Leveraging AI to Compromise IoT Device Privacy by Exploiting Hardware Imperfections

Mirza Athar Baig, *Student Member, IEEE*, Asif Iqbal, Muhammad Naveed Aman, *Senior Member, IEEE*, and Biplab Sikdar, *Senior Member, IEEE*

***Abstract*—The constrained design, remote deployment, and sensitive data generated by Internet of Things (IoT) devices make them susceptible to various cyberattacks. One such attack is profiling IoT devices by tracking their packet transmissions. While existing methods mitigate these attacks using pseudonymous identities, we propose a novel attack strategy that exploits the physical layer characteristics of IoT devices. Specifically, we demonstrate how an attacker can leverage features extracted from device transmissions to identify packets originating from the same device. Once identified, the attacker can isolate the device's signals and potentially determine its physical location. This attack exploits the fact that microcontroller clock variations exist across devices, even within the same model line. By extracting transmission features and training Machine Learning (ML) models, we accurately identify the originating device of the packets. This study reveals inherent privacy vulnerabilities in IoT systems due to hardware imperfections that are beyond user control. These limitations have profound implications for the design of security frameworks in emerging ubiquitous sensing environments. Our experiments demonstrate that the proposed attack achieves 99% accuracy in real-world settings and can bypass privacy measures implemented at higher protocol layers. This work highlights the urgent need for privacy protection strategies across multiple layers of the IoT protocol stack.**

IMPACT STATEMENT

This paper contributes to the security of the Internet of Things (IoT) by revealing a vulnerability stemming from hardware imperfections introduced during the manufacturing process of IoT devices. In particular, we exploit the variations in the internal clock of embedded devices to identify the source of a data packet. The IoT has become ubiquitous and given the private nature of data, it is crucial to guarantee privacy protection. The current state-of-art for privacy management is based on anonymization using pseudonym identities or other mechanisms at the network layer. However, the proposed attack shows that the privacy of IoT devices can be compromised using physical layer information irrespective of network layer mechanisms. Thus, this paper highlights the importance of cross layer approaches for IoT security. The convergence of hardware imperfections and the pursuit of verifiable anonymity is a critical juncture for research, especially considering the growing integration of the Internet of Things into sectors of critical infrastructure. The primary aim of our research is to promote the understanding of device-specific physical fingerprints and facilitate the advancement of strategies that safeguard privacy at multiple layers.

***Index Terms*—Internet of Things, Privacy Attack, Machine Learning, Device Profiling.**

This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research Development Programme, under grant FCP-NUS-RG-2022-019. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

M. A. Baig and M. N. Aman are with the School of Computing, University of Nebraska-Lincoln, 1400 R St, Lincoln, Nbraska, United States 68588. Email: mbaig4@huskers.unl.edu, naveed.aman@unl.edu.

A. Iqbal and B. Sikdar are with the Department of Electrical and Computer Engineering, National University of Singapore, 4 Engineering Drive 3, Singapore 117583. Email: aiqbal@nus.edu.sg, bsikdar@nus.edu.sg.

I. INTRODUCTION

The Internet of Things (IoT) is envisioned as the enabler of many new services and applications, such as smart cities, agriculture, healthcare, industries, etc. The simple and low-cost nature, extremely large number, remote location, and their use in critical infrastructure make IoT devices an attractive target for cybercriminals. One of the major applications of IoT devices is control and decision making. Therefore, trust in the fidelity of data generated by IoT devices is crucial for the success of IoT based control and decision making systems. Most IoT devices are embedded systems made up of microcontrollers. An IoT device has various hardware subsystems, including a microprocessor, memory, security, and radio subsystems, as shown in Figure 1.

Figure 1 illustrates an IoT device's clock-dependent subsystems. The microprocessor subsystem incorporates the CPU, memory, buses, and interfaces for computation and control. The security subsystem is responsible for executing cryptographic operations whereas the radio subsystem is responsible for facilitating wireless communication by utilizing various components, such as amplifiers and modulators. The process of digitizing sensor inputs in an analog-to-digital converter involves sampling at specific intervals that is determined by the clock. Frequency dividers are utilized to generate lower frequency clocks for individual subsystems by dividing a high-frequency source clock. Figure 1 shows that a clock source may be used to produce different clock periods or frequencies according to application specific needs using a frequency divider.

The CPU, bus, and peripherals of a microcontroller use its internal clock to coordinate their operations. The clock is crucial to performance because it determines how quickly the CPU executes its instructions. The central processor of

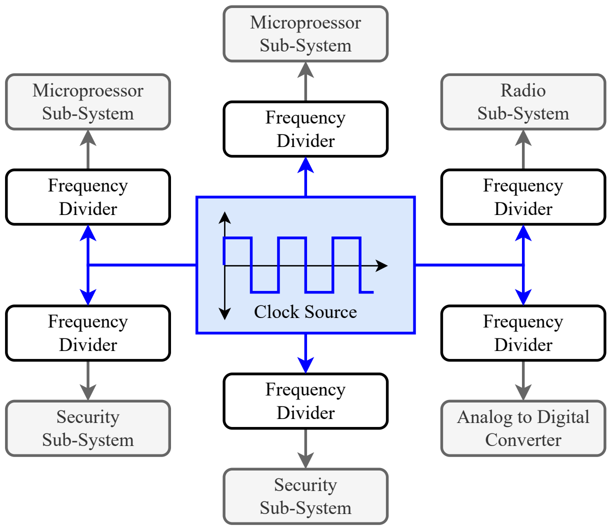


Fig. 1: IoT device sub-systems and reliance on a clock source.

a microcontroller can be viewed as a synchronized series of logic units, each one serving a specific purpose. If the clock is running too slow, the processing time will increase. On the other hand, if the clock runs too fast, there may not be enough time to complete the required operations before the next set begins. Similarly, any large error in the clock speed will have unforeseen effects on the internal operations of the microcontroller. It has been shown that the internal clock varies from one microcontroller to another, even if we consider the same type of microcontroller manufactured in the same facility [1], [2]. This is because industrial processes cannot control variations at the microscopic level. Although these variations in the internal clock may not be significant enough to cause errors, in this paper we show that they have enough entropy to be used for IoT device identification.

A. Related Work

Many researchers have discussed the impact of clock variations on system performance and mitigation techniques. The authors in [3] examine the modeling of various delay and jitter mechanisms in systems that close the communication network loop. As an example, the plant behavior is used to illustrate constant and time-varying delays. An event-triggered real-time scheduling technique for networked embedded devices based on event triggers is described in [4]. A modeling technique for random time delays in distributed control systems and a scheme for dealing with unpredictable time delays are described in [5]. [6] provides an overview of time synchronization in sensor networks. The authors of [7] address the subject of jitter control in time-triggered real-time systems, as delays or jitter in real-time embedded systems may affect system stability. The authors in [8] compare various fault-tolerant clock synchronization techniques.

Similarly, the authors in [9] describe an inventive hardware-based, extremely reliable clock generator to circumvent the shortcomings of algorithm-based fault-tolerant clock synchronization systems. Further discussion of clock synchronization issues in embedded systems and multiprocessor systems can be found in [10]–[14]. Although the significance of clock synchronization, clock delay, and clock jitter is highlighted, there is no extensive consideration of the effects of clock period variation on system behavior in the frequency domain. In a recent work by [1], frequency domain analysis is used to explore the effect of clock/sampling period drift on discrete time algorithms with open and closed loops. They described how the results of discrete-time algorithms can be impacted by clock variations due to manufacturing flaws. The above discussion shows that microcontrollers suffer from clock variations, which are exploited in this paper to attack the privacy of IoT devices.

Many research studies have utilized Radio Frequency Fingerprinting (RFF) techniques for device authentication. The underlying concept involves extracting unique features from wireless communication components to serve as device signatures for authentication purposes. Deep Learning (DL) approaches have demonstrated effectiveness in addressing classification problems of this nature, as evident from previous studies. In particular, some studies have employed DL-based RFF protocols that utilize base-band IQ samples as input to the system [15]–[17]. However, these studies indicate that significant performance degradation can occur when dealing with dynamic propagation environments and time gaps between training and testing data [18], [19]. These factors should be considered when designing and evaluating DL-based RFF systems to ensure their robustness and reliability in real-world scenarios. Thus, the existing RFF techniques are highly sensitive to channel and environmental factors such as mutipath interference, temporal and spatial variations, etc.

Prior research has explored time-based fingerprinting techniques that exploit timing imperfections in CPU clocks and system components for device identification. Authors in [29] demonstrated a method that times the execution of CPU instruction sequences and JavaScript APIs to fingerprint general-purpose computers and web browsers. However, their approach requires running code locally on the target device or remotely through web APIs, and primarily targets timing sources in high-performance CPUs and DRAM modules. Sánchez et al. [20] employed behavioral fingerprinting to identify single-board computers. They exploited the deviation between the GPU and CPU cycle counters and employed XGBoost to classify individual devices. However, their technique requires physical access to the GPU and CPU cycle counter. In [21], the authors employed programmable switches to introduce in-network fingerprinting using standard packet metadata. However, it requires access to the programmable switches which may not be possible in many cases.

