 45 Raspberry Pi devices running identical software images. The proposed LSTM-CNN architecture achieves an average F1-Score of +0.96, correctly identifying all the devices with a +0.80 True Positive Rate.
- The threat model definition of the adversarial situations that might affect a self-contained hardware behavior-based individual device identification solution. It encompasses the complete data lifecycle, from the internal fingerprint generation to its evaluation, usually using ML/DL techniques.
- The analysis of the impact of different context- and ML/DL-focused adversarial evasion attacks on the individual identification framework. In terms of context attacks, this analysis shows that temperature does not have a big impact on the performance of the identification solution, and that other context conditions, such as kernel interruptions, can be mitigated during data collection. Regarding ML/DL evasion attacks, the state-of-the-art approaches are successful when performing a targeted attack during evaluation, achieving up to 0.88 attack success rate, and performing a successful device spoofing.
- The application of different defense techniques aiming to improve the model robustness against the previous adversarial attacks. The defense techniques selected are adversarial training and model distillation. The results show that the combination of both techniques reduces the attack impact to ≈0.18 success rate in the worst case. Additionally, state-of-the-art robustness metrics such as empirical robustness or loss sensitivity also show an effective increase.

The code, data, and DL models associated with the previous results and experiments are publicly available in [20] for reproducibility purposes.

The remainder of this article is structured as follows. Section 2 gives an overview of hardware-based individual device identification, context-focused attacks, and ML/DL-focused attacks. Section 3 describes the ML- and hardware-based device fingerprinting solution for individual device identification. Section 4 gives an overview of the threat model that an IoT device identification solution suffers. Section 5 gives an overview of the implementation and impact of the adversarial attack on device identification. Next, Section 6 describes how the application of defense mechanisms enhances the solution resilience against adversarial attacks. Section 7 contrasts the key takeaways and acknowledges limitations. Finally, Section 8 gives an overview of the conclusions extracted from the present work and future research directions.

2. Related work

This section gives the insights required to understand the concepts used in the following sections and reviews the main works in the literature associated with the present one.

2.1. Hardware-based individual device identification

In [7], the authors compared the deviation between the CPU and GPU cycle counters in Raspberry Pi devices to perform individual identification of 25 devices. The identification was performed using XGBoost, achieving a 91.92% True Positive Rate (TPR). Similarly, [16] performed identical device identification using GPU performance behavior and ML/DL classification algorithms. Accuracy between 95.8% and 32.7% was achieved in nine sets of identical devices, including computers and mobile devices. Only the impact of temperature changes was verified.

Sanchez-Rola et al. [21] identified +260 identical computers by measuring the differences in code execution performance. They employed the Real-Time Clock (RTC), which includes its own physical oscillator, to find slight variations in the performance of each CPU. In [8], the author compared the drift between the CPU time counter, the RTC chip, and the sound card Digital Signal Processor (DSP) to identify identical computers. Finally, Deb Paul et al. [22] were able to uniquely identify 20 IoT devices by using an external sensor to leverage the delay

in the signals going through the printed circuit boards (PCBs) of the device. They also verified the identification stability under different temperature conditions, emulating context-attacks.

Other works have explored the usage of Physical Unclonable Functions (PUFs) for IoT device identification [23]. PUFs are digital fingerprinting technologies that leverage inherent manufacturing variations as unique identifiers. It generates hardware-specific cryptographic keys, enhancing security by making the devices virtually impossible to clone or impersonate. However, PUFs are out of the scope of this work, as their usage requires the addition of new hardware components or the low-level modification of the hardware components or firmware, limiting the real-world scalability and usability of the solutions based on this technology [7].

2.2. Context-focused attacks

In hardware-based identification solutions, the context in which the identification code or tasks are executed might influence the collected data and, therefore, the results achieved. In this sense, the temperature can affect the frequency of crystal oscillators or hardware load might introduce delays due to the scheduling between processes. Therefore, a malicious attacker could change these context conditions to affect the identification, making it unusable or generating measurements that mimic another device.

The works described in the previous section briefly discussed context issues that may affect the identification process. [7] showed that device rebooting and other processes running in the device impacted the identification results if proper process isolation mechanisms for data collection were not implemented. Besides, they checked that usual temperature changes based on device load did not affect the results. [16] demonstrated that environment temperatures between 26.4 °C and 37 °C did not affected to the identification results. However, rebooting had an impact on the identification, dropping the results to 50.3% accuracy. The authors also leave voltage variations as a future line to evaluate. In [21], the authors evaluated the identification application under different CPU loads and temperatures, with positive results in both cases. Finally, [8] only mentioned temperature impact analysis as part of future work and no context-based experiments were performed.

As can be seen, none of the previous works on device fingerprinting and identification based on hardware performance behavior has extensively explored the impact that context-focused attacks may have on their results.

2.3. ML/DL-focused adversarial attacks

Adversarial ML/DL [11] is a research field that seeks to develop not only accurate models, but also highly robust models against tampering. It studies the possible attacks against ML/DL models as well as the defense techniques that can secure these. There exist a wide variety of taxonomies for the attacks that an ML/DL model may suffer. In this sense, attacks can be classified in: (i) *Poisoning*, when the model is attacked during training using malicious samples; (ii) *Evasion*, when the model evaluation process is attacked, trying to fool a legitimately trained model; and (iii) *Model Extraction* attacks, where the attacker tries to infer the model based on its predictions.

In this work, the focus is on evasion attacks, as the intention is to full a model trained for device identification, making it misclassify a malicious device for the legitimate one. Here, the main types of attacks are: *non-targeted attacks*, when the objective is just to misclassify a sample to any different class that is not the original one, and *targeted attacks*, when the objective is to evaluate the malicious sample as a concrete objective class. Several evasion attacks can be found as the most common ones in the literature:

- As one of the first evasion attacks for DL, Goodfellow et al. [24] proposed the Fast Gradient Sign Method (FGSM). This attack performs one-step updates in the adversarial sample following the direction of the gradient loss, trying to move the sample into the boundary of a different class. The equation characterizing FGSM can be seen as:

$$X_{adv} = X + \epsilon * sign(\nabla_x J(X, Y)) \tag{1}$$

  where ϵ is a parameter defining the size of the perturbation update and $\nabla_x J(X, Y)$ is the gradient loss function for the sample X.
- Basic Iterative Method (BIM) [25] is a improvement over FGSM by including iterative optimization. This is, applying FGSM several times with small perturbation steps.
- Momentum Iterative Method (MIM) [26] integrates momentum into the iterative FGSM or BIM, avoiding local minimum or overfitting influence in the generated adversarial samples.
- Projected Gradient Descent (PGD) [27] is a generalization of BIM that has no constraints on the iteration steps.
- DeepFool L_2 attack [28] minimizes the Euclidean distance between the original and the adversarial samples by estimating the model decision boundary using a linear classifier.
- Jacobian-based Saliency Map Attack (JSMA) [29] is another common attack in the literature. It uses the Jacobian matrix [30] of the model to find the sensitivity direction of the model and perform feature selection to minimize the number of characteristics modified from the original data sample.
- Boundary Attack [31] generates a random adversarial sample and then performs optimizations in the $L_2 - norm$ of the perturbation to make the sample similar to the original legitimate vector, but maintaining the misclassification result.
- Carlini&Wagner (C&W) Attack [32] proposes a optimization-based adversarial sample generation. It can be applied to three distance metrics: L_0, L_2, L_∞. L_0 measures the number of features to be modified, L_2 measures the Euclidean distance between a benign and adversarial sample, and L_∞ measures the maximum change to any feature.
- Generative Adversarial Network (GAN)-based attack [33] uses GAN models to generate realistic adversarial samples able to fool the classifier.

Numerous defense mechanisms have arisen against the previous attacks and others that may be found in the literature [34]. The objective of these countermeasures is to make the models resistant to adversarial samples. Generally, these can be classified into detection and robustness methods, depending if the aim is to detect crafted malicious samples prior to evaluation or make the model resistant to the evaluation of these, respectively. Besides, defense mechanisms can be attack-specific or attack-agnostic, depending on whether they are focused on improving resilience against a specific attack.

One of the most extended defense techniques to avoid evasion attacks is *Adversarial training* [35], where malicious samples are employed for model training, avoiding the impact of the attacks that generated those samples. *Knowledge distillation* [36] has also been applied for robustness improvement at training. This technique seeks to generate smaller models using the base model outputs as features, hence making the knowledge of the larger model more accessible and efficient to use. It can improve the model robustness by generating smoother decision boundaries and less sensitivity to adversarial samples [37].

2.3.1. Robustness metrics

Robustness metrics provide a quantitative measure to gauge the stability and resilience of neural networks against adversarial perturbations and input variations. The main metrics found in the literature are:

- CLEVER score [38], denoted as Cross Lipschitz Extreme Value for Network Robustness, measures the smallest perturbation required to alter the classification result by utilizing the local Lipschitz constant [39]. The higher the Lipschitz constant, the more sensitive the network is to input perturbations.
- Loss sensitivity [40] calculates the largest variation of the output of a neural network under a small change in its input. Overall, it quantifies the smoothness of a model [41]. A model is considered smoother when there is minimal variation in its output. In mathematical terms, loss sensitivity is depicted by the gradient of the loss function concerning the input data. The gradient magnitude reveals how the loss changes in response to variations in the input. Eq. (2) calculates Loss sensitivity (g), where $\mathcal{L}$ represents the loss function. A smaller value of g (i.e., a smaller variation in the output for perturbed inputs) indicates that the model is smoother and potentially more robust to input variations or adversarial attacks.

$$g = \left\| \frac{\partial \mathcal{L}}{\partial x} \right\|_1 \tag{2}$$

- Empirical robustness, as defined in [28], quantifies the average smallest disturbance necessary to alter a model prediction. This is mathematically represented in Eq. (3), where C stands for a trained classifier, ρ denotes an untargeted attack, and X represents the test data. Initially, adversarial inputs, $\rho(x_i)$, are generated, and the classifier is evaluated against these inputs. Notably, the equation only accounts for the adversarial inputs that successfully deceive the model. Hence, only the indices $I \in 1, 2, n$ where $C(x_i) \neq C(\rho(x_i))$ are considered. The selection of appropriate attacks is intricate, typically relying on the success rate and computational efficiency of FGSM, C&W, and DeepFool attacks.

$$ER(C, \rho, X) = \frac{1}{|I|} \sum_{i \in I} \frac{\|\rho(x_i) - x_i\|}{\|x_i\|} \tag{3}$$

- *Confidence Score* measures the likelihood of accurate sample predictions. It assesses the consistency of predictions; a model with more stable predictions is deemed more robust [41]. Eq. (4) calculates the confidence score as the average of precision scores across all thresholds. Here, T represents the labels, and $Thrs$ denotes the probabilities that a vector is classified correctly.

$$Confidence = \frac{1}{|Thrs|} \Sigma_T \frac{TP}{TP + FP + FN} \tag{4}$$

Each metric offers a unique perspective on robustness, ranging from leveraging local Lipschitz constants to quantifying model smoothness and evaluating the average minimal perturbations required to alter model predictions.

2.3.2. Adversarial ML in IoT identification and security

In this sense, [17] is the closest work to the one at hand. The authors analyzed the impact of different non-targeted and targeted adversarial attacks (FGSM, BIM, PGD and MIM) over a CNN implemented for radiofrequency-based individual device identification. Similarly, Namvar et al. [18] evaluated the resilience of network-based IoT identification ML solutions against adversarial samples generated with FGSM, BIM, and JSMA. They showed how classifier models giving +90% accuracy decrease their performance to 75%–55% when exposed to maliciously crafted samples. From a different perspective, Benegui and Ionescu [42] evaluated the impact of adversarial samples over ML/DL models for user identification based on motion sensors, achieving near to 100% attack success rates with FGSM, JSMA, DeepFool, and Boundary Attacks. Later, [43] demonstrated that a GAN-based attack has more impact than the previous attacks in the user identification context.

From a more generic perspective, [15,44] reviewed the threat of adversarial attacks in ML solutions applied in network security. They proved the high impact of adversarial attacks over ML-based security systems, highlighting the need for more research on attack and defense methods in the area.

Table 1 shows a comparison between the different solutions reviewed in this section. It can be seen how none of the previous works combines the application of context- and ML/DL-focused attacks. Besides, some solutions require external sensors or components to perform the hardware-based identification. Finally, the ML/DL-focused attack papers in the context of device or user identification have not explored the reward from the defense mechanisms available in the literature. Therefore, the present work solves a gap in the literature, providing useful insights in the impact of attack and defense techniques on the context of hardware- and ML/DL-based individual device identification.

3. Individual device identification

The present section describes the ML/DL framework implemented for hardware-based individual device identification. It sets the baseline results for later attack and defense technique impact analysis. This approach follows the higher privilege principle [45]. This is, using the highest privileges for the data collection and processing software deployed in the sensor, isolating it from other processes that might potentially tamper the process. Even though this countermeasure is taken, in the next sections, it is assumed that an attacker could tamper with the solution.

3.1. Dataset collection and preprocessing

This subsection describes how the hardware behavior of the device is monitored from the device itself in order to generate the fingerprint representing its internal characteristics.

3.1.1. Dataset collection

For individual device identification based on hardware behavior, the imperfections in the chips contained in the device should be monitored to be later evaluated. As seen in Section 2, this task has usually been tackled in the literature by comparing components using different crystal oscillators or base frequencies, as deviations in the performance of these components can be noticed from the device itself.

To implement the individual device identification framework, a dataset leveraging metrics related to the hardware components contained in some devices was required. The dataset was named LwHbench and more details are available in [19]. In this sense, the dataset collected performance metrics from CPU, GPU, Memory, and Storage from 45 Raspberry Pi devices from different models for 100 days, enough time to accurately model the performance of the hardware contained in each device. Different functions were executed in these components, using other hardware elements (running at different frequencies) as references for performance measurement. Table 2 summarizes the set of functions monitored. These functions represent a list of common operations executed in each component, trying to measure its performance. Note that other similar operations could be leveraged in the data collection process.

The dataset contains per device model: 505 584 samples from 10 RPi 1B+, 784 095 samples from 15 RPi4, 547 800 samples from 10 RPi3 and 548 647 samples from 10 RPiZero. During the data collection process, several countermeasures were taken to avoid the effect of noise introduced by other processes running in the devices: fixed component frequency, so the hardware performance is stable; kernel level priority, so other processes cannot interrupt the execution of the code; code executed in an isolated CPU core (in multi-core devices), so other processes are not present in the CPU trying to get resources; and the disabling of memory address randomization, so the memory performance is not degraded by accessing at random points of the stack. Besides, the dataset was collected under several temperature conditions, allowing the impact analysis of this context characteristic in the component performance.

Table 1
Comparison of previous works on Context and ML/DL-focused adversarial attacks in identification solutions.

