CD^2A: Continuous Device-to-Device Authentication Exploiting Crystal Oscillator Impurities

Muthupavithran Selvam
City, University of London
London, UK
muthupavithran.selvam@city.ac.uk

Zeba Khanam
British Telecom Security Research
London, UK
zeba.2.khanam@bt.com

Amit Kumar Singh
University of Essex
Essex, UK
a.k.singh@essex.ac.uk

Zhan Cui
British Telecom Security Research
London, UK
zhan.cui@bt.com

Rajarajan Muttukrishnan
City, University of London
London, UK
r.muttukrishnan@city.ac.uk

***Abstract*—Every day, on average, eight cybercrimes targeting IoT networks occur, leading to a cumulative loss of $10 million. The main reason for these attacks is the ability of unauthorized devices to gain access to IoT networks by replicating the hardware and software configurations of authorized devices. To tackle this pressing issue, cryptographic keys are used to authenticate devices in IoT networks. Given the requirements of this process, authentication is performed once at the beginning. However, this makes devices susceptible to cyber-attacks like spoofing, Sybil attacks, distributed denial-of-service (DDoS), and Advanced Persistent Threats (APT). To address this, we propose a novel Continuous Device-to-Device Authentication (CD^2A) framework based on two components: 1) Identity Establishment and 2) Continuous Authentication. In the Identity Establishment phase, we use manufacturing imperfections to model unique device behaviours. A novel device fingerprint algorithm is proposed that leverages impurities of built-in components of the device, like crystal oscillators, and is measured in a graphical processing unit (GPU) by isolating each core at a time in the central processing unit (CPU). In the Continuous Authentication phase, we implement a dynamic timeline to establish device identity at regular intervals. Each device is continuously authenticated by using machine learning techniques. To protect devices from cyber-attacks like spoofing, Sybil attacks, DDoS, and APT, we track device legitimacy by calculating the Device Authentication Score (DAS) and the Device Risk Factor (DRF) in view of varying security risks. We evaluate the CD^2A framework on an IoT system with 11 devices. The CD^2A framework achieves an average authentication accuracy of 99.96% and 99.85% when used in tandem with CatBoost and XGBoost machine learning algorithms, respectively.**

***Index Terms*—IoT networks, Continuous Device Authentication, and Device fingerprinting.**

I. Introduction

The Internet of Things (IoT) has experienced significant growth across various application domains, becoming a crucial component of daily life for consumers and operational processes for enterprises. As IoT systems become integral to critical infrastructure, they have increasingly attracted malicious actors who use various techniques to gain unauthorized access to network devices. Major security breaches in recent years have often resulted from unauthorized devices exploiting system vulnerabilities by spoofing or replicating hardware and software configurations of legitimate devices [1]. A barrage of cybersecurity attacks like device spoofing [2], unauthorised device deployment [3], and Sybil Attacks [4] have emerged. In these attacks, threat actors masquerade the identity of the authorised device to infiltrate the network. Traditionally, device-to-device (D^2D) authentication has relied on cryptographic protocols such as Public Key Infrastructures (PKI), Pre-Shared Keys (PSK), and the Challenge-Handshake Authentication Protocol (CHAP) [5]–[7]. However, these protocols are computationally extensive to deploy on IoT devices. To this end, lightweight authentication techniques like Lightweight Directory Access Protocol (LDAP) and Lightweight Authentication Protocol to Secure End and Edge Devices (LAPE2D) have been proposed. However, they are vulnerable to other sophisticated security attacks like side channel and injection attacks [8]. These vulnerabilities highlight the need for continuous device-to-device authentication (CD^2A). Existing research on continuous authentication often employs bio-metrics and behaviour

patterns [9] to authenticate users. Some studies have explored continuous device-to-device authentication by implementing iterative methods, which repeat the authentication process at regular intervals using session IDs and token generation. However, they inherit the weaknesses of static D2D authentication techniques, making them vulnerable to many cyber attacks and bringing overhead to resource-constrained IoT devices [10] [11].

Our work took inspiration from user continuous authentication where the user's unique behavioural identity is the core basis for gaining access [12]–[15]. Our proposed system does not replace static authentication. Instead, it serves as an additional layer of security for the dynamic IoT environment. Following the initial authentication, our Continuous Device-to-Device Authentication (CD^2A) system takes over, continuously verifying a device's authenticity based on its unique behaviour. This approach operates in tandem with existing IoT static authentication methods, enhancing overall security by ensuring that IoT devices are authenticated at the beginning of the process and maintaining their secure status throughout the device's operation.