In contrast, our work focuses on passive remote fingerprinting of resource-constrained wireless IoT devices by analyzing their over-the-air transmissions. We exploit unique hardware imperfections in the internal clock sources of low-power microcontrollers that get imprinted on the wireless signals during clock-dependent operations like modulation. In [22],

	[15]–[17]	[20]	[21]	[22]	[23]–[28]	**Our Work**
Remote Fingerprinting	✗	✗	✓	✗	✗	✓
No Physical Access Needed	✓	✓	✗	✗	✓	✓
Robust to Environmental Factors	✗	✓	✓	✓	✗	✓
Exploits Hardware Imperfections	✓	✓	✗	✓	✗	✓
High Accuracy in Real-World Conditions	✗	✓	✗	✗	✗	✓
Computational Complexity	High	High	Medium	Low	Low	Low
Effective Feature Extraction	✗	✓	✗	✓	✗	✓

TABLE I: Comparative analysis of existing works and our proposed method.

authors presented fingerprinting technique that utilized electromagnetic emanations from the processor clock of simple IoT devices without wireless interfaces. However, this technique requires close physical proximity of the IoT device which may not be possible in the real world.

Existing studies on data privacy for IoT devices include the following: The privacy properties of IoT services are discussed in [30]. A survey on the taxonomy for the security and privacy issues of Bluetooth low energy (BLE)-based IoT devices is presented in [31]. A survey on current progress for privacy support in IoT blockchains is presented in [32]. A study of privacy preservation strategies and solutions proposed thus far in the literature, as well as the IoT levels at which each solution addresses privacy and their resistance to privacy breaching attacks, was carried out in [33]. In another work, [34], the authors present a methodology for handling data privacy preservation in IoT networks. The authors in [35] use differential privacy to add noise to sensor measurements. Similarly, differential privacy is used to protect location privacy in [36].

The authors in [37] use a random disturbance mechanism to provide location privacy. Redactable Signature Schemes (RSS) were proposed to delete privacy-sensitive parts of the signed data of IoT systems in [38]. However, this approach cannot avoid the inherent uncontrollable variation. The authors in [39] proposed deep learning models for privacy preservation in healthcare IoT devices. The risk of sharing private data by IoT devices is estimated in [40]. A hierarchical two-layer and three-player game framework was used for data privacy preservation in context-aware IoT applications in [41]. The authors in [42], propose an IoT-cloud-enabled healthcare data system incorporating a searchable encryption method with forward privacy and verifiability. Most of the other existing work on IoT privacy preservation use random pseudonym identities [23]–[28]. A comparative summary of the methods discussed above and their features are provided in Table I.

Although encryption and network-layer defenses continue to be crucial, we contend that inherent and unalterable hardware variations can offer an alternative method for device fingerprinting. The rationale behind our methodology stems from the imperative for integrated physical and logical privacy strategies, as the sole reliance on techniques such as pseudonymity proves inadequate. Through the demonstration of exploiting clock skew to identify devices, we uncover concrete weaknesses in the privacy of IoT systems that encompass both physical characteristics and higher network layers. This highlights the significance of incorporating cross-layer perspectives in the process of developing resilient privacy safeguards.

The significance of conducting research at the convergence of hardware imperfections and the pursuit of verifiable anonymity is paramount, particularly in light of the increasing integration of the IoT into critical infrastructure sectors. The objective of our research is to enhance comprehension of device-specific physical fingerprints and stimulate the development of multi-layer privacy preservation strategies.

B. Contributions

A major security requirement for IoT devices is privacy, i.e., the source of data should not be traceable by an adversary. In these types of attacks, an adversary should not be able to identify or correlate packets from a specific IoT device. The existing work on privacy concentrates on pseudonymous identities, where an IoT device uses temporary identifiers generated using random numbers. Despite being effective at the network layer, these methods could be vulnerable to physical layer attacks. In this paper, we show a physical layer attack to identify IoT devices using modulated signals. In particular, we exploit the variation in the internal clock of microcontrollers to identify multiple packets from the same source IoT device.

The overall operation of the proposed attack is shown in Figure 2. The adversary sniffs wireless packets sent from multiple IoT devices and then based on extracted features (frequency peaks in this paper), the packets are separated according to source using a JS-divergence based algorithm. After a sufficient number of packets are accumulated from a source IoT device, a machine learning based classifier is trained. Consequently, the trained classifier is used to identify packets from various sources.

This paper focuses on revealing a new type of attack on the privacy of IoT devices. The proposed attack uses wireless characteristics of transmitted signals, specifically frequency peaks, to identify packets from a specific IoT device. The major contributions of this paper are as follows:

i. Propose a JS-divergence based novel algorithm to separate packets based on the source of the packets.
ii. Use machine learning models to train a classifier based on the packets collected using the JS-divergence based separation algorithm.
iii. Evaluate the effectiveness of the proposed attack using actual hardware under ideal, simulated noisy, and real-world scenarios.

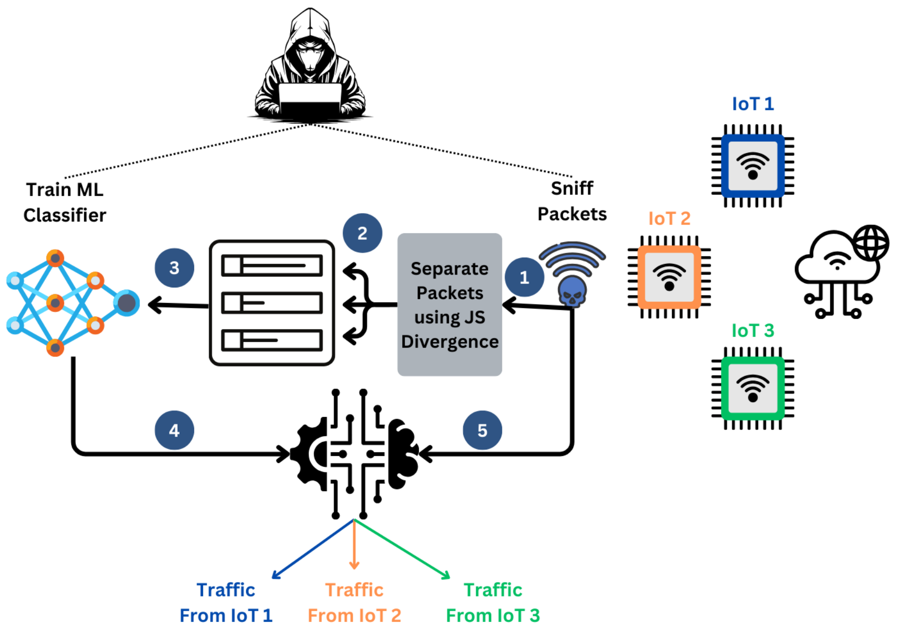


Fig. 2: Overall operation of the proposed attack.

The rest of the paper is organized as follows: Section II presents the network model and assumptions considered in this paper. Section III outlines the threat model employed in this paper. Section IV describes the proposed attack strategy. Section V presents the experiment setup, while the experimental results are discussed in Section VI. The future research directions are summarized in Section VII and Section VIII concludes the paper.

TABLE II: Notations and acronyms.

Acronym	Definition
IoT	Internet of Things
BLE	Bluetooth Low Energy
FFT	Fast Fourier Transform
ML	Machine Learning
SVM	Support Vector Machine
ROC	Receiver Operating Characteristics
JS	Jensen-Shannon
SNR	Signal-to-Noise Ratio
RFID	Radio Frequency Identification
Symbol	**Definition**
$S_n(t)$	transmitted signal
$A_n(t)$	amplitude of n^{th} sub-carrier
ω_n	carrier frequency
$\theta_k(t)$	phase for the n^{th} sub-carrier
fc	Carrier frequency
P_b	sub-carrier peaks for reference base signal
P_t	sub-carrier peaks for test signal
$\mathcal{A}$	Vector containing peaks for the same device
$\mathcal{B}$	Vector containing peaks for the different device
N	Number of samples
M	Number of classes
μ	Mean
w	window size for inputs of JS-Divergence
σ	Standard deviation

II. Network Model and Assumptions

The network model considered in this paper is shown in Figure 3. In this network model, we have two IoT devices connected to a gateway through wireless links. The adversary sniffs the wireless packets and tries to find similarities between packets according to the source, i.e., the objective of the adversary is to separate packets from Alice and Bob according to the source.

The following assumptions are made in this paper:

i. Every IoT device uses a single internal clock source.
ii. The adversary can listen to the wireless channel and obtain packets transmitted by all parties within the wireless network.
iii. Every packet transmitted by an IoT device has a preamble. This is a realistic assumption because every packet in TCP/IP has a preamble [43].

III. Threat Model

The adversary employs a cheap software-defined radio (SDR) that can sniff packets over the air. The adversary does not have direct access to the devices in order to tamper with or hijack them. The adversary passively analyzes wireless signals to track devices and invade their privacy. We assume that the radio protocol meets public IoT standards [44]. Although the attacker cannot decipher encrypted payloads, it can capture and extract information from the physical waveforms. Threats from remote passive wireless eavesdroppers are the main concern of this research.

IV. Proposed Attack

LTE-enabled devices, such as smartphones and tablets are already pervasive while LTE has become the standard for IoT

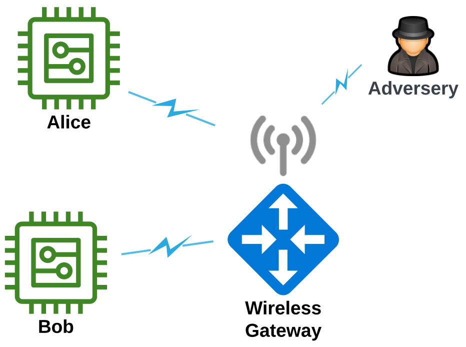


Fig. 3: The network model.

device communication. LTE is designed to efficiently transmit packets of information with low latency (a few milliseconds) and includes modern communication technologies such as Multiple Input Multiple Output (MIMO) and turbo codecs. LTE is based on Orthogonal Frequency Division Multiplexing (OFDM), with each subframe composed of multiple OFDM symbols. LTE subframes have 72 to 1200 sub-carriers with bandwidths ranging from 1.4 MHz to 20 MHz.

