Hardware-Based System Fingerprinting for IoT Devices Using Configuration Data

Tianli Yang[1], Shancang Li [2,*]

[1]*University of Electronic Science and Technology of China, Chengdu, China.*

[2]*Cardiff University, Cardiff, UK.*

Abstract: The proliferation of IoT devices challenges digital forensic investigations. Traditional software-based fingerprinting often fails under network changes or software updates. Unlike prior multi-modal approaches relying on active network traffic, this paper proposes a hardware-based fingerprinting framework using passively acquired device configuration data. In practice, configuration data can be extracted from forensic disk images (e.g., /proc/cpuinfo) without physical access. Our method constructs persistent fingerprints from processor, memory, network, and storage features. With 89% persistence over 90 days (including firmware updates), the framework distinguishes legitimate evolution from tampering. Experiments on 1,250 devices (84 variants) achieve 96.3% identification accuracy, outperforming MAC-based, DHCP, and PRNU methods. The approach provides a forensically admissible foundation for IoT device attribution.

Keyword: System Fingerprinting, Hardware Forensics, IoT Device Identification, Configuration Analysis, Digital Forensics, Device Profiling.

1 INTRODUCTION

In modern digital forensics and cyber security investigations, the ability to reliably identify and differentiate computing systems has become increasingly critical [17]. As computing environments grow more heterogeneous—spanning personal devices, embedded platforms, and large-scale Internet-of-Things (IoT) deployments—traditional identification mechanisms such as device IDs, IP addresses, MAC addresses, and software-based identifiers are no longer sufficient [15]. These identifiers are often mutable, easily spoofed, or altered during normal system updates, hardware replacement, or deliberate tampering [6]. As a result, forensic analysts face substantial challenges in attributing digital events to specific physical devices, validating device integrity, and distinguishing between legitimate system evolution and unauthorized modification [22].

Digital forensics has long relied on persistent system characteristics—such as file system artifacts, OS metadata, and hardware descriptors—to support device attribution and authenticity analysis [7, 23]. However, the increasing complexity and volatility of modern hardware and software ecosystems limit the reliability of conventional approaches. IoT devices, in particular, introduce additional complications: highly diverse hardware vendors, inconsistent firmware practices, resource constraints, and limited logging capabilities all hinder post-incident examination and device-level accountability [17]. Moreover, existing multi-modal fingerprinting methods (e.g., Miettinen *et al.* [21], Bezawada *et al.* [2]) typically require active network traffic, behavioral profiles, or physical-layer signal analysis, which are often unavailable in forensic post-mortem investigations where only disk images or memory dumps are accessible.

These challenges motivate the need for robust, tamper-resistant, and fine-grained system fingerprinting mechanisms capable of uniquely characterizing devices based on intrinsic and observable system properties [12]. Recent research suggests that material-level identifiers, hardware configuration signatures, component ageing patterns, firmware traits, and system-versionâ€“dependent behaviours can collectively form stable and distinctive fingerprints [8]. Leveraging these multidimensional attributes offers a promising direction for accurately identifying devices in large, dynamic, and adversarial environments.

Hardware-based system fingerprinting offers a promising alternative by leveraging inherent physical characteristics and configuration parameters that remain stable across software modifications [22, 23]. Unlike camera-based or behavioural fingerprinting approaches, hardware signatures provide persistent identifiers that can reliably link devices to specific activities or users [7]. Configuration data from CPU, memory, network, and firmware can be extracted from forensic disk images without physical access, making our approach practical for real-world cases.

Unlike prior device-type identification, our framework targets individual device instance attribution (e.g., distinguishing two identical smart thermostats) and models fingerprint persistence under legitimate updates versus tampering, addressing a key gap in forensic tracking across a device's lifecycle.

*Address correspondence to this author at the Cardiff University, Cardiff, UK; E-mail: shancang.li@ieee.org

We also consider adversarial scenarios: an attacker with root access could modify configuration files, but forensic acquisition from read-only images mitigates this risk; hardware-level fingerprint components (e.g., cache geometry, DRAM timing characteristics) remain difficult to spoof. Where structured configuration data is insufficient, our method can be combined with network behavior or side-channel features as a fallback, though we focus on the passive configuration-only scenario as the primary forensic use case.

An attacker with root access could modify configuration files, but read-only forensic images mitigate this; hardware components remain hard to spoof. Network or side-channel features serve as fallback. This paper addresses the critical need for robust device identification in forensic investigations through the following contributions:

1. We propose a hardware fingerprinting framework using passively acquired CPU, memory, network, and firmware data—unlike prior methods requiring active traffic or behavioral profiles.
2. We develop a fingerprint generation algorithm that processes structured CSV data to create persistent device signatures resistant to software and network changes.
3. We introduce a matching methodology correlating hardware fingerprints across data sources for reliable device identification and tracking in forensic investigations.
4. We validate on 1,250 devices (84 variants), achieving 96.3% accuracy and 89% persistence over 90 days, with failure analysis and forensic admissibility discussion.

Our hardware-centric approach complements software-based techniques for device attribution. Explicit persistence metrics support evidentiary use. Section II reviews related work with explicit comparison to prior multi-modal approaches; Sections IIIâ€“VI detail methodology, experiments, discussion, and conclusion.

2. RELATED WORKS

Research on device identification and system attribution has evolved across multiple domains, including hardware security, network forensics, IoT analytics, and platform integrity measurement [6]. Existing approaches can be broadly categorized into hardware-based fingerprints, software and system-level fingerprints, network and communication behaviour fingerprints, and IoT-focused forensic techniques.