II. Related Works

In recent years, a new area of research has gained attention: device fingerprinting. This approach aims to create a behavioural identity for devices, enhancing security by uniquely identifying each device based on its inherent characteristics and behaviours. The most commonly used feature is Channel State Information (CSI), which refers to the characteristics of a wireless signal that uniquely identifies a device when it connects to an IoT network. When a device establishes a communication link with a router, the CSI properties are analyzed to create a device fingerprint. This fingerprinting process involves measuring various channel characteristics such as amplitude, phase, and frequency response, which are unique to each device due to hardware imperfections [16]–[19]. It was noted that CSI-based device fingerprint properties are susceptible to environmental noise that interferes with the signal. Also, as the number of devices increased in the network, the router became a bottleneck for authentication. Existing work has extended CSI-based device identification to the domain of CSI-based D2D authentication [20] by introducing a protocol with two primary phases: 1) Mutual authentication and 2) Continuous Authentication. In Mutual Authentication, devices exchange and verify identities using CSI-derived session keys. Each device employs a dynamic function in the Continuous Authentication phase to ensure secure communication. However, for mobile IoT devices, CSI-derived keys are impractical because CSI changes and is affected by environmental noise as the device moves.

In addition to CSI-based device fingerprinting, crystal oscillator-based impurities have been used to establish the unique identity of devices [21] [22]. These impurities cause imperfect frequency outputs in various chip components (such as the CPU, GPU, USB driver, and Ethernet), which manifest as performance differences [23], which can be measured as a clock cycle. One of the early works [24] employed Crystal oscillator impurities to fingerprint devices like personal computers. This involved executing a cyclic task in the CPU and measuring its performance using dynamic time wrapping. This task was divided into sub-tasks and scheduled to different cores. However, precise performance measurement was challenging due to variable frequency and process scheduling constraints. Subsequent studies [25] and [26] addressed these limitations by isolating one CPU core and maintaining a fixed frequency. The cycle count was then measured in the GPU. Although this method mitigated noise measurement issues, it introduced the risk of the isolated core becoming unavailable during the identification process. Additionally, The functions and processes for measuring the imperfections could affect fingerprint generation and the system's reliability. Over time, components can degrade due to device ageing and other environmental factors, compromising the device's identity. Creating a unique device fingerprint using crystal oscillator impurities measured with an isolated single-core CPU and GPU in a multicore system, combined with unreliable functions such as random number and hash generators, introduces additional randomness into the fingerprint development. Consequently, generating a fingerprint using this method will be highly unreliable.

To our knowledge, no existing literature has modelled the base crystal oscillator impurities of devices for continuous authentication of IoT devices. We propose a novel continuous device-to-device authentication (CD^2A) algorithm that addresses the drawbacks and issues of previous device fingerprinting solutions by proposing improved and reliable device fingerprint generation. Our novel framework consists of two phases: 1) Identity Establishment and 2) Continuous Authentication. In the first phase, using an improved methodology, a unique device fingerprint is created by leveraging the devices' base crystal oscillator impurities and the impurities of each core of the CPU and GPU. In the second phase, device identity is authenticated periodically by sequentially matching the features extracted from comparing each CPU core with the GPU. The device's behaviour changes will determine the device risk, and a

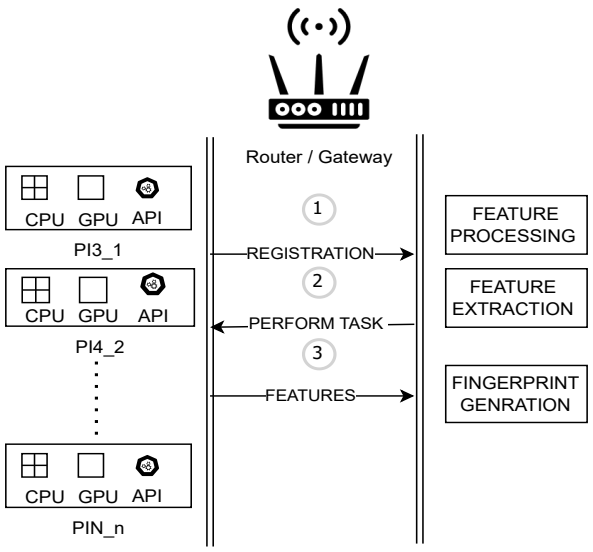

Fig. 1. Workflow for Identity Establishment

dynamic authentication policy is applied continuously.

III. Proposed CD^2A Framework

The proposed framework considers a client-server architecture with a central gateway acting as a server and the devices that need to be authenticated acting as clients, shown in Fig. 1. Given the nature of our novel framework with no prerequisite computational resources, our gateway can be a router, server or edge device. The proposed system assumes that the edge device can execute a code with root privileges and access to the hardware registers. The following sections explain the working of the CD^2A algorithm.

