IoT-ID: Deterministic Device Identity from Hybrid Network Fingerprinting

Anna Letycia[1], Joner Assolin[2] Diego Kreutz[2], Rodrigo Miani[1]

[1]Universidade Federal de Uberlândia (UFU)
[2]Universidade Federal do Pampa (UNIPAMPA)
anna.fernandes@ufu.br,jonerassolin.aluno@unipampa.edu.br
diegokreutz@unipampa.edu.br,miani@ufu.br

Abstract. *This paper presents IoT-ID, a deterministic fingerprinting system for IoT device identification based on network traffic analysis. By combining passive and active measurements into a canonical multi-layer representation and applying cryptographic hashing, IoT-ID derives stable and reproducible device identities without relying on training data. The integration of application-layer metadata with transport-layer signatures resolves collisions between devices with indistinguishable network stacks, enabling consistent identification across heterogeneous environments. Results confirm the feasibility and effectiveness of deterministic fingerprinting, while highlighting limitations in non-IoT systems due to privacy mechanisms and software homogeneity.*

1. Introduction

The rapid proliferation of IoT devices in operational networks has increased the complexity of device identification and asset management. Traditional identifiers such as IP and MAC addresses are unreliable because they can be dynamically reassigned, spoofed, or obfuscated by privacy mechanisms, hindering the construction of consistent and auditable inventories. In this context, device fingerprinting has emerged as a non-intrusive alternative based on observable network characteristics or intrinsic device properties.

Despite significant advances, existing fingerprinting approaches still present fundamental limitations. Many rely on machine learning models to classify devices from traffic patterns or behavioral features [Safi et al. 2022, Sánchez et al. 2021]. Although effective for coarse-grained classification, these methods are probabilistic, depend on representative datasets and retraining, and often struggle to distinguish devices at fine granularity, such as specific models or instances [Wan et al. 2023]. In parallel, hardware- and network-based features are often unstable, being affected by environmental conditions, firmware updates, workload dynamics, and runtime variability [Kumar and Paul 2023, Zhang et al. 2025, Safi et al. 2022]. In addition, there is still no clear consensus on which attributes remain consistently discriminative across heterogeneous settings [Sheng et al. 2025].

Current approaches also fail to guarantee uniqueness and determinism. Devices sharing similar communication stacks may produce indistinguishable fingerprints when only transport-layer or statistical features are considered, leading to collisions and ambiguity [Wan et al. 2023]. Even hybrid methods remain dependent on probabilistic inference and do not ensure consistent identities across repeated observations. The absence of canonical representations further limits reproducibility and comparability across experiments and deployments [Safi et al. 2022].

To address these limitations, this work presents IoT-ID[1] as an initial step toward deterministic and reproducible device identification in IoT environments. The proposed approach systematically combines discriminative fields extracted from active scanning tools (e.g., *nmap*) and passive fingerprinting tools (e.g., *p0f*) to construct canonical representations that generate unique fingerprints in conventional WiFi IoT networks. Unlike prior approaches, IoT-ID does not rely on training data, emphasizing interpretability, reproducibility, and consistency across executions.

IoT-ID also adopts a temporal perspective: fingerprints are considered valid at a given time instant t and may evolve due to firmware, software, or environmental changes. As part of the broader IoTEdu[2] architecture, the tool stores historical signatures, allowing such changes to be interpreted as meaningful events rather than noise. This supports not only device identification but also continuous monitoring and security analysis in IoT WiFi networks.

2. Related Work

IoT device identification through fingerprinting has been widely explored using network traffic characteristics, behavioral patterns, and intrinsic device properties. Traffic-based approaches typically rely on features such as packet sizes, inter-arrival times, and protocol signatures to build behavioral profiles [Wan et al. 2023, Aksoy and Gunes 2019]. Although passive and learning-based methods can be effective in controlled settings [Aksoy and Gunes 2019], they usually depend on representative datasets, retraining, and traffic stability, and often fail to distinguish devices with similar communication stacks.

