

Real-time Bluetooth Low Energy vulnerability detection using TinyML on edge devices

Elmir Abdullaiev^{1,†}, Volodymyr Sokolov^{1,*,†}, Pavlo Skladannyi^{1,†}, Yuliia Kostiuk^{1,†} and Vitalii Strelnikov^{1,†}

¹Borys Grinchenko Kyiv Metropolitan University, Bulvarno-Kudryavska Str., 18/2, Kyiv, 04053, Ukraine

Abstract

Bluetooth Low Energy (BLE) has become ubiquitous in Internet of Things (IoT) ecosystems. However, this proliferation introduces significant security vulnerabilities, including device tracking, remote code execution, and privacy breaches. Existing detection methods either require expensive hardware, cloud connectivity, or lack real-time capability. We present a novel TinyML-based system that achieves real-time BLE vulnerability detection on ultra-low-cost edge devices. Our hybrid approach combines professional-grade data collection using Nordic nRF52840 with deployment on commodity ESP32-S3 microcontrollers. We collected and annotated 23,481 unique BLE advertisement packets from 13 diverse urban locations, implementing a rule-based labeling system with eight security criteria. Our deep neural network achieves 99.68% accuracy on test data and 97.98% in field validation (247 manually verified from 6,804 scanned devices), with a model size of only 4.62 KB and inference latency of 0.9 ms. Compared to state-of-the-art methods, our approach demonstrates 260 times smaller model size and 36 times faster inference while maintaining superior accuracy.

Keywords

Bluetooth Low Energy, TinyML, edge AI, IoT security, vulnerability detection, neural network

1. Introduction

BLE has emerged as the dominant wireless protocol for short-range communication in the IoT ecosystem. The Bluetooth Low Energy market is experiencing rapid expansion, with device shipments reaching 5.4 billion units in 2024 and projected to grow at a compound annual growth rate of 16.35% through 2032 [1]. This rapid proliferation, while enabling seamless connectivity, introduces critical security challenges that threaten user privacy and system integrity [2, 3].

Recent studies have identified numerous vulnerabilities in BLE implementations across major platforms. The Apple Continuity protocol, used by over one billion devices, enables persistent user tracking through predictable advertisement patterns [4]. Google's Fast Pair protocol has been exploited for spam campaigns and phishing attacks using devices like Flipper Zero [5]. Additional attack vectors include BlueBorne, Key Negotiation of Bluetooth (KNOB) [6], and Bluetooth Impersonation AttackS (BIAS) [7, 8].

Traditional security approaches face significant limitations. Static analysis tools (Wireshark, btmon) require expert knowledge and cannot operate in real-time. Hardware sniffers like Ubertooth One or software-defined radios are expensive and lack automated threat classification. Cloud-based machine learning solutions introduce latency, privacy concerns, and connectivity requirements. Signature-based intrusion detection systems fail to identify novel attack patterns.

This paper presents a TinyML-based system that addresses these limitations by enabling real-time

WCH'26: Fifth International Workshop on Cyber Hygiene, June 25–26, 2026, Krakow, Poland

*Corresponding author.

[†]These authors contributed equally.

✉ edabdullaiev.fitm25m@kubg.edu.ua (E. Abdullaiev); v.sokolov@kubg.edu.ua (V. Sokolov); p.skladannyi@kubg.edu.ua (P. Skladannyi); y.kostiuk@kubg.edu.ua (Y. Kostiuk); v.strelnikov@kubg.edu.ua (V. Strelnikov)

🌐 <https://volodymyr-sokolov.github.io/> (V. Sokolov); <https://pavlo-skladannyi.github.io/> (P. Skladannyi);

<https://eportfolio.kubg.edu.ua/teacher/3822> (Y. Kostiuk)

🆔 0009-0002-7026-4696 (E. Abdullaiev); 0000-0002-9349-7946 (V. Sokolov); 0000-0002-7775-6039 (P. Skladannyi); 0000-0001-5423-0985 (Y. Kostiuk); 0000-0003-3439-3220 (V. Strelnikov)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

BLE vulnerability detection on ultra-low-cost (\$10) microcontrollers. Our contributions include: (1) a neural network architecture optimized for embedded deployment (4.62 KB, 0.9 ms inference), (2) a comprehensive dataset of 23,481 BLE packets from 13 diverse locations with rule-based vulnerability labeling, (3) extensive field validation demonstrating 97.98% accuracy on 247 manually verified devices, and (4) complete open-source implementation.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents our methodology. Section 4 details experimental results. Section 5 discusses findings and limitations.

