RF Fingerprinting for LoRa Device Authentication: Dataset Collection and Characterization

Priya Gautam[1], Merugu Jahnavi[1], Shubham Pandey[1], and Hari Prabhat Gupta[1]
Uma Maheswara[2], Kavin Thangadorai[2], and Michael Baddeley[2]
[1]Dept. of CSE, Indian Institute of Technology (BHU) Varanasi, INDIA
[2] Technology Innovation Institute, Abu Dhabi, UAE
{priyagautam.rs.cse24, merugu.jahnavi.cse21, shubhampandey.rs.cse21, hariprabhat.cse}@iitbhu.ac.in
{uma.maheswara, kavin.thangadorai, michael.baddeley}@@tii.ae

***Abstract*—We present a novel dataset for LoRa device authentication using Radio Frequency Fingerprinting (RFFI), addressing IoT security challenges in resource-constrained environments. Our dataset captures hardware-specific signal characteristics from 23 LoRa devices through three complementary representations: raw IQ samples, FFT spectra, and time-frequency spectrograms. Collected using a USRP B200 receiver with GPS synchronization, the data incorporates both coarse and fine Carrier Frequency Offset (CFO) estimations for enhanced feature analysis. The dataset supports deep learning approaches while maintaining an authentication-focused partition (20 training/3 testing devices) to evaluate real-world generalization. The primary contribution of this work is a multi-modal RF fingerprinting dataset specifically designed for LoRa networks. Beyond the core dataset, we further contribute integrated CFO annotation enabling hybrid authentication methods. Importantly, we also establish a reproducible experimental framework using software-defined radio that supports future research. Together, these contributions facilitate advancements in physical-layer security, lightweight device authentication, and robust wireless fingerprinting systems.**

***Index Terms*—Carrier Frequency Offset estimation, Deep learning, In-phase and Quadrature (IQ) samples, Long Range (LoRa) authentication, Radio Frequency fingerprinting, spectrogram analysis, Wireless security**

I. Introduction

With the exponential growth of Internet of Things (IoT) and wireless communication systems, ensuring secure device authentication has become increasingly critical. Traditional cryptographic authentication schemes often require significant computational resources, which may not be feasible for low-power and low-cost devices such as those deployed in LoRa (Long Range) networks [1], [2]. To address this challenge, Radio Frequency Fingerprinting (RFFI) has emerged as a promising physical-layer security technique that exploits inherent hardware imperfections in the transmitted signals for device identification and authentication.

RFFI techniques rely on the subtle, unique distortions introduced by components such as power amplifiers, oscillators, and filters within the transmitter. These distortions manifest in the signal's IQ samples, frequency characteristics, and time-frequency patterns. By capturing and analyzing these features, it becomes possible to distinguish between devices even when they use the same communication protocol and transmit identical data.

In this work, we present a comprehensive dataset preparation framework for deep learning-based RFFI, tailored specifically to LoRa devices. Our dataset includes three distinct signal representations—raw IQ samples, FFT results, and spectrograms—to extract rich and diverse features for training and evaluation. The signals were captured using USRP B200 hardware in GNU Radio, with a total of 23 different LoRa devices participating in the data collection. Among these, three devices were reserved exclusively for device authentication testing, while the remaining were used to train a Convolutional Neural Network (CNN) model for classification.

Additionally, Carrier Frequency Offset (CFO) values were estimated using both coarse and fine estimation techniques, and stored in a separate CFO database for future use in hybrid classification schemes. These CFO values serve as complementary features that further enhance the robustness of the identification process.

By providing a structured and multi-representational dataset along with an authentication-focused experimental setup, this work lays the groundwork for more accurate and practical deep learning-based RF fingerprinting systems in real-world LoRa environments.

The key contributions of this work include:

1) **Comprehensive LoRa Device Dataset Collection:**We present a novel RF dataset comprising IQ samples collected from 23 different LoRa-based devices using a USRP receiver. The data was acquired under controlled conditions with consistent transmission intervals, enabling reproducible analysis.
2) **Multi-Representation Signal Preprocessing:**Each transmission instance is represented in three distinct formats — raw IQ samples, FFT-transformed frequency-domain data, and time-frequency spectrograms using STFT — to support diverse signal processing and learning approaches.
3) **CFO-Aware Data Annotation:** Coarse and fine Carrier Frequency Offsets (CFO) were estimated for each device during transmission, and a dedicated CFO database was maintained per device. This enables researchers to ex-

This work was supported by SSRC Secure Comms, WAKEUP (LoRa) Project, Funded by Technology Innovation Institute (TII) Research institute in Abu Dhabi, United Arab Emirates and C3iHub, Cybersecurity and Cybersecurity of Cyber-physical Systems, IIT Kanpur.

plore frequency-offset-related features without requiring additional synchronization.