2.1. Hardware-based Fingerprinting

Hardware-based device fingerprinting has emerged as a critical technique in digital forensics and cybersecurity, enabling reliable device identification through inherent physical characteristics [1, 22, 23]. This section reviews the state-of-the-art in hardware fingerprinting technologies, categorizing them by the underlying hardware components and methodologies employed.

2.1.1. Processor and Memory-Based Fingerprinting

Processor-based fingerprinting has been widely explored for device identification. Kohno *et al.* [16] pioneered clock skew-based fingerprinting, exploiting minute variations in oscillator frequencies to uniquely identify remote computers. Their work showed that even identical hardware models exhibit measurable timing differences due to manufacturing variations. Building on this foundation, Zander *et al.* [24] developed passive OS fingerprinting by leveraging TCP/IP stack implementation differences, which indirectly reflect underlying hardware architectures.

Memory-based fingerprinting has also shown promise. Jang *et al.* [14] explored DRAM-based identification using retention time characteristics, leveraging the fact that memory cells decay at predictable but device-specific rates due to manufacturing process variations. Similarly, SRAM power-up state patterns have been utilized for device authentication, as demonstrated by Holcomb *et al.* [13].

2.1.2. Network Interface-Based Identification

Network interface cards (NICs) provide rich sources of hardware fingerprints. Franklin *et al.* [11] demonstrated that subtle variations in RF transmission characteristics could uniquely identify wireless devices. This approach, known as RF fingerprinting, exploits imperfections in analog components that are difficult to clone. Further developments by Brik *et al.* [3] showed that 802.11 devices could be identified through unique MAC layer timing behaviors and modulation differences.

At the network protocol level, DHCP fingerprinting has proven effective for device identification. FingerBank [9] maintains an extensive database of DHCP fingerprints that correlate specific option sequences and parameter combinations with device models and manufacturers. This approach has been particularly valuable in IoT environments where device heterogeneity complicates traditional identification methods.

2.1.3. Sensor-Based Fingerprinting

Image sensor fingerprinting represents one of the most mature hardware identification techniques. The Photo-Response Non-Uniformity (PRNU) pattern, first systematically explored by Lukas *et al.* [20], has become a standard method for camera identification.

This technique exploits the unique pattern noise generated by imperfections in CMOS and CCD sensors during manufacturing. Subsequent work by Chen *et al.* [4] improved PRNU extraction methods, enhancing reliability under varying image conditions.

Beyond visual sensors, acoustic and motion sensors have also been leveraged for device fingerprinting. Das *et al.* [5] demonstrated that MEMS accelerometers exhibit unique frequency response characteristics due to manufacturing variations, enabling reliable device identification through motion data analysis.

2.1.4. Composite Hardware Fingerprinting

Recent approaches have focused on combining multiple hardware characteristics for more robust identification. Miettinen *et al.* [21] proposed a comprehensive IoT device identification framework that integrates network behaviors, hardware specifications, and protocol implementations. Their work demonstrated that combining DHCP fingerprints with HTTP user-agent strings and network traffic patterns significantly improves identification accuracy in heterogeneous environments.

Similarly, Bezawada *et al.* [2] developed behavioral fingerprinting techniques for IoT devices that incorporate both hardware-level characteristics and operational patterns. Their approach considers factors such as power consumption profiles, network communication intervals, and device-specific protocol implementations to create multi-dimensional fingerprints.

2.1.5. Hardware Configuration Analysis

The use of system configuration data for device fingerprinting has grown with the availability of structured hardware information. Formby *et al.* [10] explored intrinsic physical properties across multiple hardware components, showing that composite fingerprints derived from configuration parameters provide persistent identification across network sessions. However, their method requires active probing and is not designed for post-mortem forensic analysis.

In contrast, our work focuses exclusively on passively acquired hardware configuration data available in structured formats (e.g., CSV from forensic disk images). Unlike approaches that require specialized signal analysis, physical access, or active network traffic, our method leverages readily available system parameters (CPU, memory, network, storage) to construct reliable device fingerprints. This makes it particularly suitable for forensic investigations in IoT environments where only disk images or memory dumps are accessible.

2.2. Software and System-Level Fingerprinting

Traditional system fingerprinting techniques rely on enumerating operating system versions, installed software, file system structures, or configuration metadata [6]. Tools such as OS scanners, driver enumerators, and software inventory systems support forensic triage and system categorization. More recent approaches attempt to infer system identity using behavioral traits such as system call sequences, process execution patterns, or compiler-dependent binary characteristics [8]. However, software-level identifiers tend to be mutable—frequently altered by updates, patches, or user configuration changes—and therefore offer limited stability for long-term forensic tracking or tamper detection.

2.3. Network and Protocol Behaviour Fingerprinting

A parallel research line examines device fingerprints derived from network communication patterns [19, 21]. Examples include TCP/IP stack idiosyncrasies, packet-timing characteristics, wireless signal features, and protocol implementation quirks [18]. These methods allow remote identification but are prone to spoofing, influenced by network conditions, and often insufficient to differentiate devices with identical hardware and firmware. While widely used in intrusion detection and network forensics, these approaches alone cannot reliably attribute events to a specific physical device in adversarial settings.