A. Identity Establishment

As shown in Fig. 1, the identity establishment process begins with the device sending a handshake request to the server after the initial static authentication to begin the registration to start CD^2A. The server responds by sending a client application for the device to start the process. The client application performs a series of steps on the client device. It aims to create a stable environment suitable for feature collection.

First, it selects the components from which to collect the features. Once the hardware component is identified, the next step is to ensure its stability for feature extraction, which is achieved by stabilising the operating frequencies of the CPU core. The CPU and GPU may adapt according to system load or energy-saving requirements, making it impossible to compare performance variations directly. Therefore, the software and hardware running the feature collection must be isolated (separation of cores). Additionally, kernel interrupts need to be disabled to reduce interference and noise generated by other processes in the system.

The next step is to create a function or series of operations to be executed on the devices to measure their behaviour in parallel and determine the possible unique variations between them. This was achieved using no-operation loops executed on isolated cores. Studies have utilised functions such as random number generators, hash generators, and sleep cycles [25] and [26] to extract the behavioural performance. However, these methods introduce additional noise to the performance variation, particularly with random number and hash generators. Moreover, these studies were conducted on a single CPU core. Other limitations include core availability, feature degradation over time, and multi-core systems. In our framework, we employ a no-operation loop on an isolated CPU core to measure the cycle count in the GPU, repeating this process across all available cores in the device. Unlike previous approaches, this method avoids interference with fingerprint generation and enhances performance variation by combining data from each CPU and GPU core. Consequently, this contributes to more reliable and accurate fingerprinting.

Longer execution times for functions or series of operations can indeed accentuate component variations. However, they also result in extended fingerprint generation times and disrupt the device's regular operations. This issue was noted in [25], where cyclic tasks were performed on the CPU core to measure features. Another study [22] measured a one-second cyclic task using a real-time clock (RTC). Following previous research, we opted to run a no-operation loop on each core for a time interval of 120 seconds, measuring the cycle counter values as clock cycles in the GPU. This process was repeated five times to gather sufficient data points in each core to construct the digital fingerprint.

1) Building Digital Fingerprint: Algorithm 1 elucidates our novel digital fingerprint algorithm. We discovered that 800 data points is the most efficient choice through experiments. If the number of data points is too small, the variation in the device caused by imperfections will be minimal, making pattern detection difficult. Conversely, more data points would extend the process duration, so the optimal number depends on the type of imperfection being measured. In our series of experiments, we aimed to identify the optimal combination of data points and no-operation loop durations.

We systematically varied the number of data points from 100 to 1,000 and adjusted the no-operation loop durations between 1 and 5 minutes. Each combination was rigorously tested to evaluate performance metrics. Our findings indicate that the configuration of 800 data points paired with a 2-minute no-operation loop duration yielded the most favourable results, striking an ideal balance between processing time and data handling efficiency. This combination was found to optimize both efficiency and performance. As noted in [26], system reboots can impact fingerprint generation by up to 50% due to the randomness introduced during these reboots. Therefore, in our study, we rebooted the system after collecting every 800 data points to incorporate the effects of the reboot. This process was repeated five times to populate enough data points to train the model, focusing on one core at a time to ensure that the other cores in the system could operate normally. This approach allowed each core to fully capture the performance variations caused by the imperfection in the crystal oscillator. Collecting variations through each CPU and GPU core will enable us to account for imperfections in every core. In our novel framework, we combine the extracted features from each core, which enhances performance variation and improves the reliability of the device fingerprint using our method.

After data collection, feature preprocessing is essential to eliminate noise. We used a sliding window methodology to select data points from the features collected on client devices. According to [27], data collection was performed on an isolated single core in the GPU ten times with a variable sliding window to extract limited statistical features from the raw data. However, grouping by varying window sizes can sometimes result in highly correlated and redundant information, negatively impacting fingerprint generation.

In our model, we determined that while the window size can be varied, it should not be grouped. The optimal window size depends on the specific model and the type of feature engineering applied to the clock cycles extracted in the client devices. We considered extended statistical features such as Mean, Sum, Maximum Value, Interquartile Range (IQR), Interaction (Standard Deviation × Mean), Skewness, and Kurtosis to gain a deeper understanding. Additionally, we captured two positional features using the 25th and 75th percentiles.

Once the desired feature set for generating the fingerprint has been identified, it is essential to select the appropriate model, as discussed in [28]. Due to the fixed number of devices, we will utilise a classification model for our proof-of-concept (POC) system. This approach will help accurately distinguish between the devices based on their unique fingerprints.

After selecting the appropriate method, the machine learning (ML) model is trained, and the metrics for fingerprinting are defined. Numerous ML classification algorithms are considered: support vector machine (SVM), Quantitative Descriptive Analysis (QDA), gradient boosting, bagging, K-nearest neighbour (KNN), decision tree, extra tree, random forest, XGBoost, and CatBoost.