Work	Platform and Objective	Attack type	Attack technique	Defense	Results
si te [8] (2007)	Computer identification	✗	✗	✗	Computer identification based on the comparison of three physical oscillators using t-test statistic
[21] (2018)	Computer identification	Context-focused	CPU load, temperature	Process isolation	265 computers uniquely identified. No effect from CPU load and temperature
[16] (2022)	Computer and mobile identification	Context-focused	Temperature changes and device reboot	✗	95.8% and 32.7% accuracy in nine sets of identical devices. Accuracy drop with device rebooting
[22] (2022)	IoT device identification	Context-focused	Temperature and voltage changes	✗- ML attacks theoretically considered	20 IoT devices identified using the delay in the PCB signals. Temperature and voltage changes impact analyzed.
[7] (2023)	IoT device identification	Context-focused	Temperature changes and device rebooting	Process isolation	91.92% average TPR in 25 devices. No effects from temperature changes and device rebooting
[42] (2020)	User identification	DL-focused	Non-targeted and targeted attacks (FGSM, JSMA, DeepFool, Boundary)	✗	Near to 100% attack success over CNNs models with different depths (from 4 to 12 layers)
[43] (2021)	User identification	DL-focused	GAN-based attack	✗	GAN-generated samples were more effective than FGSM, Deepfool and Boundary when performing adversarial attacks
[18] (2021)	IoT device identification	ML/DL-focused	Non-targeted attacks (FGSM, BIM, JSMA)	✗	Accuracy decreased from +90% to 75%–55%
[17] (2021)	IoT device identification	ML/DL-focused	Non-targeted and targeted attacks (FGSM, BIM, PGD and MIM)	✗	Proven vulnerability to targeted attacks with +80% attack success rate.
This work (2023)	IoT device identification	Context and ML/DL-focused	Context: Temperature changes, CPU load, device rebooting ML/DL: FGSM, BIM, MIM, PGD, JSMA, Boundary Attack, C&W	Process isolation, Adversarial training, Model distillation	+0.96 average F1-Score. Resilience to temperature and process-based context attacks. ML/DL evasion attack resilience improved using model distillation and adversarial training.

Table 2
Features available in LwHBench dataset [19].

Component	Function	Monitored feature
–	timestamp	Unix timestamp
	temperature	Device core temperature
CPU	1 s sleep	GPU cycles elapsed during 1 s CPU sleep
	2 s sleep	GPU cycles elapsed during 2 s CPU sleep
	5 s sleep	GPU cycles elapsed during 5 s CPU sleep
	10 s sleep	GPU cycles elapsed during 10 s CPU sleep
	120 s sleep	GPU cycles elapsed during 120 s CPU sleep
	string hash	GPU cycles elapsed during a fixed string hash calculation
	pseudo random	GPU cycles elapsed while generating a software pseudo-random number
	urandom	GPU cycles elapsed while generating 100 MB using */dev/urandom* interface
	fib	GPU cycles elapsed while calculating Fibonacci number for 20 using the CPU
GPU	matrix mul	CPU time taken to execute a GPU-based matrix multiplication
	matrix sum	CPU time taken to execute a GPU-based matrix summation
	scopy	CPU time taken to execute a GPU-based graph shadow processing
Memory	list creation	CPU time taken to generate a list with 1000 elements
	mem reserve	CPU time taken to fill 100 MB in memory
	csv read	CPU time taken to read a 500 kB *csv* file
Storage	read × 100	100 CPU time measurements for 100 kB storage read operations
	write × 100	100 CPU time measurements for 100 kB storage write operations

Table 3
Feature set extracted for validation.

Operation collected	Python code function	Sliding windows	Statistics extracted	No. features
10 s sleep	*time.sleep(10)*			40
120 s sleep	*time.sleep(120)*			40
string hashing	*hashlib.sha256(str)*	10 Sliding windows.		40
urandom	*os.urandom()*	Group sizes:	Minimum,	40
matrix mul	*vc.cond_mul()*	10, 20, 30, 40,	maximum,	40
matrix sum	*vc.cond_add()*	50, 60, 70,	mean,	40
list creation	*list.append()*	80, 90, 100	median	40
memory reserve	*cgroup.set_memory()*			40
CSV read	*pandas.read_csv()*			40
1st storage read	*os.read()*			40
1st storage write	*os.read()*			40
Total				440

3.1.2. Data preprocessing

As the first preprocessing technique and following the approach of [7], sliding-window-based feature extraction was performed per device, extracting statistical features such as median, average, maximum, minimum, and summation. The reasoning behind this preprocessing is that the distribution of raw feature values from each device may overlap due to the limited variability in the component performance. Therefore, statistical values such as median or average help to differentiate between partially overlapping distributions. Only some of the available raw features were selected for this step, as keeping a low feature number helps to lighten the ML/DL model training. Table 3 describes the set of features extracted from the dataset of each device.

In addition to sliding windows, it was decided to directly evaluate the raw data vectors without the sliding window processing described above. The reasoning behind this approach is that having a large dataset of raw values can work well in the case of DL models, which can automatically extract internal insights from the data. In this approach, only timestamp and temperature features were filtered, using the rest of the values (215 values in total) as features for the models.

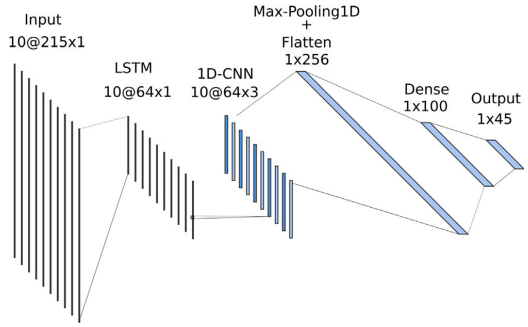


Fig. 1. LSTM-1DCNN architecture proposed.

Finally, it is also decided to perform a time series-based evaluation, concatenating together the available samples in groups of 10 vectors. This grouping technique allows the application of time series DL methods such as LSTM and 1D-CNN models [46,47]. These models can extract complex trends in the data that may achieve better results than the isolated processing and evaluation of individual samples.

3.2. LSTM-1DCNN architecture

This work proposes a client–server framework that leverages an LSTM-1DCNN neural network architecture for the classification of the performance samples obtained from the device. These models have shown good performance in very varied time series scenarios, such as human activity recognition [48], gold price forecast [49], or DNA protein binding [50]. The data generated and preprocessed in the device are sent to a server for model training and later device evaluation and identification.

The network architecture combines LSTM and 1D-CNN layers to extract patterns in the series fed as input. The main benefit of this approach is that combines the recursion patterns extracted by the LSTM layer, due to its memory capabilities, with the space patterns extracted by the 1D-CNN layer, as kernels are applied to close features to derive more complex ones.

Fig. 1 describes the neural network architecture explained above, depicting the size of each layer. The LSTM layer is configured to return sequences, so the 1D-CNN layer can be applied afterward in those sequences. After the 1D-CNN layer, Max-Pooling is applied. Finally, a fully connected layer of 100 neurons is added before the last layer with 45 outputs, one per device. In the implementation, the LSTM layer has 64 neurons, the 1D-CNN layer uses *ReLU* is used as activation function in the hidden layers and *ADAM* is used as optimizer during training (Table 4 show the complete list of hyperparameters tested).

3.3. Classification-based device identification performance

Once the two data preprocessing approaches were applied to generate two datasets, one with raw values and another with sliding-window-based features, the next step was to compare the proposed LSTM-1DCNN model with the most common ML/DL classification approaches. In [7], the authors directly applied ML classifiers using CPU and GPU-related statistical features similar to the ones described in the previous section. Moreover, LSTM and 1D-CNN networks were also tested for the time series approaches. Finally, a more complex multi-input network that combined one LSTM and one 1D-CNN input layer was also implemented for comparison, this model is denoted as Multi_1DCNN_LSTM. The experiments were performed in a server equipped with an *AMD EPYC 7742* CPU and an *NVIDIA A100* GPU.

Table 4
Classification algorithms and hyperparameters tested against the proposed architecture.

Model	Hyperparameters tested
Naive Bayes	No hyperparameter tunning required
k-NN	$k \in [3, 20]$
SVM	$C \in [0.01, 100]$, $gamma \in [0.001, 10]$ $kernel \in \{'rbf', 'linear', 'sigmoid', 'poly'\}$
AdaBoost	$n\_estimators \in [10, 100]$
XGBoost	$lr \in [0.01, 0.3]$, $max\_depth \in [3, 15]$ $min\_child\_weight \in [1, 7]$, $gamma \in [0, 0.5]$, $colsample\_bytree \in [0.3, 0.7]$
Decision Tree	$max\_depth \in [None, 5, 10, 15, 20]$ $min\_samples\_split \in [2, 3, 4, 5]$
Random Forest	$number\_of\_trees \in [50, 1000]$ $max\_depth \in [None, 5, 10, 15, 20]$ $min\_samples\_split \in [2, 3, 4, 5]$
MLP	$n\_layers \in [1, 3]$, $neurons\_layer \in [100, 500]$, $batch\_size \in [32, 64, 128, 256, 512]$ $activation = relu$, $optimizer = [SGD, adam, adamax]$
1D-CNN	$filters = [16, 32, 64, 128]$, $kernel\_size = [3, 5, 7]$, $n\_layers = [1, 2, 3]$, $optimizer = [SGD, adam, adamax]$
LSTM	$neurons = [10, 100]$, $n\_layers = [1, 2, 3]$, $optimizer = [SGD, adam, adamax]$
Multi_1DCNN_LSTM	$input\_layers = [2, 3]$, $cnn\_filters = [16, 32, 64, 128]$, $cnn\_kernel\_size = [3, 5, 7]$, $lstm\_neurons = [10, 100]$ $n\_layers = [1, 2, 3]$, $optimizer = [SGD, adam, adamax]$

Table 4 describes the algorithms and hyperparameters tested against the proposed architecture. Besides, for the algorithms requiring data normalization, *QuantileTransformer* [51] was applied, as the data from the different device models had different distributions based on their hardware capabilities. 80% of the data was used for training and cross-validation, while 20% was used for testing. The train/test splitting was done without vector shuffling to avoid that possible order correlation in the vectors might influence the results.

Table 5 depicts the classification results for each algorithm (with its best hyperparameter setup) in both of the generated datasets. The performance metrics are Accuracy, average Precision, average Recall, and average F1-Score: (TP: True Positives, TN: True Negatives, FP: False Positives, FN: False Negatives)

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall\ or\ TPR = \frac{TP}{TP + FN} \tag{7}$$

$$F1 - Score = \frac{2 \times precision \times recall}{precision + recall} \tag{8}$$

It can be seen that the LSTM-1DCNN model is the classification model with the best classification performance, achieving around 0.96 in all the reported metrics in the case of the usage of raw data features. It also shows how the time series approaches using DL-based models are the ones with the best performance, achieving +0.93 in all the reported metrics and improving XGBoost, which was the model with the best performance in similar literature solutions. Besides, Fig. 2 shows the confusion matrix for each device. It can be appreciated that all devices show +0.80 TPR (True Positive Rate), therefore having positive identification of all of them.

The comprehensive experimentation conducted in this study underscores the superior performance of the LSTM-1DCNN model in device identification tasks, particularly when utilizing raw data features. This model not only outperformed traditional machine learning classifiers but also demonstrated a significant improvement over the best-performing models cited in related literature. From this experiment, it is interesting to observe that the use of raw data features

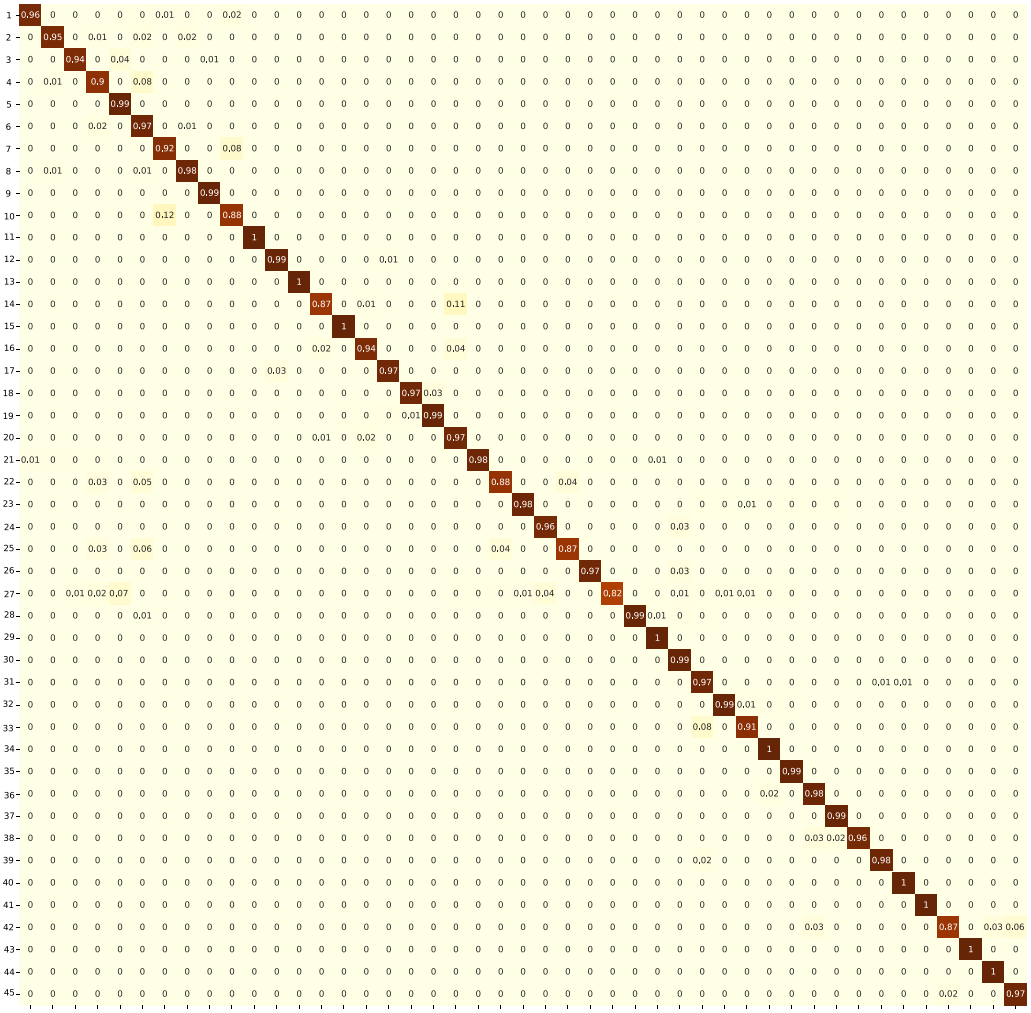


Fig. 2. Individual device identification confusion matrix.

Table 5
Baseline classification model performance.

Model	Raw data features				Sliding-window features			
	Accuracy	Avg. precision	Avg. recall	Avg. F1	Accuracy	Avg. precision	Avg. recall	Avg. F1
	Single vector approaches							
Naive Bayes	0.4569	0.4735	0.4569	0.4473	0.6829	0.6935	0.6829	0.6719
k-NN	0.4526	0.4679	0.4526	0.4472	0.5274	0.5410	0.5274	0.5285
SVM	0.7838	0.7955	0.7829	0.7849	0.7375	0.7434	0.7318	0.7297
AdaBoost	0.0705	0.0060	0.0705	0.0110	0.0706	0.0060	0.0706	0.0110
XGBoost	0.9059	0.9173	0.9056	0.9087	0.7498	0.7655	0.7498	0.7461
Decision Tree	0.7816	0.7896	0.7825	0.7837	0.6932	0.7045	0.6932	0.6910
Random Forest	0.8549	0.8664	0.8542	0.8570	0.7487	0.7615	0.7487	0.7430
MLP	0.8895	0.8960	0.8880	0.8899	0.6840	0.6988	0.6758	0.6749
	Time series approaches (10 values)							
1D-CNN	0.9428	0.9453	0.9428	0.9428	0.6941	0.7170	0.6941	0.6862
LSTM	0.9346	0.9430	0.9346	0.9346	0.7225	0.7345	0.7225	0.7147
LSTM_1D-CNN	**0.9602**	**0.9626**	**0.9602**	**0.9602**	0.7149	0.7287	0.7149	0.7080
Multi_1DCNN_LSTM	0.9535	0.9553	0.9535	0.9535	0.6784	0.6947	0.6784	0.6700

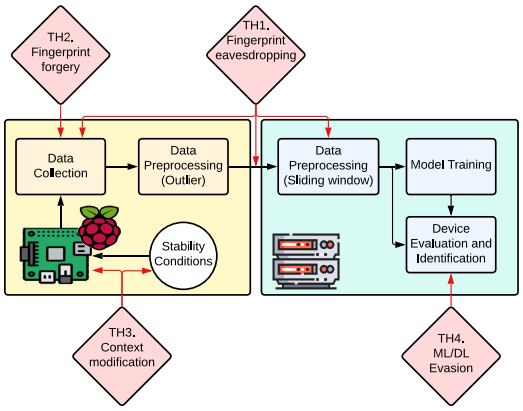


Fig. 3. Threat impact on the different steps of the identification process.

instead of sliding-window ones achieved better results in most of the classification models, something that contrasts with previous works in the literature [7]. This can be a consequence of using a larger dataset than the ones used previously, which also includes information about memory and storage ([7] only included features regarding CPU and GPU).