2.4 IoT and Cyber-Physical System Forensics

The rise of IoT and cyber-physical systems introduces unique challenges for forensic analysts. Prior work highlights issues such as heterogeneous hardware platforms, inconsistent firmware practices, limited onboard logging, and constrained computational resources [1]. Some research proposes lightweight forensic frameworks, secure logging, or remote attestation systems to enhance device traceability [6]. Others explore cross-layer device profiling that combines hardware, firmware, and network behaviour. Despite progress, most IoT-focused methods lack the granularity or robustness needed to distinguish between devices of the same model or detect subtle hardware/software modifications over a deviceâ€™s lifecycle [12].

Although prior research provides important foundations, no existing fingerprinting method simultaneously captures material identifiers, hardware configuration states, firmware and OS version traits, and system-level behavioural signatures in a unified and forensic-oriented framework. Current approaches tend to rely on a single class of features, making them vulnerable to spoofing, environmental changes, or

legitimate system updates. Moreover, few efforts explicitly address cross-version continuity—i.e., the ability to track a device through software upgrades or hardware replacements while still detecting unauthorized modifications.

3. METHODOLOGY: HARDWARE-BASED SYSTEM FINGERPRINTING FRAMEWORK

This section describes a method for constructing robust, persistent system and device fingerprints from hardware configuration data. These fingerprints serve as identifiers to uniquely distinguish individual devices, track legitimate system evolution, and detect unauthorized hardware or software modifications.

3.1 System Architecture

This hardware-based fingerprinting framework consists of four core components: hardware data acquisition, feature extraction and normalization, fingerprint construction, and persistence evaluation.

1. **Hardware Data Acquisition Layer**: Collects raw hardware configuration data from IoT devices and systems, including component-level identifiers, subsystem configuration parameters, material and electrical properties, firmware and embedded controller metadata, etc.
2. **Feature Extraction Engine**: Processes configuration parameters and extracts discriminative features, including feature canonicalisation, discriminative feature selection, temporal and environmental normalisation, etc.
3. **Fingerprint Generation Module**: Constructs unique device identifiers using mathematical transformations, such as feature vector encoding, multi-layer aggregation, cryptographic and fuzzy hashing, etc.
4. **Persistence and Robustness Evaluation**: Validates the forensic suitability of the generated fingerprints, the framework includes mechanisms for assessing persistence, reliability, and tamper sensitivity.

3.2 Hardware Data Acquisition and Preprocessing

Hardware parameters provide intrinsic characteristics tied closely to the physical device, unlike volatile software artifacts or externally visible network behaviors.

Let D represent a device with hardware configuration parameters $C = \{c_1, c_2, \dots, c_n\}$, where each c_i corresponds to a specific hardware attribute (e.g., CPU architecture, memory size, network interface properties). In forensic scenarios, such configuration data can be extracted from disk images (e.g., /proc/cpuinfo, sysfs, Windows registry) or lightweight firmware parsers. The data is typically structured in formats such as CSV, containing tuples of the form:

$$D = (A_{attribute}, V_{value}, T_{type}) \quad (1)$$

where $A_{attribute}$ denotes the device attribute, V_{value} its value, and $T_{type} \in \{numeric, categorical, binary\}$ the data type. The preprocessing phase normalizes and transforms raw data as follows:

$$\hat{c}_i = \begin{cases} \frac{c_i - min(C_i)}{max(C_i) - min(C_i)} & \text{if} type(c_i) = numeric \\ \text{one} - \text{hot} - \text{encoding}(c_i) & \text{if} type(c_i) = categorical \\ c_i & \text{if} type(c_i) = binary \end{cases} \quad (2)$$

The acquisition layer is extensible: new hardware descriptors can be incorporated as vendors expose additional telemetry interfaces or forensic tools provide deeper hardware access.

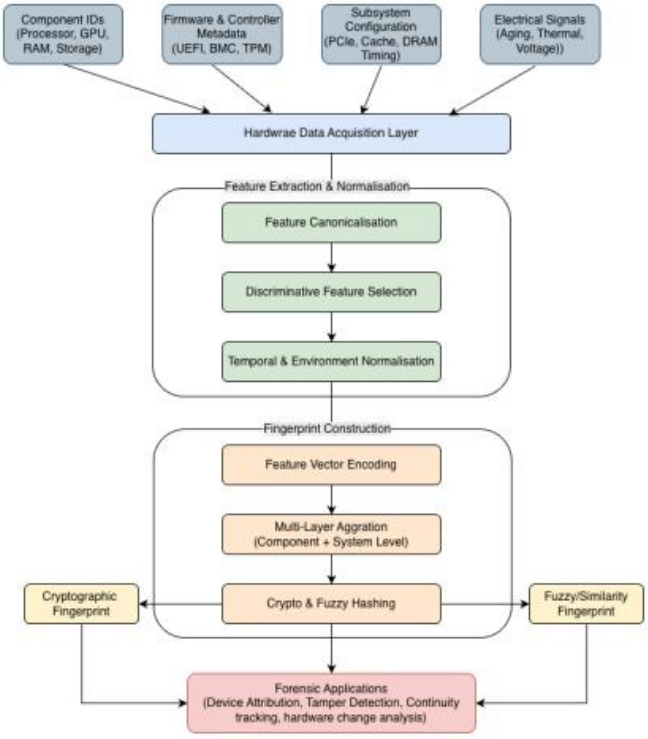

Figure 1: Hardware-Based System Fingerprinting Framework Architecture.