4) **Authentication-Focused Partitioning:** A subset of three devices was reserved for device authentication testing, while the remaining 20 devices were used to generate training data, making the dataset suitable for evaluating generalization and cross-device performance.
5) **Open Applicability for Future Research:**The dataset is designed to support a range of research areas including RF fingerprinting, physical-layer authentication, anomaly detection, and lightweight IoT security. It is compatible with both classical and deep learning-based workflows.

The rest of the report is organized as follows: the next section presents the experimental setup and instrumentation. Sections III and IV describe the environment setup and the software and hardware Setup, respectively. Sections V, VI, and VII cover the Dataset Overview, Quick Start Guide, and details about the CFO Dataset. Finally, the conclusion is presented in Section VIII.

II. Experimental Setup and Instrumentation

The experimental framework for this RF fingerprinting research incorporates a carefully selected combination of hardware and software components, each chosen to meet specific requirements for signal acquisition, processing, and analysis. The system architecture follows a modular design philosophy, enabling independent optimization of each subsystem while maintaining interoperability through standardized interfaces. This section details the complete toolchain, from RF hardware through to data analysis pipelines.

A. Hardware Platform

The hardware configuration consists of three primary components: a high-performance software-defined radio receiver, multiple LoRa transmitter nodes, and supporting test equipment. This setup enables precise capture and analysis of device-specific RF characteristics.

- **USRP B200mini-i Software Defined Radio Receiver**:
  - *Manufacturer*: Ettus Research (now NI)
  - *Interface*: USB 3.0 SuperSpeed (5 Gbps)
  - *Frequency Range*: 70 MHz - 6 GHz
  - *Instantaneous Bandwidth*: 56 MHz
  - *ADC Resolution*: 12-bit @ 61.44 MS/s
  - *RF Frontend*:
    - Programmable gain: 0-76 dB in 1 dB steps
    - Noise figure: 5-7 dB typical
    - I/Q imbalance: 0.2 dB amplitude, 1 phase
  - *Configuration*:
    - Center frequency: 868.3 MHz (EU LoRa band)
    - Sampling rate: 2 MS/s (complex baseband)
    - Antenna: VERT900 vertical dipole (3 dBi gain)

  The USRP B200 serves as the primary RF acquisition platform, capturing raw I/Q samples with sufficient dynamic range to resolve subtle device-specific artifacts. The receiver was synchronized to a 10 MHz GPS-disciplined oscillator (Stanford Research Systems FS725) to maintain absolute frequency reference accuracy 50 ppb.
- **LoRa Transmitter Nodes**:
  - *Device Composition*:
    - 15 commercial LoRa nodes (MultiTech mDot)
    - 8 custom nodes based on Semtech SX1276
  - *Key Parameters*:
    - Frequency: 868.3 MHz ±20 ppm
    - Output power: +14 dBm ±0.5 dB
    - Modulation: LoRa (SF7, 125 kHz BW)
    - Preamble: 8 symbols (fixed length)
    - Packet interval: 2sec (synchronized via external trigger)
  - *Hardware Variations*:
    - 3 different PCB designs
    - 2 distinct TCXO manufacturers (Abracon vs. Epson)
    - Mixed PA configurations (SKY65405 vs. RFX2401C)

  The 23-node testbed provides sufficient device diversity to evaluate fingerprinting robustness against real-world hardware variations while maintaining controlled transmission parameters. Fig. 1 illustrates such nodes.

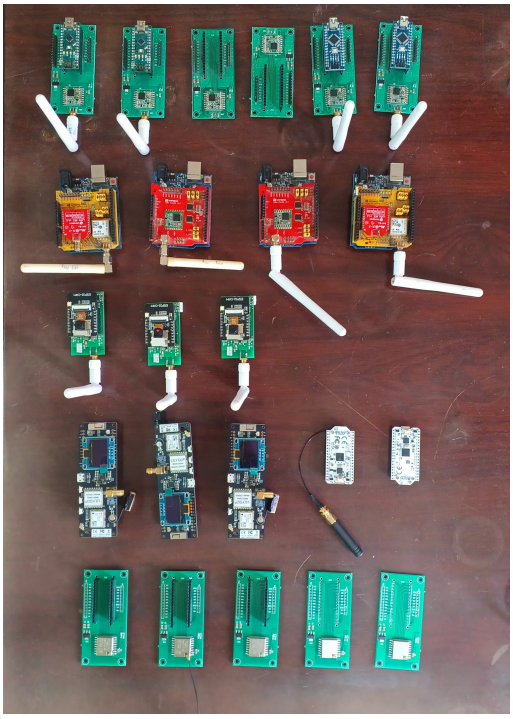

Fig. 1: Complete hardware testbed showing (A) USRP receiver with reference clock, (B) LoRa device array in anechoic chamber, and (C) test controller with power monitoring. The inset shows spectral characteristics of simultaneous transmissions from 5 nodes.