4. Threat model

This section details the threat model faced by an ML/DL-based device identification solution based on internal hardware performance monitoring. In this sense, an attacker may try to affect the two different sides of the identification: (i) the hardware generating the data or (ii) the ML/DL models in charge of the data evaluation. Fig. 3 reflects the hardware-based identification process and where the different threats can disturb the solution.

- *TH1. Fingerprint eavesdropping and hijacking*. An adversary could read the data composing a fingerprint, either at the level of in-device data collection, communication, or during processing (in a server or the device itself), and then use it in another device to impersonate the identity of the first. This threat implies a reduced knowledge of the fingerprint generation process and the functions and components used during the process. This threat primarily exploits the vulnerabilities in data transmission and storage. An attacker might employ techniques like packet sniffing to capture data during transmission. They could also exploit weak encryption methods or even unencrypted data storage to access the fingerprint data. Once the data is accessed, it can be replayed or used in another device to impersonate the original device. The attacker might also exploit weak authentication protocols or lack of multi-factor authentication to gain unauthorized access.
- *TH2. Fingerprint forgery*. Since the components and frequencies of the devices are public, an attacker with knowledge about the functions that are executed to generate the fingerprint could try to generate a new one that resembles that of a legitimate device. This threat would be triggered possibly on a trial/error or brute force basis. This threat requires thorough knowledge of the implementation of the fingerprint generation process and the values composing the fingerprint. This threat involves a deep understanding of the device hardware and software components. An attacker might use tools to monitor the device performance metrics, such as CPU usage, memory allocation, and power consumption, to reverse engineer the fingerprint generation process. They might also exploit public documentation or even insider information to gain knowledge about the specific algorithms and processes used. Once they have this knowledge, they can craft or modify fingerprints to impersonate legitimate devices.
- *TH3. Context modification*. As the fingerprint is based on data collected from the performance of the execution of certain tasks in the software, an attacker may try to modify the conditions under which the fingerprint is generated. This can neglect to successfully recognize a legitimate device or generate fingerprints that pretend to mimic another device. The context can be modified from several perspectives, for example raising the device temperature (using external tools or exhaustively using the hardware) or introducing software that may add kernel interruptions in the fingerprint collection program, which should be isolated from these interactions as much as possible. This threat exploits the environmental and operational conditions under which the fingerprint is generated. For instance, an attacker might use external heaters or coolers to manipulate the device temperature. They could also run resource-intensive tasks to change the device performance metrics. On the software side, they might introduce malware or other software that interrupts or alters the fingerprint collection process. For instance, a malware that causes frequent CPU spikes could distort the fingerprint. Additionally, rebooting the device frequently or altering its clock speed can also impact the fingerprint generation.
- *TH4. ML/DL evaluation evasion*. In ML/DL-based solutions, an attacker with enough knowledge or access to the evaluation model can be able to craft malicious data samples to fool the ML/DL solution. These samples can target and impersonate a specific device following a trial and error approach or using a targeted attack. This threat capitalizes on the vulnerabilities in ML/DL models. An attacker, with knowledge of the model architecture and parameters, can craft adversarial samples that the model misclassifies. Techniques like gradient ascent on the input data or perturbing the input data in a way that the model output changes can be employed. The attacker might also exploit transferability, where adversarial samples crafted for one model can fool another model. They could use tools and libraries specifically designed for crafting adversarial attacks, or even exploit weak spots in the model architecture, like layers with fewer neurons or weak activation functions. In this sense, several adversarial attacks have been proposed in the literature as shown in Section 2.

Therefore, a proper individual device identification solution has to consider and evaluate the previous threats in order to ensure correct functioning and attack resilience. In essence, while ML/DL-based device identification solutions offer advanced capabilities, they are not immune to threats. Ensuring robustness against these threats requires a multi-faceted approach, encompassing secure data transmission and storage, robust fingerprint generation processes, resilient ML/DL models, and continuous monitoring and updating of the system to counter emerging threats.

5. Adversarial attacks

This section shows the results of the different adversarial attacks tested on the previous ML/DL-based device identification model. These adversarial attacks reflect the implementation of the threats described in the previous section. The objective is to measure how vulnerable the model is to these attacks if an adversary wants to impersonate a given device or disrupt the identification process. The threats reflecting the tested attacks are: *TH2, TH3, and TH4*. *TH1. Fingerprint eavesdropping and hijacking* is assumed to be solved by using encryption in all communications and the higher privilege principle in all the processes of the device and server.

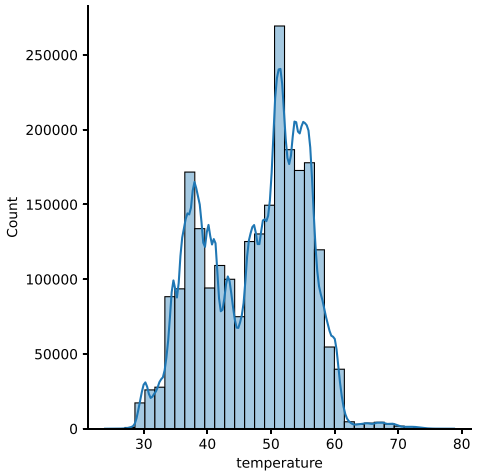


Fig. 4. Temperature distribution in the collected dataset.

5.1. Identification disruption attack (TH3)

The objective of an identification disruption attack is to deny the identification of a legitimate device by performing a context-based attack. Therefore, it is implementing *TH3. Context modification*. Here, the adversary seeks to modify the device hardware performance metrics by changing the device environmental and contextual conditions. Then, the device can no longer be identified properly, and the service is affected. In this attack, the objective is not the generation of adversarial samples that mimic a different device, but the generation of noisy samples that make the identification process of the legitimate device not possible. This experiment evaluates how resilient the identification solution is to context and environmental changes.

Based on the results in previous research [7,16], the dataset was collected considering context stability conditions as mentioned in Section 3. These conditions ensured that the collected data was not affected by other processes in the device. Therefore context-based attacks leveraging factors such as device rebooting or kernel interruption from other processes are not successful because these were already considered during data collection. Moreover, [7] proved that if the stability and isolation measurements are not included during data collection, the hardware-based identification becomes unstable and does not work properly when the context changes.

Regarding temperature, an attacker could try to rise the device temperature by externally interacting with the device with a heat source or by extensively executing resource-demanding tasks in the device. To represent this issue during data collection, the environmental conditions were modified by adding heatsinks to the components and turning on/off fans attached to the devices during data collection. In this sense, Fig. 4 shows the temperature distribution in the samples contained in the dataset. It can be seen that the temperature during data collection varied from 30 °C to +60 °C.

The temperature conditions were randomly varied during data collection. Therefore, the train and test datasets employed in Section 3 do not have a temperature-based bias. However, an attacker might induce new temperature conditions not seen during fingerprint generation to disrupt the identification service. To test this attack, the base dataset of each device was ordered based on the temperature and then divided into train and test samples following an 80/20 ordered split. Then, a new model was generated to compare its performance with the one selected in the previous section. This experiment was repeated both in ascending and descending order. Note that temperature was only used for data ordering and not as a feature.

Table 6
Temperature-based context attack.

Model	Accuracy	Avg. precision	Avg. recall	Avg. F1-Score	Min. TPR
Baseline order	0.9602	0.9626	0.9602	0.9602	0.8045
Temperature ascending order	0.9621	0.9652	0.9621	0.9619	0.6387
Temperature descending order	0.9394	0.9480	0.9392	0.9402	0.6682

Table 6 compares the baseline model with the ones trained by ordering the samples according to the temperature they were collected at. It can be seen how evaluating samples generated at new temperatures does not significantly affect the model performance, with only a 0.03 decrease in the average of the metrics in the case of descending order and even a slight improvement for ascending order. However, the minimum TPR of the evaluated devices (the metric employed for threshold-based identification) is reduced to 0.6682 and 0.6387, respectively. This drop to around 0.65 is observed in two devices in both configurations. Although all the devices can still be identified by setting a threshold in the 0.50 TPR value, these results show that some devices can be more affected by temperature variations.

This experiment demonstrated that temperature conditions do not excessively impact the device identification performance, as the average performance does not degrade when evaluating data generated under temperatures different from the ones during training. The results of the temperature-based context attack experiment revealed that while the identification model performance was not significantly affected by new temperature conditions, there was a notable decrease in the minimum TPR for some devices. This indicates that while the majority of devices remained identifiable, certain devices were more susceptible to temperature variations. This issue, for some devices, can lead to wrong identification if the TPR-based threshold is defined at a high value. Therefore, it can be concluded that the identification approach is resilient to temperature conditions but an eye should be kept to ensure that all the devices meet the performance requirements.

5.2. Device spoofing attacks (TH2, TH4)

In the device spoofing attacks, the adversary performs an evasion attack over the already trained ML/DL model, modifying the evaluated data to change the model outputs, so a malicious device is identified as a legitimate one. In this setup, complete knowledge of the model by the attacker was assumed, therefore having a *white-box* evasion attack. This attack fulfills both *TH2. Fingerprint forgery* and *TH4. ML/DL evaluation evasion*, as malicious fingerprint samples are generated in order to fool the model during evaluation. This attack could also occur as a consequence of a side-channel attack over the data collection process where the adversary is able to modify the samples according to his objective.

As the objective was to fool the device identification model, only targeted attacks make sense to evaluate how easy it is to impersonate other devices. In this sense, one device from each RPi model present in the dataset is selected as the "target class" (constant for attack hyperparameter optimization). Then, the samples from the rest of the devices from each model are used to impersonate that device. Concretely, the selected targets are:

- RPiZero with MAC *80:1f:02:f1:e3:e0*
- RPi1 with MAC *b8:27:eb:87:a7:ce*
- RPi3 with MAC *b8:27:eb:dc:61:2f*
- RPi4 with MAC *dc:a6:32:e4:48:9e*

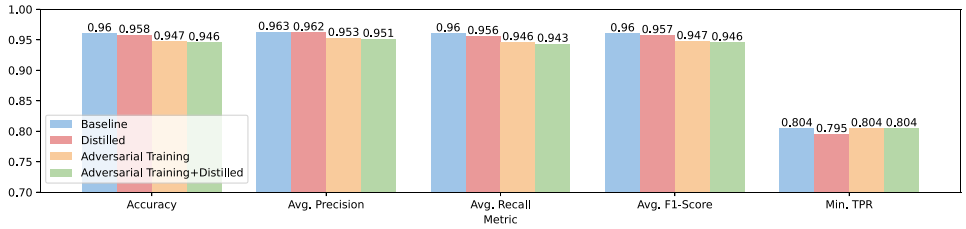


Fig. 5. Performance metrics of the robust models on the legitimate test set.

Table 7
Adversarial attack results.

Attack	Attack success rate	Time
FGSM, $\epsilon = 0.05$	0.3056	8.79 s
BIM, $\epsilon = 0.5$	0.8823	752.64 s
MIM, $\epsilon = 0.05$	0.8537	793.97 s
PGD, $\epsilon = 0.6$	0.8823	748.06 s
NewtonFool, $eta = 0.1$	0.0994	1168.94 s
C&W, L_2	0.1766	63 734.18 s
C&W, L_{inf}	0.0834	142 087.72 s
JSMA, $\theta = 0.1$	Fails	–
Boundary attack	0.2507	379 232.28 s

Table 8
Attack ASR on the robust models.

Attack	Baseline model	Distilled model	Adversarial training	Adversarial training + Distilled
FGSM, $\epsilon = 0.05$	0.3056	0.2725	0.2704	0.1561
BIM, $\epsilon = 0.5$	0.8823	0.3024	0.1482	0.1631
MIM, $\epsilon = 0.05$	0.8537	0.7950	0.1918	0.1784
PGD, $\epsilon = 0.6$	0.8823	0.2741	0.1155	0.1235
NewtonFool	0.0994	0.0600	0.0846	0.0839
C&W, L_2	0.1766	0.1190	0.0953	0.0952
C&W, L_{inf}	0.0834	0.0835	0.0848	0.0841
JSMA, $\theta = 0.1$	Fails	Fails	Fails	Fails
Boundary attack	0.2507	0.0989	0.0886	0.0861

For the implementation of the attacks, the Adversarial Robustness Toolbox (ART) [52] is employed, as it provides straightforward implementations for the attacks detailed in Section 2. Attack Success Rate (ASR) was considered as the metric for the experiments. In targeted attacks, this metric can be defined as the accuracy of the adversarial samples on the malicious labels. Besides, as the use case was related to device identification using performance-based metrics, the distance between benign and adversarial samples was irrelevant. Note that this metric would be important in other use cases, such as image recognition, where the adversarial and benign samples should not be distinguishable by a human.

The attacks selected to be tested are the ones explained in Section 2. However, the DeepFool attack is discarded as it is only untargeted. For this reason, a variant called NewtonFool [53] is used in this work. For each attack, an iteration in its main hyperparameters has been performed to find the most successful configuration (the one with a higher ASR). Table 7 shows the ASR results for each attack together with the execution time of the adversarial sample generation.

Different target devices were also tested with similar results to the reported in Table 7. Note that the exact results may vary if other devices were selected as the target, but the objective was to measure the model vulnerability to adversarial attacks. In this sense, FGSM, BIM, and MIM attacks show an ASR over 0.85. All these attacks achieve a +0.50 success in all the devices employed as adversaries. Therefore, these attacks would fully compromise an identification solution setting a threshold in the 0.50 TPR. In contrast, the NewtonFool attack cannot generate adversarial samples complex enough to target the selected class and only generates the misclassification of the data used as a base for crafting adversarial samples. This experiment demonstrated how the model was vulnerable to targeted adversarial evasion attacks, with over 0.85 ASR in some cases. These attacks could perform device spoofing if he/she has enough knowledge about the model or enough trial and error evaluations.

This experiment underscored the model susceptibility to targeted adversarial evasion attacks. With certain attacks achieving an ASR of over 0.85, it is evident that an adversary, equipped with sufficient knowledge about the model or through iterative evaluations, can successfully execute device spoofing. This revelation underscores the importance of fortifying identification models against such adversarial threats to ensure the security and integrity of device identification processes.

6. Defense techniques

This section analyzes how defense techniques can improve the model robustness against ML/DL evasion attacks. ART [52] was also used to implement the defense techniques for the model-focused attacks, as it includes several model-focused defense techniques. Note that this defense section focuses on device spoofing attacks because the defenses for context-based attacks were already applied during data collection as explained in the previous sections.

The first defense approach applied was to perform adversarial training, using crafted samples as part of the dataset used for model generation. In this sense, untargeted adversarial samples were generated using FGSM, PGD and BIM attacks and concatenated to the original training dataset. Then, a new model was trained from scratch using the new training dataset. Besides, defensive model distillation [37] was also applied over the baseline model to compare the robustness of each resulting model. Finally, the model trained using adversarial samples was also distilled. This combination had unstable behavior during model generation, requiring several attempts to avoid gradient explosion issues. Fig. 5 shows the results of each defense model when evaluating the legitimate test dataset (when no attack is present).