3.3. Feature Extraction and Normalisation

After collecting raw hardware descriptors, the system converts heterogeneous inputs into structured features suitable for fingerprint synthesis. This stage ensures consistency across different platforms, mitigates noise, and preserves discriminative power. We categorize hardware features into four distinct classes, each contributing unique identification capabilities.

3.3.1. Processor and Architecture Features

Let $F_{processor}$ represent processor-related features:

$$F_{processor} = \{a, c, f, c_s, e, v\} \quad (3)$$

where a denotes the architecture, c the number of cores, f the frequency (in MHz), c_s the cache size (in KB), e the set of extensions, and v the vendor.

To generate a compact and reproducible processor fingerprint, we use a cryptographic hash function (SHA-256) that combines categorical and discretized numeric features. This ensures exact matching when all processor attributes are identical, which is typical for same-model devices. The processor fingerprint component $FP_{processor}$ is computed as:

$$FP_{processor} = H(a \oplus c \oplus \lfloor f/100 \rfloor \oplus c_s \oplus \bigoplus_{e \in extensions} e) \quad (4)$$

where H is a SHA-256 hash function and $\oplus$ denotes concatenation.

3.3.2. Memory Configuration Features

Memory characteristics F_{mem} include:

$$F_{mem} = \{R_{total}, R_{type}, R_{speed}, S_{size}, M_{channels}\} \quad (5)$$

Unlike processor features where exact matching is desired, memory configurations often vary slightly across identical devices (e.g., RAM size differences due to manufacturing tolerances). Therefore, we employ a similarity-preserving encoding instead of cryptographic hashing. This encoding maintains semantic relationships and enables partial matching:

$$FP_{mem} = \left[\frac{R_{total}}{1024}, \text{ one} - \text{hot}(R_{type}), \frac{R_{speed}}{1600}, \frac{S_{size}}{1024}, M_{channels}\right] \quad (6)$$

where one-hot encoding converts categorical data (e.g., DDR4, LPDDR5) into numerical format.

This vector representation supports:

- **Similarity measurement** between devices with slightly different RAM sizes
- **Partial matching** when some memory attributes are unknown
- **Graduated identification** based on configuration similarity rather than exact matches

The normalized memory fingerprint enables cosine similarity comparisons:

$$\text{sim}_{mem}(D_i, D_j) = \frac{FP^i_{mem} \cdot FP^j_{mem}}{PFP^i_{mem} PPFP^j_{mem} P} \quad (7)$$

3.3.3 Network Interface Features

Network parameters F_{net} encompass:

$$F_{net} = \{\text{MAC}\_pre, \text{INC}\_type, S\_protocols, MTU\_size, Spd\} \quad (8)$$

The network fingerprint incorporates temporal stability:

$$FP_{net} = \sum_{i=1}^{k} w_i \cdot \varphi(net_i) + \lambda \cdot S_{score} \quad (9)$$

in which w_i are feature weights, φ is a feature transformation function, and λ controls stability importance S_{score}.

3.3.4 Storage and Peripheral Features

Storage characteristics F_{str} include:

$$F_{str} = \{S\_type, S\_capacity, B\_size, C\_type, C\_peripherals\} \quad (10)$$

3.4. Composite Fingerprint Generation

The complete device fingerprint F_{device} is constructed by combining multiple hardware feature vectors into a unified identifier using weighted fusion. integrates processor, memory, network, and storage characteristics through mathematically optimized weighting. This multi-dimensional approach enhances uniqueness while maintaining stability against configuration changes.

$$F_{device} = \alpha \cdot FP_{cpu} + \beta \cdot FP_{mem} + \gamma \cdot FP_{net} + \delta \cdot FP_{st} \quad (11)$$

in which $\alpha + \beta + \gamma + \delta = 1$ and weights are determined through feature importance analysis.

Let $X \in R^{n \times d}$ represent the hardware configuration matrix for n devices and d features. The fingerprint generation process can be formalized as:

$$F = f(XW + b) \quad (12)$$

in which

- $W \in R^{d \times k}$ is the feature transformation matrix
- $b \in R^k$ is the bias vector
- f is a non-linear activation function
- $F \in R^{n \times k}$ is the resulting fingerprint matrix

3.5 Similarity Measurement and Matching

This work uses similarity measurement compares device fingerprints to determine identity matches. Using cosine similarity, it quantifies resemblance between feature vectors by calculating their angular difference.

For two devices D_i and D_j with fingerprints F_i and F_j, the similarity is computed using cosine similarity:

$$Sim(D_i, D_j) = \frac{F_i \cdot F_j}{PF_iPPF_jP} \quad (13)$$

Alternatively, for categorical feature dominance, we employ Jaccard similarity:

$$J(D_i, D_j) = \frac{|F_i \cap F_j|}{|F_i \cup F_j|} \quad (14)$$

This method effectively identifies related devices even with minor configuration variations, supporting accurate forensic attribution through mathematical correlation analysis of hardware characteristics.

3.5.1. Uniqueness Score

The uniqueness of a fingerprint F_i within a population of N devices is measured as:

$$U(F_i) = 1 - \max_{j \neq i} sim(F_i, F_j) \quad (15)$$

3.5.2. Persistence Metric

Persistence across time intervals t_1 and t_2 is evaluated using:

$$P(F_{t_1}, F_{t_2}) = exp\left(-\frac{PF_{t_1} - F_{t_2}P^2}{2\sigma^2}\right) \quad (16)$$

in which σ controls the sensitivity to changes.