A time domain OFDM signal for each sub-carrier can be represented with the following equation:

$$S_n(t) = A_n(t)e^{j|\omega_n t+\theta_n(t)},\tag{1}$$

where, $S_c(t)$, $A_k(t)$, ω_k and $\theta_k(t)$ represent the transmitted signal, amplitude, carrier frequency, and phase for the n^{th} sub-carrier. Although the amplitude and phase can vary from symbol to symbol, they remain constant for a single symbol's duration. For N sub-carriers, the complex transmitted signal is given by:

$$S_s(t) = \frac{1}{N}\sum_{n=0}^{N-1} A_n(t)e^{j|\omega_n t+\theta_n(t)|},\tag{2}$$

where $\omega_n = \omega_0 + n\Delta\omega$, while ω_0 and $\Delta\omega$ represent the carrier frequency and sub-carrier spacing, respectively. Thus, this can be viewed as a superposition of multiple sub-carrier signals representing individual FDM signals.

A block diagram to generate each subframe of an LTE frame is shown in Figure 4. We observe that the output relies heavily on the carrier frequency f_c, i.e., any jitter in f_c would be propagated to all the sub-carriers. A typical spectrum for an OFDM signal is shown in Figure 5. This shows that any variation in f_c would in turn affect the sub-carrier peaks. To classify incoming packets as coming from the same or different sources, we employ a hierarchical binary classification approach that involves computing the distance between packet feature distributions using the *Jensen-Shannon (JS) divergence* [46]. JS divergence is a method of measuring the similarity between two probability distributions. It is a symmetric and finite version of the Kullback-Leibler divergence (KL divergence), which is a measure of the difference between two probability distributions. The JS divergence ranges from 0 to $\ln(2)$, with

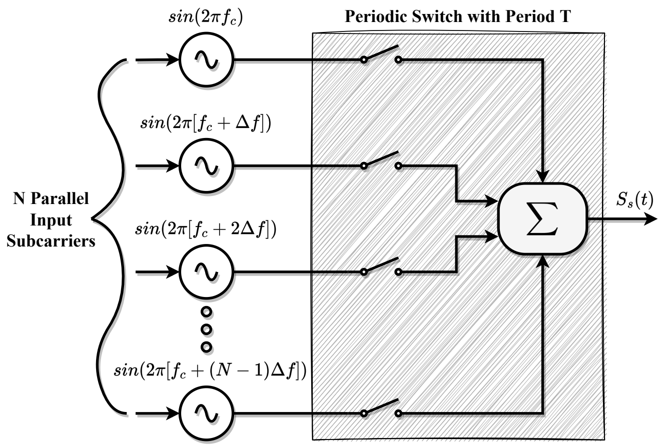

$sin(2\pi f_c)$
Periodic Switch with Period T
$sin(2\pi[f_c+\Delta f])$
N Parallel Input Subcarriers
$sin(2\pi[f_c+2\Delta f])$
$sin(2\pi[f_c+(N-1)\Delta f])$
Σ
$S_s(t)$

Fig. 4: Block diagram for OFDM generation.

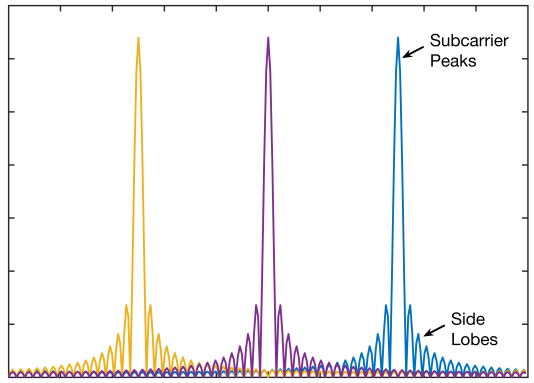


Fig. 5: OFDM signal frequency spectra [45].

0 indicating that the two distributions are identical and a value of $\ln(2)$ indicating maximum divergence. A simple threshold δ can then be used to classify the origin of the packets based on the distance computation.

The algorithm for the proposed attack is summarized in Algorithm 1 and the steps are as follows:

Algorithm 1: Proposed algorithm to detect packets from the same source.

```
Input: $\mathbf{P}_b = [P_1, P_2, \cdots, P_N]^T$, $thresh$,
       $\mathbf{P}_t = [P'_1, P'_2, \cdots, P'_N]^T$, $\mathcal{A} = \phi$, $\mathcal{B} = \phi$
1 $I \leftarrow$ JS divergence$(\mathbf{P}_b, \mathbf{P}_t)$
2 if $I < thresh$ then
3     $\mathcal{A} = [\mathcal{A} \quad \mathbf{P}_t]$: $H_1$ - Same Device
4 else
5     $\mathcal{B} = [\mathcal{B} \quad \mathbf{P}_t]$: $H_0$ - Different Device
6 end
Output: $\mathcal{A}, \mathcal{B}$
```

1) Sniff a new data frame/packet and find the sub-carrier peaks using the bandpass filter bank technique over a

specified window length [47]. Save the sub-carrier peaks in a vector,

$$\mathbf{P}_b = \begin{bmatrix} P_1 & P_2 & \cdots & P_N \end{bmatrix}. \tag{3}$$

2) For each new data frame sniffed, do the following:
   a) Calculate the sub-carrier peaks as before and save them to a vector,

$$\mathbf{P}_t = \begin{bmatrix} P'_1 & P'_2 & \cdots & P'_N \end{bmatrix}. \tag{4}$$

   b) Use JS divergence [46] to compute distance between the distributions of $\mathbf{P}_b$ and $\mathbf{P}_t$ to test the following hypothesis on $\mathbf{P} = [\mathbf{P}_b\ \mathbf{P}_t]$:

      H_0: Higher JS divergence score. The elements of $\mathbf{P}$ are random and belong to different sources.

      H_1: Lower JS divergence score. The elements of $\mathbf{P}$ are not random, i.e., they are from the same source.

   c) The null hypothesis H_0 refers to the case in which the distributions of the two vectors $\mathbf{P}_b$ and $\mathbf{P}_t$ have higher statistical diversity and are therefore random, i.e., the new packet is regarded as not coming from the same source. The purpose of the JS divergence test is to determine whether the sub-carrier peak distributions of the current and prior windows are consistent. If distributions are consistent, the JS divergence test rejects the null hypothesis, and the new packet is considered to come from the same source.

3) If the null hypothesis is rejected, save P_t in container $\mathcal{A}$, otherwise save it in $\mathcal{B}$.

If the goal is to identify G distinct sources, then the above steps can be repeated $G-1$ times on packets stored in $\mathcal{B}$.

To train an ML model to classify packets from different sources, an attacker needs to first generate a training dataset, i.e., separate packets coming from different sources. For this purpose, the Algorithm 1 is employed for identifying packets from the same source. However, it alone cannot effectively classify the packets into their respective labeled classes. Consequently, machine learning classifiers are utilized to perform the classification task. Once the adversary accumulates a sufficient number of packets from various sources, a machine learning (ML) model is trained using this data. This trained model can subsequently be employed to classify packets from these sources in future instances.

V. Experiment Design

This section describes the experimental setup and implementation details. The system model includes multiple transmitter (TX) devices that communicate data to a single receiver (RX) via a wireless channel, as seen in Figure 6. The TX devices are simulated using software-defined radios such as PlutoSDR and HackRF, which transmit an OFDM based LTE waveform and have settings specifically designed for device fingerprinting analysis. The RX device employs a PlutoSDR that has been configured with automatic gain control and matched filtering to record wireless signals.

The experiment is repeated under three scenarios.

1) **Lab Environment (S1):** In this scenario, we simulated ideal conditions under controlled environment within the lab as shown in Figure 6.
2) **Controlled Noisy Channel Simulation (S2):** To evaluate the robustness of our fingerprinting technique under adverse channel conditions, we simulated the whole experiment within a controlled lab environment by adding synthetic noise to the collected wireless signal samples. Specifically, we considered two commonly used noise models:
   a) Additive White Gaussian Noise (AWGN): This models the effect of thermal noise in the receiver circuitry. We added AWGN noise with varying signal-to-noise ratio (SNR) levels of 10dB to the received signal samples.
   b) Rayleigh Fading: This models the multi-path fading effects in wireless channels caused by reflections and obstructions. We applied a Rayleigh fading channel model with scale parameters of 1.0 to induce varying fading intensities.
3) **Real-World Scenario (S3):** To evaluate our technique in practical conditions, we repeated the experiment in an open parking area with an unobstructed path between the transmitter and receiver SDRs separated by 10m.

The wireless channel introduces various impairments, such as path loss, fading, and noise. Despite these challenges, the consistent hardware-based clock skew variations among TX devices enable the identification of individual devices, even in the presence of these channel effects. Figure 7 shows a block diagram for the experimental workflow.