Next, we fine-tune the hyperparameters and train the ML model, which must be trained to achieve optimal results. Once trained, the model classifies each device based on its unique behavioural fingerprint, establishing its unique identity.

Algorithm 1: Device Fingerprint

Input: C_i^j: Value of cycle counter register for core i at time instance j

Output: $DF = \{DF_i \mid \forall i \in N\}$: Digital Fingerprint DF for a device with N cores

begin

Phase 1:Feature Collection in Client Device

for *each core i gather desired number of feature points* **do**

Run no-operation loop for S seconds;

Measure Cycle count register C_i^j at current time instance j;

$C_i = C_i \bigcup C_i^j$;

if *time elapsed == X secs* **then**

Reboot Device;

Isolate and Stabilise each Core

Phase 2: Feature Processing and Extraction in Server

Fix your window size W;

i = 0;

for *each Core Features C_i* **do**

%Extract Features Points by Sliding Window% $F = C_i[i : i + W]$

Normalise data points in F using min-max scalar;

df : Calculate set of features discussed in section III-A1

$DF_i = DF_i \bigcup df$

B. Continuous Authentication

After establishing each device's identity, our model continuously monitors changes in device behaviour using fingerprint. Based on these changes, the system determines the device risk level. A study by [29] introduced a system known as authentication phases. ,

which categorised devices into low-, medium-, and high-priority groups based on their deployment locations. This framework utilised static and continuous authentication (tokens), with the intervals between continuous authentications varying by group. However, this system lacked dynamic authentication and relied on repeated lightweight static authentication. It also shared the limitations and vulnerabilities discussed in Section I.

To address these limitations and effectively utilise our identity establishment method section III-A, we developed a continuous device authentication system suitable for a rapidly changing environment. Our continuous authentication consists of two main components: the Device Authentication System (DAS) and the Device Risk Factor (DRF), which determine each device's authentication timeline.

1) Device Authentication Score (DAS): The Device Authentication System (DAS) provides a real-time measure of how closely a device's current behaviour aligns with the behaviour established during the training phase. A DAS score of 100% indicates perfect alignment with expected behaviour, while lower scores represent increasing deviation levels. This score is derived by examining key performance metrics such as precision, recall, F1 score, and true positive rate (TPR).

The difference between these metrics' values for the established identity and the ongoing behaviour is used to quantify any behavioural change. The DAS allows for a consistent and uniform evaluation across different metrics by expressing this difference as a percentage change.

The DAS is computed using the following equation:

$$\text{DAS} = 100\% - \frac{1}{n}\sum_{i=1}^{n}\left(\frac{M_{\text{test\_}i} - M_{\text{train\_}i}}{M_{\text{train\_}i}} \times 100\right) \quad (1)$$

Where:

- $M_{\text{test\_}i}$ is the combined summation value of the i-th performance metric (Precision, Recall, F1-Score and TPR) during the test phase.
- $M_{\text{train\_}i}$ is the corresponding value for the training phase.
- $n = 4$ is the total number of evaluated metrics.

Equation 1 captures the percentage change in each performance metric between the test data and training data and computes the average percentage change across all metrics. The final DAS score is obtained by subtracting this aggregate change from 100%, yielding a single indicator of the device's behavioural consistency.

2) Device Risk Factor (DRF): This section outlines the Device Risk Factor (DRF), which is derived from the Device Authentication Score (DAS). This DRF value is then used to group the device into four categories: low, medium, high, and unauthenticated. These categories determine the frequency of continuous authentication required for each device, as illustrated in Fig. 2. Once device identity is established, the initial continuous authentication will begin immediately, followed by the computation of the DAS and DRF.

An earlier study by [29] suggested that the risk group of a device depended on where the device was deployed. However, adjusting authentication depending on where it was deployed posed a huge security risk. In our method, we use the DRF to dynamically determine the risk for each device based on its behaviour rather than its deployment location. This approach enhances security by focusing on the device's actual performance and behaviour.

- **LOW DRF** ($80 \leq \text{DAS} \leq 100$): Devices operate as expected with minimal risk, requiring less frequent authentication (e.g., every two hours), reducing system load while maintaining security.
- **MEDIUM DRF** ($60 \leq \text{DAS} < 80$): Devices show minor deviations, triggering more frequent authentication than low-risk devices to enhance security without reducing efficiency.
- **HIGH DRF** ($50 \leq \text{DAS} < 60$): Devices display significant behavioural changes, undergoing frequent and continuous authentication to detect potential threats promptly.
- **UN-AUTHORISED** ($\text{DAS} \leq 50$): Devices are flagged for immediate disconnection or to perform initial static authentication again due to possible security breaches, isolated, and reported to administrators.