It can be seen how the main performance metrics were not degraded in an impactful manner. Only ≈0.02 performance decrease was noticed in accuracy and average precision, recall, and F1-Score. Besides, the minimum TPR was maintained at 0.8 for the adversarial training model and its distilled version. Once it was verified that the robustness techniques did not decrease the identification performance, the next step was to verify if the new models were robust against the evasion attacks. Table 8 shows the ASR for the different attacks when applied to each model (the baseline one and the ones including robustness techniques).

The combined approach of adversarial training and distillation consistently outperformed other techniques, registering the lowest ASR in seven out of the eight successful attacks. Most notably, the ASR for the most potent attacks on the baseline model, namely BIM, MIM, and PGD, witnessed a significant drop from around 0.85/0.88 to a range of 0.12/0.18. This reduction places the ASR below the 0.5 TPR threshold earmarked for device identification, highlighting the efficacy of the combined defense approach.

Table 9
Robustness evaluation results.

Metric	Baseline model	Distilled model	Adversarial training	Adversarial training + Distilled
CLEVER untargeted, *radius = 8, norm = 2*	*Avg.*:0.0056 *Dev.*:0.0052	*Avg.*:0.0058 *Dev.*:0.0059	*Avg.*:0.0045 *Dev.*:0.0033	*Avg.*:0.0052 *Dev.*:0.0043
CLEVER targeted, *radius = 8, norm = 2*	*Avg.*:0.0218 *Dev.*:0.0363	*Avg.*:0.0239 *Dev.*:0.0305	*Avg.*:0.0210 *Dev.*:0.0243	*Avg.*:0.0240 *Dev.*:0.0394
Loss sensitivity	5.1391	4.7920	6.8399	6.7756
Empirical robustness, *FGSM* $\epsilon = 0.05$	0.0616	0.0613	0.0648	0.0639

Avg.: Average, *Dev.*: Standard Deviation.

The combination of adversarial training and distillation has proven to be a potent defense mechanism against ML/DL evasion attacks. This research underscores the importance of continuously refining and enhancing defense techniques to stay ahead of evolving adversarial threats, ensuring the security and reliability of ML/DL models in real-world applications.

6.1. Robustness metrics

There also exist some additional metrics that evaluate how robust a model is by analyzing its parameters and outputs. Therefore, it is relevant to analyze the state-of-the-art metrics in this sense to quantify how the application of robustness techniques improved the model.

ART [52] includes the following metrics regarding model robustness: Cross Lipschitz Extreme Value for nEtwork Robustness (CLEVER) score [38], Loss sensitivity [40], and Empirical robustness [28]. Table 9 shows the values for these metrics using the test dataset as samples for evaluation. Note that CLEVER score is a metric calculated per sample. Therefore, the average and standard deviation are given in the table.

Although there is not a very great change on these metrics, and even the results for CLEVER untargeted are worse in the adversarial trained models than in the base model, it can be seen how in the case of CLEVER targeted, the score rises 10% from 0.0218 to ≈0.024 in both distilled models. Loss sensitivity score is increased from 5.1391 to 6.8399 and 6.7756 in the adversarial trained and adversarial trained +distilled models, respectively. Finally, Empirical robustness is slightly increased from 0.0616 to 0.0648 and 0.0639, a ≈5%.

While the improvements in robustness metrics might appear subtle, they are indicative of the potential benefits that robustness techniques can bring to the table. Especially in the realm of adversarial attacks, even marginal enhancements in robustness can be crucial in thwarting potential threats. This analysis underscores the importance of continuously refining and employing robustness techniques, ensuring that ML/DL models remain resilient in the face of evolving adversarial challenges.

7. Discussion

This section articulates the constraints inherent to the proposed solution and delivers key insights gleaned from the performed study. Based on the set of experiments conducted in this work and after the comparison with the literature, some important insights and conclusions can be extracted as lessons learned but also as limitations. The list of lessons learned is as follows:

- The use of raw data features provided better results for most of the classification models, as compared to sliding-window features, which contrasted with the previous works. This may occur because the LSTM-1DCNN model can use underlying correlations and information in the raw data to learn more accurate patterns. Another possible reason could be the use of a larger dataset that also includes information about memory and storage.
- Temperature conditions did not excessively impact the device identification performance. The average performance did not degrade when evaluating data generated under temperatures different from the ones during training. However, for some devices, it could lead to wrong identification if the TPR-based threshold is defined at a high value.
- The initial device identification model was found vulnerable to targeted adversarial evasion attacks, with over 0.85 ASR in some cases. These attacks could potentially compromise the identification solution by setting a threshold in the 0.50 TPR.
- The application of defense techniques, specifically adversarial training and model distillation, improved the robustness of the device identification model against ML/DL evasion attacks. The model combining adversarial training and distillation offered the best robustness against such attacks.

In contrast, the following limitations are observed and should be addressed in future research in the area:

- While the device identification model performed well under the context-based attack, some devices could still be more affected by temperature variations, leading to potential misidentification if a more impactful attack is performed. One way to address this limitation is to collect data from a wider range of temperatures during the training process.
- The model that combined adversarial training and distillation had unstable behavior during model generation, requiring several attempts to avoid gradient explosion issues. Such instability necessitated multiple attempts to generate a functioning model, significantly escalating the resources and time required in the model generation process. Furthermore, this instability could potentially result in the generation of less accurate or less generalized models.
- The degradation in performance on benign samples is a trade-off that must be considered when using robustness techniques. While adversarial training and model distillation improved the robustness of the model, the performance metrics were slightly degraded, with about a 0.02 performance decrease in accuracy, and average precision, recall, and F1-Score.
- The robustness techniques may not be effective against all types of evasion attacks. Untested attacks, such as GAN-based methods, could have a high ASR if full access to the identification model is available. Therefore, active iteration of the defense techniques is necessary as attack methods evolve.

8. Conclusions and future work

The explosion in IoT device deployment has motivated the development of new device identification solutions based on hardware behavior and ML/DL processing. However, these solutions face adversarial attacks that try to evade their functionality. This work explored the performance of hardware behavior-based device identification. For

that, the LwHBench dataset containing samples from 45 Raspberry Pi devices running identical software images was used to train ML/DL classifiers in charge of performing individual identification of each device. A DL model combining LSTM and 1D-CNN layers offered the best performance with an average F1-Score of 0.96, identifying all the devices by setting a threshold in +0.80 TPR. This model improved the performance of previous approaches in the literature. Afterward, a temperature-based attack and nine ML/DL evasion attacks were executed to measure the model performance degradation. In this case, the baseline model was robust against temperature context changes. However, some ML/DL evasion attacks successfully fooled the identification system, reaching up to 0.88 attack success rates and demonstrating its vulnerability to these attacks. Finally, model distillation and adversarial training defense techniques were applied during the model training, improving the model resilience to the ML/DL evasion attacks. These techniques improved the model robustness, being the combination of adversarial training and model distillation the best defense approach. Only a ≈0.02 decrease was noticed in accuracy, precision, recall, and F1-Score metrics, without a decrease in the minimum TPR, which is the metric used for setting the threshold for device identification.

In future work, more adversarial attack and defense techniques, such as the ones based on generative models, will be applied to fully improve the solution robustness. Besides, it is planned to add trust metrics in the individual device identification framework, in-depth evaluating the fairness and robustness of the predictions. Another research perspective to be tested is the fully distributed model generation, leveraging federated learning to avoid data sharing and centralization.

CRediT authorship contribution statement

Pedro Miguel Sánchez Sánchez: Methodology, Data curation, Software, Investigation, Formal analysis, Software, Writing – original draft. **Alberto Huertas Celdrán:** Conceptualization, Writing – original draft, Writing – review & editing, Resources. **Gérôme Bovet:** Writing – review & editing, Supervision, Funding acquisition. **Gregorio Martínez Pérez:** Writing – review & editing, Supervision, Project administration.

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

The data is published in a previous article and freely available to other researchers.

Acknowledgments

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

References

[1] J. Wang, M.K. Lim, C. Wang, M.-L. Tseng, The evolution of the Internet of Things (IoT) over the past 20 years, Comput. Ind. Eng. 155 (2021) 107174.
[2] K. Shafique, B.A. Khawaja, F. Sabir, S. Qazi, M. Mustaqim, Internet of things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios, IEEE Access 8 (2020) 23022–23040.
[3] A.H. Celdrán, J. von der Assen, K. Moser, P.M.S. Sánchez, G. Bovet, G.M. Pérez, B. Stiller, Early detection of cryptojacker malicious behaviors on IoT crowdsensing devices, in: NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, 2023, pp. 1–8.
[4] A. Srivastava, S. Gupta, M. Quamara, P. Chaudhary, V.J. Aski, Future IoT-enabled threats and vulnerabilities: State of the art, challenges, and future prospects, Int. J. Commun. Syst. 33 (12) (2020) e4443.
[5] P.M. Sánchez Sánchez, J.M. Jorquera Valero, A. Huertas Celdrán, G. Bovet, M. Gil Pérez, G. Martínez Pérez, A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1048–1077.
[6] Y. Meidan, M. Bohadana, A. Shabtai, J.D. Guarnizo, M. Ochoa, N.O. Tippenhauer, Y. Elovici, ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis, in: Proceedings of the Symposium on Applied Computing, 2017, pp. 506–509.
[7] P.M.S. Sánchez, J.M.J. Valero, A.H. Celdrán, G. Bovet, M.G. Pérez, G.M. Pérez, A methodology to identify identical single-board computers based on hardware behavior fingerprinting, J. Netw. Comput. Appl. 212 (2023) 103579.
[8] T.J. Salo, Multi-factor fingerprints for personal computer hardware, in: MILCOM 2007-IEEE Military Communications Conference, 2007, pp. 1–7.
[9] Y. Liu, J. Wang, J. Li, S. Niu, H. Song, Machine learning for the detection and identification of internet of things devices: A survey, IEEE Internet Things J. 9 (1) (2021) 298–320.
[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, 2013, arXiv preprint arXiv:1312.6199.
[11] K. Sadeghi, A. Banerjee, S.K. Gupta, A system-driven taxonomy of attacks and defenses in adversarial machine learning, IEEE Trans. Emerg. Top. Comput. Intell. 4 (4) (2020) 450–467.
[12] F. Suya, S. Mahloujifar, A. Suri, D. Evans, Y. Tian, Model-targeted poisoning attacks with provable convergence, in: International Conference on Machine Learning, PMLR, 2021, pp. 10000–10010.
[13] H. Kwon, Y. Kim, K.-W. Park, H. Yoon, D. Choi, Multi-targeted adversarial example in evasion attack on deep neural network, IEEE Access 6 (2018) 46084–46096.
[14] M. Al-Rubaie, J.M. Chang, Privacy-preserving machine learning: Threats and solutions, IEEE Secur. Priv. 17 (2) (2019) 49–58.
[15] O. Ibitoye, R. Abou-Khamis, A. Matrawy, M.O. Shafiq, The threat of adversarial attacks on machine learning in network security–a survey, 2019, arXiv preprint arXiv:1911.02621.
[16] T. Laor, N. Mehanna, A. Durey, V. Dyadyuk, P. Laperdrix, C. Maurice, Y. Oren, R. Rouvoy, W. Rudametkin, Y. Yarom, Drawnapart: A device identification technique based on remote GPU fingerprinting, 2022, arXiv preprint arXiv:2201.09956.
[17] Z. Bao, Y. Lin, S. Zhang, Z. Li, S. Mao, Threat of adversarial attacks on DL-based IoT device identification, IEEE Internet Things J. 9 (11) (2021) 9012–9024.
[18] A. Namvar, C. Thapa, S.S. Kanhere, S. Camtepe, Evaluating the security of machine learning based IoT device identification systems against adversarial examples, in: International Conference on Service-Oriented Computing, Springer, 2021, pp. 800–810.
[19] P.M.S. Sánchez, J.M.J. Valero, A.H. Celdrán, G. Bovet, M.G. Pérez, G.M. Pérez, LwHBench: A low-level hardware component benchmark and dataset for Single Board Computers, Internet Things 22 (2023) 100764.
[20] P.M. Sánchez Sána.namvar@student.unsw.edu.auchez, Adversarial_identification, 2023, https://github.com/sxz0/Adversarial_Identification. [Online; accessed 13-July-2023].
[21] I. Sanchez-Rola, I. Santos, D. Balzarotti, Clock around the clock: Time-based device fingerprinting, in: 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 1502–1514.
[22] S.D. Paul, F. Zhang, P. SLPSK, A.R. Trivedi, S. Bhunia, RIHANN: Remote IoT hardware authentication with intrinsic identifiers, IEEE Internet Things J. 9 (24) (2022) 24615–24627.
[23] A. Shamsoshoara, A. Korenda, F. Afghah, S. Zeadally, A survey on physical unclonable function (PUF)-based security solutions for Internet of Things, Comput. Netw. 183 (2020) 107593.
[24] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, 2014, arXiv preprint arXiv:1412.6572.
[25] A. Kurakin, I.J. Goodfellow, S. Bengio, Adversarial examples in the physical world, in: Artificial Intelligence Safety and Security, Chapman and Hall/CRC, 2018, pp. 99–112.
[26] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, J. Li, Boosting adversarial attacks with momentum, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9185–9193.
[27] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, 2017, arXiv preprint arXiv:1706.06083.
[28] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, Deepfool: a simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2574–2582.
[29] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, A. Swami, The limitations of deep learning in adversarial settings, in: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2016, pp. 372–387.
[30] K.J. Waldron, S.-L. Wang, S.J. Bolin, A study of the Jacobian matrix of serial manipulators, J. Mech. Transm. Autom. Des. 107 (2) (1985) 230–237.

[31] W. Brendel, J. Rauber, M. Bethge, Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, 2017, arXiv preprint arXiv:1712.04248.

[32] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: 2017 Ieee Symposium on Security and Privacy (Sp), Ieee, 2017, pp. 39–57.

[33] W. Hu, Y. Tan, Generating adversarial malware examples for black-box attacks based on GAN, 2017, arXiv preprint arXiv:1702.05983.

[34] I. Rosenberg, A. Shabtai, Y. Elovici, L. Rokach, Adversarial machine learning attacks and defense methods in the cyber security domain, ACM Comput. Surv. 54 (5) (2021) 1–36.

[35] E. Wong, L. Rice, J.Z. Kolter, Fast is better than free: Revisiting adversarial training, 2020, arXiv preprint arXiv:2001.03994.

[36] G. Hinton, O. Vinyals, J. Dean, et al., Distilling the knowledge in a neural network, 2015, arXiv preprint arXiv:1503.02531, 2(7).

[37] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in: 2016 IEEE Symposium on Security and Privacy (SP), IEEE, 2016, pp. 582–597.

[38] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, L. Daniel, Evaluating the robustness of neural networks: An extreme value theory approach, 2018, arXiv preprint arXiv:1801.10578.

[39] G. Wood, B. Zhang, Estimation of the Lipschitz constant of a function, J. Global Optim. 8 (1996) 91–103.

[40] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M.S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, et al., A closer look at memorization in deep networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 233–242.

[41] F. Yu, et al., Interpreting and evaluating neural network robustness, in: 2019 International Joint Conferences on Artificial Intelligence, 2019, pp. 4199–4205.

[42] C. Benegui, R.T. Ionescu, Adversarial attacks on deep learning systems for user identification based on motion sensors, in: International Conference on Neural Information Processing, Springer, 2020, pp. 752–761.

[43] N. Pourshahrokhi, M. Smith-Creasey, M. Ghassemian, S. Kouchaki, Generative adversarial attacks on motion-based continuous authentication schemes, in: 2021 14th International Conference on Security of Information and Networks (SIN), Vol. 1, IEEE, 2021, pp. 1–6.

[44] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, M. Colajanni, Modeling realistic adversarial attacks against network intrusion detection systems, Digit. Threats: Res. Pract. (DTRAP) 3 (3) (2022) 1–19.