More advanced solutions combine active and passive data collection or incorporate hardware-level features to improve discriminative power [Wan et al. 2023, Kohli et al. 2024]. However, they add complexity, may require direct device access, and remain sensitive to environmental variations. Overall, existing approaches focus on classification or probabilistic inference rather than on stable and reproducible device identities [Chowdhury and Abas 2022, Sánchez et al. 2021]. As summarized in Table 1, no current solution consistently provides unique, stable, and deterministic fingerprints.

Table 1. Comparison of fingerprinting approaches

Work	Type of Collection	ML	Attribute Source	Deterministic
[Gao et al. 2010]	Passive	✗	RF / Temporal	✗
[Aksoy and Gunes 2019]	Passive	✓	Network Traffic	✗
[Wan et al. 2023]	Hybrid	✓	Network Traffic	✗
[Kohli et al. 2024]	Active	✓	Hardware (SRAM)	✗
IoT-ID - this work	**Hybrid**	✗	**Network Traffic**	✓

3. IoT-ID: Architecture

In contrast to prior fingerprinting approaches based on probabilistic inference, training data, or single-layer observations, IoT-ID is designed to construct **deterministic, reproducible, and multi-layer device identities**. The architecture addresses key limitations

[1] https://github.com/GT-IoTEdu/sbrc26_fingerprint
[2] https://gt-iotedu.github.io

identified in the literature: (i) lack of uniqueness in devices with indistinguishable network stacks, (ii) instability of behavioral features, and (iii) absence of canonical representations for reproducible fingerprint generation. To this end, IoT-ID integrates active and passive measurements and applies a canonicalization process that converts volatile observations into stable and comparable identifiers.

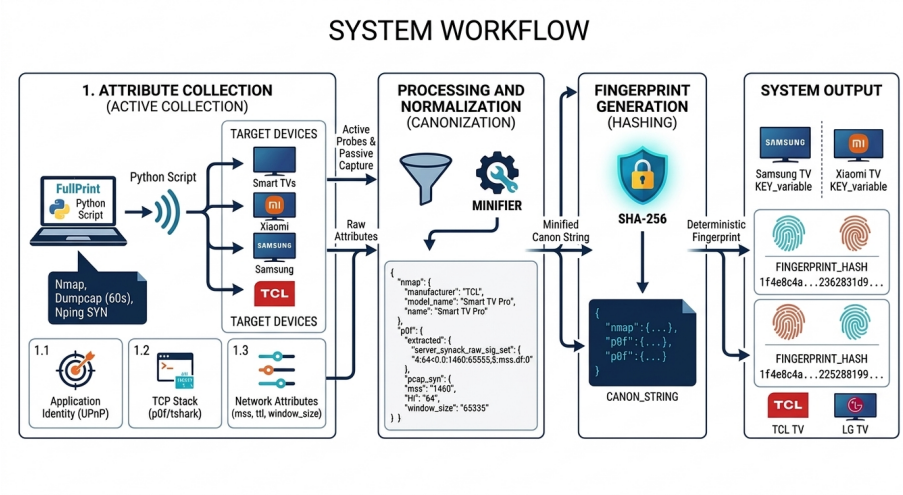

Figure 1. Overview of the fingerprinting tool architecture

As shown in Figure 1, the IoT-ID pipeline is organized into four phases. In **active probing and device discovery**, IoT-ID combines declarative and behavioral sources. Metadata is collected via UPnP (*Universal Plug and Play*), capturing application-layer attributes such as `manufacturer` and `model_name`. In parallel, `nping` triggers TCP SYN responses from the target device, enabling feature extraction even in lowtraffic or idle scenarios.

In **traffic capture and observation**, network traffic is collected with `dumpcap` over a configurable time window. Rather than relying on aggregate behavioral statistics, IoT-ID extracts protocol-level and negotiation-specific attributes, reducing sensitivity to workload variability and improving stability across repeated executions. Next, in **feature extraction (multi-layer representation)**, the captured PCAP is processed with `p0f` and `tshark` to obtain transport-layer signatures and packet-level attributes. This phase produces a multi-layer representation that combines application-layer metadata (L7), such as manufacturer, model, and service descriptors, with transport-layer signatures (L3/L4), including MSS, TTL, window size, TCP options, and SYN/SYN-ACK signatures. This hybrid view allows IoT-ID to distinguish devices that would remain ambiguous if only one layer were considered.