The value of the Device Risk Factor (DRF) will determine whether T_1(authentication interval as indicated in Fig. 2) changes to longer, medium, short, or no intervals at all. As shown in Fig. 2, each block or segment in the timeline represents an authentication event. As the risk level increases, continuous authentication methods become more frequent. This structured approach allows for a responsive security protocol where the rigour of authentication directly correlates with the device's assessed risk.

In summary, our proposed framework provides robust, continuous device-to-device authentication with varying levels of risk for devices in an ever-changing IoT environment. We ensure high security and reliability by dynamically adjusting the authentication frequency based

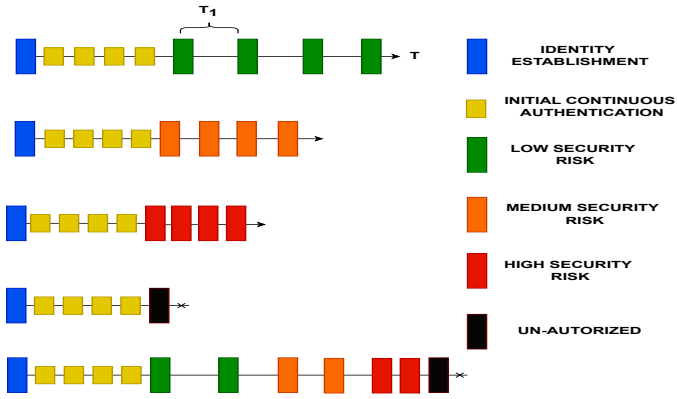

Fig. 2. Dynamic Authentication Timeline

on the Device Risk Factor (DRF). This approach focuses on the device's actual performance and behaviour, enhancing overall security and adapting to the dynamic nature of IoT environments.

IV. Validation of CD^2A Framework

1) Experimental Setup: The proposed proof-of-concept (POC) utilised 11 devices: three Raspberry Pi 3 Model B+ units and eight Raspberry Pi 4 Model B units. These devices were configured using identical desktop-less operating systems in a non-graphical environment and interconnected via a single router to a central server. The system's flexibility allows it to be deployed across various router or server configurations without restrictions, depending on the deployment needs.

The first step involved installing the client application on all Raspberry Pi devices. To create an ideal environment for feature collection, the devices were configured to run at their maximum frequency without overclocking by configuring `force_turbo=1`. Each core of the Pis is isolated for accurate cycle count measurement using the configurations `isolcpus=Core_Number`, `nohz_full=Core_Number`, and `RCU_nocbs=Core_Number` (ref. Section III-A).

Once the ideal condition is set, the client application begins feature collection by executing the No-operation loop for 120 seconds on one isolated CPU core at a time while measuring the cycle counter values as clock cycles in the GPU for 800 data points. After each time, the system will be rebooted. This process is repeated five times on each CPU core of the Pis sequentially and saved in comma-separated values (CSV) files that are securely sent to the server application.

TABLE I
Feature Set Extracted for Identity Establishment

Operation Performed	Data Points	Feature Extraction	Engineered Features
No-operation loop 120s	16,000 (each device) 4,000 (each core)	Statistical values	Mean, sum, 25th, 75th, max, IQR, interaction, skewness, Kurt.

2) Identity Establishment Validation: After receiving the data in the server application, we combine the collected features from all cores and apply noise reduction, outlier detection, min-max scalar and feature engineering to extract deep statistical features using a rolling window size between 800 and 1000, depending on the model. Ten algorithms were fine-tuned and applied to the feature set as shown in Table II to identify the optimum performing model for our proposed CD2A framework. Typically, model predictions are made on a per-vector basis, which means they cannot be directly used to make decisions during device evaluation and identification. Therefore, it is common practice to set a performance-based threshold to determine if a device is legitimate. This threshold can be established using various methods, such as recognising 50% of the values as legitimate, as demonstrated by [22]. Important metrics to consider at this stage include accuracy and true positive rate (TPR) [30].

From Table II, it can be seen that the best-performing models are Catboost with an average TPR of 99.96% and XGBoost with an average TPR of 99.85%. Although both models exhibit detection accuracies, they are computationally intensive due to the nature of gradient boosting. Significant memory and training time are required if the number of devices changes or the identity establishment needs to be retrained. Additionally, if the system needs to run on a router, it needs to be resource-optimised.

Considering performance and optimisation, Extra Tree is considered an optimum choice as it has an average TPR of 97.86%. At the same time, it is faster to train and less resource-intensive than CatBoost and XGBoost. The fourth-best performing model, the Decision Tree, is extremely fast and resource-efficient, making it ideal for lightweight applications or smaller datasets. We have compared TPRs for all the classification algorithms considered in this study refer to Fig. 3.