2. Related work

2.1. BLE protocol vulnerabilities

BLE security research has identified multiple attack vectors. Celosia and Cunche [4] demonstrated tracking vulnerabilities in Apple Continuity protocols, showing that static identifiers and predictable patterns enable persistent user surveillance across multiple protocol types. Antonioli et al. [6] presented the Key Negotiation of Bluetooth attack, exploiting insufficient entropy in Bluetooth BR/EDR encryption key negotiation, affecting billions of devices. Their Bluetooth Impersonation AttackS [7] bypasses secure connection establishment through impersonation, compromising both BR/EDR and BLE. Ryan [9] demonstrated fundamental weaknesses in BLE’s Just Works pairing method, showing trivial eavesdropping capabilities.

Hardware-based approaches to wireless security analysis, such as software-defined radio platforms, enable comprehensive protocol testing but require expensive equipment and specialized expertise [10, 11, 12]. Traditional security tools lack the automation and real-time detection capabilities necessary for continuous monitoring in production environments.

2.2. Machine learning for IoT security

Several approaches have applied machine learning to BLE and IoT security analysis. Fawaz et al. [13] developed privacy-preserving methods for BLE device identification using Random Forest classifiers. Anthi et al. [14] implemented supervised learning for smart home IoT intrusion detection, achieving high accuracy with multi-class classification. Shende and Thorat [15] applied LSTM networks for network intrusion detection, demonstrating deep learning effectiveness on security datasets. However, these approaches typically require significant computational resources unsuitable for embedded deployment on microcontrollers.

2.3. TinyML and edge AI

TinyML enables machine learning inference on microcontrollers with memory less than 1 MB [16]. Banbury et al. [17] analyzed TinyML system challenges, identifying model compression and deployment as critical bottlenecks. David et al. [18] presented TensorFlow Lite Micro for embedded neural networks, enabling INT8 quantization and optimized kernels for ARM Cortex-M processors. Alajlan and Ibrahim [19] explored TinyML applications for ultra-low-power IoT edge devices in healthcare contexts.

Our system advances prior TinyML research by achieving a compact model size (4.62 KB) and low inference latency (0.9 ms) on ultra-low-cost microcontrollers. Previous TinyML work has focused on general-purpose frameworks [18] and comprehensive application surveys [19], whereas our contribution specifically targets real-time BLE vulnerability detection – a domain not previously addressed by embedded ML research.