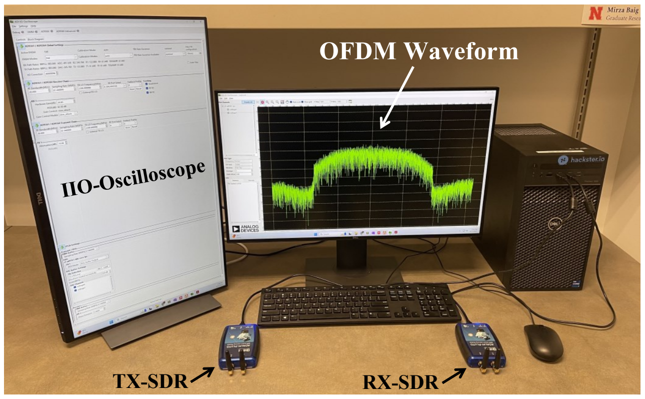


Fig. 6: Experimental setup.

A. Data acquisition

The experimental setup consists of one receiver (RX) and multiple transmitters (TX). We used analog devices, ADALM PLUTO® Software Defined Radios (PlutoSDR) as the receiver and multiple PlutoSDR and HackRF One as transmitters. The plutoSDR is based on the Analog Devices Highly Integrated RF Agile Transceiver, AD9363. It has a homodyne or Direct-Conversion Receiver (DCR) utilizing the same Local Oscillator (LO) for both RX and TX. The LO is used as the carrier frequency f_c. This f_c is selected in the UHF band with an

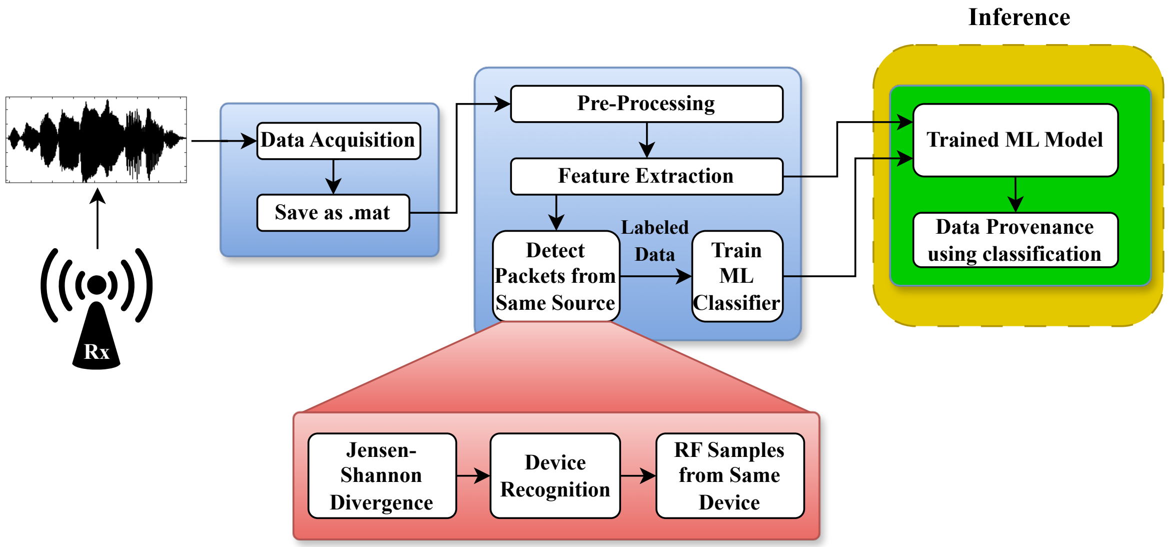


Fig. 7: Experiment block diagram.

1860 MHz frequency value. The same f_c and sampling rate is used for all RX and TX modules. The same parameters are used in every RX and TX module for consistency.

The HackRF uses a more stable crystal oscillator as it's clocking source and can be used to transmit and receive frequencies from 1MHz to 6GHz. We used one RX-TX pair for the data acquisition and repeated this process for all the TX devices as shown in Figure 6. The TX transmits a pre-computed LTE quadrature waveform with a 5MHz bandwidth.

For each device, we recorded 50×10^6 time domain samples as training and testing disjoint datasets, which were used for feature extraction as discussed next.

B. Feature Extraction

The collected samples are packaged in frames of 10,000 samples per frame, leading to a matrix of size $5,000 \times 10,000$. These frames are used to extract features that are used to train the ML classification models. The PlutoSDR employs a Temperature-Compensated Crystal Oscillator (TCXO) that is used to derive the master reference clock for SDR operations and processing. Each SDR module has its own independent TCXO i.e, an independent reference clock with a ± 25 ppm tolerance. Besides this, the hardware intrinsic also cause clock variations that can be exploited to find module-specific characteristics. As described in Section IV, the sub-carrier frequency peaks are exploited to identify packets from the same source. Thus, the sub-carrier frequency peaks are extracted as features from the LTE waveform and then used to train the classifiers.

The sub-carrier peak frequencies are selected as features because: 1) They directly relate to the internal clock source generating the OFDM waveform since any variations in the clock will manifest as shifts in the sub-carrier peak locations; 2) Hardware imperfections introduce uncontrollable variations in the internal clocks across different device instances, causing the peak locations to shift uniquely for each device; 3) The highest magnitude peaks are most resilient to noise and channel distortions compared to lower magnitude peaks. Thus, the peak frequencies effectively capture device-specific "fingerprints" arising from manufacturing imperfections in the clock sources.

The frequency peaks are extracted per frame using an FFT-based polyphase spectrum estimation method [47]. This method uses an analysis filter bank to split the broadband signal $x[n]$ into multiple narrow subbands, $y_0, y_1, \cdots, y_{M-1}$ where M is the number of frequency bands in the filter bank.

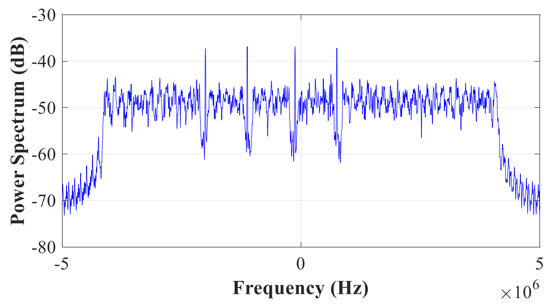
After the broadband input signal is split into multiple narrow bands, the spectrum estimator computes the power in each narrow band using the equation 5. Each z_i value becomes an estimate of the power over that narrow frequency band, i.e.,

$$z_i = \frac{1}{L} \sum_{m=0}^{L-1} |y_i[m]|^2, \quad \forall i = 0, 1, \cdots, M-1 \tag{5}$$

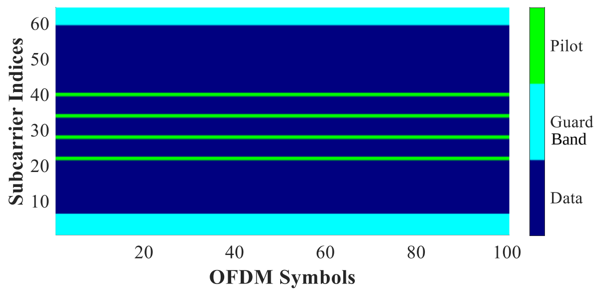
where z_i is the power value of the i^{th} band, and L represents the length of the narrow band signal y_i. The final output is the matrix $\mathbf{X}$ of size $5,000 \times M$. The filter bank approach produces a spectral estimate with a higher resolution, a more accurate noise floor, and more precise peaks with low or no spectral leakage [47]. The frequencies found in rows of $\mathbf{X}$ are ranked according to their Received Signal Strength Indicator (RSSI). The frequency components with the highest RSSI values are used as features, i.e.,

$$f_{peak}^{j} = argmax_N |RSSI(f^j)|, \quad \forall j = 1, \cdots, M \tag{6}$$

where f^j is the j^{th} row of $\mathbf{X}$, and $RSSI(.)$ is the RSSI of a given frequency obtained by spectral analysis of the original signal. In this paper, the IoT devices are represented by the PlutoSDR and HackRF, while the top $N = 10$ peak frequencies are utilized as features. We observed that the peaks deviate from one IoT device to another due to clock



(a) Frequency spectrum of single data frame.



(b) OFDM data frame with 4 pilots.

Fig. 8: Frequency domain analysis.

deviations caused by hardware manufacturing imperfections. The spectrum analysis was performed using the DSP System Toolbox of MATLAB® 2022a by Mathworks.

C. Supervised Learning Setup

The purpose of this experiment was to classify the packets originating from the six test devices. Therefore, each classifier was trained using six classes. We considered the most common ML models, i.e., fine k-Nearest Neighbors (k-NN), Decision Tree, Logistic Regression, Random Forest, and fine Support Vector Machines (SVM) as classifiers for device identification. Note that these classification models are not too complex and can be used for multi-class classification. The dataset was normalized to scale the feature values between 0 and 1. For the k-NN classifier, the value of k was set to five. Additionally, we used Euclidean distances and equal distance weights as the distance metrics. We trained the decision tree classifier using one hundred splits, with Gini's Diversity Index (GDI) used as the criterion for each split. For SVM, the RBF kernel was used. All kernel functions use standardized data with a box constraint level set to 1. A random forest classifier was trained using the GDI criterion and 8 estimators. All these models were trained and tested using Python's Scikit-Learn [48] library.

The dataset was split into training (75%) and test (25%) sets. The training set was further split using 5-fold cross-validation for tuning models' hyperparameters. Once the models were trained, their performance was evaluated on the held-out test set using metrics such as accuracy, F1-score, precision, recall, and AUC score. Confusion matrices were also generated to analyze errors.