3) Continuous Authentication Validation: After establishing device identities, we continuously collect features and monitor the real-time change in device behaviour to perform authentication to calculate the Device Authentication Score (DAS) and Device Risk Factor (DRF), as shown in Table III. Low-risk devices un-

TABLE II
CLASSIFICATION ALGORITHMS AND THRESHOLDS

Algorithm	Hyperparameters	Avg TPR
SVM	C=0.1, degree=3, gamma=1, kernel='linear'	0.2436
Gradient Boosting	learning_rate=0.2, max_depth=3, min_samples_leaf=4, min_samples_split=10, n_estimators=100, subsample=0.8	0.3599
QDA	reg_param=0.0, tol=0.0001	0.6788
Bagging	bootstrap=True, bootstrap_features=False, max_features=1.0, max_samples=0.5, n_estimators=200	0.7891
Random Forest	bootstrap=False, max_depth=30, min_samples_leaf=1, min_samples_split=2, n_estimators=200	0.9416
KNN	metric='euclidean', n_neighbors=3, weights='uniform'	0.9586
Decision Tree	criterion='entropy', max_depth=None, max_features=None, min_samples_leaf=1, min_samples_split=2	0.9613
Extra Trees	bootstrap=False, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=200	0.9786
XGBoost	learning_rate=0.2, max_depth=3, min_samples_leaf=4, min_samples_split=10, n_estimators=100, subsample=0.8	0.9985
CatBoost	border_count=128, depth=10, iterations=500, l2_leaf_reg=1, learning_rate=0.1	0.9996

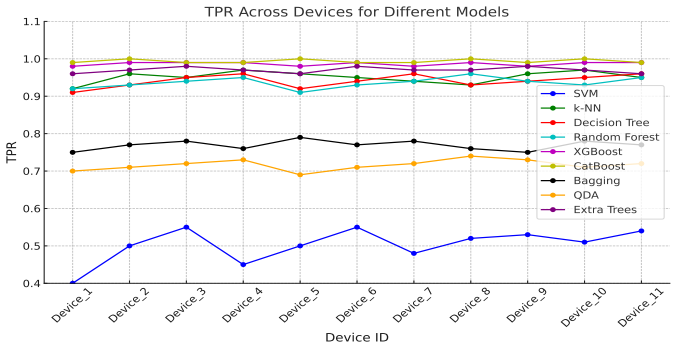

Fig. 3. Average True Positive Rate for each device

dergo less frequent authentication, as shown in Figure 2. Devices are positioned on the authentication timeline based on their DRF, with appropriate policies (e.g., low, medium, high risk, or unauthenticated) applied. These results determine the frequency of authentication or unauthentication, enabling dynamic, continuous device-to-device authentication in an IoT environment.

In Table IV, we present a comparison of the results obtained from our framework and those from the state-of-the-art, along with the features considered by different

TABLE III
DEVICE AUTHENTICATION SCORE AND RISK FACTOR

Device	Precision of Change (%)	Device Authentication Score (DAS)	Device Risk Factor (DRF)
Device 0	0.02	99.98%	LOW
Device 1	0.02	99.98%	LOW
Device 2	0.00	100.00%	LOW
Device 3	2.34	97.66%	LOW
Device 4	3.05	96.95%	LOW
Device 5	3.08	96.92%	LOW
Device 6	2.82	97.18%	LOW
Device 7	2.04	97.96%	LOW
Device 8	3.34	96.66%	LOW
Device 9	2.92	97.08%	LOW
Device 10	2.79	97.21%	LOW

TABLE IV
EVALUATION OF THE CD^2A SYSTEM

Device	Precision of Change (%)	Device Authentication Score (DAS)	Device Risk Factor (DRF)
Device 0	0.02	99.98%	LOW
Device 1	0.02	99.98%	LOW
Device 2	0.00	100.00%	LOW
Device 3	2.34	97.66%	LOW
Device 4	99.72	0.28%	UN-AUTH
Device 5	3.08	96.92%	LOW
Device 6	5.32	52.24%	HIGH
Device 7	2.04	97.96%	LOW
Device 8	99.59	0.41%	UN-AUTH
Device 9	2.92	97.08%	LOW
Device 10	2.79	97.21%	LOW
Spoofing	74.07	25.93%	UN-AUTH

approaches. The author [24] selected only the CPU cores separately to generate the fingerprint using thread affinity. The features were collected statistically based on the time taken to perform small operations on each core, and the Local Outlier Factor (LOF) was used as an anomaly detection method for each device. However, the author did not consider that a reboot could affect the device's fingerprint and did not establish an ideal environment to collect the features, making repeatability impossible and not suitable for building an authentication mechanism. Studies [25] and [26] addressed this limitation by introducing the stable condition to measure the features. However, they only addressed one core, and the operations used a random number generator and hash generator, which introduced more noise into the fingerprint. Additionally, imperfections in each CPU core and GPU were not considered. Despite this, they

