

Contents

1. Introduction and Research Context	1
1.1 Background	1
1.1.1 Current Developments in 5G Security and Fake Base Station Detection	2
1.2 Significance of the Research	2
1.3 Brief Literature Summary Leading to the Research/Literature Gap	3
1.4 Significance of the Research	3
2. Problem Statement and Research Gap	3
2.1 Research Problem.....	3
2.2 Literature Support for the Gap	4
3. Research Aim and Objectives	4
3.1 Aim.....	4
3.2 Objectives (SMART).....	4
4. Literature Review	5
4.1 Categories of FBS Detection.....	5
4.2 Limitations of Existing Work.....	7
4.3 Research Gap and Contribution.....	7
5. Research Methodology	7
5.1 Research Design and Approach.....	7
5.2 Selection of Methods and Materials (Justification).....	8
5.3 Data Collection.....	8
5.4 Data Preprocessing and Feature Engineering.....	9
5.5 Evaluation and Comparison	9
5.6 Ethical Considerations.....	9
6. Project Plan and Resources	9
6.1 Timeline and Milestones	9
6.2 Resources.....	10
7. Expected Outcomes, Impact and Limitations.....	10
7.1 Expected Outcomes	10
7.2 Impact.....	10
7.3 Limitations.....	11

References	12
------------------	----

FIGURE 1. USER EQUIPMENT (UE) ATTACHING TO A ROGUE (FAKE) GNODEB INSTEAD OF THE LEGITIMATE BASE STATION	5
FIGURE 2. SECURITY THREAT OF BASE STATION	6
FIGURE 3. PROPOSED EDGE-BASED FBS DETECTION PIPELINE FROM SDR SIGNAL CAPTURE TO ML- BASED DETECTION AND OPTIONAL BLOCKCHAIN LOGGING	7
FIGURE 4. ML PIPELINE FOR FAKE BASE STATION DETECTION FROM DATA COLLECTION TO DEPLOYMENT	9

Research Proposal

An AI-Based Low-Cost System for Detecting Fake 5G Base Stations Using Open 5G Signal Measurements

Author: Sayem Akther Talukdar

Student ID: 2502919

Program: MSC In Cybersecurity & Digital Forensics

Module: Research Methods and Professional Development

1. Introduction and Research Context

1.1 Background

Fake base stations (FBS) are devices that emulate legitimate cellular base stations, causing user equipment within range to connect to them as if they were part of the authorized network [1], [3]. They exploit the fact that UEs typically attach to the strongest available signal and that critical broadcast and early handshake messages in 4G and 5G are transmitted without authentication [1], [4], [5]. Attackers can thereby request permanent or subscription-concealed identifiers (IMSI or SUCI), redirect traffic, or force downgrade to vulnerable 2G/3G protocols [3], [5]. The radio resource control (RRC) protocol has also been shown to be exploitable for prolonged denial-of-service and information leakage until later 5G protections apply [2], [5]. With mobile connectivity ubiquitous in society, the need to detect and mitigate FBS threats is well recognized in both industry and academia [6], [7], [9].

Current developments in 5G security have not eliminated the FBS threat. Specification-based detectors such as SMDFbs achieve high accuracy (around 98%) by modelling normal base-station behavior as a state machine [1]. Machine learning (ML) approaches using reference signal received power (RSRP) and related measurement features have demonstrated 75–96% detection rates with low false-positive percentages [2], [3], [6]. Crowdsourced systems such as SeaGlass and FBS-Radar enable large-scale monitoring but depend on fleet operators and centralized infrastructure [7], [10]. Nevertheless, these solutions typically require expensive hardware, large or proprietary datasets, or complex protocol modelling, and few are evaluated on resource-constrained edge devices such as single-board computers [1], [2], [6]. There is growing interest in low-cost, deployable FBS detectors that can be operated by smaller network operators or community groups [6], [11]. Current debates center on the trade-off between detection accuracy and deployment cost, and on the need for open, reproducible datasets to train lightweight models suitable for edge devices [2], [3], [6].