VI. Results and Discussion

This section is divided into four subsections. In the first subsection, we discuss the TX data generation process and analyze the RX data signals for each device. In the second subsection, we analyze the performance of the proposed Algorithm 1 in terms of its ability to separate data packets coming from different sources. In the third subsection, we use the datasets generated using Algorithm 1 to train ML classifiers and present their classification scores on the test datasets. Finally, in the fourth section, we discuss the computational complexity and analysis of the proposed algorithm.

A. Data generation

For prototyping purposes, the experimental OFDM frame specifications were simplified from LTE. A 10 MHz spectrum is utilized by the 64 sub-carriers as opposed to up to 20 MHz for LTE. Pilot tones and cyclic prefixes are two essential elements that are kept in the simplified design. The objective is to demonstrate device fingerprinting via clock skew as opposed to testing against every potential LTE configuration. Note that using a simplified waveform presents a lower bound on the performance. An OFDM frame was created using the MATLAB software. The frame consisting of 64 sub-carriers with a total bandwidth of 10 MHz. The frame was then transmitted using the PlutoSDR devices, which had a sampling rate of 10 MHz. The sub-carrier spacing was set to 156.25 kHz, with 11 sub-carriers designated as guard bands and 4 as pilots. The input data bits were modulated using 8-point Quadrature Amplitude Modulation (QAM). These settings were chosen to be closest to the standard 10 MHz LTE waveform. Figures 8(a) and 8(b) show the spectrum and OFDM sub-carrier mapping of a single data frame, respectively.

Figure 9 illustrates the block diagram of the proposed technique. The input RF data coming from the ADC is sequenced into the Features Extraction Unit (FEU) to extract peak frequency features from the data. The FEU calculates frequency spectrum peaks as explained in V-B. As a proof of concept, we utilize the data acquired in the laboratory environment to analyze the features. Figure 10 shows the number of unique frequencies found for each feature. We observe that the first 4 features have the least number of unique frequencies as compared to the rest and are thus consistent. This is expected as, in the simulated OFDM frame, there were 4 fixed pilot sub-carriers with constant unit amplitudes as compared to the rest of the data sub-carriers. The lower-variance features have an impact on fingerprinting because they have a lower ability to distinguish between different devices. The features with greater variance, however, have more entropy to distinguish between devices. Although devices 1-3 show more unique frequencies than devices 4-6, we observe that these are sufficient for identifying the devices nonetheless.

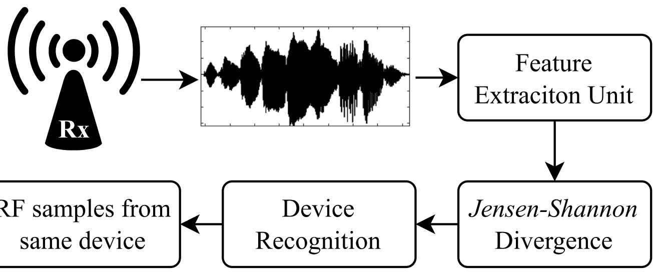


Fig. 9: JS divergence compute unit.

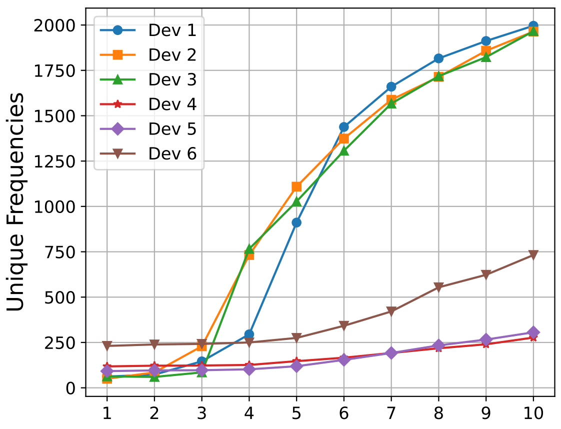


Fig. 10: Number of unique frequencies found for each feature in lab environment.

Figure 11 shows the empirical Cumulative Distribution Function (CDF) of all the frequencies present in the overall spectrum. The differences among the devices in CDF show that they have slightly different frequencies, caused by the clock sources of the TX and the RX. Since the RX is same for this experiment, the difference can be attributed to the TX devices. Note that PlutoSDRs utilize a temperature-compensated crystal oscillator, while HackRFs use a crystal oscillator. TCXOs are less stable than crystal oscillators; hence, PlutoSDRs' lead to larger frequency drifts. Due to their more stable clock, HackRFs can pick up more frequencies at the lower end of the spectrum, where signals are weaker. Because frequency drift is less prone to miss weaker signals, HackRFs miss them less.

Figure 12 shows the frequency distribution for the six TX devices within the 1857-1863 MHz band. The densities are multi-modal, with each mode centered around a specific pilot frequency. A density diagram illustrates the number of devices found at each frequency. The four peaks shown in the graph are due to the four pilot signals used in the waveform. Ideally, we should not have anything in the graph except for the four pilot frequencies but due to the clock variations, each TX device generates a range of frequencies other than the pilot frequencies. The more unstable the clock, the wider the range of frequencies as shown in Figure 12. Although

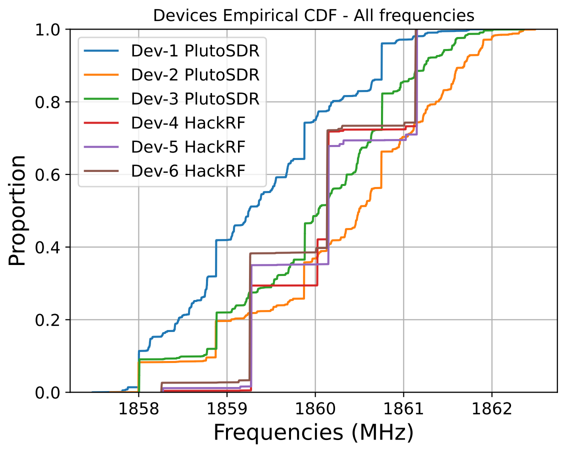


Fig. 11: Devices empirical CDF for the entire spectrum in lab environment.

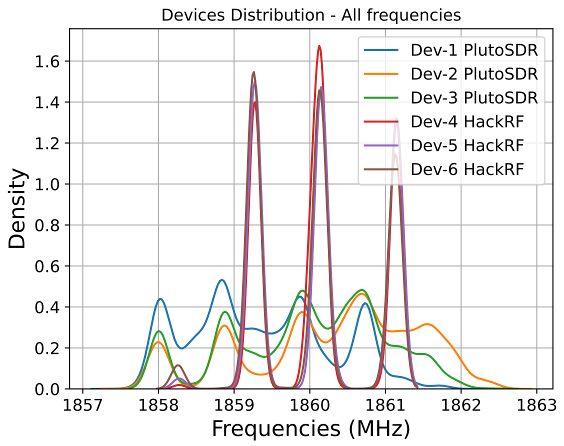


Fig. 12: Frequency distribution for each device in lab environment.

the transmitted waveform was the same for each device, the received frequency profiles are not the same, and thus, we should be able to use these profiles as hardware fingerprints for packet differentiation.

B. Performance Analysis

Once the feature matrix for the received packet stream has been computed, we use the proposed algorithm to separate packets coming from different devices. To do so, we pass the computed feature frames (from laboratory environment setting) to the JS divergence compute unit, which performs hierarchical binary classification on the frames using Algorithm 1. A sliding window approach is used in the JS divergence compute unit with a minimum stride of 1 frame. Each window w has f frames, where F shows the total number of frames of the data. If the incoming frames have a low divergence score compared to the previously collected frames, it is likely that they are coming from the same device. The divergence algorithm calculates the probability that a test statistic will