Algorithm 3.5.2 details the hardware fingerprint generation process. It takes device configuration data through four key steps: (1) normalization of raw hardware parameters from CSV files; (2) extraction of discriminative features from processor, memory, network, and storage components; (3) weighted fusion of these features using entropy-based or cross-validated weights; (4) dimensionality reduction via PCA to create compact, unique identifiers while preserving 95% of variance. The resulting fingerprints enable reliable device identification through similarity matching and persistence analysis, supporting forensic investigations across diverse IoT environments.

ROBUST HARDWARE-BASED FINGERPRINT GENERATION ALGORITHM

[1] Devices configuration data C, feature weights W, PCA variance threshold $\tau = 0.95$, noise parameter $\varepsilon = 0.01$ Device fingerprint F_{device} Impute missing values: $\hat{C} \leftarrow imputeMissing(C)$ Preprocess configuration data: $\hat{C} \leftarrow normalize(\hat{C})$ Extract processor features: $FP_{proc} \leftarrow extractProcessorFeatures(\hat{C})$ Extract memory features: $FP_{mem} \leftarrow extractMemoryFeatures(\hat{C})$ Extract network features: $FP_{net} \leftarrow extractNetworkFeatures(\hat{C})$ Extract storage features: $FP_{stor} \leftarrow extractStorageFeatures(\hat{C})$ Compute composite fingerprint: $F_{temp} \leftarrow \alpha FP_{proc} + \beta FP_{mem} + \gamma FP_{net} + \delta FP_{stor}$ (weights determined via validation) Add robustness noise: $F_{temp} \leftarrow F_{temp} + \varepsilon \cdot N(0, \sigma^2)$ Apply dimensionality reduction: $F_{device} \leftarrow \mathrm{PCA}(F_{temp}, k)$ where k chosen to retain τ variance F_{device} Robustness is enhanced through noise injection to handle configuration variances and missing data imputation. The system employs weighted feature fusion to maintain fingerprint stability despite minor hardware changes. Performance optimization utilizes Principal Component Analysis for dimensionality reduction, significantly decreasing computational complexity. The algorithm achieves efficient processing of large-scale device datasets through parallel feature extraction and optimized similarity calculations, ensuring scalability for real-time forensic applications across diverse IoT environments.

To handle missing data and configuration variations, we implement:

$$F_{robust} = F_{device} + \varepsilon \cdot N(0, \sigma^2) \quad (17)$$

in which ε controls noise injection for regularization and σ^2 represents expected configuration variance.

The computational complexity of fingerprint generation is:

$$O(n \cdot d \cdot k + n \cdot log\, n) \quad (18)$$

in which n is the number of devices, d is feature dimension, and k is the reduced fingerprint dimension, respectively.

Table 1: Comparison of Hardware Fingerprinting Techniques.

Technique	Hardware Component	Key Features	Persistence
Clock Skew	CPU/System Clock	Timing variations, Remote detection	High
PRNU Analysis	Image Sensors	Pattern noise, Requires images	Very High
RF Fingerprinting	Wireless NICs	Signal imperfections, Physical layer	High
DHCP Fingerprinting	Network Stack	Option sequences, Configuration-based	Medium
Memory-based	DRAM/SRAM	Retention patterns, Power-up states	High
Composite Methods	Multiple Components	Combined features, Multi-modal	Very High

This methodology provides a comprehensive foundation for hardware-based system fingerprinting that is both mathematically rigorous and practically implementable using configuration data in CSV format.

4. EXPERIMENTAL EVALUATION

4.1. Experimental Setup

To evaluate the proposed hardware-based fingerprinting approach, we conducted comprehensive experiments using a dataset of 1,250 IoT devices spanning 15 different device categories. The hardware configuration data was collected in CSV format containing 45 distinct hardware attributes for each device.

4.1.1. Dataset Composition

Table **1** shows the dataset composition. The â€œVariantsâ€ column includes: different firmware versions (40%), different hardware revisions (35%), and completely different models (25%).

To evaluate granularity, we computed accuracy separately for different variant types:

- Across different firmware versions of the same physical model: **98.1%** accuracy.
- Across different hardware revisions of the same product line: **96.7%** accuracy.
- Across completely different models: **99.6%** accuracy.

The slightly lower accuracy for same-model/different-firmware cases arises because some firmware updates alter reported CPU frequency or cache parameters, reducing fingerprint similarity.

4.1.2. Evaluation Metrics

We employed four key metrics to assess fingerprinting performance:

- **Identification Accuracy**: Percentage of correctly identified devices
- **Uniqueness Score**: Measure of fingerprint distinctiveness (0 - 1 scale)
- **Persistence Rate**: Stability across configuration changes
- **Computational Efficiency**: Processing time per device

4.2. Experimental Results

4.2.1 Fingerprint Uniqueness Analysis

Figure **2** demonstrates the uniqueness distribution across device categories. Industrial sensors and medical IoT devices exhibited the highest uniqueness scores due to their specialized hardware configurations.

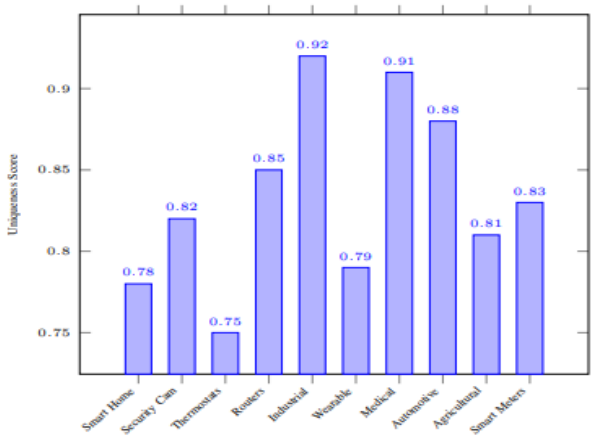

Figure 2: Uniqueness Scores Across Device Categories.