- **Arduino-Based Test Controller**:
  - *Base Platform*: Arduino MKR WAN 1310
  - *Core Specifications*:

    - ∗ MCU: SAMD21 Cortex-M0+ (48 MHz)
    - ∗ LoRa IC: Murata CMWX1ZZABZ (SX1276-based)
    - ∗ Timing accuracy: ±1 ppm (with external 32.768 kHz crystal)
  - *Test Pattern Generation*:
    - ∗ Fixed payload: 32-byte pseudorandom sequence
    - ∗ Transmission scheduling: TDMA with 10 ms guard intervals
    - ∗ Power monitoring: INA219 current sensor (0.5% accuracy)

  The controller provides precise timing synchronization across all DUTs while logging power consumption metrics for correlation with RF features.

B. Software Framework and Processing Pipeline

The software architecture was designed to support real-time signal acquisition, processing, and analysis through an integrated toolchain.

1) Signal Processing Framework:

- **GNU Radio Companion (v3.10.7.0)**:
  - *Configuration*:
    - ∗ UHD source block @ 2 MS/s sampling rate
    - ∗ 256 MB circular buffer for continuous acquisition
    - ∗ Stream tags for packet boundary detection
  - *Custom Python Blocks*:
    - ∗ Real-time IQ capture (32-bit complex float)
    - ∗ CFO estimation:
      - · Coarse: FFT peak detection (2048-point)
      - · Fine: Phase gradient method (0.1 Hz resolution)
    - ∗ CFO compensation using NCO with PID control
    - ∗ Packet detection via normalized cross-correlation (threshold = 0.85)
  - *Performance*: Sustained 1.8 MS/s processing on Intel i7-1185G7
- **Python Analysis Stack (v3.12.1)**:
  - *Core Libraries*:
    - ∗ NumPy v1.26.4: Vectorized operations and FFT implementation
      - · 2048-point FFT with Blackman-Harris window
      - · Complex64 dtype for memory efficiency
    - ∗ Matplotlib v3.8.2: Visualization and analysis
      - · Spectrograms (256-sample window, 50% overlap)
      - · Constellation diagrams with EVM calculation
    - ∗ SciPy v1.11.4: Signal processing
      - · STFT with 1 Hz resolution
      - · FIR filtering (Kaiser window, $\beta = 8$)
      - · Peak finding (prominence ¿ 3σ)
  - *Custom Modules*:
    - ∗ RF fingerprint extraction (500-sample preamble analysis)
    - ∗ Device classification (SVM and CNN implementations)
    - ∗ Statistical analysis of device features
- **Development Environment**:
  - Visual Studio Code (v1.85.1)
    - ∗ Python extension with linting (pylint)
    - ∗ Jupyter notebook integration
    - ∗ GNU Radio Companion plugin
  - Version Control: Git v2.43 with LFS for large binaries
  - Documentation: Sphinx v7.2.6 with MathJax support

2) Data Management System:

- **IQ Data Storage**:
  - Format: Binary (complex64)
  - Organization: Per-device directories with capture logs
- **Feature Database**:
  - JSON schema v7:
    - ∗ Device metadata (ID, hardware version)
    - ∗ CFO statistics (mean, σ, min/max)
    - ∗ Spectral features (5 dominant peaks)
    - ∗ Temporal features (rise/fall times)
  - Query interface via Python pandas v2.1.3
- **Training Data**:
  - FFT magnitude: HDF5 format (100,000 x 1024)
  - Spectrograms: PNG (1280×720) + NumPy arrays
  - Labels: One-hot encoded (23 classes)

3) Computing Environment:

- **Ubuntu 22.04.3 LTS**:
  - Kernel: Linux 5.15.0-86-generic (RT patches)
  - Filesystem: XFS with 4k block size
  - USRP UHD driver v4.4.0 (FPGA images v12.3)
  - CPU governor: performance mode
- **Anaconda Environment**:
  - Python 3.12.1 base environment
  - Conda-forge channel priority
  - Dependency pinning (exact versions)
  - Docker image for reproducibility
    - ∗ Size: 4.7 GB (includes CUDA 12.1)
    - ∗ Hash: SHA-256 verified

III. Environment Setup

The experiments were conducted in a controlled indoor environment to ensure consistent signal propagation and minimize the effects of external interference or environmental variability. A long table—approximately 8 feet in length—served as the physical layout for the setup, with the transmitter and receiver placed at opposite ends to establish a stable line-of-sight (LoS) communication path.

At the transmitting end, a LoRa module was connected to a PC via an Arduino board. The Arduino was programmed using the Arduino IDE to periodically transmit data packets at fixed intervals, emulating a continuous and consistent source of RF signals. These LoRa transmissions traversed the indoor environment, inherently carrying device-specific hardware imperfections essential for RF fingerprinting.

On the receiving side, a USRP (Universal Software Radio Peripheral) device was connected to a separate PC running GNU Radio. The USRP was configured to continuously monitor the incoming LoRa transmissions. Upon reception, the raw IQ (In-phase and Quadrature) samples were captured in complex format and processed in real-time using a custom-designed GNU Radio flowgraph. This processing included Carrier Frequency Offset (CFO) estimation, compensation, and storage of both raw and processed IQ data.