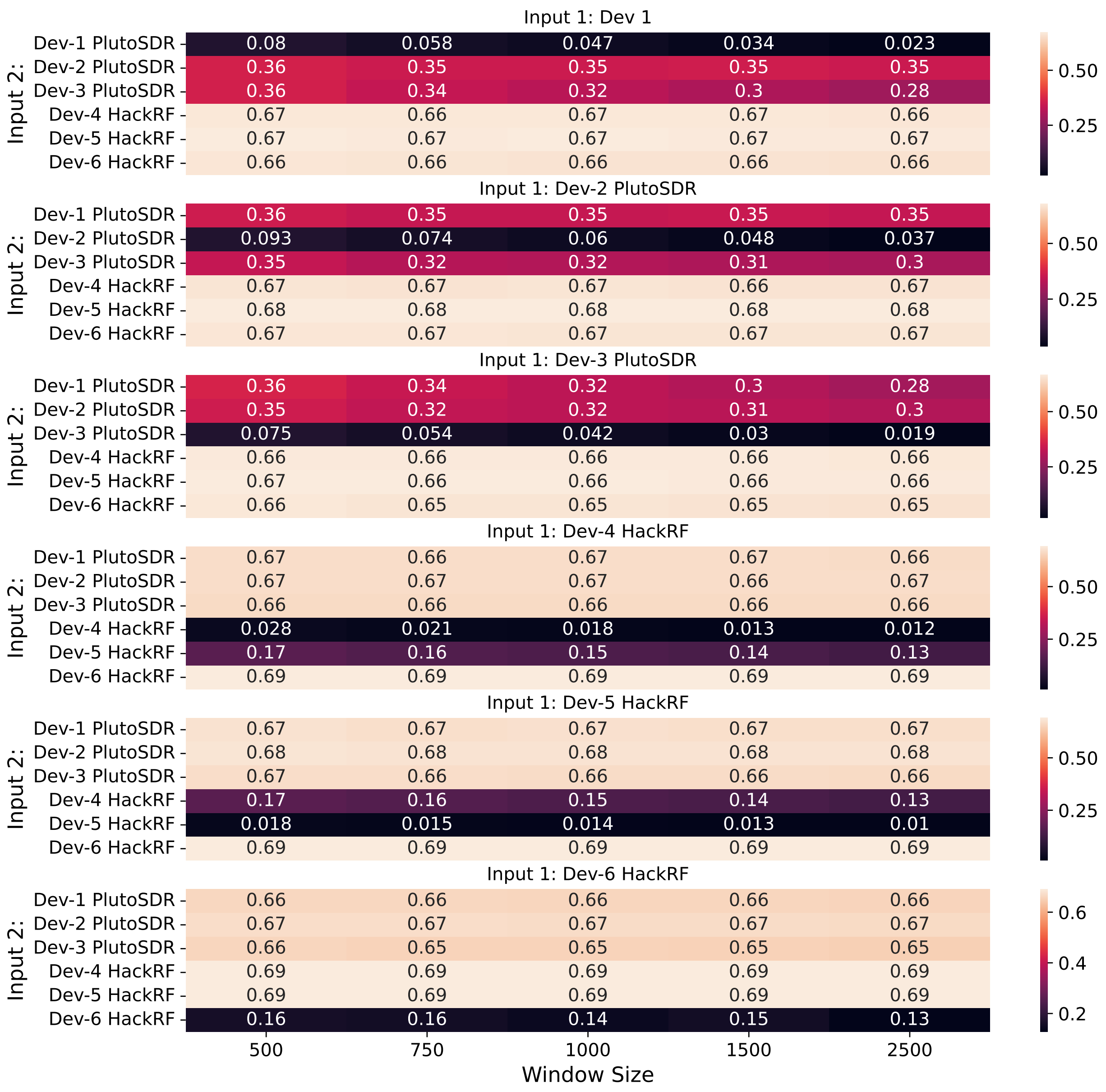


Fig. 13: Mean JS divergence scores.

be observed to be as extreme as or more extreme than its observed value under the assumption that the null hypothesis is true. These JS Divergence scores can be used to identify if the incoming packets originated from the same source or not.

To verify the sensitivity of the JS divergence scores, we computed the distance between input frames, coming from the same or different devices, over window sizes between 50 and 250 frames (500 and 2500 samples, respectively). The mean scores of this test are shown in Figure 13 and the histogram of all raw divergence scores is shown in Figure 14. Figure 13 highlights that the JS divergence test was found to be relatively insensitive to the window size used. The results suggest that the window size has a minimal impact on the divergence score. Moreover, the distribution of JS divergence scores, as shown in Figure 14, is tri-modal (scores for frames coming from the same vs. different devices) and has very low overlap; thus, an appropriate threshold can be used here for the binary classification problem.

To analyze the classification accuracy of Algorithm 1, we used it to classify the received data from the three PlutoSDR devices. The test was repeated using multiple window sizes and various threshold levels. The results are summarized in Figure 15. Figure 15 (a) shows the classification accuracy with respect to the various thresholds used on the JS divergence score highlighting the robustness against threshold selection as a threshold of 0.1 - 0.3 (see Figure 13) would give us around 99% accurate classification. Figure 15 (b) shows the classification accuracy for various window sizes

TABLE III: Summary of the performance of different classifiers.

	Precision			Recall			F1-score			AUC-Score			Accuracy		
	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3
Random Forest	0.9650	0.9516	0.9981	0.9650	0.9516	0.9981	0.9649	0.9516	0.9981	0.9972	0.9948	0.9994	0.9650	0.9516	0.9981
Decision Tree	0.9483	0.9396	0.9976	0.9483	0.9396	0.9976	0.9483	0.9396	0.9976	0.9738	0.9396	0.9980	0.9483	0.9396	0.9976
Logistic Regression	0.4440	0.2900	0.5888	0.4498	0.2900	0.5888	0.4246	0.2900	0.5888	0.8127	0.7406	0.8385	0.4498	0.2900	0.5888
k-NN	0.5789	0.3259	0.8211	0.5656	0.3259	0.8211	0.5642	0.3259	0.8211	0.8584	0.7160	0.9171	0.5657	0.3259	0.8211
SVM	0.5789	0.3773	0.8669	0.5656	0.3773	0.8669	0.5642	0.3773	0.8669	0.9176	0.8145	0.9582	0.5656	0.3773	0.8669

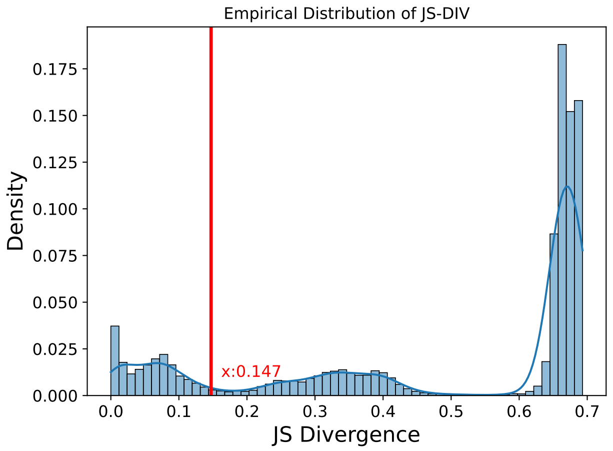


Fig. 14: Empirical distribution of JS divergence scores for all window sizes and all devices.

with a constant threshold value. Increasing the window size decreases the JS divergence score, as discussed above, hence improving the classification. These results demonstrate that the proposed algorithm is robust and does not require precise tuning of parameters such as the window size and classification threshold. It works effectively for a wide range of parameter values.

C. Classifier Performance

The feature training datasets from all devices obtained using Algorithm 1 were used to train multiple ML classifiers, and their effectiveness is evaluated through various metrics such as accuracy, F1-score, precision score, recall score, AUC score, and confusion matrices. A summary of these metrics for the different classifiers under different environment conditions studied in this paper on test datasets is presented in the table III.

We observe that the classifiers based on random forest and decision tree classification models can identify an IoT device with precision and accuracy of more than 96% for ideal closed lab environment and 99 % in real-world scenario. Sampling with the same precision, recall, and F1-score means that the model classifies positive and negative samples with the same accuracy, i.e., the ratio of true positives to false positives is the same as the ratio of true positives to false negatives. This is a desirable outcome, as it indicates that the model is correctly identifying both positive and negative samples. However, the logistic regression, k-NN, and SVM based classifiers have significantly degraded performance on all of the performance metrics. This performance degradation can be attributed to the fact that our data is not linearly separable and the features are correlated. If the features are highly correlated, it can lead to multicollinearity, which can affect a model's performance. The data must be linearly separable in order for logistic regression, RBF SVM, and k-NN to work. This means that a linear boundary must be able to separate the data into various classes. If the data is not linearly separable, these models may not perform well. Thus, this shows that the nature of the data is non-linear, and a linear classifier may not be appropriate.

A confusion matrix, which illustrates the true positives, true negatives, false positives, and false negatives for each of the three test IoT devices is provided in Figure 16 for each of the classifiers. The confusion matrices summarize the performance of the classifiers for lab environment. The decision tree classifier has the best performance among all. It has only one misclassification, where it misidentified device 1 as device 2. When compared to existing techniques, our approach outperforms those referenced as [15], [16], and [22], which reported accuracies of 96.4%, 95%, and 95.1%, respectively. Additionally, our approach demonstrates robustness, as it does not require close physical proximity to the victim device and is independent of the wireless protocols used.

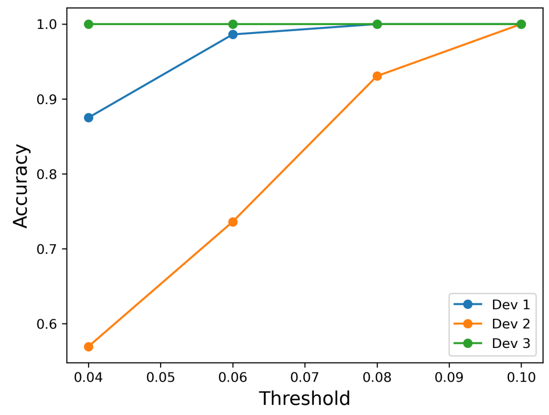
D. Computational Complexity

The worst case running time computational complexity for the different stages of Algorithm 1 is as follows:

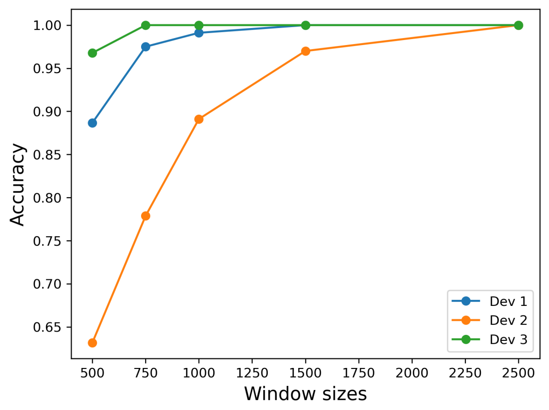
1) Feature extraction involves applying an FFT to the signal, which is $O(N\texttt{log}N)$ for N samples [43].
2) Computing the Jensen-Shannon divergence between two feature vectors x and y with length N is $O(N)$ [28].
3) The threshold for comparison and classification is $O(1)$.