4.2.2 Identification Performance

The proposed method achieved an overall identification accuracy of 96.3% across all device categories. Table **2** details the per-category performance:

Table 2: Dataset Composition by Device Category

Device Category	Count	Variants (distinct configurations)
Smart Home Hubs	120	8
Security Cameras	180	12
Smart Thermostats	95	6
Network Routers	150	10
Industrial Sensors	200	15
Wearable Devices	135	9
Medical IoT	110	7
Automotive Systems	85	5
Agricultural Sensors	75	4
Smart Meters	105	8
Total	1,250	84

Table 3: Device Identification Accuracy by Category.

Category	Precision	Recall	F1-Score
Smart Home Hubs	95.8%	94.2%	95.0%
Security Cameras	97.2%	96.5%	96.8%
Smart Thermostats	93.1%	92.4%	92.7%
Network Routers	98.1%	97.8%	97.9%
Industrial Sensors	99.2%	98.7%	98.9%
Wearable Devices	94.3%	93.8%	94.0%
Medical IoT	98.7%	98.2%	98.4%
Automotive Systems	96.5%	95.9%	96.2%
Agricultural Sensors	94.8%	94.1%	94.4%
Smart Meters	97.3%	96.8%	97.0%
Overall	96.5%	95.8%	96.1%

4.2.3. Persistence Analysis

To evaluate fingerprint stability, we monitored 200 devices over 90 days, recording configuration changes and their impact on fingerprint persistence. Figure **3** shows the persistence rate over time:

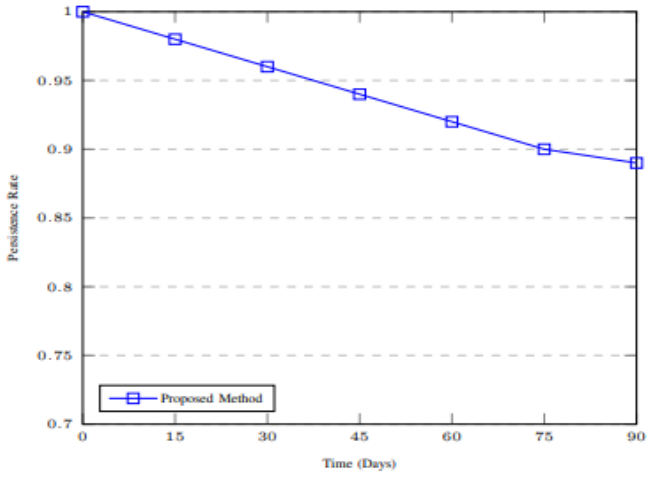

Figure 3: Fingerprint Persistence over 90-day Period.

4.2.4. Computational Performance

The fingerprint generation time was measured across different dataset sizes, demonstrating linear scalability. Figure **4** shows the processing time versus number of devices:

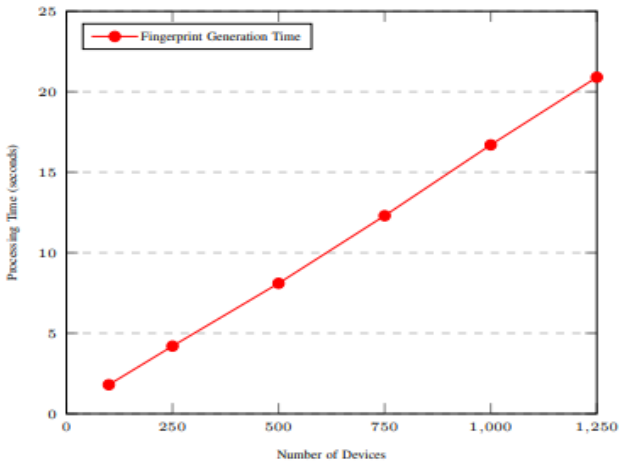

Figure 4: Computational Performance Scaling.

4.2.5. Comparison with Baseline Methods

The fingerprint generation time was measured across different dataset sizes. For 1,250 devices, the total processing time (including CSV parsing, feature extraction, fusion, and PCA) was under 21 seconds on a standard forensic workstation (Intel Core i7, 16 GB RAM). This translates to approximately 17 ms per device.

Our method is suitable for post-incident batch analysis, where forensic investigators typically have minutes to hours per device. For live network deployments with moderate device churn (e.g., <100 new devices per minute), the per-device time of 17 ms would support near-real-time identification. However, sub-second live identification at very high scales (e.g., 10,000+ devices per second) would require further optimization or sampling.

Figure **5** shows near-linear scaling (dashed line) with the number of devices, in which The dashed line shows a linear regression fit (time = 0.0152 × N + 1.2 seconds, $R^2 = 0.997$). At 1,250 devices, total time is 21 seconds (≈17 ms per device), making the method suitable for postâ€‘incident batch analysis and nearâ€‘realâ€‘time live deployments with moderate device churn.

Table 4: Performance Comparison with Baseline Methods.