To ensure communication reliability and experimental repeatability, parameters such as bandwidth (e.g., 250 kHz) and spreading factor (e.g., SF = 7) were carefully synchronized between the transmitter and receiver. The static positioning of both devices throughout the experiment further ensured a consistent RF environment with minimal multipath distortion.

To better understand the experimental configuration and ensure clarity in the description of device placements and signal propagation paths, a schematic representation of the setup was created. This visualization helps illustrate the controlled environment in which the experiments were conducted and highlights the critical spatial arrangements necessary for achieving consistent and reliable measurements. Fig. 2 illustrates the physical layout of the experimental setup, highlighting the positions of the transmitter, receiver, and the surrounding space used for signal propagation.

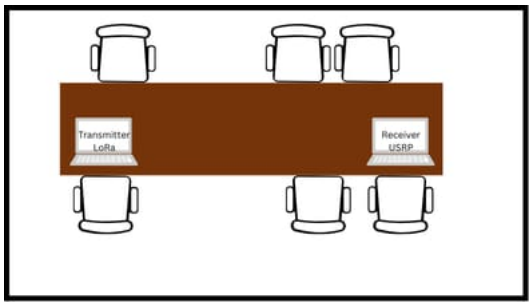

Fig. 2: Schematic representation of the experimental setup, showing the spatial arrangement of the transmitter and receiver across a long table in a controlled indoor environment. The layout ensures a stable line-of-sight (LoS) communication path for signal propagation and minimizes the impact of external interference and multipath effects.

To complement the schematic view, a prototype of the actual experimental setup was also assembled and documented. This prototype showcases the real hardware, connections, and environment in which the measurements were carried out, providing a practical perspective on the experimental arrangement. Fig. 3 presents the prototype of the experimental setup, capturing the actual devices, wiring, and configuration used during data collection.

In this experiment, a total of 23 LoRa devices, spanning different models and manufacturers, were utilized to serve as the transmitters, providing a diverse range of hardware characteristics essential for robust RF fingerprinting analysis. A USRP B200 software-defined radio platform was employed at the receiver end to perform high-fidelity signal acquisition and real-time IQ sample collection. This combination of multiple heterogeneous LoRa transmitters and a flexible, high-resolution receiver setup ensured comprehensive coverage of device-specific variations and enhanced the reliability of the experimental results.

- **Transmitter**: The LoRa-based transmitter was implemented using a LoRa device connected to a PC via an Arduino board. This setup was programmed through the Arduino IDE to periodically transmit data packets at fixed intervals, thereby emulating a consistent and continuous RF signal source. As the LoRa signals propagated across the indoor environment, they interacted with surrounding objects and structural elements, embedding device-specific hardware imperfections into the signal. These imperfections are crucial for enabling effective RF fingerprinting.
- **Receiver**: The receiving side consisted of a USRP (Universal Software Radio Peripheral) B200 connected to a PC running GNU Radio. The USRP was configured to continuously monitor and capture the incoming LoRa transmissions. The received signals were stored as raw IQ (In-phase and Quadrature) samples in complex format. Further signal processing was performed, employing techniques such as the Fast Fourier Transform (FFT) and Short-Time Fourier Transform (STFT) to extract discriminative features. The receiver setup was critical in capturing both time-domain and frequency-domain representations necessary for constructing a robust RF fingerprinting dataset.

Key Questions:

To gain deeper insights into the design choices and experimental methodology, the following key questions were considered throughout the study. These questions aim to critically evaluate the consistency, synchronization, and feature extraction processes that underpin the success of RF fingerprinting in a controlled environment:

1) *How does the fixed-interval packet transmission from the Arduino-controlled LoRa transmitter influence the consistency of the captured IQ data at the receiver?*
2) *How critical is the synchronization between the Arduino-transmitted LoRa signals and the USRP sampling configuration in ensuring reliable RF fingerprint collection?*
3) *How does the choice of processing techniques, such as FFT and STFT, enhance the ability to extract unique features from the received IQ samples for RF fingerprinting?*

IV. Software and Hardware Setup

This section outlines the hardware configuration and software environment setup necessary for the signal acquisition, processing, and authentication framework. The primary hardware used includes multiple USRP devices for data acquisition and various IoT nodes for authentication tests. The software stack was built around GNU Radio within a Conda-managed workspace to enable modular, flexible development and testing.

Fig. 3: Photograph of the experimental prototype, showing the real-world implementation of the transmitter, receiver, and associated components in the controlled indoor environment.

A. Configuring USRP and Installing GNU Radio

To initiate the signal acquisition process, the USRP device was configured, and GNU Radio was installed within a Conda-managed environment. Initially, Miniconda was installed, and a new environment was created and activated using `conda activate <env_name>`. The system packages were updated using the commands `sudo apt-get update` and `sudo apt-get dist-upgrade`.