[45] C. Rossow, C.J. Dietrich, C. Grier, C. Kreibich, V. Paxson, N. Pohlmann, H. Bos, M. Van Steen, Prudent practices for designing malware experiments: Status quo and outlook, in: 2012 IEEE Symposium on Security and Privacy, IEEE, 2012, pp. 65–79.

[46] Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: LSTM cells and network architectures, Neural Comput. 31 (7) (2019) 1235–1270.

[47] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, D.J. Inman, 1D convolutional neural networks and applications: A survey, Mech. Syst. Signal Process. 151 (2021) 107398.

[48] K. Xia, J. Huang, H. Wang, LSTM-CNN architecture for human activity recognition, IEEE Access 8 (2020) 56855–56866.

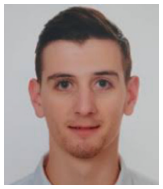
[49] Z. He, J. Zhou, H.-N. Dai, H. Wang, Gold price forecast based on LSTM-CNN model, in: 2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), IEEE, 2019, pp. 1046–1053.

[50] Y. Zhang, S. Qiao, S. Ji, Y. Li, DeepSite: bidirectional LSTM and CNN models for predicting DNA–protein binding, Int. J. Mach. Learn. Cybern. 11 (4) (2020) 841–851.

[51] M.M. Ahsan, M.P. Mahmud, P.K. Saha, K.D. Gupta, Z. Siddique, Effect of data scaling methods on machine learning algorithms and model performance, Technologies 9 (3) (2021) 52.

[52] M.-I. Nicolae, M. Sinn, M.N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, B. Edwards, Adversarial robustness toolbox v1.2.0, 2018, CoRR, 1807.01069.

[53] U. Jang, X. Wu, S. Jha, Objective metrics and gradient descent algorithms for adversarial examples in machine learning, in: Proceedings of the 33rd Annual Computer Security Applications Conference, 2017, pp. 262–277.



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.

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.

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.

6

Single-board Device Individual Authentication based on Hardware Performance and Autoencoder Transformer Models

Title:	Single-board Device Individual Authentication based on Hardware Performance and Autoencoder Transformer Models
Authors:	Pedro Miguel Sánchez Sánchez, Alberto Huertas Celdrán, Gérôme Bovet, Gregorio Martínez Pérez
Journal:	Computers & Security
JIF:	5.6 Q2 (2022)
Publisher:	Elsevier
Volume:	137
Number:	
Pages:	103596
Year:	2024
Month:	February
DOI:	10.1016/j.cose.2023.103596
Status:	Published

Abstract

The proliferation of the Internet of Things (IoT) has led to the emergence of crowdsensing applications, where a multitude of interconnected devices collaboratively collect and analyze data. Ensuring the authenticity and integrity of the data collected by these devices is crucial for reliable decision-making and maintaining trust in the system. Traditional authentication methods are often vulnerable to attacks or can be easily duplicated, posing challenges to securing crowdsensing applications. Besides, current solutions leveraging device behavior are mostly focused on device identification, which is a simpler task than authentication. To address these issues, an individual IoT device authentication framework based on hardware behavior fingerprinting and Transformer autoencoders is proposed in this work. To support the design, a threat model details the security problems faced when performing hardware-based authentication in IoT. This solution leverages the inherent imperfections and variations in IoT device hardware to differentiate between devices with identical specifications. By monitoring and analyzing the behavior of key hardware components, such as the CPU, GPU, RAM, and Storage on devices, unique fingerprints for each device are created. The performance samples are considered as time series data and

used to train outlier detection transformer models, one per device and aiming to model its normal data distribution. Then, the framework is validated within a spectrum crowdsensing system leveraging Raspberry Pi devices. After a pool of experiments, the model from each device is able to individually authenticate it between the 45 devices employed for validation. An average True Positive Rate (TPR) of 0.74 ± 0.13 and an average maximum False Positive Rate (FPR) of 0.06 ± 0.09 demonstrate the effectiveness of this approach in enhancing authentication, security, and trust in crowdsensing applications.

Keywords

Device Behavior Fingerprinting · Device Authentication · Transformer · Behavioral Data · Hardware Fingerprinting · Autoencoder

Computers & Security 137 (2024) 103596

Contents lists available at ScienceDirect

Computers & Security

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

Single-board device individual authentication based on hardware performance and autoencoder transformer models

Pedro Miguel Sánchez Sánchez [a,*], Alberto Huertas Celdrán [b], Gérôme Bovet [c], 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 authentication
Transformer
Behavioral data
Hardware fingerprinting
Autoencoder

ABSTRACT

The proliferation of the Internet of Things (IoT) has led to the emergence of crowdsensing applications, where a multitude of interconnected devices collaboratively collect and analyze data. Ensuring the authenticity and integrity of the data collected by these devices is crucial for reliable decision-making and maintaining trust in the system. Traditional authentication methods are often vulnerable to attacks or can be easily duplicated, posing challenges to securing crowdsensing applications. Besides, current solutions leveraging device behavior are mostly focused on device identification, which is a simpler task than authentication. To address these issues, an individual IoT device authentication framework based on hardware behavior fingerprinting and Transformer autoencoders is proposed in this work. To support the design, a threat model details the security problems faced when performing hardware-based authentication in IoT. This solution leverages the inherent imperfections and variations in IoT device hardware to differentiate between devices with identical specifications. By monitoring and analyzing the behavior of key hardware components, such as the CPU, GPU, RAM, and Storage on devices, unique fingerprints for each device are created. The performance samples are considered as time series data and used to train outlier detection transformer models, one per device and aiming to model its normal data distribution. Then, the framework is validated within a spectrum crowdsensing system leveraging Raspberry Pi devices. After a pool of experiments, the model from each device is able to individually authenticate it between the 45 devices employed for validation. An average True Positive Rate (TPR) of 0.74±0.13 and an average maximum False Positive Rate (FPR) of 0.06±0.09 demonstrate the effectiveness of this approach in enhancing authentication, security, and trust in crowdsensing applications.

1. Introduction

The widespread adoption of the Internet of Things (IoT) has led to the emergence of crowdsensing applications, where many IoT devices collaboratively gather and analyze data from the environment (Rajendran et al., 2018). Many of these applications rely on single-board computers due to their reduced price and relatively good performance. These applications offer tremendous potential in diverse domains, such as environmental monitoring, urban planning, healthcare, and transportation. However, ensuring the authenticity and integrity of the data collected by these devices is critical for reliable decision-making and maintaining trust in the system (Capponi et al., 2019).

The openness and distributed nature of crowdsensing systems make them susceptible to Sybil attacks and collusion among malicious entities (Yu, 2020). Sybil attacks involve adversaries creating multiple fake identities to gain control over the system or manipulate the collected data. Collusion among malicious entities can also lead to coordinated attacks or data manipulation. Implementing identity verification mechanisms, reputation systems, and distributed consensus algorithms is required in order to prevent and detect such attacks (Zhong et al., 2019).

Traditional authentication methods for IoT devices, such as cryptographic protocols or unique identifiers, are often susceptible to various attacks and vulnerabilities (Wang et al., 2020). Moreover, devices with identical specifications can be easily duplicated or impersonated, posing

* Corresponding author.
E-mail addresses: pedromiguel.sanchez@um.es (P.M. Sánchez Sánchez), huertas@ifi.uzh.ch (A. Huertas Celdrán), gerome.bovet@armasuisse.ch (G. Bovet), gregorio@um.es (G.M. Martínez Pérez).

https://doi.org/10.1016/j.cose.2023.103596
Received 28 August 2023; Received in revised form 19 October 2023; Accepted 10 November 2023
Available online 17 November 2023

a significant challenge to maintaining trust and security in crowdsensing applications. To address these limitations, novel approaches are required that leverage the unique characteristics of IoT devices to establish their authenticity. These methods can be seen as an additional layer in the authentication security of IoT scenarios.

One of the directions proposed in the literature to solve these issues is leveraging hardware manufacturing imperfections in order to uniquely identify each device in the environment (Sánchez et al., 2021). What elevates the efficiency of this approach is the integration of Machine Learning (ML) and Deep Learning (DL) techniques for the processing of collected hardware behavior data. These cutting-edge computational methodologies facilitate the analysis, classification, and prediction of the enormous amounts of complex, high-dimensional data generated by IoT devices (Al-Garadi et al., 2020). Particularly, they can adeptly capture patterns and dependencies in this data, enabling effective anomaly detection and thereby facilitating the identification of devices or activities that deviate from established norms. The combination of hardware manufacturing imperfections and ML/DL techniques has been evidenced to provide remarkable results in the context of device identification (Sanchez-Rola et al., 2018; Sánchez et al., 2023b). However, authentication poses a more complex issue: discerning whether a device is authentic or not, but without taking into account the data distributions of other devices.

Therefore, there are still many challenges present related to hardware-based individual authentication leveraging ML/DL techniques: (*i*) most of the solutions available in the literature cover device identification and not in authentication (Sánchez et al., 2023c), trying to differentiate a device between a set of known devices instead of uniquely verify its identity; (*ii*) novel DL methods such as attention Transformers have not been applied yet in this field (Sánchez et al., 2022), but could improve current results as it is happening in other fields; (*iii*) solutions are usually implemented in simulated or isolated environments, and not integrated into real-world applications (Zhang et al., 2019); (*iv*) most of the solutions relying on ML/DL follow a classification-based approach as they focus on identification, which is not practical in dynamic scenarios or when the number of devices is high (Al-Naji and Zagrouba, 2022).

To solve the previous challenges, the main contributions of the present work are:

- A framework that leverages Transformer-based autoencoder models and hardware performance fingerprinting for the individual authentication of single-board computer devices. This framework leverages CPU, GPU, RAM and Storage components to measure their performance and find manufacturing variations that enable the differentiation between devices based on their performance. In this sense, the data from the legitimate device are taken as normal samples modeling its performance distribution, while samples from other devices should be detected as outliers or anomalies.
- The deployment of the framework in a real-world spectrum crowdsensing platform based on Raspberry Pi devices, namely ElectroSense. In total, 45 devices are utilized in the scenario: 15 Raspberry Pi 4, 10 Raspberry Pi 3, 10 Raspberry Pi 1, and 10 Raspberry Pi Zero. This deployment demonstrates the practical applicability of the framework and its compatibility with different versions of Raspberry Pi devices. It also provides valuable insights into the real-world challenges and considerations in implementing such a sophisticated authentication system, contributing to the broader field of IoT security.
- The validation of the framework authentication performance in the deployed scenario. After data collection, an average True Positive Rate (TPR) of 0.74±0.13 and an average maximum False Positive Rate (FPR) of 0.06±0.09 are achieved, improving other state-of-the-art models such as LSTM and 1D-CNN networks. This validation not only confirms the effectiveness of the proposed framework but also sets a new benchmark in the field. Besides, a second validation approach details how the solution can be adapted to new device models with different hardware components. The detailed analysis and comparison with other models provide a comprehensive understanding of the strengths and potential areas for further optimization, paving the way for future research and development in hardware-based authentication. The validation code for the performed experiments is available at Sánchez Sánchez (2023).

The remainder of this article is structured as follows. Section 2 gives an overview of hardware-based individual authentication and background on transformer usage for anomaly detection. Section 3 explains the threat model faced by the proposed solution. Section 4 describes the Transformer and hardware-based device fingerprinting solution for individual authentication of single-board devices. Section 5 gives an overview of the crowdsensing platform employed for validation, the data collection process, and the experimental results when performing the authentication. Finally, Section 7 gives an overview of the conclusions extracted from the present work and future research directions.

2. Related work

This section reviews the key literature relevant on individual device authentication through hardware performance fingerprinting and transformer-based anomaly detection.

2.1. Individual device authentication and identification

The present work focuses on hardware-based single-board device authentication using the performance behavior of the components self-contained in the device and anomaly detection DL algorithms. Arafin and Qu (2021) discussed several examples of hardware-based authentication that use memory access latency, instruction execution latency, and clock skew to authenticate devices, users, and broadcast signals used for navigation. In Sánchez et al. (2023b), the authors compared the deviation between the CPU and GPU cycle counters in Raspberry Pi devices to perform individual identification of 25 devices. The identification was performed using XGBoost, achieving a 91.92% True Positive Rate (TPR). In continuing work (Sánchez et al., 2022), the same authors improved the results to an average F1-Score of +0.96 and a minimum TPR of 0.8 using a time series classification approach based on LSTM and 1D-CNN combination. Similarly, (Laor et al., 2022) performed identical device identification using GPU performance behavior and ML/DL classification algorithms. Accuracy between 95.8% and 32.7% was achieved in nine sets of identical devices, including computers and mobile devices. Sanchez-Rola et al. (2018) identified +260 identical computers by measuring the differences in code execution performance. They employed the Real-Time Clock (RTC), which includes its own physical oscillator, to find slight variations in the performance of each CPU. In Salo (2007), the author compared the drift between the CPU time counter, the RTC chip, and the sound card Digital Signal Processor (DSP) to identify identical computers. Other works have also explored hardware-based authentication applications using physical properties of computing hardware such as main memory, computing units, and clocks. Shrivastava et al. (2022) proposed a high-performance Field Programmable Gate Arrays (FPGA) based secured hardware model for IoT devices using the Advanced Encryption Standard (AES) algorithm. They compared the performance of two FPGAs and found that the Spartan-6 FPGA provides better throughput and less time delay for IoT devices.

Other works have explored the usage of Physical Unclonable Functions (PUFs) for IoT device identification (Shamsoshoara et al., 2020). However, PUFs are out of the scope of this work, as it is centered on hardware behavior fingerprinting based on device performance, avoiding the usage of new hardware elements or the modification of the device specifications.

Finally, some solutions are also available in the industry, leveraging hardware characteristics for IoT device identification. Numerous

Table 1
Comparison of the closest works on ML/DL-focused hardware-based device identification and authentication.

Work	Scenario	Approach	Algorithm/Model	N Devices	Results
(Salo, 2007)	Computer identification	Statistical correlation	Pair-based identification	38	Computer identification based on the comparison of three physical oscillators using t-test statistic
(Sanchez-Rola et al., 2018)	Computer identification	Statistical correlation	Mode-based statistics	265	All computers uniquely identified. No effect from CPU load and temperature
(Laor et al., 2022)	Computer and mobile identification	Classification	CNN	9	95.8% and 32.7% accuracy in nine sets of identical devices. Accuracy drops with device rebooting
(Sánchez et al., 2023b)	IoT device identification	Classification	XGBoost	25	91.92% average TPR. No effects from temperature changes and device rebooting
(Sánchez et al., 2022)	IoT device identification	Classification	LSTM + 1D-CNN	45	0.96 average F1-Score. Resilience to temperature and ML/DL evasion attacks.
This work (2023)	IoT device authentication	Anomaly Detection	Transformer	45	All devices authenticated. 0.74 average TPR and 0.06 average maximum FPR

hardware-based authentication solutions for IoT devices have been introduced to enhance security. Intel Enhanced Privacy ID (EPID) provides a mechanism for device authentication while ensuring privacy, making certain that devices connecting to networks are genuine Intel products (Intel, 2021). ARM TrustZone technology partitions devices into secure and non-secure zones, offering a foundational layer for security solutions (ARM, 2021). Cisco Trust Anchor module (TAm) embeds a hardware module in products to guarantee device integrity and authenticity right from manufacturing to deployment (Cisco, 2021). Microsoft has ventured into this space with Azure Sphere, which incorporates custom silicon security technology for comprehensive IoT security (Microsoft, 2021). NXP A71CH is a secure element designed to provide a root of trust at the integrated circuit level for IoT devices (NXP, 2021). Infineon OPTIGA Trusted Platform Module (TPM) offers hardware-based security functions, facilitating device authentication (Infineon, 2021). Microchip CryptoAuthentication devices are tailored to protect against various security threats by offering robust cryptographic solutions (Microchip, 2021). Rambus CryptoManager IoT Device Management is a turnkey solution designed to provide end-to-end security, including device attestation and hardware-based security (Rambus, 2021). Lastly, GlobalPlatform Device Trust Architecture (DTA) standardizes the use of secure components in IoT devices to protect digital services and data (GlobalPlatform, 2021). While hardware-based industrial authentication solutions for IoT devices bolster security, they come with challenges. These include higher costs, increased deployment complexity, computational overhead on devices, reduced flexibility for updates, scalability concerns in vast networks, vulnerabilities to physical attacks, supply chain integrity issues, interoperability problems among different manufacturers, potential long-term hardware degradation affecting performance, and the risk of vendor lock-in due to proprietary solutions.