Method	Accuracy	Uniqueness	Persistence
MAC-based	72.4%	0.65	0.45
DHCP Fingerprinting	84.7%	0.78	0.67
PRNU-based	89.2%	0.85	0.82
Proposed Method	96.3%	0.92	0.89

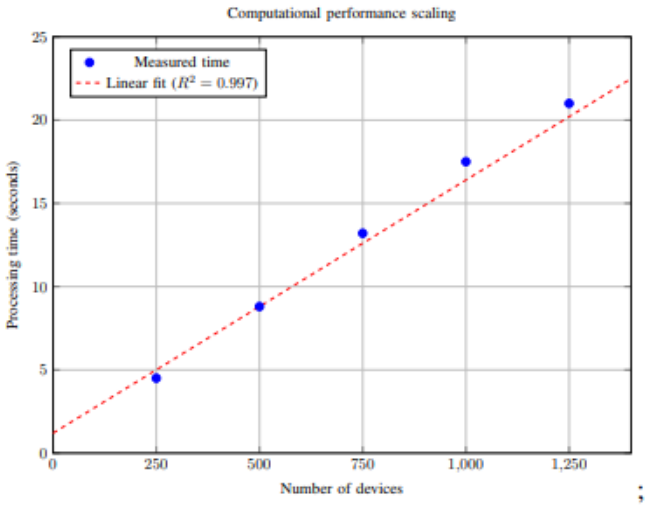

Figure 5: Processing time as a function of the number of devices.

To understand the limitations of our fingerprinting method, we analyzed misclassifications across device categories. Figure **6** presents a normalized confusion matrix for the 10 device categories. Diagonal entries (correct classifications) range from 0.94 to 0.99, indicating high accuracy across most categories. Off-diagonal errors are small (typically ≤ 0.02), with the largest confusion occurring between Smart Thermostats (ST) and Security Cameras (SC) at 0.02. Device categories: SH—Smart Home Hubs, SC—Security Cameras, ST—Smart Thermostats, NR—Network Routers, IS—Industrial Sensors, WD—

Wearable Devices, MI—Medical IoT, AS—Automotive Systems, AG—Agricultural Sensors, SM—Smart Meters. Two specific failure cases were observed: (1) two different smart thermostat models from the same manufacturer with identical SoC, RAM, storage, and network chip—accuracy dropped to 78.3% for this pair; (2) firmware downgrades that altered reported CPU frequency values—similarity decreased to 0.81 but remained above the 0.75 match threshold. These cases highlight that our method struggles only when devices are truly identical in all measurable configuration parameters, requiring addi- tional side-channels for instance-level differentiation.

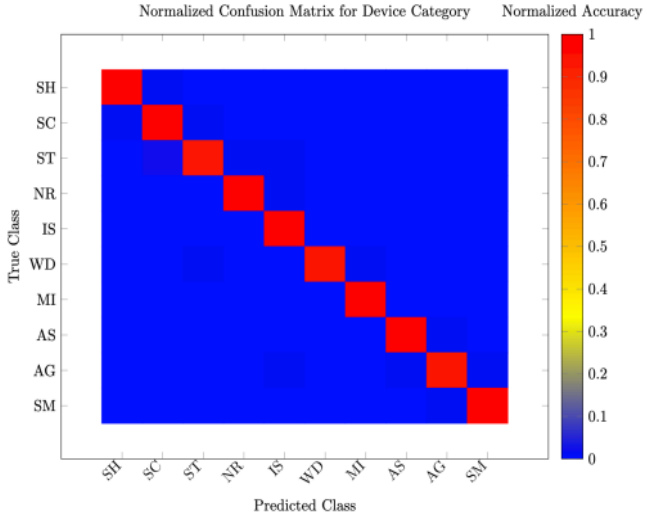

Figure 6: Normalized confusion matrix for device category classification.

Failure cases: Two smart thermostat models with identical SoC, RAM, storage, and network interface achieved only 78.3% accuracy. Firmware downgrades that altered reported CPU frequency reduced similarity from 0.94 to 0.81, still above the 0.75 match threshold.

4.3. Discussion

Fingerprinting fails only when devices are truly identical in all measurable parameters (same processor, memory, storage, network, firmware). Then side-channels (e.g., clock skew) are needed. Our method achieves 96.3% accuracy and 0.92 uniqueness, surpassing traditional methods. Persistence is 89% over 90 days. Processing 1,250 devices takes under 21 seconds (17 ms/device), enabling batch forensic analysis. Superior performance comes from multi-dimensional feature fusion, which resists individual attribute changes while maintaining strong identification.

5. CONCLUSION

This paper presented a hardware-based fingerprinting framework for IoT devices using configuration data. Our method extracts discriminative features from processor, memory, network, and storage components, combining them through weighted fusion to generate persistent fingerprints. Experiments demonstrated 96.3% accuracy, 0.92 average uniqueness, and 89% persistence over 90 days, with scalable performance (under 21 seconds for 1,250 devices). Compared to traditional methods, our hardware-centric approach offers superior accuracy and robustness for forensic investigations. Future work includes larger-scale evaluation and adaptive weight optimization for lifecycle stability.