Finally, in **canonicalization and deterministic hashing**, the extracted attributes are normalized to remove temporal noise and non-deterministic variations, then transformed into a canonical representation in a consistent and ordered form. The fingerprint at time t is defined as $\mathrm{FP}_t = \mathcal{H}(\mathcal{C}(\mathcal{A}_t))$, where $\mathcal{A}_t$ is the set of observed attributes, $\mathcal{C}(\cdot)$ is a deterministic canonicalization function, and $\mathcal{H}(\cdot)$ is a cryptographic hash function (e.g., SHA-256). Guided by the attribute taxonomy in Table 2, the canonicalization distinguishes stable from dynamic features, enabling reproducible fingerprints under equivalent conditions while preserving sensitivity to meaningful device changes over time.

Attribute Taxonomy, Temporal Stability, and Security Semantics

To address instability issues observed in prior work, IoT-ID distinguishes between **static** and **dynamic** attributes, as summarized in Table 2. Static attributes, such as manufacturer, model name, MSS, and baseline TTL, form the identity core of the Canon String and support long-term consistency. Dynamic attributes, including TCP window size, `p0f` signatures, and protocol options, capture software-dependent characteristics that may evolve with firmware, kernel, or configuration updates.

Rather than treating such variations as noise, IoT-ID represents them explicitly over time. A fingerprint is therefore considered valid at a given time instant t, representing the observed device state at that moment. When a subsequent fingerprint FP_{t+1} differs from FP_t, IoT-ID records a *state transition* rather than treating the change as an error.

IoT-ID does not classify a transition as legitimate or malicious on its own; that judgment belongs to a higher-level policy layer. Its role is to make transitions visible, auditable, and attributable by identifying which attributes changed, at which layer, and when. Changes restricted to dynamic attributes are consistent with expected software evolution, whereas changes in static attributes indicate potentially relevant structural deviations that may require further investigation. This distinction enables IoT-ID to support both lifecycle monitoring and security analysis without conflating the two.

Security model. IoT-ID is designed to expose four classes of security-relevant events through fingerprint transitions: (i) changes in static attributes, which may indicate device replacement, spoofing, or hardware-level tampering; (ii) changes in dynamic attributes, which should be correlated with maintenance events; (iii) fingerprint disappearance, which may indicate removal, network reconfiguration, or denial of service; and (iv) the introduction of a fingerprint collision, which is expected for legitimate additions of the same device class but may be suspicious otherwise.

Class-level fingerprint collisions among identical devices are not a weakness but a deliberate property. They support group-based vulnerability management, while the MAC address, maintained separately by IoTEdu, preserves instance-level accountability. Together, deterministic fingerprints and MAC addresses provide class- and instance-level views of device identity.

Table 2. Attribute Taxonomy for Canonical IoT Fingerprinting

Layer	Source	Attribute	Can change?	Justification / Description
L7	`nmap`	`manufacturer`	No	The manufacturer is a static hardware property.
L7	`nmap`	`model_name`	No	The physical model remains unchanged after updates.
L7	`nmap`	`server`	Yes	Firmware updates often modify the UPnP server or SDK version.
L7	`nmap`	`name`	Yes	Typically a user-defined or application-defined string.
L4	`p0f`	`server_synack_raw_sig_set`	Yes	Kernel changes (e.g., Linux 4.x to 5.x) alter TCP signatures.
L4	`p0f`	`client_syn_raw_sig_set`	Yes	Follows the same kernel-dependent behavior.
L3/L4	`pcap_syn`	`mss`	No	Typically fixed by hardware and network interface MTU.
L3/L4	`pcap_syn`	`window_size`	Yes	Firmware updates may modify TCP buffer configurations.
L3/L4	`pcap_syn`	`ttl`	No	The base value (e.g., 64) is consistent across most Linux/Android systems.
L3/L4	`pcap_syn`	`ws`	Yes	The *Window Scale* factor may be enabled or adjusted in newer versions.
L4	`pcap_syn`	`sack_perm`	Yes	SACK support may be introduced through performance improvements.
L4	`pcap_syn`	`ts_present`	Yes	*Timestamp* support depends on kernel configuration.