1.1.1 Current Developments in 5G Security and Fake Base Station Detection

In recent years, the rapid deployment of 5G networks has introduced significant improvements in mobile communication, including higher data speeds, lower latency, and enhanced security mechanisms. One notable advancement is the use of Subscription Concealed Identifiers (SUCI), which aims to protect user identity privacy compared to older systems such as IMSI used in previous generations [4]. Additionally, stronger encryption protocols and authentication procedures have been integrated into 5G architecture to reduce common security risks [2], [5]. However, despite these improvements, several vulnerabilities still exist, particularly during the early stages of network communication. Initial broadcast messages and connection procedures are not fully authenticated, which creates opportunities for attackers to deploy fake base stations (FBS), also known as IMSI catchers or Stingrays [2], [5]. These malicious entities can trick user devices into connecting to them, enabling data interception, identity exposure, or forced network downgrades [3].

To address these challenges, recent research has increasingly focused on machine learning-based detection techniques. These approaches analyse radio signal features such as RSRP, RSRQ, and signal-to-noise ratio to identify abnormal patterns that may indicate the presence of a fake base station [3], [6]. At the same time, there is a growing trend towards lightweight and edge-based solutions, where detection systems are deployed on low-cost devices such as Raspberry Pi. This shift aims to make detection more accessible and scalable, especially for smaller organisations or community-level monitoring [6]. Despite these developments, most existing solutions still rely on expensive hardware, large datasets, or computationally intensive models. As a result, there remains a clear need for affordable, efficient, and real-time detection systems that can operate on resource-constrained devices without compromising accuracy [2], [6].

1.2 Significance of the Research

Despite significant advancements in 5G security mechanisms, a critical gap remains in the availability of low-cost, real-time fake base station detection systems suitable for deployment on resource-constrained edge devices. Existing detection approaches often rely on expensive hardware, large-scale datasets, or computationally intensive models, which limit their practical deployment in real-world environments [1], [2], [6].

Fake base stations (FBS), also known as IMSI catchers or Stingrays, exploit weaknesses in early-stage communication procedures of cellular networks. Although 5G introduces improved security features such as encrypted identifiers and enhanced authentication, initial broadcast messages and connection setups remain vulnerable to manipulation [2], [5]. As a result, malicious entities can impersonate legitimate base stations and force user devices to connect, leading to risks such as identity exposure, data interception, and network downgrading [3].

Current detection solutions face several challenges. Specification-based approaches require complex protocol modelling and deep knowledge of network internals, making them difficult to implement in low-cost environments [1]. Machine learning-based methods, while effective, often depend on large datasets and high-performance computing resources, limiting their suitability for edge deployment [2], [6]. Additionally, many existing systems do not evaluate critical

performance factors such as inference latency, CPU and memory usage, which are essential for real-time detection on embedded devices.

These limitations highlight a clear research problem: the lack of an efficient, affordable, and real-time fake base station detection system that can operate effectively on low-cost edge hardware while maintaining acceptable detection accuracy and performance.

1.3 Brief Literature Summary Leading to the Research/Literature Gap

A brief synthesis of the literature leads directly to this gap. Specification-based methods (e.g. [1]) and ML-based methods (e.g. [2], [3], [6]) achieve high accuracy but rely on costly or complex setups; crowdsourced approaches (e.g. [7], [10]) require large-scale infrastructure. None of these provide a low-cost, edge-deployable solution with open datasets and explicit latency and resource evaluation. Therefore, the present research focuses on filling that gap by building a Raspberry Pi-based detector with an open dataset and documented performance trade-offs.

1.4 Significance of the Research

This research is significant as it addresses a critical gap in existing fake base station detection systems by focusing on low-cost and real-time solutions suitable for edge devices. The study contributes to the academic field by exploring the balance between detection accuracy, computational efficiency, and deployment feasibility. Furthermore, it provides practical insights into the use of lightweight machine learning techniques for enhancing cellular network security.