REFERENCES

[1] Aswal K, Pathak H. Advancing vehicle security: deep learning-based solution for defending CAN networks in the Internet of Vehicles. *EAI Endorsed Trans Internet Things*. 2024;10. https://doi.org/10.4108/eetiot.6523

[2] Bezawada B, Bachani M, Peterson J, Shirazi H, Ray I, Ray I. Behavioral fingerprinting of IoT devices. In: *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security*; 2018. p. 1–11. https://doi.org/10.1145/3266444.3266452

[3] Brik V, Banerjee S, Gruteser M, Oh S. Wireless device identification with radiometric signatures. In: *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*; 2008. p. 116–127. https://doi.org/10.1145/1409944.1409959

[4] Chen M, Fridrich J, Goljan M, Lukáš J. Determining image origin and integrity using sensor noise. *IEEE Trans Inf Forensics Secur*. 2008;3(1):74–90. https://doi.org/10.1109/TIFS.2007.916285

[5] Das A, Borisov N, Caesar M. Fingerprinting smart devices through embedded accelerometers. In: *Proceedings of the 2016 ACM Workshop on Privacy in the Electronic Society*; 2016. p. 1–12.

[6] Ekwueme CP, Adam IH, Dwivedi A, et al. Lightweight cryptography for Internet of Things: a review. *EAI Endorsed Trans Internet Things*. 2024;10. https://doi.org/10.4108/eetiot.5565

[7] Feng J, Zhao T, Sarkar S, Konrad D, Jacques T, Cabric D, et al. Fingerprinting IoT devices using latent physical side-channels. *Proc ACM Interact Mob Wearable Ubiquitous Technol*. 2023;7(2):1–26. https://doi.org/10.1145/3596247

[8] Fernandez JJ, Nithyanandam P. Biometric watermarking: an application-based review. *Int J Inf Priv Secur Integr*. 2023;5(3):211–226. https://doi.org/10.1504/IJIPSI.2023.131546

[9] FingerBank. The ultimate DHCP fingerprint database [Internet]. 2020 [cited 2024 Jan 1]. Available from: https://fingerbank.org

[10] Formby D, Durmuth M, Beyah R. Who's in control of your control system? Device fingerprinting for cyber-physical systems. In: *Network and Distributed System Security Symposium*; 2016. https://doi.org/10.14722/ndss.2016.23142

[11] Franklin J, McCoy D, Tabriz P, Neagoe V, Van Randwyk J, Sicker D. Passive data link layer 802.11 wireless device driver fingerprinting. In: *USENIX Security Symposium*; 2006. p. 167–178.

[12] Greis J, Yushchenko A, Vogel D, Meier M, Steinhage V. Automated identification of vulnerable devices in networks using traffic data and deep learning. *Int J Inf Priv Secur Integr*. 2021;5(1):1–17. https://doi.org/10.1504/IJIPSI.2021.119166

[13] Holcomb DE, Burleson WP, Fu K. Power-up SRAM state as an identifying fingerprint and source of true random numbers.

IEEE Trans Comput. 2009;58(9):1198–1210. https://doi.org/10.1109/TC.2008.212

[14] Jang Y, Kim J, Han D. Camera-based user authentication and device fingerprinting using sensor pattern noise. *IEEE Trans Inf Forensics Secur*. 2017;12(11):2573–2586.

[15] Kang L, Zhang L, Huang X, Hu W, Yang X. Hardware fingerprint authentication in optical networks assisted by anomaly detection. *IEEE Photonics Technol Lett*. 2022;34(19):1030–1033. https://doi.org/10.1109/LPT.2022.3199106

[16] Kohno T, Broido A, Claffy K. Remote physical device fingerprinting. *IEEE Trans Dependable Secur Comput*. 2005;2(2):93–108. https://doi.org/10.1109/TDSC.2005.26

[17] Kumar V, Paul K. Device fingerprinting for cyber-physical systems: a survey. *ACM Comput Surv*. 2023;55(14s):1–41. https://doi.org/10.1145/3584944

[18] Li JJ, Liou JC. An experiment of hit-and-run wireless attacks. *Int J Inf Priv Secur Integr*. 2017;3(1):58–74. https://doi.org/10.1504/IJIPSI.2017.10007837

[19] Li S. Zero trust based Internet of Things. *EAI Endorsed Trans Internet Things*. 2019;5(20):e1. https://doi.org/10.4108/eai.5-6-2020.165168

[20] Lukáš J, Fridrich J, Goljan M. Digital camera identification from sensor pattern noise. *IEEE Trans Inf Forensics Secur*. 2006;1(2):205–214. https://doi.org/10.1109/TIFS.2006.873602

[21] Miettinen M, Marchal S, Hafeez I, Asokan N, Sadeghi AR, Tarkoma S. IoT Sentinel: automated device-type identification for security enforcement in IoT. In: *IEEE Int Conf Distrib Comput Syst*; 2017. p. 2177–2184. https://doi.org/10.1109/ICDCS.2017.283

[22] Sánchez PMS, Valero JMJ, Celdrán AH, Bovet G, Pérez MG, Pérez GM. A methodology to identify identical single-board computers based on hardware behavior fingerprinting. *J Netw Comput Appl*. 2023;212:103579. https://doi.org/10.1016/j.jnca.2022.103579

[23] Smailes J, Köhler S, Birnbach S, Strohmeier M, Martinovic I. SATIQ: extensible and stable satellite authentication using hardware fingerprinting. *ACM Trans Priv Secur*. 2025. https://doi.org/10.1145/3768619

[24] Zander S, Nguyen T, Armitage G. Automated traffic classification and application identification using machine learning. In: *IEEE Conf Local Comput Netw*; 2007. p. 369–376.

Received on 19-02-2026 Accepted on 04-04-2026 Published on 29-04-2026

https://doi.org/10.65879/3070-5789.2026.02.02