TABLE V
COMPARISON WITH LITERATURE

Works	Hardware	Stability	Operations	Feature	Output	CD2A	Limitation
[24]	CPU	No stability	Short function	Raw values	69% device identification, no reboot considered, Dynamic Time warping	No	No Stability, No Reliability, Fast Degradation
[25], [26]	CPU, GPU	Core isolation, Fixed frequency	Sleep cycle, Random number, Hatstring	Limited statistical feature, grouped window size	91.9% TPR, XGBoost	No	No Stability, No Reliability, No Efficiency, High Resource Usage, Large Data Points, Single Core
Our Work	CPU, GPU	Core isolation, Fixed frequency, Core combination	No-operation loop with multi-core	Variable window size statistical feature	97.86% TPR, ExtraTree	Yes	-

achieved good results in the TPR classification algorithm, with a 91.9% boosting algorithm. However, they did not consider the limitations of their system.

4) Evaluation of System: To evaluate the performance and security of our CD^2A system, we designed three distinct scenarios:

1) **Label Swapping**: In the first scenario, we swapped the labels of devices 4 and 8 to observe the system's response. As shown in Table IV, these devices were unauthenticated, demonstrating that the model can accurately classify devices even after label swapping.
2) **Adversarial Noise**: In the second scenario, we introduced varying levels of adversarial noise (ranging from 10% to 60%) to device 6. For noise levels below 40% the Device Risk Factor (DRF) ranged from low to medium, while noise levels above 50% resulted in high. This indicates that the system can recognize the device despite feature perturbations and detects anomalies.
3) **Synthetic Spoofing**: In the third scenario, we generated synthetic data to simulate a spoofed device by combining the mean values of actual devices with standard deviation noise. A spoofing attack on device 7 resulted in unauthenticated, as shown in Table IV. This confirms the model's ability to detect synthetic devices as potential threats.

These tests validate the security, reliability, and robustness of our CD^2A system.

In our work, we addressed all these limitations by applying core combination, reducing reboot variability, and feature collection by comparing each core to the GPU and executing the no-operation loop in an ideal environment. Our test bed consisted of a limited number of devices, all from the same manufacturer and with identical configurations, which resulted in a very low false positive rate (FPR). However, we acknowledge that in a more extensive set of varying devices, the FPR may increase, although we were unable to test this scenario. Despite this limitation, our results were superior to prior research, achieving a true positive rate (TPR) of 99.96% in CatBoost, 99.85% in XGBoost, and 97.86% in ExtraTree.

V. CONCLUSION AND FUTURE WORKS

This paper introduces a framework for continuous device-to-device authentication CD^2A using devices' unique hardware behaviours. This mechanism calculates the Device Authentication Score (DAS) and Device Risk Factor (DRF) values to identify the device risk and adjust the authentication timeline. If the DRF value falls below 50%, the device is deemed insecure and requested to perform the initial static authentication again or disconnected from the network, and the administrator is notified. This system's proof of concept (POC) demonstrated its effectiveness using data from 11 Raspberry Pi devices, analyzed with ten classification models. The Catboost model performed best, achieving a true positive rate of 0.96. Our dynamic continuous authentication adapts to devices based on their behavioural risk levels. In future work, we will test our methodology in various situations and with a larger number of devices.

REFERENCES

[1] A. Ehret, E. D. Rosario, C. Schwicking, K. Gettings, and M. A. Kinsy, "Reconfigurable hardware root-of-trust for secure edge processing," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, 2021, pp. 1–7.

[2] M. R. Nosouhi, K. Sood, M. Grobler, and R. Doss, "Towards spoofing resistant next generation iot networks," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1669–1683, 2022.

[3] V. T. Amos, "Identifying unauthorized devices on vlans using software-defined networks," Ph.D. dissertation, Monterey, CA; Naval Postgraduate School, 2019.

[4] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[5] A. Boldyreva, M. Fischlin, A. Palacio, and B. Warinschi, "A closer look at pki: Security and efficiency," in *Public Key Cryptography – PKC 2007*, T. Okamoto and X. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 458–475.

[6] P. Eronen and H. Tschofenig, "Rfc 4279: Pre-shared key ciphersuites for transport layer security (tls)," USA, 2005.

[7] G. Leduc, "Verification of two versions of the challenge handshake authentication protocol (chap)," *Annales des Télécommunications*, vol. 55, no. 1-2, January 2000.