4. Experimental Setup and Data Collection

4.1. Experimental Environment and Evaluated Devices

The experimental evaluation was conducted in a controlled and reproducible environment using a VirtualBox-based virtualization platform. The host system consists of an Intel

CPU with 2 GB of dedicated RAM, while network traffic was captured through the Intel 82540EM interface (`enp0s3`) configured in promiscuous mode to ensure full visibility of local communications. Data collection was performed on Linux Mint 22.3 (Zena), running kernel version `6.14.0-37-generic`, minimizing external variability during fingerprint acquisition. Table 3 summarizes the evaluated devices, including manufacturer, model, and operating system or kernel version when available. The dataset comprises heterogeneous device classes commonly found in IoT environments, such as smart TVs, TV boxes, printers, and general-purpose computing devices, enabling the evaluation of IoT-ID under realistic conditions, including homogeneous network stacks and varying levels of device observability.

Table 3. Evaluated devices

Device	Manufacturer	Model	Qty.
Game Console	Microsoft	Xbox One	1
IP Camera	TP-Link	C500 (Model A)	3
IP Camera	Vlx LED EXcelente	Speed Dome Solar (Model B)	1
Router	FiberHome	HG6143D	1
Router	TP-Link	EC220-G5	1
Smart Bulb	Avant NEO	RGB E27	1
Smart TV	TCL	UnionTV	1
Smart TV	TCL	Smart TV Pro	1
Smart TV	Samsung	UN32J4303	2
Smart TV	Samsung	QN55Q60TAGXZD	1
TV Box	Xiaomi	MiTV-AESP0	1
Wi-Fi Printer	HP	Deskjet 4640	1

4.2. Fingerprint Generation and Outputs

Fingerprint generation is performed by `iot_id_fingerprint.py`, which integrates active probing, traffic capture, feature extraction, and canonicalization into a unified pipeline. The tool receives as input the target IP, capture duration, output directory, and network interface, as illustrated below:

```
iot_id_fingerprint.py runs [IP] --seconds 60 --iface <INTERFACE>
```

The tool is designed to run in a Linux environment (Ubuntu 20.04 or later recommended) with Python 3.8+, requiring access to a local network where target devices are present. Administrative privileges (`sudo`) are necessary to enable packet capture and active probing. The execution depends on standard network analysis tools, including `nmap`, `nping`, `dumpcap`, `tshark`, and `p0f`, which must be available in the system path.

When executed, the command orchestrates the complete fingerprinting pipeline, combining active scanning using Nmap, controlled packet capture via dumpcap, TCP SYN probing using nping, passive fingerprint extraction with p0f, and feature extraction from PCAP using tshark. The extracted attributes are then canonicalized and hashed using SHA-256 to produce a deterministic fingerprint. Optional parameters such as `--mode` (`target`, `default`, or `network`) and `--canon_policy` (`stable` or `rich`) allow configuring the discovery strategy and the attribute set used prior to hashing.

Each execution produces a structured output directory containing all intermediate and final artifacts. The outputs include the raw bundle (`fingerprint.json`), the canonical representation (`features_canon.json`/`.txt`), and the final fingerprint (`fingerprint_sha256.txt`). These artifacts are stored under directories following

the pattern `runs/<IP_timestamp>/`, enabling traceability and reproducibility of each execution.

In a typical usage scenario, the user first identifies active devices on the local network and then executes the fingerprinting pipeline for a selected target. The tool captures network behavior over a fixed interval and produces a deterministic fingerprint that remains stable across multiple executions under consistent conditions. This workflow enables the practical application of IoT-ID in real environments, supporting device identification, monitoring, and comparison across heterogeneous IoT devices.

By construction, identical inputs always generate identical fingerprints, ensuring reproducibility and reinforcing the deterministic nature of IoT-ID. The tool is publicly available as an open-source repository with documentation, dependencies, an explicit license, and defined execution workflows to support reproducibility. While strict replication depends on access to identical physical devices, the methodology ensures consistent fingerprints for the same device and distinct identities across different devices under comparable conditions.