Support for USRP hardware was installed by adding the Ettus Research UHD PPA repository using `sudo add-apt-repository ppa:ettusresearch/uhd`, followed by the installation command `sudo apt-get install libuhd-dev libuhd003 uhd-host`.

The connectivity and recognition of the USRP device were verified by executing `uhd_usrp_probe`. In case the device was not detected, the firmware and FPGA images were downloaded using `sudo /usr/lib/uhd/utils/uhd_images_downloader.py`. Upon successful detection, running `uhd_usrp_probe` again displayed the connected device's details.

For LoRa signal processing, the necessary blocks were installed within the Conda environment using `conda install -c tapparelj -c conda-forge gnuradio-lora_sdr`. GNU Radio itself was installed via `sudo apt install gnuradio`, and its graphical interface was launched with the command `gnuradio-companion`.

Table I lists the devices and their respective models used for signal acquisition experiments. Table II presents the devices designated for authentication studies.

TABLE I: Device Inventory

Device	Model
1–4	Nano
5–7	Dragino
8–10	ESP32
11–13	Lillygo
14–15	Heltec
16–19, 0	Nano 433MHz

TABLE II: Devices Selected for Authentication

Device	Model
20	Dragino
21–22	Nano

B. Flowgraph Design in GNU Radio

A customized GNU Radio flowgraph was developed to facilitate real-time signal reception and processing. Several key blocks were integrated to define critical parameters and operations, particularly for Carrier Frequency Offset (CFO) estimation and compensation.

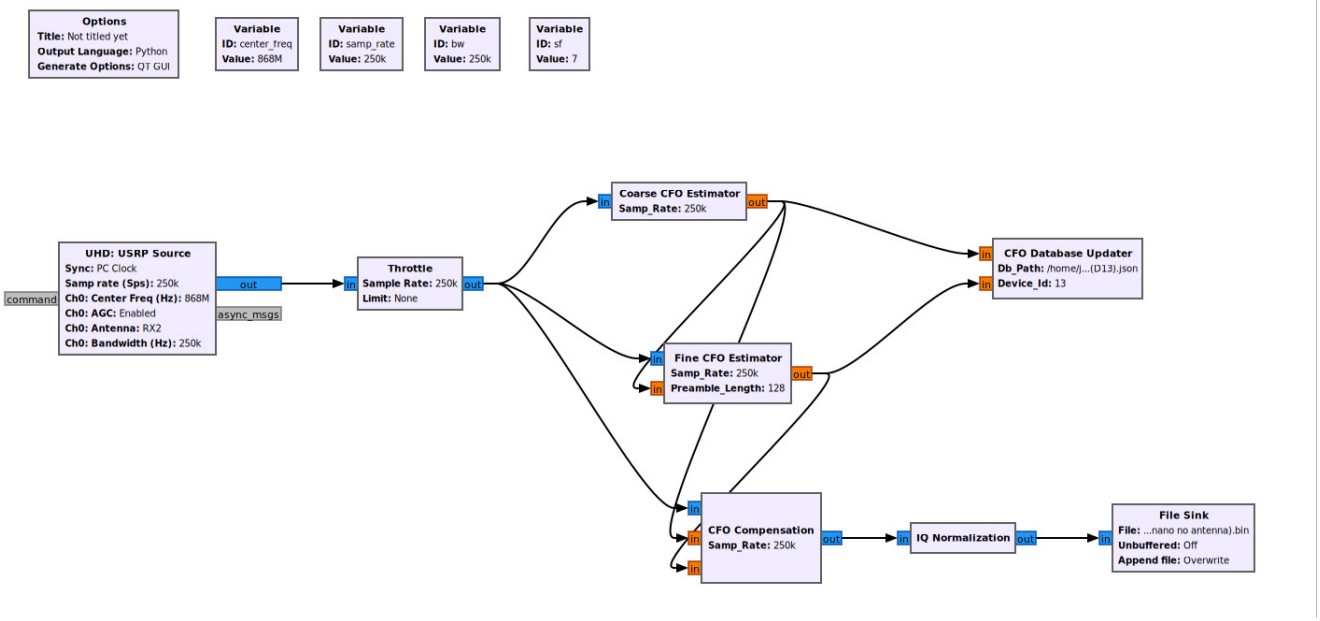

Fig. 4: Custom Flowgraph for Signal Acquisition and Processing in GNU Radio

The `UHD: USRP Source` block interfaced with the USRP to receive complex IQ samples. Variable blocks were used to configure parameters such as sample rate, bandwidth, and spreading factor. A `Throttle` block was included to control the data flow rate, preventing CPU overload during simulation runs.

The signal was processed through custom Python blocks, each handling a specific function: Coarse CFO Estimation, Fine CFO Estimation, CFO Compensation, IQ Normalization, and CFO Database Updating. The CFO estimation blocks extracted the frequency offset induced by hardware imperfections, and the compensation block corrected for these offsets. Normalized IQ samples were then stored using a `File Sink` block. The CFO Database Updater maintained per-device metrics such as minimum, maximum, mean, and range of CFO estimates. Figure 4 illustrates the designed flowgraph.