The potential applications of this research include:

- Telecom security monitoring for detecting rogue base stations in real-world environments
- Community-based detection systems that enable low-cost monitoring without reliance on large infrastructure
- IoT and edge security systems where resource-constrained devices require efficient threat detection mechanisms

2. Problem Statement and Research Gap

2.1 Research Problem

There is a lack of low-cost, real-time FBS detection systems that (i) run on resource-constrained hardware such as single-board computers, (ii) are supported by open datasets and reproducible methods, and (iii) report inference latency and resource usage necessary for real-time deployment. Existing solutions fall into one or more of the following categories: dependence on costly spectrum analyzers and antennas; reliance on proprietary or large-scale datasets; use of complex protocol state machines that are difficult to scale; or deployment of heavy ML models on powerful servers rather than edge devices [1], [2], [6], [7]. Few studies explicitly evaluate detection models on platforms such as the Raspberry Pi or document end-to-end latency and CPU/memory consumption, which are critical for practical, real-time monitoring [1], [2], [3].

2.2 Literature Support for the Gap

Park et al. [1] present SMDFbs, a specification-based detector that achieves approximately 98% accuracy with low overhead. However, the approach requires full protocol state-machine modelling and is not designed for low-cost edge deployment.

Mubasshir et al. [2] presents FBS Detector, an ML framework that successfully detects over 96% of fake base stations and multi-step attacks with an average false-positive rate of approximately 2.96%. The framework uses need large datasets and powerful hardware, and inference workload is not optimized for embedded device.

Nakarmi et al. [3], Jin et al. [6], and Islam et al. ML is found to perform well for FBS detection on RSRP-based and related measurement features [5]. These works, however, do not covers deployment on Raspberry Pi and open datasets release for light weight classifiers.

Ney et al. [7] and Li et al. [10] and crowdsourced systems (SeaGlass and FBS-Radar) that allow large-scale discovery of FBS. These rely on fleet or operator infrastructure, and thus are not practical for small communities or low-cost deployments.

The combination of these findings create a clear research gap: affordable, edge-deployable FBS detection with open data, lightweight models and documented trade-offs in latency and resources.

3. Research Aim and Objectives

3.1 Aim

This paper presents the design, implementation and evaluation of a low-cost system to detect fake base stations in 4G/5G networks based on a Raspberry Pi and software defined radio. This goal is articulated specifically as a response to the knowledge gap identified: that there exists no currently available low-cost, edge-deployable FBS detection with open data and documented latency and resource usage [1], [2], [3]. Srsran/ns-3 will be used to collect open 5G signal measurements from simulated environments, with features engineered from measurement reports, and lightweight ML models trained for high accuracy and low latency in classifying genuine vs malicious base stations. Another goal is to release an open labelled dataset and implementation guidelines, enabling reproducibility and extensiveness of the work by other researchers and practitioners. Where applicable, the project will investigate whether a lightweight private blockchain can be utilized to allow for tamper-evident logging of detection events similar to principles found in IoT-based blockchain security [8].

3.2 Objectives (SMART)

To achieve the aforementioned goal, the following objectives are established. All are specific, measurable, achievable, relevant to the research gap and time bound according to a 15-week project schedule.

Objective	Description	Link to Research Gap
O1	Conduct a literature survey (Weeks 1–2) and compile at least 10 recent journal or conference sources on FBS attacks and detection methods.	Grounds the work in current research and formalises the gap.