5. Experimental Results

The effectiveness of the proposed *IoT-ID* pipeline is evaluated using a dataset of 15 heterogeneous devices, including smart TVs, routers, cameras, a gaming console, and embedded IoT devices. The results reveal three distinct classes of behavior: (i) class-consistent fingerprints, (ii) cross-device signature collisions at L3/L4, and (iii) generic or low-entropy fingerprints.

Table 4 summarizes the extracted attributes and observed behavior for all devices. Across five consecutive executions per device, identical canonical representations and SHA-256 fingerprints were obtained, confirming the stability and reproducibility of the proposed method under consistent network conditions.

Table 4. Consolidated fingerprint attributes and observed behavior

Device	L7 (nmap)		L3/L4 (pcap_syn)			L4 (p0f)		Behavior
	manufacturer	model	mss	window	ttl	server sig	client sig	
Game Console	Microsoft	Xbox One	1460	65535	128	-	`4:128+0:0:1460:65535,...`	Unique
IP Camera Model A (1)	TP-Link	C500	1440	14400	64	-	-	Generic
IP Camera Model A (2)	TP-Link	C500	1440	14400	64	-	-	Class
IP Camera Model A (3)	TP-Link	C500	1440	14400	64	-	-	Class
IP Camera Model B	Vlx LED EXcelente	Speed Dome Solar	-	-	64	-	-	Limited
Router FiberHome	FiberHome	HG6143D	1460	29200	64	-	-	Generic
Router TP-Link	TP-Link	EC220-G5	1460	14600	64	`mss*10,...`	-	Collision (Cross-device)
Smart Bulb	Avant NEO	RGB E27	-	-	255	-	-	Limited
Smart TV Samsung (1)	Samsung	UN32J4303	1460	14600	64	`mss*10,...`	-	Class
Smart TV Samsung (2)	Samsung	UN32J4303	1460	14600	64	`mss*10,...`	-	Class
Smart TV Samsung 55"	Samsung	QN55Q60TAGXZD	1460	29200	64	`mss*20,...`	-	Unique
Smart TV TCL (Smart TV Pro)	TCL	Smart TV Pro	1460	65535	64	`4:64+0:0:1460:65535,...`	-	Unique
Smart TV TCL (UnionTV)	TCL	UnionTV	1460	65535	64	`4:64+0:0:1460:65535,...`	-	Unique
TV Box Xiaomi	Xiaomi	MiTV-AESP0	1460	65535	64	`4:64+0:0:1460:65535,...`	-	Collision (L3/L4)
Wi-Fi Printer	HP	Deskjet 4640	1460	8760	64	`mss*6,...`	-	Unique

Class-consistent fingerprints. Devices sharing identical hardware and software configurations produce identical fingerprints, forming well-defined classes. This behavior is observed in the two Samsung UN32J4303 TVs, which exhibit identical L3/L4 attributes and identical p0f signatures, resulting in the same canonical representation and SHA-256 hash. A similar pattern is observed among IP cameras (1, 2, and 4), which share identical network parameters (MSS, TTL, and window size), indicating homogeneous firmware and network stacks. These results confirm that IoT-ID reliably captures device classes.

Cross-device collisions at L3/L4. The evaluation also reveals that distinct devices may share identical transport-layer characteristics. Notably, the Xiaomi MiTV and both

TCL TVs exhibit identical MSS (1460), TTL (64), window size (65535), and identical p0f signatures. Under traditional passive fingerprinting approaches, these devices would be indistinguishable. Additionally, the TP-Link router shares similar TCP characteristics with one of the Samsung TVs (window size 14600 and MSS scaling), demonstrating that such overlaps can occur even across device categories. IoT-ID resolves these ambiguities by incorporating application-layer attributes, producing distinct fingerprints despite identical L3/L4 behavior.

Generic and low-entropy devices. Some devices exhibit limited or non-distinctive attributes, resulting in generic fingerprints. The FiberHome router and multiple IP cameras expose only basic TCP/IP characteristics without application-layer metadata, reducing their identifiability. Similarly, each smart bulb and IP camera has an incomplete attribute set (e.g., missing MSS or window size), resulting in low-entropy fingerprints. These cases highlight the inherent limitations of passive fingerprinting in constrained IoT environments.