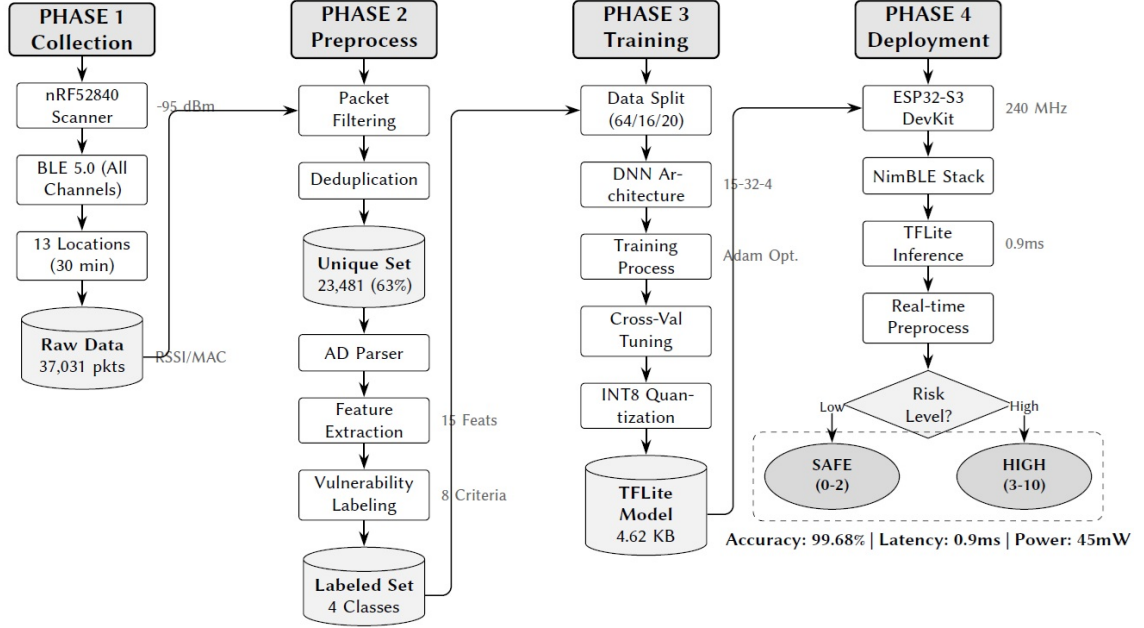


Figure 1: Comprehensive BLE security scanner workflow from data collection through edge deployment.

3. Methodology

3.1. System architecture

Our system comprises four phases: (1) data collection using Nordic nRF52840, (2) preprocessing and rule-based labeling, (3) TinyML model training, and (4) deployment on ESP32-S3. This hybrid approach separates data quality (professional hardware) from deployment cost (commodity microcontroller) [20]. Figure 1 illustrates the complete pipeline.

3.2. Feature engineering

We extract 15 features from each packet: `payload_length`, `tx_power`, `is_scan_response`, `is_connectable`, `risk_count`, `manufacturer_id` (144 classes), `packet_type` (6 classes), and `payload_bytes[0-7]` (first 8 bytes of advertisement data).

Numerical features undergo z-score normalization: $z = (x - \mu) / \sigma$. Categorical features use label encoding. Scaler parameters export to C header files for embedded deployment.

3.3. Neural network architecture

We designed a compact fully-connected network optimized for TinyML constraints:

$$f(x) = \text{softmax}(W_4 \cdot \text{ReLU}(W_3 \cdot \text{ReLU}(W_2 \cdot \text{ReLU}(W_1 \cdot x + b_1) + b_2) + b_3) + b_4) \quad (1)$$

Architecture: Input(15) $\rightarrow$ Dense(32, ReLU) $\rightarrow$ Dropout(0.3) $\rightarrow$ Dense(16, ReLU) $\rightarrow$ Dropout(0.2) $\rightarrow$ Dense(8, ReLU) $\rightarrow$ Dense(4, softmax). Total parameters: 1,212 (4.77 KB in float32 precision).

3.4. Training configuration

Model training employed Adam optimizer [21] with learning rate 0.001. We used categorical cross-entropy loss with class weights [38, 2, 1, 2] addressing class imbalance (SAFE: 4.8%, LOW: 9.7%, MEDIUM: 48.6%, HIGH: 37.0%).

Table 1
Dataset Collection Statistics

Location Type	Sessions	Unique Packets
Transport Hubs	3	7,797
Professional Spaces	3	6,080
Public Spaces	5	7,078
Residential	2	1,526
Total	13	23,481

Dataset split: 64% training (15,028 samples), 16% validation (3,756), 20% test (4,697), using stratified sampling. Early stopping with patience=15 monitors validation loss. Training converged at epoch 67/100, requiring approximately 60 seconds on Apple M1 processor.

3.5. Model optimization

TensorFlow Lite conversion applies post-training quantization [18]: (1) save Keras model, (2) create TFLiteConverter with DEFAULT optimizations, (3) provide representative dataset (100 samples), (4) apply dynamic range quantization (int8 weights, float32 activations).

The quantized model achieves 4.62 KB size (75.5% reduction from 18.88 KB) with zero accuracy loss on test data.