O2	Select and set up hardware (Raspberry Pi 4, RTL-SDR or HackRF) and software (srsRAN, ns-3) to simulate genuine and fake gNodeBs (Week 3).	Enables low-cost, controlled data collection.
O3	Collect at least 5,000 labelled measurement reports (RSRP, RSRQ, SNR, TA, PCI, MNC) under varied conditions; extract and normalize features (Weeks 4–5).	Produces an open dataset and feature set for lightweight ML.
O4	Train and compare five lightweight classifiers (decision tree, random forest, k-NN, SVM, LightGBM) using k-fold cross-validation & hyper parameter tuning; evaluate with accuracy, precision, recall, F1-score and ROC-AUC (Weeks 6–7).	Delivers justified methods and measurable performance.
O5	Deploy the best-performing model on the Raspberry Pi; measure inference latency (target <200 ms per sample) and resource usage (CPU, memory) (Weeks 8–9).	Addresses real-time edge deployment and resource constraints.
O6	Compare cost–accuracy trade-offs with SMDFBs [1] and FBSDetector [2]; optionally design a lightweight blockchain for logging (Week 10).	Positions the contribution against state of the art.
O7	Complete documentation, dataset release, and discussion of ethical considerations and limitations (Weeks 11–15).	Ensures professional dissemination and reproducibility.

4. Literature Review

4.1 Categories of FBS Detection

Fake base stations (FBS), also known as IMSI-catchers or Stingrays, are malicious devices that impersonate legitimate cellular base stations to deceive nearby user equipment (UE) into



Figure 1. User equipment (UE) attaching to a rogue (fake) gNodeB instead of the legitimate base station

connecting to them. They exploit the fact that mobile devices attach to the strongest signal and that early broadcast and handshake messages in 4G/5G networks are not fully authenticated [1], [3]. Once connected, attackers can request sensitive identifiers such as IMSI or SUCI, redirect signaling, or force downgrade to insecure legacy networks [1], [4]. In addition, weaknesses in the Radio Resource Control (RRC) protocol can enable denial-of-service and information leakage [5]. Although 5G improves security, early-stage procedures remain vulnerable [2].

1. **Specification-based detection.** Park et al. [1] propose SMDFbs, which derives behaviour rules from the normal operation of 5G base stations and implements them as a state machine. The system achieves detection accuracy of around 98% with low overhead. Constructing and verifying such protocol state machines is, however, complex and may require access to network internals; the approach is not aimed at low-cost edge deployment.



Figure 2. Security Threat Of Base Station

2. **Rule-based and signal-strength methods.** The early methods have been based on thresholds or rules about signal strength and broadcast identifiers. These are fairly easy to deploy but not very effective against attackers using legitimate physical cell identifiers (PCIs) [3]. Ali and Fischer [9] detect FBS presence on the UE side through higher-order noise statistics by measuring deviations from ideal Gaussian noise, showing that they can catch rogue transmissions based on signal statistics.
3. **Machine learning-based detection.** Nakarmi et al. [3] use ML for RSRP-based and location-related features, achieving 75–95% FBS identification for about 0.5% false positives. Jin et al. [6] deploy LightGBM to classify base stations using measured parameters with high precision. Islam et al. [5] construct a multi-featured anomaly-detection system, reporting good performance. Mubasshir et al. [2] present an ML framework (FBSDetector) capable of detecting more than 96% of fake base stations and

multi-step attacks. These methods are powerful but often depend on large datasets and high-end hardware, and are not evaluated on single-board computers.

4. **Crowdsourced and large-scale detection.** SeaGlass [7] uses sensors in rideshare vehicles to gather cellular measurements and monitor for IMSI-catcher signatures at city scale. FBS-Radar [10] uses large-scale crowdsourced data to uncover FBS and estimate their locations. Such systems demonstrate the value of large-scale monitoring but require coordination with operators and significant infrastructure, and may raise privacy concerns.

4.2 Limitations of Existing Work

Although specification-based and ML detectors achieve high accuracy, they typically require specialized equipment or proprietary datasets [1], [2]. SMDFbs relies on a full protocol state machine that may not scale to diverse network configurations [1]. ML models such as those in FBS Detector depend on large network traces and are not optimized for small embedded devices [2]. Crowdsourced systems need fleet or operator coordination and centralized processing [7], [10]. There is limited research on deploying FBS detectors on resource-constrained hardware such as the Raspberry Pi, and few studies report latency or energy usage, which are critical for real-time edge applications [1], [2], [3].