Device-specific signatures. Other devices exhibit distinctive patterns that enable clear identification. The Xbox One is characterized by a higher TTL (128) and a client-side SYN signature that includes advanced TCP options (e.g., window scaling). The Wi-Fi printer presents a non-standard TCP configuration, including a window size derived from $MSS \times 6$ and the absence of the DF flag, both of which are characteristic of embedded or RTOS-based stacks.

Overall, the results demonstrate that multi-layer canonicalization effectively balances generalization and discrimination: identical devices are consistently grouped into classes, while heterogeneous devices are correctly distinguished even under identical transport-layer conditions.

5.1. Discussion

The results in Table 5 confirm a key property of deterministic fingerprinting: devices with identical manufacturer, model, firmware, and network stack parameters produce the same SHA-256 fingerprint. For example, *TV Samsung (1)* and *TV Samsung (2)* share the fingerprint `1f4e8c4a...`, while the three units of IP Camera Model A, instances (1), (2), and (3), share `c61f2883...`. These repetitions are therefore expected outcomes of equivalent attribute sets rather than accidental collisions.

Table 5. Devices and their corresponding SHA-256 fingerprints

Device	SHA-256 Fingerprint
TV TCL (UnionTV)	994f131342176c20415565dab0adf2666b160422d3df1511c0a37ae135985add
Router FiberHome	14407342dc69a801f52f6cead97d00e5260b7206072f145e6fde19e07cb1f157
Smart Bulb	993e9afc860d624a332822679ea4efc3490c2734ba84ad160c8d36abf0d546f7
TV TCL (Smart TV Pro)	e20c48257b98e86fa11d7c4444e7e5da7176a1b328719ea5f46f831951392d51
TV Samsung (1)	1f4e8c4af2be567b929cf5b7409c57d648ea062262eeab566e11e32c1b437e94
TV Xiaomi	6397f4729927379fde73d5c1ea234ef1070d3785b4f271c562fa4cb688be2d48
TV Samsung 55”	410528d091cb14cea300d79e74348087fc3a1b351584c919650c3a6f24ec18d1
Printer	93ef878c8f2de4eab5a5c780b1dc146d28df337116d8e93019bd9f4cc78c41df
Xbox One	50f1736613c079a70b2d094c680cd8f9027adeb34ca7e90e7c4322a73492eb01
TV Samsung (2)	1f4e8c4af2be567b929cf5b7409c57d648ea062262eeab566e11e32c1b437e94
Router TP-Link	dce0ee76d22f60bec93ce4f6478a9ffe439b2b6e914bafbafe0048685402b2cc
IP Camera Model A (1)	c61f28839b696b65307cef2db341e976fadf10012d9c0f01f6a490b1e3e5742f
IP Camera Model A (2)	c61f28839b696b65307cef2db341e976fadf10012d9c0f01f6a490b1e3e5742f
IP Camera Model A (3)	c61f28839b696b65307cef2db341e976fadf10012d9c0f01f6a490b1e3e5742f
IP Camera Model B	5b60133e2971f571f7a4ceadd55041a78771b65189e64c0f891eb687ab37bc74

Within the IoTEdu architecture, this is a deliberate design choice rather than

a limitation. The fingerprint acts as a class identifier, while the MAC address preserves instance-level accountability. This dual granularity supports both asset tracking and group-based management. Operationally, devices sharing the same fingerprint can be managed as a homogeneous group. For example, all devices associated with `c61f2883...` (IP cameras) can be jointly assessed for vulnerabilities or firmware updates, reducing overhead in large-scale deployments.

The canonical model separates stable from dynamic attributes, making state transitions interpretable. Manufacturer and model define the invariant identity core, whereas MSS and observed TTL are treated as context-dependent indicators rather than anomalies by definition. In this work, baseline TTL denotes the reference value, or expected range, previously observed for a device in a given network context. Dynamic attributes, such as TCP window size, `p0f` signatures, and service strings, may evolve with software updates. This separation supports continuous monitoring while preserving transition visibility for operator validation.