Considering M packets and N sub-carrier peaks extracted, the overall computational complexity can be calculated as follows:

1) Inputs for the Algorithm 1 require calculating FFT. So they have complexity of $O(M\texttt{log}M \cdot N)$
2) Step 1 loops through set of feature vector P with JS-divergence costs of $O(N)$ for a total computational cost of $O(P \cdot M)$.
3) Step $2-6$ run in $O(1)$.

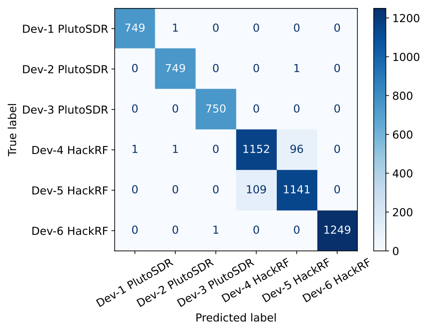


(a) Accuracy vs multiple thresholds with a single window size of 2500 samples.



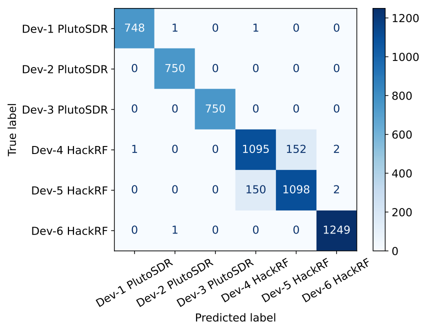
(b) Accuracy vs multiple window sizes with a constant threshold of 0.1.

Fig. 15: Binary classification: the data frames are classified into two groups: "same device" or "different devices".



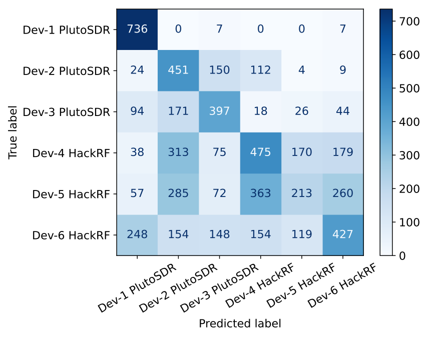
(a) Random Forest

True label \ Predicted label	Dev-1 PlutoSDR	Dev-2 PlutoSDR	Dev-3 PlutoSDR	Dev-4 HackRF	Dev-5 HackRF	Dev-6 HackRF
Dev-1 PlutoSDR	749	1	0	0	0	0
Dev-2 PlutoSDR	0	749	0	0	1	0
Dev-3 PlutoSDR	0	0	750	0	0	0
Dev-4 HackRF	1	1	0	1152	96	0
Dev-5 HackRF	0	0	0	109	1141	0
Dev-6 HackRF	0	0	1	0	0	1249



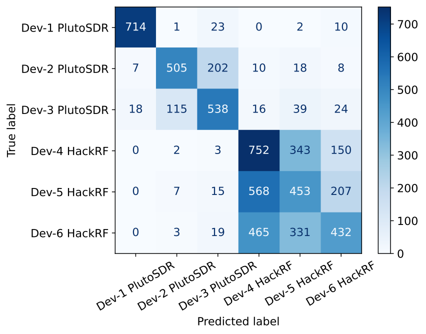
(b) Decision Tree

True label \ Predicted label	Dev-1 PlutoSDR	Dev-2 PlutoSDR	Dev-3 PlutoSDR	Dev-4 HackRF	Dev-5 HackRF	Dev-6 HackRF
Dev-1 PlutoSDR	748	1	0	1	0	0
Dev-2 PlutoSDR	0	750	0	0	0	0
Dev-3 PlutoSDR	0	0	750	0	0	0
Dev-4 HackRF	1	0	0	1095	152	2
Dev-5 HackRF	0	0	0	150	1098	2
Dev-6 HackRF	0	1	0	0	0	1249



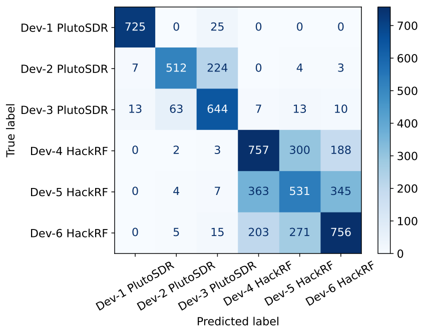
(c) Logistic Regression

True label \ Predicted label	Dev-1 PlutoSDR	Dev-2 PlutoSDR	Dev-3 PlutoSDR	Dev-4 HackRF	Dev-5 HackRF	Dev-6 HackRF
Dev-1 PlutoSDR	736	0	7	0	0	7
Dev-2 PlutoSDR	24	451	150	112	4	9
Dev-3 PlutoSDR	94	171	397	18	26	44
Dev-4 HackRF	38	313	75	475	170	179
Dev-5 HackRF	57	285	72	363	213	260
Dev-6 HackRF	248	154	148	154	119	427



(d) k-NN

True label \ Predicted label	Dev-1 PlutoSDR	Dev-2 PlutoSDR	Dev-3 PlutoSDR	Dev-4 HackRF	Dev-5 HackRF	Dev-6 HackRF
Dev-1 PlutoSDR	714	1	23	0	2	10
Dev-2 PlutoSDR	7	505	202	10	18	8
Dev-3 PlutoSDR	18	115	538	16	39	24
Dev-4 HackRF	0	2	3	752	343	150
Dev-5 HackRF	0	7	15	568	453	207
Dev-6 HackRF	0	3	19	465	331	432



(e) Support Vector Machines

True label \ Predicted label	Dev-1 PlutoSDR	Dev-2 PlutoSDR	Dev-3 PlutoSDR	Dev-4 HackRF	Dev-5 HackRF	Dev-6 HackRF
Dev-1 PlutoSDR	725	0	25	0	0	0
Dev-2 PlutoSDR	7	512	224	0	4	3
Dev-3 PlutoSDR	13	63	644	7	13	10
Dev-4 HackRF	0	2	3	757	300	188
Dev-5 HackRF	0	4	7	363	531	345
Dev-6 HackRF	0	5	15	203	271	756

Fig. 16: Confusion matrices for various classifiers.

Thus, the overall worst-case running time is given by $O(M \log M \cdot N + P \cdot M)$. Since $P < M$, it simplifies to $O(M \log M \cdot N)$. This shows that the proposed attack can be carried out with low computational complexity.

E. Ethical Considerations and Responsible Disclosure

While unveiling new security vulnerabilities plays a crucial role in driving more robust and resilient system designs, it must be done with prudence to prevent potential misuse by malicious actors. We understand the weight of responsibility that comes with disclosing attack methodologies against devices used in critical infrastructure sectors like IoT devices. From an ethical standpoint, our intention is not to aid nefarious purposes, but to raise awareness about hardware-based fingerprinting vulnerabilities that could severely undermine user privacy. By responsibly disclosing our findings through this publication, we aim to initiate an open discourse and prompt the research community, industry practitioners, and policymakers to prioritize the development of multi-layered defenses.

To mitigate potential misuse risks, we have anonymized all sensitive information related to specific device configurations and omitted low-level implementation details that could enable straightforward recreations of the attack. Furthermore, Section VII include possible defenses and mitigation techniques for this attack to be explored in future.

VII. Future directions

While this work highlights a fundamental vulnerability arising due to hardware imperfections, there are several potential strategies that could be explored to defend against such privacy attacks. Since the attack relies on uniqueness in clock characteristics, introducing intentional dithering or obfuscation of the clock signal could help mitigate device fingerprinting. However, this may impact timing-sensitive operations.

Clock calibration and controlled jitter injection are two suggestions to counteract fingerprinting attempts by adversaries that use clock variations. PUFs could reduce hardware signature leakage while improving synchronization protocols to allow for more accurate device clock alignment. The data that can be accessible for fingerprinting can be limited using a variety of methods at the PHY, MAC, and network levels. Moreover, engineering absolutely identical hardware behavior across devices will remain an open challenge.

Applying physical layer transformations like Direct Sequence Spread Spectrum (DSSS) or frequency hopping could distort the sub-carrier peak locations and make fingerprinting more difficult. Redesigning wireless protocols to avoid long preambles and minimize imperfection-based modulations could reduce fingerprinting vulnerabilities at the cost of increased overhead. Processing data locally and sending obfuscated/encrypted data over wireless links could prevent privacy leakage from physical signals. Combining multiple techniques like obfuscation, shielding, and edge computing could provide stronger multi-layered defenses against fingerprinting.

VIII. Conclusion

This paper presented a new way to attack the privacy of IoT devices. The proposed attack exploits the variation in the internal clock of microcontrollers to identify the source of packets using statistical diversity among the sub-carrier frequency peaks. The diversity among successive packets is determined using the Jensen–Shannon divergence. Once the adversary gathers sufficient data packets, it uses them to train a machine learning classifier. We demonstrate the effectiveness of the proposed attack using an experimental setup consisting of PlutoSDR and HackRF nodes. Experimental results show that the proposed attack can identify the source of a data packet with approximately 100% accuracy.

To the best of our knowledge, this is the first work which uncovers the vulnerability of using hardware imperfections in the internal clock of IoT devices. However, this paper also demonstrates device fingerprinting through clock skew analysis, opening up future applications in authentication, counterfeit detection, and surveillance. Our results also demonstrate the need for cross-layer privacy mechanisms. Further research can explore exploiting hardware imperfections for security goals while also guarding against potential privacy violations.

References

[1] S. S. Kuruppu and A. Shibilski, "Clock variation impact on digital control system performance," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2019, pp. 0353–0358.