4.3 Research Gap and Contribution

This review highlights a gap between high-accuracy but resource-intensive FBS detectors and the need for low-cost, deployable solutions with open data and explicit evaluation of latency and resources. This project will design a detection system that runs on a Raspberry Pi, produce a labelled dataset from open-source simulators, evaluate multiple lightweight classifiers, and measure inference latency and resource usage. An optional component will explore tamper-evident logging via a lightweight blockchain design [8], to strengthen trust in generated alerts.

5. Research Methodology

5.1 Research Design and Approach

The research adopts a quantitative, experimental design in a controlled laboratory environment, consistent with established FBS detection studies that use measurement data and ML classification [2], [3], [6]. Genuine and fake gNodeB transmissions will be generated using open-source 5G/LTE simulators (srsRAN and/or ns-3). A Raspberry Pi equipped with a USB SDR will

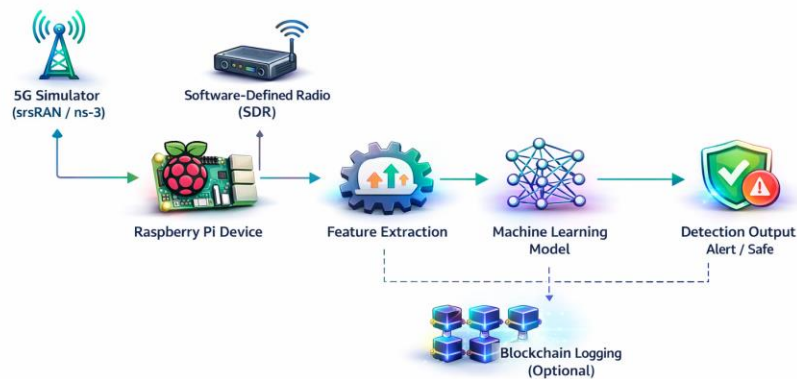


Figure 3. Proposed edge-based FBS detection pipeline from SDR signal capture to ML-based detection and optional blockchain logging

capture measurement reports; no live public networks or real subscriber data will be used, minimizing ethical and safety risks.

5.2 Selection of Methods and Materials (Justification)

Hardware and software. A Raspberry Pi 4 with at least 4 GB RAM will serve as the edge device; a USB SDR (RTL-SDR or HackRF) will capture radio signals. A laptop running Ubuntu will host the base-station simulator. This choice is justified by the project aim of low-cost, reproducible deployment; all components are widely available and open-source software will be used throughout [1], [2], [6].

Algorithms. Five lightweight classifiers—decision tree, random forest, k-nearest neighbours (k-NN), support vector machine (SVM), and Light GBM—will be implemented in Python using scikit-learn (and LightGBM). These are chosen because they are suitable for resource-constrained deployment, allow comparison of accuracy versus complexity, and have been successfully applied in related FBS detection work [3], [6]. Hyper parameters will be tuned via grid search with k-fold cross-validation ($k = 5$). The dataset will be split into training (70%) and test (30%) sets to ensure generalizable performance estimates.

Evaluation metrics. Accuracy, precision, recall, F1-score and area under the ROC curve (ROC-AUC) will be used to evaluate model performance. Analyze false-positive/false-negative using confusion matrices. These metrics are suitable for binary classification and weigh missed detections against false alarms [2], [3], [6].

Deployment and profiling. The winning model (by ROC-AUC) will then be exported (possibly with joblib) and uploaded to the Raspberry Pi. To build an inference pipeline, the whole implementation will be performed in Python. Resource usage (CPU, memory) and end-to-end latency will be measured with standard libraries (e.g. time, psutil), with a target performance of under 200 ms per sample in order to facilitate the real-time monitoring.

Optional: blockchain logging. A lightweight private blockchain design will be proposed for logging detection events based on the work done by Dorri et al followed, if time remains. [8]. To avoid energy-intense consensus (such as proof-of-work) and ensure low overhead on the RaspberryPi, the design will not include it.