The evaluation also highlights the boundaries of the proposed approach. IoT-ID is designed for IoT environments, where devices typically expose constrained and relatively stable network stacks. In non-IoT settings, however, standardized stacks and privacy-preserving mechanisms may reduce the observable feature space, leading to generic or inconsistent fingerprints. This trade-off between identifiability and privacy is particularly relevant in enterprise environments and motivates complementary temporal or behavioral approaches.

In addition to these scope limitations, IoT-ID operates primarily at the device-class level and depends on observable network behavior. Devices with sparse or event-driven communication may yield incomplete fingerprints, making profile updates opportunistic and potentially delayed. Moreover, network topology, intermediary devices, and filtering mechanisms may alter observable attributes. These aspects do not invalidate the approach, but define important boundaries of the current evaluation.

6. Conclusion and Future Work

This paper presented IoT-ID[3], a deterministic fingerprinting approach for IoT device identification based on a hybrid pipeline that combines application-layer metadata with transport-layer signatures. Its main contribution is a canonical representation that enables reproducible and auditable device identities via cryptographic hashing. The results show that transport-layer attributes alone are insufficient in environments with homogeneous network stacks, whereas multi-layer canonicalization resolves important ambiguities. They also show that device identity emerges from cross-layer correlation, although non-IoT environments remain challenging due to privacy mechanisms and software homogeneity. Future work includes temporal and learning-based techniques, expanded discovery protocols, TLS/SSL metadata, broader validation in real-world enterprise environments, and a longitudinal evaluation with repeated observations over time, including shutdown/restart cycles, reconnections, and software changes, in order to better characterize fingerprint stability and temporal variability.

[3]`https://github.com/GT-IoTEdu/sbrc26_fingerprint`

Acknowledgments

This work was partially supported by the Brazilian National Research and Education Network (RNP) through the GT-IoTEdu project, approved under the 2025 Advanced Services R&D Program, and by the Brazilian National Council for Scientific and Technological Development (CNPq) under Grants 409743/2025-9 and 408432/2024-1.

References

Aksoy, A. and Gunes, M. H. (2019). Automated iot device identification using network traffic. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–7.

Chowdhury, R. R. and Abas, P. E. (2022). A survey on device fingerprinting approach for resource-constraint iot devices: Comparative study and research challenges. *Internet of Things*, 20:100632.

Gao, K., Corbett, C., and Beyah, R. (2010). A passive approach to wireless device fingerprinting. In *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, pages 383–392.

Kohli, V., Aman, M. N., and Sikdar, B. (2024). An intelligent fingerprinting technique for low-power embedded iot devices. *IEEE Transactions on Artificial Intelligence*, 5(9):4519–4534.

Kumar, V. and Paul, K. (2023). Device fingerprinting for cyber-physical systems: A survey. *ACM Computing Surveys*, 55(14s):1–41.

Safi, M., Dadkhah, S., Shoeleh, F., Mahdikhani, H., Molyneaux, H., and Ghorbani, A. A. (2022). A survey on iot profiling, fingerprinting, and identification. *ACM Transactions on Internet of Things*, 3(4):1–39.

Sheng, C., Zhou, W., Han, Q.-L., Ma, W., Zhu, X., Wen, S., and Xiang, Y. (2025). Network traffic fingerprinting for iiot device identification: A survey. *IEEE Transactions on Industrial Informatics*.

Sánchez, P. M. S., Valero, J. M. J., Celdrán, A. H., Bovet, G., Pérez, M. G., and Pérez, G. M. (2021). A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets. *IEEE Communications Surveys & Tutorials*, 23(2):1048–1077.

Wan, S., Li, Q., Wang, H., Li, H., and Sun, L. (2023). Devtag: A benchmark for fingerprinting iot devices. *IEEE Internet of Things Journal*, 10(7):6388–6399.

Zhang, J., Ardizzon, F., Piana, M., Shen, G., and Tomasin, S. (2025). Physical layer-based device fingerprinting for wireless security: From theory to practice. *IEEE Transactions on Information Forensics and Security*.