Table 1 compares the closest works in the literature with the present one. Although several works have worked in the combination of ML/DL techniques and hardware fingerprinting for device identification, a notable gap persists in the literature with respect to addressing the unique challenges of device authentication via an anomaly detection approach. Contemporary studies have primarily employed classification models, which serve to identify devices from a set pool of labels. However, these models are inadequate for the authentication problem. The task of authentication involves more than simple device recognition - it requires a system capable of detecting deviations from an expected hardware behavior, a task for which anomaly detection models, rather than traditional classification models, are better suited. Consequently, there is a significant need to investigate the potential of DL-based anomaly detection models, such as Transformer models, in the realm of device authentication.

2.2. Transformer-based anomaly detection in IoT security

The application of Transformer models in anomaly detection has recently gained momentum, recognizing their ability to extract meaningful features from sequential data effectively. Anomaly detection in time-series data, in particular, has seen significant advancements through the adoption of Transformer models (Choi et al., 2021). Their proficiency in capturing temporal dynamics makes them an excellent choice for tasks that involve detecting irregularities in time-bound sequences (Tuli et al., 2022).

In the field of IoT security, Transformer-based autoencoders have been employed to address high-dimensional and complex dependencies issues by leveraging the self-attention mechanism and the encoder-decoder architecture. Chen et al. (2021) proposed a framework called GTA that learns a graph structure among sensors and applies graph convolution and Transformer-based modeling to detect anomalies in multivariate time series. Kozik et al. (2021) proposed a hybrid time window embedding method with a Transformer-based classifier to identify compromised devices in IoT-networked environment. Tuli et al. (2022) proposed TranAD, a deep Transformer network that uses attention-based sequence encoders to perform anomaly detection and diagnosis for IoT data streams. These works demonstrate the effectiveness and efficiency of Transformer-based models for anomaly detection in IoT security.

However, the performance of Transformer-based anomaly detection in individual device authentication has not been explored yet, remaining as a practical field where the performance of these novel models can improve the state-of-the-art approaches.

3. Threat model

The primary concern in the single-board device authentication scenario is an adversarial actor attempting to integrate an unauthorized device into a sensitive setting, like an industry, by masquerading as or impersonating a legitimate device. This threat can be approached from multiple angles:

- *TH1. Device impersonation* (Marabissi et al., 2022). The foremost security challenge is when an adversarial entity substitutes a genuine device with a malicious device that mirrors its software characteristics. In this case, the adversary deploys identical legitimate software credentials but incorporates malevolent processes and features.
- *TH2. Sybil* (Rajan et al., 2017). A singular device (or multiple) might attempt to create numerous authentications to transmit deceptive data from many mimicked devices. The vulnerability of a system to Sybil attacks hinges on (i) the simplicity of creating authentications; (ii) whether the system uniformly handles all entities, and (iii) the extent to which the system approves of entities lacking a trust linkage to a recognized trustworthy entity.

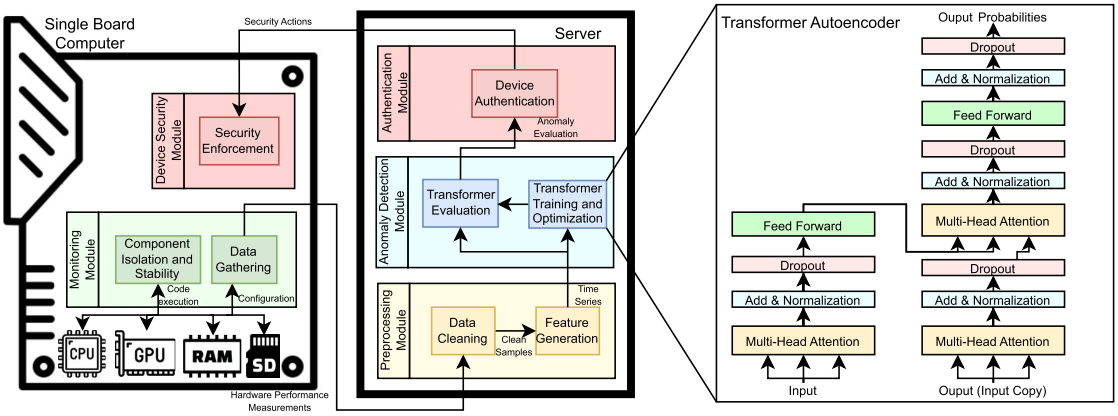


Fig. 1. Individual device authentication framework. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

- *TH3. Replay Attacks* (Feng et al., 2017). Attackers can capture authentication tokens or messages and replay them later to gain unauthorized access or to disrupt the network operations.
- *TH4. Physical Attacks* (Stellios et al., 2021). Many IoT devices are deployed in environments that lack stringent security, leaving them vulnerable to physical tampering. Malicious actors can directly access these devices to extract sensitive cryptographic keys or implant malicious hardware/software components. The direct physical access grants attackers a high level of control, making these attacks particularly devastating.
- *TH5. Advanced persistent threat* (Chen et al., 2022). This threat emerges as an outcome of the preceding one. A rogue device set up in the environment may be capable of extracting data from the situation and other devices or initiating more aggressive assaults like vulnerability scanning and/or Denial of Service (DoS) attacks. Additionally, contemporary attacks typically incorporate evasion methods that conceal their operations from software-centric behavior observation security mechanisms (Li and Li, 2020).

Traditional software-based authentication methods, while effective in some scenarios, have shown vulnerabilities in the face of sophisticated threat models where certificates or software identifiers can be cloned. Therefore, solving the previous threat model is the main objective of the proposed solution, complementing traditional authentication systems based on software. By capitalizing on inherent cycle skew and performance disparities in hardware -even among identical IoT devices- this approach can establish a unique, tamper-resistant identity for each device. These intrinsic hardware traits offer not just a shield against software-based incursions but also a robust defense against physical intrusions. Additionally, by folding hardware performance metrics into the authentication matrix, the solution can seamlessly cater to the diverse performance spectra of IoT devices, facilitating efficient authentication processes, even for those with resource constraints.

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 authentication tasks can be executed. This condition guarantees that device management is maintained during a possible attack. If this control is lost, it would be assumed that the device is infected or has some error.

4. Individual device authentication framework

This section elucidates the DL framework implemented for the purpose of hardware performance fingerprinting. The framework performs device fingerprinting based on performance deviations that show hardware manufacturing imperfections. An autoencoder Transformer model, a state-of-the-art approach in DL-based time series processing, is leveraged for the authentication of individual devices.

The framework is designed in a modular manner, where different components are combined in a stacked layout, from the hardware behavior monitoring to the DL-based evaluation and authentication. Due to the reduced processing capabilities of single-board computers, the framework follows a client-server architecture, where the components related to data collection and device configuration are deployed locally in the device, and the server processes the data and performs the model training and evaluation. Fig. 1 illustrates the different modules composing the framework and the pipeline followed by the data until an authentication decision is made. Five modules compose the framework: *(i)* Monitoring, *(ii)* Preprocessing, *(iii)* Anomaly Detection, *(iv)* Authentication, and *(v)* Device Security.

4.1. Monitoring module

The *Monitoring Module* is in charge of the interaction with the hardware components and the monitoring of their performance. Besides, it sends the collected data to the server for its processing and evaluation. It contains two components: *Component Isolation and Stability* and *Data Gathering*.

4.1.1. Component isolation and stability

One of the key conditions to perform fingerprinting based on hardware performance is to ensure that the components selected for monitoring are running under stable conditions that enable the characterization of the small performance variations in the components due to manufacturing imperfections (Sánchez et al., 2023b). Therefore, this component is in charge of configuring the CPU, GPU, RAM and SD Card, the selected hardware components. It sets fixed running frequency for the components, isolate the components to avoid kernel interruptions, and disables some component optimizations that might affect the stability of the performance, such as memory address randomization.

4.1.2. Data gathering

This component is in charge of collecting the performance measurements by executing different tasks in the selected hardware components. In the case of single-board computers, the available hardware elements are the CPU, GPU, RAM and storage (typically SD card). As proposed in the literature (Sánchez et al., 2023b), the hardware monitoring is done by using the in-device elements as a reference for the performance measurements. For example, GPU performance is measured in CPU cycles, and CPU performance when executing a code is

measured using the elapsed GPU cycles. The reasoning for this approach is that the component itself is not able to measure the deviations in its performance specification without an external cycle or time counter.

4.2. Preprocessing module

The *Preprocessing Module* plays the pivotal role of a bridge between the raw data gathered by the *Monitoring Module* and the *Anomaly Detection Module*, where the data is employed to train the DL models and evaluate the device. The main tasks of this module encompass data cleaning and feature generation.

4.2.1. Data cleaning

This component is responsible for filtering and cleaning the raw performance metrics. Any missing, inconsistent, or erroneous data are identified and filtered, thus preparing the dataset for further processing.

4.2.2. Feature generation

This component focuses on feature extraction and engineering based on the cleaned data. First, it performs normalization of each one of the metrics gathered. Afterward, it is in charge of transforming the raw data into a format suitable for the Transformer model. A key aspect of this process is the concatenation of samples into groups of vectors, which facilitates time series-based analysis.

4.3. Anomaly detection module

The *Anomaly Detection Module* is the heart of the authentication framework, tasked with training and evaluating the Transformer-based autoencoder model. The Transformer-based autoencoder is a variant of the Transformer model, which was originally proposed for natural language processing tasks. The key component of the Transformer architecture is the self-attention mechanism, which models the interactions between the elements in the input sequence (Vaswani et al., 2017). More in detail, the self-attention mechanism computes a weighted sum of the input elements for each position in the sequence. The weight assigned to each input element is determined by its relevance to the position being considered. Formally, the self-attention can be computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where Q, K, and V are matrices representing the queries, keys, and values, respectively, and d_k is the dimensionality of the keys. In multi-head attention, this operation is done h times with different learned linear projections of the original Q, K, and V matrices.

In the autoencoder variant of the Transformer model, the same sequence is provided as both the input and the target output of the model. The Transformer-based autoencoder learns to reconstruct the input sequence, which allows it to capture the underlying structure of the sequence data.

The encoder and decoder are both composed of several identical layers. Each layer contains two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network, using ReLU as activation function. The output of each sub-layer is then passed through a residual connection and layer normalization.

In the context of device authentication, the Transformer-based autoencoder is trained to reconstruct the normal behavior of each device. Once the model is trained, it can be used to detect anomalies by comparing the reconstruction error of a new sequence with a predefined threshold. A high reconstruction error indicates that the new sequence is significantly different from the normal behavior, which could suggest a possible intrusion.

The two components forming this module, in charge of the Transformer-based autoencoder training for each device, are:

4.3.1. Transformer training and optimization

This component takes the processed data and trains a Transformer model for each device. This model, adept at reconstructing input data, establishes a profile of standard device behavior, thereby becoming proficient at detecting anomalies or deviations from the norm. This phase also involves the optimization of model parameters for each device independently to ensure the best performance. Then, the best model for each device is stored to be later used. The training and optimization process is iterative and may require several exploratory iterations to find the combination that meets all the properties needed in the generated fingerprint.

4.3.2. Transformer evaluation

Upon completion of the training phase, the model is subject to deployment for live data evaluation. The model predictive capability is tested against the values collected from the device after deployment. Then, the output of the Transformer will be employed in the *Authentication Module* to determine if a device is the legitimate one and grant allow him to remain deployed in the network. Any deviations from the established profile may trigger further investigation or immediate action, depending on the *Device Security Module*.

4.4. Authentication module

The *Authentication Module* makes the final decision regarding device authentication based on the evaluation results coming from the previous module. It integrates the anomaly detection results with other contextual information, such as device history or network behavior, to make a more informed decision. This module may also include additional verification steps or multi-factor authentication to enhance security.

4.4.1. Device authentication

This component is charged with the essential task of making the final authentication decision based on the anomaly detection results. Anomalies, interpreted as potential indications of device tampering or misuse, inform the authentication decision. A device may be authenticated and granted network access, or it may be rejected, depending on the analysis of these anomalies.

4.5. Device security module

The *Device Security Module* serves as an additional layer of security, overseeing the enforcement of security measures. It works in conjunction with the *Authentication Module* to provide a comprehensive security solution for IoT devices after authentication.

4.5.1. Security enforcement

This component ensures the enforcement of necessary security rules or protocols based on the *Authentication Module* decision. If a device is authenticated, it is granted access to the network. If a device is deemed unauthenticated, this component ensures the device is isolated from the network, safeguarding the integrity of the IoT system. This module also reports any security issues, such as repeated authentication failures, to a central authority for further investigation. Moving target defense (MTD) techniques are a suitable approach for this module, as they focus on changing the device configuration according to the mitigation actions required. Some examples of these techniques is the removal of files, dynamic network connection filtering, among others.

5. Framework validation

This section succinctly lays out the overall validation methodology, from leveraging the ElectroSense spectrum crowdsensing platform to data collection and preprocessing crucial for the analysis. The specifics of data gathering and the processes of cleaning, normalization, and

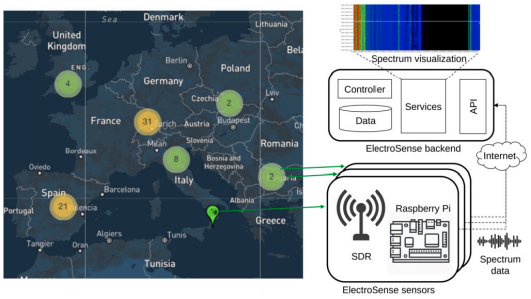


Fig. 2. ElectroSense crowdsensing platform diagram.

transformation are explained. Finally, the Transformer-based Anomaly Detection model approach is validated in this real-world scenario, measuring its effectiveness. Note that the validation focuses on the data collection, monitoring, and DL parts of the framework. The development of advanced authentication rules and security measures is out of the scope of this work.

5.1. ElectroSense spectrum crowdsensing platform

The IoT spectrum sensors utilized in this research are a part of the ElectroSense network (Rajendran et al., 2018), an open-source, crowdsensing platform that collects radio frequency spectrum data with the aid of low-cost sensors. The platform, which capitalizes on a collaborative crowdsensing approach, enables the monitoring and collection of spectrum data. The core of this platform is the Raspberry Pi, a compact and cost-effective single-board computer, that when attached to software-defined radio kits and antennas can function as a versatile spectrum sensor. Such assembly of spectrum sensors by individual users contributes to the broad reach and comprehensive data collection capability of the ElectroSense platform.

Once the sensors have collected the data, it is then sent to the ElectroSense backend platform, which is responsible for its storage, processing, and analysis. This meticulous processing and analysis facilitate the provision of a suite of services. These services extend beyond mere spectrum occupancy monitoring, delving into areas such as transmission optimization and decoding. This range of services provided by ElectroSense not only bolsters the understanding of spectrum utilization but also opens up avenues for innovative optimization and enhancement strategies in the field of IoT. Fig. 2 depicts a diagram of the ElectroSense platform.