5.3 Data Collection

The simulator will then take and capture MEASUREMENT reports on the Raspberry Pi, it will log timestamps and labels whether it is genuine or fake. The data collection process consists of: (i) synthesizing data using the srsRAN/ns-3 simulators in controlled conditions; (ii) recording performance metrics from the system, such as RSRP, RSRQ, SNR, TA, PCI and MNC for each sample; and (iii) labeling each record to allow supervised learning. PCI, MNC, RSRP, RSRQ, SNR and TA will be stored for each record. Data will be collected in different distance, transmit power, and free space conditions. The target is at least 5,000 labelled samples to support robust training and evaluation [3], [6].

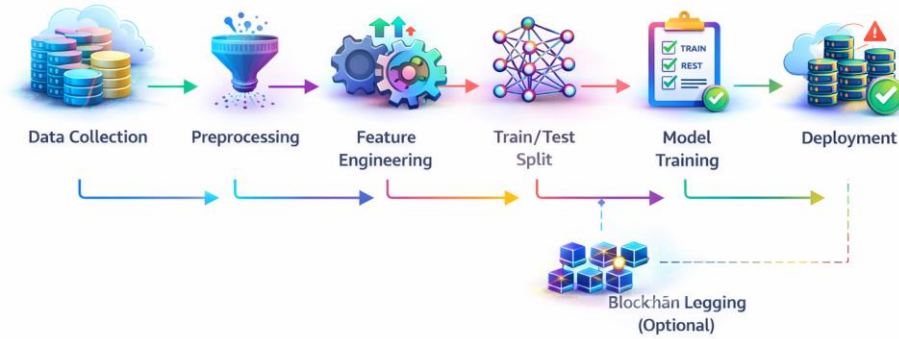


Figure 4. ML pipeline for fake base station detection from data collection to deployment

5.4 Data Preprocessing and Feature Engineering

Raw measurements will be preprocessed in Python. Statistical features (mean, variance, minimum, maximum) will be computed over sliding windows. Temporal features, such as differences between consecutive RSRP readings, will capture abrupt signal changes. Anomalous timing advance values and inconsistent broadcast identifiers will be included where applicable. Features will be normalized to zero mean and unit variance before model training [3], [6].

5.5 Evaluation and Comparison

Model performance will be evaluated on the held-out test set. Results will be compared with reported performance of SMDFbs [1] and FBSDetector [2] to contextualize the trade-offs between cost, accuracy, and deployment context. Deployment evaluation will document end-to-end latency per sample and CPU/memory usage under load.

5.6 Ethical Considerations

All experiments will be confined to a controlled lab environment using simulators. No real subscriber data will be processed. The work will not interfere with public networks. The use of open-source tools and the release of an open dataset will support transparency and reproducibility.

6. Project Plan and Resources

6.1 Timeline and Milestones

Weeks	Activity	Milestone
1–15	Literature survey; finalise sources and research gap.	M1: Literature review complete; gap documented.
3	Hardware and software setup	M2: Testbed operational.

	(Raspberry Pi, SDR, srsRAN/ns-3).	
4–5	Data collection, preprocessing, and feature engineering.	M3: Labelled dataset ($\geq 5,000$ samples) and feature set ready.
6–7	Model design, training, hyper parameter tuning, and evaluation.	M4: Best model selected; metrics reported.
8–9	Deployment on Raspberry Pi; latency and resource profiling.	M5: Edge deployment validated; latency < 200 ms target checked.
10	Comparison with literature; optional blockchain design.	M6: Comparison with [1], [2] documented.
11–15	Report writing, dataset release, and dissemination.	M7: Final report and open dataset published.

6.2 Resources

Hardware: Raspberry Pi 4 (≥ 4 GB RAM), USB SDR (RTL-SDR or HackRF), laptop for simulator (Ubuntu).

Software: srsRAN, ns-3, Python (scikit-learn, Light GBM, joblib, psutil). All software will be open source.

Environment: Lab-only experiments; no use of live public networks or real subscriber data.