Key Questions:

- How does synchronization impact the overall signal acquisition and processing performance?
- In what ways does using GNU Radio within a Conda-managed environment simplify the installation and customization of modules such as CFO estimators and LoRa processing blocks?

C. Signal Representations

For robust RF fingerprinting and deep learning-based device authentication, the raw received signals must be transformed into representations that highlight device-specific characteristics. In this work, three distinct signal representations were employed: IQ samples, FFT results, and spectrograms.

- **IQ Samples:** The IQ samples represent the time-domain complex-valued signals received by the USRP, expressed as:

$$x[n] = I[n] + jQ[n],$$

where $I[n]$ and $Q[n]$ are the in-phase and quadrature components, respectively. IQ samples preserve both amplitude and phase information, capturing hardware-induced imperfections such as distortions and frequency offsets. Stored in complex format (e.g., `-0.0052 + 0.0034j`), they offer fine-grained temporal resolution and are ideal for analyzing phase noise, CFO, and modulation artifacts.
- **FFT Results:** The Fast Fourier Transform (FFT) is applied to IQ samples to analyze the signal in the frequency domain. This reveals the distribution of signal energy across frequencies and highlights anomalies such as CFO and spectral leakage. Each FFT bin corresponds to a frequency component, with the magnitude indicating power and the phase indicating frequency behavior. The FFT transformation is performed using the following Python code:
data=np.fromfile(input_file_path,dtype=np.complex64)
fft_result = np.fft.fft(data)
This transformation helps to identify device-specific spectral signatures that are difficult to detect in the time domain.
- **Spectrograms:** Spectrograms are generated using the Short-Time Fourier Transform (STFT), offering a time-frequency view of the signal. Unlike FFT, spectrograms partition the signal into overlapping segments and apply FFT to each, showing how frequency content evolves over time. This is particularly useful for non-stationary signals

like LoRa, where linear chirps span a wide frequency range over each symbol duration. Spectrograms are visualized as 2D plots, where the x-axis represents time, the y-axis represents frequency, and pixel intensity represents signal power. Spectrogram generation is implemented as follows: for idx, packet in enumerate(iq_packets): f, t, Sxx = spectrogram(packet, fs=FS, nperseg=256). Spectrograms reveal time-dependent variations and transient features that aid in device fingerprinting.

V. Dataset Overview

The data collection process involved two main types of raw data: **CFO_Dataset** and **IQ** samples, both of which form the foundation for further analysis and model development. From the raw IQ samples, additional derived representations were generated, including:

1) **IQ to FFT**: Converts time-domain IQ samples to frequency-domain data using the Fast Fourier Transform (FFT).
2) **IQ to Spectrograms**: Generates time-frequency representations that capture how the signal's frequency components evolve over time.

he dataset comprises four directories: noitemsep,topsep=0pt

- **CFO_Dataset**: Carrier frequency offset measurements.
- **IQ**: Time-domain I/Q samples.
- **FFT**: Frequency-domain transforms.
- **Spectrograms**: Time-frequency visualizations.

A. CFO_Dataset

The **CFO_Dataset** folder, which is primarily used for device authentication tasks, contains individual files for each device under test. Each file records four key CFO-related metrics that characterize the frequency behavior of the respective device:

- **Minimum CFO**: The lowest observed value of the total CFO over all recorded transmissions.
- **Maximum CFO**: The highest observed value of the total CFO during the same period.
- **Mean CFO**: The average of all total CFO values, providing an overall sense of frequency offset behavior.
- **Threshold** (λ): Defined as half the difference between the maximum and minimum CFO, i.e.,

$$\lambda = \frac{\text{Maximum CFO} - \text{Minimum CFO}}{2}.$$

The **total CFO** is computed as the sum of the **Coarse CFO** and **Fine CFO**:

- **Coarse CFO**: Estimated by calculating the average frequency deviation over the preambles.
- **Fine CFO**: Estimated by measuring the phase difference between repeated preamble segments.

These CFO values are essential for identifying device-specific behaviors and form the core of RF fingerprinting, where each device's unique frequency characteristics are leveraged for authentication.

B. IQ Folder

The **IQ** folder contains the raw complex IQ samples collected directly from the LoRa devices. Each IQ sample consists of two components:

- **In-phase (I) component**: Represents the real part of the signal.
- **Quadrature (Q) component**: Represents the imaginary part of the signal.

These IQ samples capture the signal's amplitude and phase information, which are crucial for analyzing the time-domain characteristics of the received RF signals, such as phase noise and frequency offsets.

C. FFT Folder

The **FFT** folder contains the results of applying the Fast Fourier Transform (FFT) to the IQ samples, which converts them from the time domain to the frequency domain. This transformation helps visualize how the signal's energy is distributed across different frequencies.

Frequency-domain analysis is particularly useful for identifying anomalies such as carrier frequency offsets (CFO) and spectral leakage. Each FFT output provides insight into the signal's spectral characteristics, enabling the identification of device-specific signatures that are difficult to detect in the time-domain IQ samples.