For validation, numerous Raspberry Pi devices from different models are deployed in the crowdsensing platform in order to validate the proposed authentication framework. More in detail, the devices deployed are 15 Raspberry Pi 4 Model B, 10 Raspberry Pi 3 Model B+, 10 Raspberry Pi Model +, and 10 Raspberry Pi Zero.

5.2. Data gathering and preprocessing

The first step in the validation process is to obtain the hardware performance data from each device and preprocess it in order to be fed into the Transformer models.

5.2.1. Data gathering

The assembly of individual device authentication premised on hardware behavior hinges on the ability to monitor imperfections inherent in the device chips for subsequent evaluation. As outlined in Section 2, previous studies have primarily tackled this task by contrasting components featuring different base frequencies or crystal oscillators since deviations in these components performance can be discerned directly from the device.

Table 2
LwHBench dataset features (Sánchez et al., 2023a).

Component	Function	Feature Under Observation
-	timestamp	Unix timestamp
	temperature	Core temperature of the device
CPU	1 s sleep	Elapsed GPU cycles during 1s of CPU sleep
	2 s sleep	Elapsed GPU cycles during 2s of CPU sleep
	5 s sleep	Elapsed GPU cycles during 5s of CPU sleep
	10 s sleep	Elapsed GPU cycles during 10s of CPU sleep
	120 s sleep	Elapsed GPU cycles during 120s of CPU sleep
	string hash	Elapsed GPU cycles during computation of a fixed string hash
	pseudo random	Elapsed GPU cycles while generating a software pseudo-random number
	urandom	Elapsed GPU cycles while generating 100 MB using */dev/urandom* interface
	fib	Elapsed GPU cycles while calculating the 20th Fibonacci number using the CPU
GPU	matrix mul	Time taken by CPU to execute a GPU-based matrix multiplication
	matrix sum	Time taken by CPU to execute a GPU-based matrix summation
	scopy	Time taken by CPU to execute a GPU-based graph shadow processing
Memory	list creation	Time taken by CPU to generate a list with 1000 elements
	mem reserve	Time taken by CPU to fill 100 MB in memory
	csv read	Time taken by CPU to read a 500 kB *csv* file
Storage	read x100	100 measurements of CPU time for 100 kB storage read operations
	write x100	100 measurements of CPU time for 100 kB storage write operations

To construct the framework for individual device authentication, it was necessary to compile a dataset that utilizes metrics pertinent to the hardware components inherent in certain devices. This dataset has been christened LwHbench, and additional details can be found in Sánchez et al. (2023a). In this context, the dataset gathered performance metrics from the CPU, GPU, Memory, and Storage of 45 Raspberry Pi devices of diverse models over a span of 100 days. Various functions were executed in these components, employing other hardware elements (operating at differing frequencies) to measure performance. Table 2 provides a summary of the functions that were monitored. These functions embody a set of common operations carried out in every component, aiming to gauge their performance. It is worth mentioning that additional analogous operations could be utilized during the data gathering process. In total, 215 features formed each one of the collected data vectors.

The final dataset contains the following samples per device model: 505584 samples collected from 10 RPi 1B+ devices, 784095 samples from 15 RPi4 devices, 547800 samples from 10 RPi3 devices, and 548647 samples from 10 RPiZero devices. To collect the data, an array of countermeasures were implemented to mitigate the effect of noise introduced by other processes operating in the devices: Component frequency was kept constant, kernel level priority was enforced, the code was executed in an isolated CPU core (in multi-core devices), and memory address randomization was disabled. Moreover, the dataset was compiled under a variety of temperature conditions, facilitating the analysis of the influence this environmental feature has on component performance.

5.2.2. Preprocessing

In the preprocessing stage, the time series were generated by applying a time window over the collected samples, combining them into groups of 10 to 100 vectors. This method of grouping facilitates the implementation of time series Deep Learning (DL) approaches and is adjusted to other literature works (Sánchez et al., 2022). These models possess the ability to uncover intricate trends within the data, potentially leading to superior results compared to the standalone processing

Table 3
Anomaly detection time series models and hyperparameters tested.

Model	Hyperparameters
General Parameters	$epochs = [10, 20, 50]$, $batch\_size = [32, 128, 256, 512]$
1D-CNN	$filters = [16, 32, 64, 128]$, $kernel\_size = [3, 5, 7]$, $n\_layers = [1, 2, 3]$
LSTM	$neurons = [10, 100]$, $n\_layers = [1, 2, 3]$,
LSTM_1D-CNN	$input\_layers = [2, 3]$, $cnn\_filters = [16, 32, 64, 128]$, $cnn\_kernel\_size = [3, 5, 7]$, $lstm\_neurons = [10, 100]$ $n\_layers = [1, 2, 3]$
Transformer	$dff = [32, 64, 128, 256, 1024]$, $num\_layers = [1, 2, 3]$

and evaluation of individual samples. Moreover, it also permits the utilization of attention models such as Transformers, which currently represent the pinnacle of performance in this field. For data normalization, *QuantileTransformer* (Ahsan et al., 2021) was utilized, given the variable data distributions originating from the differing hardware capabilities of each device model. The division of the data for model training and validation purposes consisted of 70% and 10% of the total, leaving the remaining 20% for testing. In order to minimize the potential impact of vector order correlations on the results, the splitting of training, validation, and test sets was performed without shuffling the samples.

5.3. Transformer-based anomaly detection validation

As detailed in Section 4, the proposed Transformer approach performs hyperparameter tuning personalized for each device. Besides, other state-of-the-art DL architectures for anomaly detection in time series are tested to compare their performance to the Transformer. The tested networks are LSTM, 1D-CNN, and a combination of both of these layouts. Table 3 provides a comprehensive overview of the examined algorithms along with their corresponding hyperparameters. For validation, a server equipped with AMD EPYC 7742 CPU, NVIDIA A100 GPU, and 180 GB of RAM is employed, and the models are implemented using *Keras* library.

In the case of the LSTM and 1D-CNN models, the time series concatenation only achieved good results when using groups of 10 vectors or smaller due to their limited memory capabilities. In contrast, the Transformer achieved good results with all the sliding window lengths from 5 to 100, with the best results obtained with 100 vectors per sliding window.

To set the anomaly detection threshold in the reconstruction of the samples fed to the autoencoder models, the 10% of the reconstruction error in the training samples is chosen as the boundary between anomaly and normal sample. Then, the validation set is employed for the hyperparameter selection by choosing the model with the higher TPR.

5.3.1. Authentication performance

For the authentication capabilities evaluation, the strategy followed is one-vs-all, where the trained transformer model evaluates the test set of the source device (normal samples) but also the test sets of the rest of the devices (anomalies or outliers). Then, the True Positive Rate (TPR) of the legitimate device is compared with all the False Positive Rates (FPRs) of the rest of the devices, checking that the TPR value is greater than all the FPRs. Note that for this approach, different data normalizations should be performed in the test sets depending on which device is employed for training as the training data distribution changes.

Table 4 shows the results of the one-vs-all authentication tests. It can be seen how only the Transformer-based approach is able to authenticate all the devices successfully. Although their average TPR is higher, LSTM and 1D-CNN networks only can identify some of the devices, offering a much lower difference between the average TPR and maximum

Table 4
Anomaly detection time series models results.

Model	Best window size	Devices Authenticated	Avg. TPR	Avg. Max. FPR
1D-CNN	10	32	0.88±0.06	0.67±0.29
LSTM	10	38	0.85±0.09	0.53±0.19
LSTM_1D-CNN	10	35	0.88±0.08	0.59±0.22
Transformer	100	45	0.74±0.13	0.06±0.09

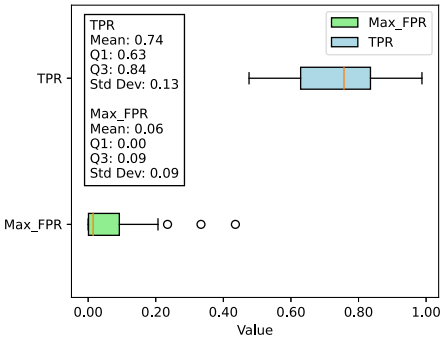


Fig. 3. TPR and maximum FPR distributions of the Transformer autoencoder.

FPR. This occurs because the FPR is much more variable in these models, and many models have a high FPR when evaluating data from other devices, while the FPR variability is smaller in the Transformer models.

Fig. 3 gives a closer look into the distributions of the TPRs and maximum FPRs of the 45 devices evaluated. It can be seen that both distributions are greatly separated, having only three cases where the maximum FPR goes over 0.20 and remains under 0.45. The TPR always stays over that value and reaches values close to 1 in some cases, having most of its values between 0.6 and 0.8. Besides, Fig. 4 shows the exact TPR and maximum FPR values for each one of the devices evaluated, having its MAC address as an identifier. In this graph can be observed that in the cases where the maximum FPR has a relatively high value (0.2 to 0.4), the TPR is way higher, guaranteeing that the authentication can be made reliably.

According to these results, a threshold-based authentication approach could be employed by the *Authentication Module* to determine the result of the authentication process. An example can be a threshold for each device with a value 0.1 lower than the TPR achieved in the validation, as it is enough to differentiate all the devices present in the deployment.

The results achieved by the anomaly detection validation have demonstrated the feasibility of the proposed framework, as it was able to uniquely authenticate 45 single-board devices with identical hardware and software specifications. These findings point towards a promising direction for individual device authentication premised on hardware behavior, demonstrating the potential of Transformer models in this sphere.

5.3.2. Resource usage

Although performance is the key characteristic to decide which model to use in the validation setup, resource usage during training and evaluation is also a critical point that should be taken into account when developing ML/DL-based solutions.

Table 5 shows the time and memory employed by the model. The training time statistics were collected using 10 epochs as the number of iterations over the training dataset. Besides, the evaluation time was obtained while evaluating the entire test dataset of the device. Finally, memory usage represents the size of the model after it has been completely trained.

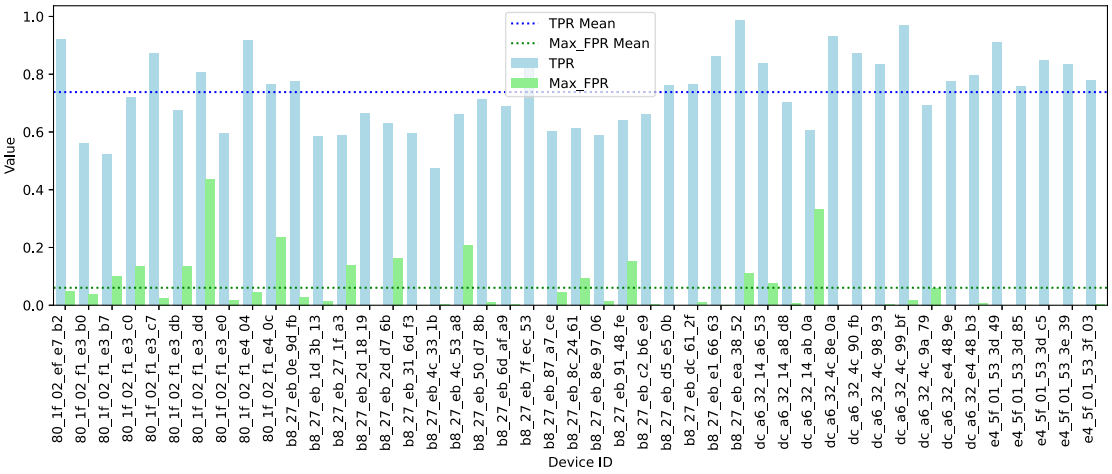


Fig. 4. Transformer autoencoder TPR and maximum FPR comparison per device.

Table 5
Resource usage of each model (per device).

Model	Training Time	Evaluation Time	Memory
1D-CNN	≈47.79 s	≈1.44 s	0.86 MB
LSTM	≈283.68 s	≈2.11 s	1.33 MB
LSTM_1D-CNN	≈306.92 s	≈2.45 s	1.83 MB
Transformer	≈157.68	≈8.93 s	7.77 MB

Each model demonstrates distinct computational characteristics in terms of training time, evaluation time, and memory usage. The 1D-CNN model stands out as the most efficient, boasting the fastest training time of approximately 47.79 seconds and the quickest evaluation time of around 1.44 seconds. Additionally, it consumes the least amount of memory, using only about 0.86 MB. This combination of speed and efficiency makes it an appealing choice for resource-limited applications.

However, the LSTM model presents a significant increase in training time, taking approximately 283.68 seconds, and a slightly longer evaluation time of roughly 2.11 seconds. Coupled with a higher memory footprint of 1.33 MB, this model may demand greater computational resources than the 1D-CNN.

Interestingly, the hybrid LSTM+1D-CNN model exhibits the highest training time among the models, approximately 306.92 seconds, and has a considerable evaluation time of about 2.45 seconds. Its memory usage is also higher, at 1.83 MB, reflecting the complexity inherent to the combination of LSTM and 1D-CNN architectures.

Lastly, the Transformer model demonstrates a more moderate training time of approximately 157.68 seconds, albeit with the longest evaluation time of all models, around 8.93 seconds. More notably, it has a significantly higher memory usage, at a substantial 7.77 MB. While this may limit its applicability in memory-constrained environments, the Transformer model may excel in terms of capturing complex data patterns or delivering superior model accuracy, which are aspects not directly portrayed in the provided table.

In conclusion, while the 1D-CNN model is undeniably efficient regarding speed and memory usage, the Transformer models might offer better performance under certain circumstances. These trade-offs between time, memory usage, and potential model accuracy ought to be taken into account when deciding on the most suitable model for a particular scenario.

5.4. Additional validation in a simulated IoT deployment

Although the solution has already been validated in a real-world ElectroSense deployment, some additional challenges arise when adapting the framework to further scenarios. One of these challenges appears when new hardware models are present and hardware performance samples have to be collected from them. In this sense, gathering cycle counters from the device components is dependent on the exact hardware component, and the procedure might vary, requiring code adaptations in the data gathering process.

ElectroSense is only compatible with Raspberry Pi hardware. Then, to explore this problem, an agriculture IoT scenario was simulated using nine additional devices from three new hardware models. Concretely, the list of devices employed was: 3 Rock64 devices, 3 RockPro64 devices, and 3 Orange Pi 2 Lite devices.

The first step in the deployment process was to adapt the CPU and GPU cycle collection in the data gathering in order to be able to obtain the metrics described in Table 2. As the GPUs in these devices come from ARM Mali family, new counters should be leveraged. Concretely, the counter labeled as *GPU_ACTIVE* was chosen from the available options (Harris, 2016). The *ARM HWCPipe* library was utilized for the collection of cycle counters (Developers, 2021). In terms of CPU-based time collection, the *perf* time was acquired similarly to the methodology for RPi devices, leveraging the *perf_counter_ns()* function. The software to facilitate GPU task execution was modified from the source found in the *ARM Compute Library* (Developer, 2021).

The data collection was executed for one week in these nine devices to have a large enough dataset for this secondary validation, around 4000 samples were collected per device in this period. Then, the data preprocessing steps were repeated as in the ElectroSense validation using Raspberry Pi devices, using a sliding window of 100 values and *QuantileTransformer* normalization. After, the hyperparameter search for the Transformer models in charge of the authentication of each device was performed.

Table 6 shows the authentication results from the nine devices employed in this validation. It can be seen that the results were even better than in the ElectroSense experiments. The TPR for all of the devices was above 0.90, while the maximum FPR stayed under 0.05 in all cases, enabling the threshold-based authentication as in the ElectroSense deployment. The improved results in this validation occurred due to the decreased number of devices from the same model being compared. In

Table 6
Validation in simulated IoT scenario.