[8] P. Bulusu, H. Shahriar, and H. M. Haddad, "Classification of lightweight directory access protocol query injection attacks and mitigation techniques," in *2015 International Conference on Collaboration Technologies and Systems (CTS)*, 2015, pp. 337–344.

[9] V. M. Patel, R. Chellappa, D. Chandra, and B. Barbello, "Continuous user authentication on mobile devices: Recent progress and remaining challenges," *IEEE Signal Processing Magazine*, vol. 33, no. 4, pp. 49–61, 2016.

[10] S. W. A. Shah, N. F. Syed, A. Shaghaghi, A. Anwar, Z. Baig, and R. Doss, "Towards a lightweight continuous authentication protocol for device-to-device communication," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 1119–1126.

[11] A. Badhib, S. Alshehri, and A. Cherif, "A robust device-to-device continuous authentication protocol for the internet of things," *IEEE Access*, vol. 9, pp. 124 768–124 792, 2021.

[12] T. J. Salo, "Multi-factor fingerprints for personal computer hardware," in *MILCOM 2007 - IEEE Military Communications Conference*, 2007, pp. 1–7.

[13] C. Benzaid, K. Lounis, A. Al-Nemrat, N. Badache, and M. Alazab, "Fast authentication in wireless sensor networks," *Future Generation Computer Systems*, vol. 55, pp. 362–375, 2016.

[14] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Authentication protocols for internet of things: a comprehensive survey," *Security and Communication Networks*, vol. 2017, no. 1, p. 6562953, 2017.

[15] A. Babaei and G. Schiele, "Physical unclonable functions in the internet of things: State of the art and open challenges," *Sensors*, vol. 19, no. 14, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/14/3208

[16] J. Huang, W. Albazrqaoe, and G. Xing, "Blueid: A practical system for bluetooth device identification," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014, pp. 2849–2857.

[17] L. Polčák and B. Franková, "Clock-skew-based computer identification: Traps and pitfalls," *Journal of Universal Computer Science*, vol. 21, no. 9, pp. 1210–1233, 2015.

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

[19] H. Jafari, O. Omotere, D. Adesina, H.-H. Wu, and L. Qian, "Iot devices fingerprinting using deep learning," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, 2018, pp. 1–9.

[20] S. W. Shah, N. F. Syed, A. Shaghaghi, A. Anwar, Z. Baig, and R. Doss, "Lcda: Lightweight continuous device-to-device authentication for a zero trust architecture (zta)," *Comput. Secur.*, vol. 108, no. C, sep 2021.

[21] G. Nakibly, G. Shelef, and S. Yudilevich, "Hardware fingerprinting using html5," *arXiv preprint arXiv:1503.01408*, 2015.

[22] I. Sanchez-Rola, I. Santos, and D. Balzarotti, "Clock around the clock: Time-based device fingerprinting," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1502–1514.

[23] Y. Xue, Y. Zhang, and H. Xiang, "Development and research progress of crystal oscillator," in *7th International Conference on Computing, Control and Industrial Engineering (CCIE 2023)*, Y. S. Shmaliy and A. Nayyar, Eds. Singapore: Springer Nature Singapore, 2023, pp. 265–279.

[24] 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 *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)*, 2019, pp. 266–270.

[25] P. M. Sánchez Sánchez, A. Huertas Celdrán, G. Bovet, and G. Martínez Pérez, "Adversarial attacks and defenses on ml- and hardware-based iot device fingerprinting and identification," *Future Gener. Comput. Syst.*, vol. 152, no. C, p. 30–42, mar 2024.

[26] P. M. Sánchez Sánchez, A. Huertas Celdrán, G. Bovet, and G. Martínez Pérez, "Single-board device individual authentication based on hardware performance and autoencoder transformer models," *Computers Security*, vol. 137, p. 103596, 2024.

[27] P. M. Sánchez Sánchez, J. M. Jorquera Valero, A. Huertas Celdrán, G. Bovet, M. Gil Pérez, and G. M. Pérez, "A methodology to identify identical single-board computers based on hardware behavior fingerprinting," *Journal of Network and Computer Applications*, vol. 212, p. 103579, 2023.

[28] J. P. Usuga Cadavid, S. Lamouri, B. Grabot, R. Pellerin, and A. Fortin, "Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0," *Journal of Intelligent Manufacturing*, vol. 31, pp. 1531–1558, 2020.

[29] R. S. Miyanaji, S. Jabbehdari, and N. Modiri, "Continuous lightweight authentication according group priority and key agreement for internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 7, Mar. 2022.

[30] J. P. U. Cadavid, S. Lamouri, B. Grabot, R. Pellerin, and A. Fortin, "Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0," *Journal of Intelligent Manufacturing*, vol. 31, no. 6, p. 1531–1558, Jan. 2020.