D. Spectrograms Folder

The **Spectrograms** folder holds time-frequency representations of the signals, generated using the Short-Time Fourier Transform (STFT). Unlike the FFT, which provides a snapshot of the frequency spectrum at a single time point, the spectrogram tracks how the signal's frequency content evolves over time.

Spectrograms are visualized as two-dimensional images, with the x-axis representing time, the y-axis representing frequency, and the pixel intensity representing signal power. These time-frequency representations are critical for capturing dynamic distortions and signal characteristics that evolve throughout the transmission, such as modulation artifacts and time-dependent frequency offsets. Fig. 5 represents the dataset overview.

E. Training Dataset

Table III summarizes the basic information for each training dataset. All datasets were collected in a residential room with a line-of-sight (LOS) connection between the transmitter and receiver.

TABLE III: Details of the training datasets.

Training Dataset Path	Devices	Number of Packets per Device
Dataset/IQ	0–19	2000
Dataset/FFT	0–19	2000
Dataset/Spectrograms	0–19	2000

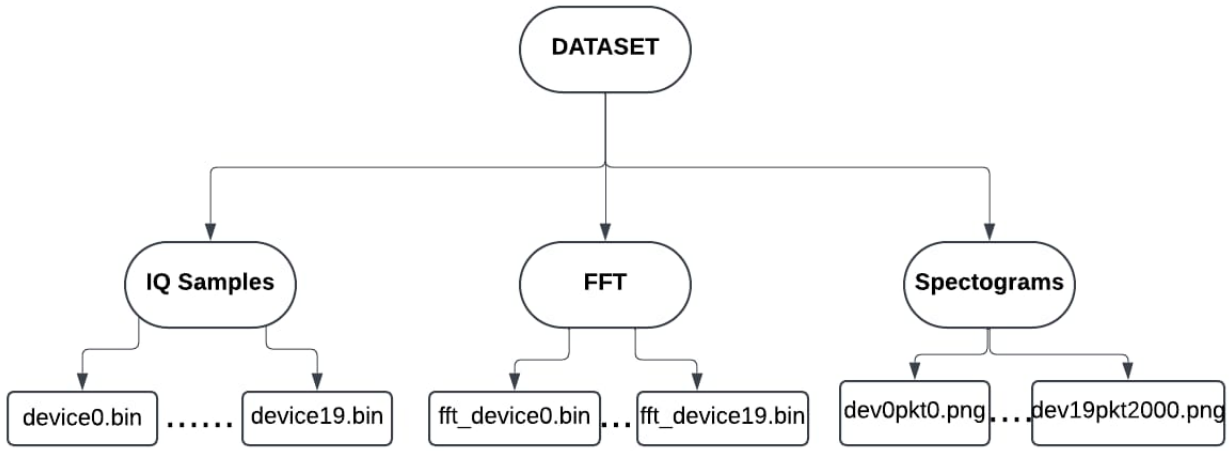

Fig. 5: Overview of the dataset organization, showing the relationship between raw data (CFO measurements), processed time-domain (IQ samples), frequency-domain (FFT transforms), and visual representations (spectrograms).

F. Test Dataset

For testing, we selected three devices specifically for authentication purposes. The IQ samples for these devices were collected in the same format as the training datasets. Table IV summarizes the information for each testing dataset.

TABLE IV: Details of the testing datasets.

Test Dataset Path	Devices	Number of Packets per Device
Dataset/IQ	20–22	2000
Dataset/FFT	20–22	2000
Dataset/Spectrograms	20–22	2000

VI. Quick Start

This section provides a step-by-step guide to quickly start working with the dataset and the training pipeline. It covers loading the IQ datasets, assigning labels, extracting packets, generating dynamic batches for training, and splitting the data into training and testing sets. By following these instructions, users can efficiently prepare the data for model training and evaluation.

A. Loading IQ Datasets and Label Assignment

After successfully downloading the dataset and installing GNU Radio, the IQ samples are stored in `.bin` files located within a folder named `IQ`. Each binary file corresponds to data collected from a specific LoRa device, with filenames following a convention that includes numeric identifiers unique to each device. These numeric identifiers are later used for assigning labels to the data.

The IQ dataset consists of multiple `.bin` files, all located in the `IQ` folder. The `extract_label()` function extracts numeric values from the filenames (e.g., `Dev1.bin` becomes label 1). These numeric values serve as class labels for the classification task.

B. Packet Extraction Logic

Each `.bin` file contains a sequence of complex IQ samples stored in `np.complex64` format. A fixed number of samples, denoted as `PACKET_SIZE = 83824`, represents one packet. The number of packets in a file is computed by dividing the total number of samples by `PACKET_SIZE`. This ensures that each packet is of uniform size and is consistently extracted across all devices.