7. Expected Outcomes, Impact and Limitations

7.1 Expected Outcomes

- A functional but low-cost FBS detection system deployed on Raspberry Pi with known accuracy, latency, and resource consumption.
- A labeled dataset of measurements reports for reproducible research.
- A comparison of cost–accuracy–latency trade-offs against those of SMDFbs [1] and FBS Detector [2].
- A playbook for replication and extension by other researchers and practitioners.
- An optional design for a lightweight private blockchain for tamper-evident logging of detection events [8].

7.2 Impact

- Right: Academic $\rightarrow$ Reported evidence supporting the feasibility of edge-deployable FBS detection; open dataset and pipeline for future work.
- Professional: A professional solution for smaller operators or community associations to analyze the radio environment.
- Societal: Contribution to user protection against IMSI-catchers cellular security threats.

7.3 Limitations

- The simulated environment may not cover all varieties of real-world channel conditions and attack strategies.
- Single-board compute restricts model complexity; performance comparisons with high-end detectors are contextual not absolute.
- If time is short, the optional blockchain part may stay in design phase.
- The focus is on passive detection based on broadcasting and measurement methods for 4G/5G, not full-core-protocol or core-network applied analysis.

References

- [1] H. Park, P. V. B. Astillo, Y. Ko, Y. Park, T. Kim, and I. You, "SMDFbs: Specification-based misbehaviour detection for false base stations," *Sensors*, vol. 23, no. 23, p. 9504, 2023. doi: 10.3390/s23239504.
- [2] K. S. Mubasshir, I. Karim, and E. Bertino, "Gotta detect 'em all: Fake base station and multi-step attack detection in cellular networks," in *Proc. 34th USENIX Security Symposium*, 2025. [Preprint].
- [3] P. K. Nakarmi, J. Sternby, and I. Ullah, "Applying machine learning on RSRP-based features for false base station detection," *arXiv preprint arXiv:2207.10999*, 2022. [Online].
- [4] M. Chlosta, D. Rupprecht, C. Pöpper, and T. Holz, "5G SUCI-catchers: Still catching them all?" in *Proc. 14th ACM Conf. Security Privacy Wireless Mobile Netw.*, 2021, pp. 359–364. doi: 10.1145/3448300.3467826.
- [5] A. Islam, S. Parvez, and S.-Y. Chang, "Anomaly detection against fake base station threats using machine learning," *J. Cybersecurity Privacy*, vol. 5, no. 4, p. 94, 2025. doi: 10.3390/jcp5040094.
- [6] J. Jin, C. Lian, and M. Xu, "Rogue base station detection using a machine learning approach," in *Proc. 28th Wireless Opt. Commun. Conf. (WOCC)*, 2019, pp. 1–5. doi: 10.1109/WOCC.2019.8770554.
- [7] P. Ney, I. Smith, G. Cadamuro, and T. Kohno, "SeaGlass: Enabling city-wide IMSI-catcher detection," *Proc. Privacy Enhancing Technol.*, vol. 2017, no. 3, pp. 39–56, 2017. doi: 10.1515/popets-2017-0027.
- [8] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, 2017, pp. 618–623. doi: 10.1109/PERCOMW.2017.7917634.
- [9] A. Ali and G. Fischer, "Enabling fake base station detection through sample-based higher order noise statistics," in *Proc. 42nd Int. Conf. Telecommun. Signal Process. (TSP)*, 2019, pp. 695–700. doi: 10.1109/TSP.2019.8769046.
- [10] Z. Li et al., "FBS-Radar: Uncovering fake base stations at scale in the wild," in *Proc. 2017 Netw. Distributed Syst. Security Symp. (NDSS)*, 2017. [Online].
- [11] I.-H. Liu, M.-H. Lee, H.-C. Huang, and J.-S. Li, "5G-based smart healthcare and mobile network security: Combating fake base stations," *Appl. Sci.*, vol. 13, no. 20, p. 11565, 2023. doi: 10.3390/app132011565.