Device Model	Device (MAC)	TPR	Max. FPR
RockPro64	86:a4:4c:5f:ff:95	0.8894	0.0417
	8a:32:38:8c:63:e6	0.9217	0.0131
	ee:db:54:9d:8a:67	0.9710	0.000
Rock64	42:58:a0:38:16:11	0.9094	0.007
	76:8f:be:c0:c5:3b	0.9318	0.000
	9a:1d:93:b3:b5:8f	0.9187	0.041
Orange Pi 2 Lite	c0:84:7d:82:4a:1e	0.9653	0.011
	c0:84:7d:82:1c:42	0.9142	0.000
	c0:84:7d:82:38:6d	0.9760	0.002

this sense, only three devices per model were present, so the similarities between the devices in the scenario were reduced.

This second validation approach serves as an example of how the proposed solution could be adapted to new scenarios where novel IoT device models are present, and code adoptions for data collection are necessary. It can be seen how the solution pipeline is still effective once the hardware monitoring is properly modified to gather the cycle counters from the monitored components. Besides, the proposal has been validated with some more devices, enhancing the scalability of the demonstrated solution.

6. Discussion

This section outlines the limitations intrinsic to the suggested approach and provides essential understanding obtained from the research carried out. Through the series of tests performed in this work, coupled with a comparative analysis with existing literature, valuable observations and conclusions emerge. These findings serve not only as lessons gained but also highlight certain restrictions and limitations. The enumeration of lessons learned is as follows:

Potential of Transformer Models for IoT Authentication. The achieved results illustrate the innovative application of Transformer models in the field of hardware-based authentication. By employing a Transformer model, the framework was able to capture complex patterns in hardware behavior of each device, demonstrating a novel approach that could pave the way for future research in security and authentication. This lesson emphasizes the adaptability and potential of Transformer models in areas beyond natural language processing.

Importance of Resource Usage Consideration. The resource analysis emphasizes the critical consideration of resource usage during training and evaluation when developing Machine Learning or Deep Learning-based solutions. Different models demonstrated distinct computational characteristics in terms of training time, evaluation time, and memory usage. For example, the 1D-CNN model was found to be the most efficient, while the Transformer model had a significantly higher memory usage. These trade-offs between time, memory usage, and potential model accuracy must be carefully weighed when selecting the most suitable model for an IoT scenario where processing resources are limited.

Versatility and Importance of Preprocessing Techniques The paper emphasizes the importance of preprocessing techniques in handling time series data. The use of methods like grouping into vectors and data normalization (using QuantileTransformer) was essential in uncovering intricate trends within the data. This lesson serves as a reminder that preprocessing is not a one-size-fits-all step but a critical and adaptable component of the data analysis process, with significant implications for the success of the modeling and authentication framework.

Adaptability to New Scenarios. Based on the results of the second validation using additional IoT devices, it can be seen how only small changes in the data collection process are necessary to adapt the solution to the hardware of new devices. The remaining Transformer-based authentication pipeline remains functional in different scenarios and is hardware agnostic, enabling the application of the solution in a wide variety of environments as an additional security layer and complimenting traditional software-based authentication.

Conversely, the subsequent constraints and limitations have been noted and warrant consideration in upcoming studies within this field:

Determining Hardware Behavior Measurements and Hyperparameter Tuning. The process of identifying the appropriate hardware behavior measurements or feature extraction for individual device authentication is complex and multifaceted. The implementation of the proposed methodology may necessitate multiple exploratory iterations to discover a combination that satisfies all the required properties in the generated fingerprint. Additionally, the Transformer model introduces further complexity due to the need for hyperparameter tuning. Finding the optimal set of hyperparameters for the Transformer model requires a meticulous search, adding to the trial-and-error nature of the process. This iterative analysis can be significantly minimized by examining the properties of the leveraged devices, including different components and operating frequencies, and by employing systematic hyperparameter optimization techniques. Since every chip inherently contains imperfections, the real challenge lies in devising accurate and effective methods to measure them and in fine-tuning the Transformer model to capture these unique characteristics. This complexity adds multiple layers of difficulty to the process and may require careful consideration, experimentation, and optimization to achieve the desired authentication accuracy.

Training and Evaluation Time. The varying training and evaluation times across different models, with the Transformer model exhibiting the longest evaluation time, present a limitation that may affect its suitability in time-sensitive applications. This constraint highlights the importance of considering both accuracy and computational efficiency in model selection and design.

Threshold Setting for Anomaly Detection. During validation, the anomaly detection threshold is set at 10% of the reconstruction error in the training samples fed to the Transformer models. This choice of threshold might have specific implications on the sensitivity and specificity of the anomaly detection as it is manually assigned.

Possible performance degradation over time. As with any hardware, the components of IoT devices may undergo wear and tear, leading to gradual changes in their performance metrics. This natural aging process can alter the cycle skew and other performance parameters that the authentication system initially learned and recognized (Halak et al., 2016). It has been experimentally verified that during the 100 days of data collection, the hardware performance has remained stable. To that end, the authentication experiments were repeated with different splits in the train/test data of each device, achieving very similar results no matter how the data was selected. However, longer periods might have a larger impact on hardware degradation.

7. Conclusions and future work

This paper proposes a framework for individual device authentication based on hardware behavior and outlier detection, which fundamentally relies on identifying inherent imperfections in the device chips. The framework, which leverages hardware behavior fingerprinting and Transformer autoencoders, establishes a unique 'fingerprint' for each device based on manufacturing imperfections in CPU, GPU, RAM, and Storage, even in those with identical specifications. These imperfections are modeled by generating a model trained with the "normal" data distribution of the hardware performance of each device. This provides a robust mechanism for device authentication, distinguishing between genuine and potentially harmful devices. The framework follows a modular design where device monitoring and security enforcement modules are deployed in the device and the data processing modules are hosted in a server with enhanced processing capabilities.

The practical implementation of this authentication framework in the ElectroSense platform demonstrates its effectiveness and real-world

applicability. After 100 days of data collection using 45 Raspberry Pi devices, the Transformer-based autoencoder approach was implemented and compared with other state-of-the-art Deep Learning architectures such as LSTM and 1D-CNN for anomaly detection in time series. Despite the competitive performance of LSTM and 1D-CNN, the Transformer model emerged as the superior method, successfully authenticating all the devices. An average True Positive Rate (TPR) of 0.74±0.13 and an average maximum False Positive Rate (FPR) of 0.06±0.09 are achieved when performing one-versus-all authentication, a more complex task than the classification-based identification performed by other solutions in the literature. From these results, it can be concluded that the proposed approach not only prevents unauthorized device intrusions but also significantly contributes to the reliability of data analysis and the overall trustworthiness of the platform.

Moving forward, this research line has room for future work and improvements. While the current study has focused on Raspberry Pi devices, further research should involve testing the proposed model with other IoT devices, expanding its scope, and ensuring its applicability across a broad range of hardware. In addition, the study has examined the model effectiveness primarily in the context of a spectrum crowdsensing platform, ElectroSense. Future investigations could explore its implementation in different types of crowdsensing applications, thereby contributing to a comprehensive understanding of the framework versatility.

CRediT authorship contribution statement

Pedro Miguel Sánchez Sánchez: Data curation, Formal analysis, Investigation, Methodology, Software, Writing – original draft. **Alberto Huertas Celdrán:** Conceptualization, Resources, Writing – original draft, Writing – review & editing. **Gérôme Bovet:** Funding acquisition, Supervision, Writing – review & editing. **Gregorio Martínez Pérez:** Project administration, Supervision, Writing – review & editing.

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 is available in a different publication referenced in the manuscript

Acknowledgement

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

References

Ahsan, M.M., Mahmud, M.P., Saha, P.K., Gupta, K.D., Siddique, Z., 2021. Effect of data scaling methods on machine learning algorithms and model performance. Technol. 9 (3), 52.

Al-Garadi, M.A., Mohamed, A., Al-Ali, A.K., Du, X., Ali, I., Guizani, M., 2020. A survey of machine and deep learning methods for Internet of things (iot) security. IEEE Commun. Surv. Tutor. 22 (3), 1646–1685.

Al-Naji, F.H., Zagrouba, R., 2022. Cab-iot: continuous authentication architecture based on blockchain for Internet of things. J. King Saud Univ, Comput. Inf. Sci. 34 (6), 2497–2514.

Arafin, M.T., Qu, G., 2021. Hardware-based authentication applications. In: Authentication of Embedded Devices: Technologies, Protocols and Emerging Applications, pp. 145–181.

ARM, 2021. Trustzone Technology. (Accessed 10 October 2023).

Capponi, A., Fiandrino, C., Kantarci, B., Foschini, L., Kliazovich, D., Bouvry, P., 2019. A survey on mobile crowdsensing systems: challenges, solutions, and opportunities. IEEE Commun. Surv. Tutor. 21 (3), 2419–2465.

Chen, Z., Chen, D., Zhang, X., Yuan, Z., Cheng, X., 2021. Learning graph structures with transformer for multivariate time-series anomaly detection in iot. IEEE Int. Things J. 9 (12), 9179–9189.

Chen, Z., Liu, J., Shen, Y., Simsek, M., Kantarci, B., Mouftah, H.T., Djukic, P., 2022. Machine learning-enabled iot security: open issues and challenges under advanced persistent threats. In: ACM Computing Surveys (CSUR).

Choi, K., Yi, J., Park, C., Yoon, S., 2021. Deep learning for anomaly detection in time-series data: review, analysis, and guidelines. IEEE Access 9, 120043–120065.

Cisco, 2021. Trust Anchor Module (tam). (Accessed 10 October 2023).

Developers, A., 2021. ARM compute library. https://arm-software.github.io/ComputeLibrary/latest/index.xhtml. (Accessed 12 September 2022). Online.

Developers, A., 2021. HWCPipe. https://github.com/ARM-software/HWCPipe. (Accessed 12 September 2022). Online.

Feng, Y., Wang, W., Weng, Y., Zhang, H., 2017. A Replay-Attack Resistant Authentication Scheme for the Internet of Things. 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), vol. 1. IEEE, pp. 541–547.

GlobalPlatform, 2021. Device Trust Architecture (dta). (Accessed 10 October 2023).

Halak, B., Zwolinski, M., Mispan, M.S., 2016. Overview of puf-based hardware security solutions for the Internet of things. In: 2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, pp. 1–4.

Harris, P., 2016. Mali: Midgard family performance counters. https://community.arm.com/arm-community-blogs/b/graphics-gaming-and-vr-blog/posts/mali-midgard-family-performance-counters/. (Accessed 30 July 2022). Online.

Infineon, 2021. Optiga Trusted Platform Module (tpm). (Accessed 10 October 2023).

Intel, 2021. Enhanced Privacy Id (epid). (Accessed 10 October 2023).

Kozik, R., Pawlicki, M., Choraś, M., 2021. A new method of hybrid time window embedding with transformer-based traffic data classification in iot-networked environment. Pattern Anal. Appl. 24 (4), 1441–1449.

Laor, T., Mehanna, N., Durey, A., Dyadyuk, V., Laperdrix, P., Maurice, C., Oren, Y., Rouvoy, R., Rudametkin, W., Yarom, Y., 2022. Drawnapart: a device identification technique based on remote gpu fingerprinting. ArXiv preprint arXiv:2201.09956.

Li, D., Li, Q., 2020. Adversarial deep ensemble: evasion attacks and defenses for malware detection. IEEE Trans. Inf. Forensics Secur. 15, 3886–3900.

Marabissi, D., Mucchi, L., Stomaci, A., 2022. Iot nodes authentication and id spoofing detection based on joint use of physical layer security and machine learning. Future Internet 14 (2), 61.

Microchip, 2021. Cryptoauthentication. (Accessed 10 October 2023).

Microsoft, 2021. Azure Sphere. (Accessed 10 October 2023).

NXP, 2021. A71ch Secure Element. (Accessed 10 October 2023), Online.

Rajan, A., Jithish, J., Sankaran, S., 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, S., Calvo-Palomino, R., Fuchs, M., Van den Bergh, B., Cordobés, H., Giustiniano, D., Pollin, S., Lenders, V., 2018. Electrosense: open and big spectrum data. IEEE Commun. Mag. 56 (1), 210–217.

Rambus, 2021. Cryptomanager Iot Device Management. (Accessed 10 October 2023).

Salo, T.J., 2007. Multi-factor fingerprints for personal computer hardware. In: MILCOM 2007-IEEE Military Communications Conference, pp. 1–7.

Sánchez, P.M.S., Celdrán, A.H., Bovet, G., Pérez, G.M., 2022. Adversarial attacks and defenses on ml-and hardware-based iot device fingerprinting and identification. ArXiv preprint arXiv:2212.14677.

Sánchez, P.M.S., Valero, J.M.J., Celdrán, A.H., Bovet, G., Pérez, M.G., Pérez, G.M., 2021. A survey on device behavior fingerprinting: data sources, techniques, application scenarios, and datasets. IEEE Commun. Surv. Tutor. 23 (2), 1048–1077.

Sánchez, P.M.S., Valero, J.M.J., Celdrán, A.H., Bovet, G., Pérez, M.G., Pérez, G.M., 2023a. Lwhbench: a low-level hardware component benchmark and dataset for single board computers. Int. Things 22, 100764.

Sánchez, P.M.S., Valero, J.M.J., Celdrán, A.H., Bovet, G., Pérez, M.G., Pérez, G.M., 2023b. A methodology to identify identical single-board computers based on hardware behavior fingerprinting. J. Netw. Comput. Appl., 103579.

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.

Sánchez Sánchez, P.M., 2023. Authentication IoT transformer. https://github.com/sxz0/Authentication_IoT_Transformer. (Accessed 8 October 2023). Online.

Shamsoshoara, A., Korenda, A., Afghah, F., Zeadally, S., 2020. A survey on physical unclonable function (puf)-based security solutions for Internet of things. Comput. Netw. 183, 107593.

Shrivastava, A., Haripriya, D., Borole, Y.D., Nanoty, A., Singh, C., Chauhan, D., 2022. High performance fpga based secured hardware model for iot devices. Int. J. Syst. Assur. Eng. Manag., 1–6.

Stellios, I., Kotzanikolaou, P., Grigoriadis, C., 2021. Assessing iot enabled cyber-physical attack paths against critical systems. Comput. Secur. 107, 102316.

Sánchez, P.M.S., Celdrán, A.H., Bovet, G., Pérez, G.M., Stiller, B., 2023c. Specforce: a framework to secure iot spectrum sensors in the Internet of battlefield things. IEEE Commun. Mag. 61 (5), 174–180.

Tuli, S., Casale, G., Jennings, N.R., 2022. Tranad: deep transformer networks for anomaly detection in multivariate time series data. ArXiv preprint arXiv:2201.07284.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30.
Wang, C., Wang, D., Tu, Y., Xu, G., Wang, H., 2020. Understanding node capture attacks in user authentication schemes for wireless sensor networks. IEEE Trans. Dependable Secure Comput. 19 (1), 507–523.
Yu, J.J.Q., 2020. Sybil attack identification for crowdsourced navigation: a self-supervised deep learning approach. IEEE Trans. Intell. Transp. Syst. 22 (7), 4622–4634.
Zhang, J., Rajendran, S., Sun, Z., Woods, R., Hanzo, L., 2019. Physical layer security for the Internet of things: authentication and key generation. IEEE Wirel. Commun. 26 (5), 92–98.
Zhong, S., Zhong, H., Huang, X., Yang, P., Shi, J., Xie, L., Wang, K., Zhong, S., Zhong, H., Huang, X., et al., 2019. Connecting human to cyber-world: security and privacy issues in mobile crowdsourcing networks. In: Security and Privacy for Next-Generation Wireless Networks, pp. 65–100.

Pedro Miguel Sánchez Sánchez received the M.Sc. degree in computer science from the University of Murcia. He is currently pursuing his PhD 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.

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.

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 1ST 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.