C. Generator for Dynamic Batch Loading

A generator is used to dynamically load packets in batches. For each batch, the generator selects packets from all device files. It reads specific segments from each file using the `f.seek()` function and loads the IQ data with `np.fromfile()`. The complex IQ signal is then split into real and imaginary components. Each processed packet is appended to the batch along with its corresponding label.

D. Packet Labeling

Each packet extracted from a file is labeled with the device number, which is extracted from the filename. This labeling is performed within the generator to ensure that each packet is correctly associated with its respective device label.

The final output from the generator is a batch containing:

- **Input Shape:** `(batch_size, PACKET_SIZE, 2)` – representing the real and imaginary parts of the IQ samples.
- **Labels:** Numeric class labels corresponding to each device.

Figure 6 illustrates the IQ samples after labeling.

E. Training and Testing Dataset Split

The dataset is split into training and testing phases, with 80% of the device files used for training and the remaining

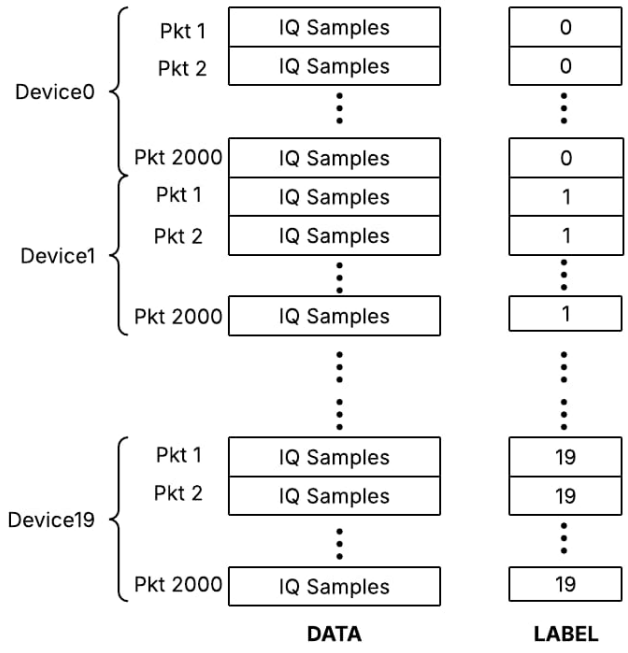

Fig. 6: Visualization of IQ samples after labeling.

20% reserved for testing. This ensures that samples from each device are present in both the training and testing datasets, providing a robust evaluation framework.

VII. About CFO_Dataset

The **CFO_Dataset** is an essential component of the RF fingerprinting dataset, specifically designed to capture the Carrier Frequency Offset (CFO) characteristics of different LoRa devices. CFO arises due to slight mismatches in the oscillators of the transmitting and receiving hardware. Since these mismatches are device-specific and relatively stable over time, CFO serves as a powerful feature for identifying and authenticating individual devices.

A. Purpose of CFO_Dataset

- **Device Fingerprinting:** CFO acts as a unique fingerprint for each device, caused by oscillator imperfections during transmission.
- **Lightweight Authentication:** Instead of relying solely on complex signal representations or deep learning outputs, CFO can be used for lightweight authentication mechanisms.
- **Hybrid Approaches:** The CFO values collected here can be fused with neural network predictions to improve robustness and reliability.

B. Collection Methodology

The CFO estimation is carried out using a GNU Radio flowgraph, which includes the following components:

- **Coarse CFO Estimator Block:** Computes a rough estimate of CFO based on the linear frequency progression of LoRa preambles.
- **Fine CFO Estimator Block:** Estimates the residual CFO using the repeating structure of the preambles.
- **CFO Database Updater Block:** Takes both coarse and fine estimates as input, adds them to calculate the total CFO, and updates a central database.

For each device, multiple packets are transmitted, and the CFO is estimated for every received packet. As new values are observed, the updater maintains four statistical values per device:

- Minimum CFO value observed across all packets.
- Maximum CFO value observed across all packets.
- Mean CFO, computed as the average of the minimum and maximum values.
- Threshold (λ), computed as half the difference between the maximum and minimum CFO values.

VIII. Conclusion

In this work, we introduced a versatile dataset preparation framework for deep learning-based RFFI, specifically designed for LoRa devices. By capturing raw IQ samples, FFT data, and spectrograms from 23 LoRa transmitters, we created a multi-representational dataset enriched with CFO values to support hybrid authentication approaches. Our reproducible experimental setup and authentication-focused partitioning enable robust model evaluation, advancing research in RF fingerprinting and IoT security. This work lays a foundation for future explorations into multi-feature fusion for scalable and resilient LoRa authentication.

References

[1] S. Pandey, P. Kumari, H. P. Gupta, D. Rai, and S. V. Rao, "A site-specific lorawan parameters selection approach with multi-loss propagation model," in *IFIP Networking*, 2024, pp. 350–358.

[2] P. Kumari, H. P. Gupta, T. Dutta, and S. K. Das, "Improving age of information with interference problem in long-range wide area networks," in *IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2022, pp. 137–146.