[2] M. Harris, *How important is your microcontroller clock source*, Altium, 2021. [Online]. Available: https://resources.altium.com/p/how-important-your-microcontroller-clock-source-0.

[3] B. Wittenmark, J. Nilsson, and M. Torngren, "Timing problems in real-time control systems," in *Proceedings of 1995 American Control Conference - ACC'95*, vol. 3, 1995, 2000–2004 vol.3.

[4] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680–1685, 2007.

[5] J. Nilsson, B. Bernhardsson, and B. Wittenmark, "Stochastic analysis and control of real-time systems with random time delays," *Automatica*, vol. 34, no. 1, pp. 57–64, 1998.

[6] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: A survey," *IEEE Network*, vol. 18, no. 4, pp. 45–50, 2004.

[7] K.-J. Lin and A. Herkert, "Jitter control in time-triggered systems," in *Proceedings of HICSS-29: 29th Hawaii International Conference on System Sciences*, vol. 1, 1996, 451–459 vol.1.

[8] P. Ramanathan and R. Butler, "Fault-tolerant clock synchronization in distributed systems," *Computer*, vol. 23, pp. 33–42, Nov. 1990.

[9] M. Tsuchimura, H. Sawada, T. Kurokawa, and Y. Koga, "Fault tolerant ic chip for crystal oscillators," in *[1991] Proceedings Pacific Rim International Symposium on Fault Tolerant Systems*, 1991, pp. 232–237.

[10] T. LeBlanc, "Shared memory versus message-passing in a tightly-coupled multiprocessor: A case study.," English (US), in *Proceedings of the International Conference on Parallel Processing*, K. Hwang, S. Jacobs, and E. Swartzlander, Eds., ser. Proceedings of the International Conference on Parallel Processing, IEEE, Dec. 1986, pp. 463–466.

[11] L. Lamport and P. M. Melliar-Smith, "Synchronizing clocks in the presence of faults," *J. ACM*, vol. 32, no. 1, pp. 52–78, Jan. 1985.

[12] J. Y. Halpern, B. Simons, R. Strong, and D. Dolev, "Fault-tolerant clock synchronization," in *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '84, Vancouver, British Columbia, Canada: Association for Computing Machinery, 1984, pp. 89–102.

[13] T. K. Srikanth and S. Toueg, "Optimal clock synchronization," *J. ACM*, vol. 34, no. 3, pp. 626–645, Jul. 1987.

[14] K. G. Shin and P. Ramanathan, “Clock synchronization of a large multiprocessor system in the presence of malicious faults,” *IEEE Transactions on Computers*, vol. C-36, no. 1, pp. 2–12, 1987.

[15] G. Shen, J. Zhang, A. Marshall, L. Peng, and X. Wang, “Radio frequency fingerprint identification for lora using deep learning,” *IEEE Journal on Selected Areas in Communications*, vol. 39, pp. 2604–2616, 8 Aug. 2021.

[16] S. Riyaz, K. Sankhe, S. Ioannidis, and K. R. Chowdhury, “Deep learning convolutional neural networks for radio identification,” *IEEE Communications Magazine*, vol. 56, pp. 146–152, 2018.

[17] I. Agadakos, N. Agadakos, J. Polakis, and M. R. Amer, “Chameleons’ oblivion: Complex-valued deep neural networks for protocol-agnostic rf device fingerprinting,” *Proceedings - 5th IEEE European Symposium on Security and Privacy, Euro S and P 2020*, pp. 322–338, Sep. 2020, Protocol-agnostic DL based RFFI.

[18] A. Al-Shawabka, F. Restuccia, S. D’Oro, *et al.*, “Exposing the fingerprint: Dissecting the impact of the wireless channel on radio fingerprinting,” *Proceedings - IEEE INFOCOM*, vol. 2020-July, pp. 646–655, Jul. 2020, ISSN: 0743166X.

[19] S. Alhazbi, S. Sciancalepore, and G. Oligeri, “The day-after-tomorrow: On the performance of radio fingerprinting over time,” May 2023. [Online]. Available: https://arxiv.org/abs/2305.05285v1.

[20] P. M. S. Sánchez, A. H. Celdrán, G. Bovet, and G. M. Pérez, “Adversarial attacks and defenses on ml- and hardware-based iot device fingerprinting and identification,” *Future Generation Computer Systems*, vol. 152, pp. 30–42, Mar. 2024, ISSN: 0167-739X. DOI: 10.1016/J.FUTURE.2023.10.011.

[21] C. Kuzniar, M. Neves, V. Gurevich, and I. Haque, “Poiriot: Fingerprinting iot devices at tbps scale,” *IEEE/ACM Transactions on Networking*, 2024, ISSN: 15582566. DOI: 10.1109/TNET.2024.3395278.

[22] J. Feng, T. Zhao, S. Sarkar, *et al.*, “Fingerprinting iot devices using latent physical side-channels,” *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 7, no. 2, Jun. 2023.

[23] S. Johar, N. Ahmad, A. Durrani, and G. Ali, “Proof of pseudonym: Blockchain-based privacy preserving protocol for intelligent transport system,” *IEEE Access*, vol. 9, pp. 163 625–163 639, 2021.

[24] J. Petit, F. Schaub, M. Feiri, and F. Kargl, “Pseudonym schemes in vehicular networks: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 228–255, 2015.

[25] I. Gutiérrez-Agüero, S. Anguita, X. Larrucea, A. Gomez-Goiri, and B. Urquizu, “Burnable pseudo-identity: A non-binding anonymous identity method for ethereum,” *IEEE Access*, vol. 9, pp. 108 912–108 923, 2021.

[26] M. Akil, L. Islami, S. Fischer-Hübner, L. A. Martucci, and A. Zuccato, “Privacy-preserving identifiers for iot: A systematic literature review,” *IEEE Access*, vol. 8, pp. 168 470–168 485, 2020.

[27] M. N. Aman, U. Javaid, and B. Sikdar, “Iot-proctor: A secure and lightweight device patching framework for mitigating malware spread in iot networks,” *IEEE Systems Journal*, pp. 1–12, 2021.

[28] M. N. Aman, M. H. Basheer, S. Dash, *et al.*, “Prom: Passive remote attestation against roving malware in multicore iot devices,” *IEEE Systems Journal*, vol. 16, no. 1, pp. 789–800, 2022.

[29] 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, Toronto, Canada: Association for Computing Machinery, 2018, pp. 1502–1514.

[30] R. Chow, “The last mile for iot privacy,” *IEEE Security & Privacy*, vol. 15, no. 6, pp. 73–76, 2017.

[31] A. Barua, M. A. Al Alamin, M. S. Hossain, and E. Hossain, “Security and privacy threats for bluetooth low energy in iot and wearable devices: A comprehensive survey,” *IEEE Open Journal of the Communications Society*, vol. 3, pp. 251–281, 2022.

[32] S. A. Wright, “Privacy in iot blockchains: With big data comes big responsibility,” in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 5282–5291.

[33] S. Imtiaz, R. Sadre, and V. Vlassov, “On the case of privacy in the iot ecosystem: A survey,” in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2019, pp. 1015–1024.

[34] R. S. Apare and S. N. Gujar, “Research issues in privacy preservation in iot,” in *2018 IEEE Global Conference on Wireless Computing and Networking (GCWCN)*, 2018, pp. 87–90.

[35] N. Fotiou, V. A. Siris, A. Mertzianis, and G. C. Polyzos, “Smart iot data collection,” in *2018 IEEE 19th International Symposium on ”A World of Wireless, Mobile and Multimedia Networks” (WoWMoM)*, 2018, pp. 588–599.

[36] C. Yin, J. Xi, R. Sun, and J. Wang, “Location privacy protection based on differential privacy strategy for big data in industrial internet of things,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3628–3636, 2018.

[37] M. Bi, Y. Wang, Z. Cai, and X. Tong, “A privacy-preserving mechanism based on local differential privacy in edge computing,” *China Communications*, vol. 17, no. 9, pp. 50–65, 2020.

[38] F. Zhu, X. Yi, A. Abuadbba, I. Khalil, S. Nepal, and X. Huang, “Cost-effective authenticated data redaction

with privacy protection in iot," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 678–11 689, 2021.

[39] H. Bi, J. Liu, and N. Kato, "Deep learning-based privacy preservation and data analytics for iot enabled healthcare," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 7, pp. 4798–4807, 2022.

[40] A. Ukil, S. Bandyopadhyay, and A. Pal, "Iot-privacy: To be private or not to be private," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2014, pp. 123–124.

[41] W. Li, T. Song, Y. Li, L. Ma, J. Yu, and X. Cheng, "A hierarchical game framework for data privacy preservation in context-aware iot applications," in *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*, 2017, pp. 176–177.

[42] K. Wang, C.-M. Chen, Z. Tie, M. Shojafar, S. Kumar, and S. Kumari, "Forward privacy preservation in iot-enabled healthcare systems," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1991–1999, 2022.

[43] A. S. Tanenbaum and D. J. Wetherall, *Computer networks*. Prentice Hall, 2011.

[44] A. Al-Fuqaha, M. Guizani, M. Mohammadi, A. Rayes, and A. Kamal, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys &amp; Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[45] S. B. Weinstein, "The history of orthogonal frequency-division multiplexing [history of communications]," *IEEE Communications Magazine*, vol. 47, no. 11, pp. 26–35, 2009.

[46] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, pp. 145–151, 1 1991, ISSN: 15579654.

[47] R. G. Lyons, *Understanding digital signal processing*, 3rd. Prentice Hall, 2011, ch. 13.20, p. 954.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.