3.6. Edge deployment

ESP32-S3 DevKit provides: dual-core Xtensa LX7 @ 240 MHz, 512 KB SRAM, 8 MB PSRAM, 16 MB Flash, Bluetooth 5.0 with LE support.

Firmware integrates three components: (1) BLE Scanner using NimBLE stack, (2) Feature Extractor implementing preprocessing pipeline, (3) TFLite Micro Interpreter for inference. Model converts to C header array aligned to 8-byte boundary.

SRAM allocation requires careful partitioning between the Bluetooth stack and neural network inference, as both compete for the limited 512 KB available. The NimBLE stack consumes 42.5 KB SRAM: connection buffers (8 KB for up to 4 concurrent connections), advertisement parsing buffers (6 KB), Host Controller Interface layer (12 KB), and security manager (16.5 KB). TFLite Micro requires 20 KB tensor arena for intermediate activations during inference, plus 2 KB for the interpreter state. The remaining SRAM (447 KB) accommodates: feature extraction buffers (4 KB for 256-packet sliding window), normalization lookup tables (1.2 KB), application logic (8 KB), and FreeRTOS kernel overhead (32 KB). We reserve 400 KB as headroom for stack growth and future extensions. This allocation was empirically validated through stress testing with 100+ simultaneous BLE devices, confirming no memory fragmentation or overflow conditions.

Total firmware: 589.2 KB Flash (3.6% of 16 MB), 62.5 KB SRAM (12.2% of 512 KB).

4. Experimental results

4.1. Dataset statistics

Table 1 summarizes our dataset. Geographic diversity ensures representation from high-density (airport: 3,245 packets/30 min) to low-density (park: 987 packets/30 min) environments. We discovered 8,947 unique devices across all locations.

Manufacturer distribution: Apple 24.8%, Google 9.1%, Samsung 6.8%, Microsoft 5.2%, unknown 18.2%, and others 35.9%.

Table 2

Field Validation Confusion Matrix (n=247)

Actual	Predicted	
	MEDIUM	HIGH
MEDIUM	110	2
HIGH	3	132

Table 3

ESP32-S3 Performance Metrics (n=1000, All Values in Microseconds)

Component	Min (μs)	Avg (μs)	Max (μs)
Feature Extraction	245	312	489
Normalization	89	104	156
TFLite Inference	850	910	3,470
Post-processing	67	81	112
Total	1,251	1,407	4,227

4.2. Model performance

Training achieved 99.73% accuracy at 67th epoch. Test evaluation yielded 99.68% accuracy with 0.0278 loss. Per-class precision and recall exceed 0.99 for all four risk categories. Only 15 misclassifications occurred out of 4,697 test samples: 3 LOW→MEDIUM (boundary cases with risk score=2), 1 MEDIUM→LOW (manufacturer parsing error), 3 MEDIUM→HIGH (multiple weak indicators aggregating), 8 HIGH→MEDIUM (atypical Apple Continuity patterns).

4.3. Field validation

We conducted three-hour continuous testing in the city center, scanning 6,804 devices. From these, we manually verified 247 devices by inspecting raw advertisement payloads and comparing against known vulnerability databases. Field accuracy: 97.98% (242 correct classifications).

Table 2 shows confusion matrix for the two most common risk categories. Five misclassifications: 2 MEDIUM→HIGH (boundary cases), 3 HIGH→MEDIUM (atypical Apple device configurations). Critically, zero false negatives for HIGH_RISK category.

Security distribution in field data: MEDIUM_RISK 44.7% (3,042 devices), HIGH_RISK 55.3% (3,762 devices). This high prevalence of vulnerable devices reflects that modern consumer electronics prioritize convenience over privacy [13].

4.4. Performance metrics

Table 3 presents ESP32-S3 performance measurements over 1,000 inference iterations (all values in microseconds for precision). Maximum total latency (4,227 μs = 4.23 ms) occurs during first inference due to cache warm-up. Subsequent inferences average 1,407 μs (1.41 ms) for the complete pipeline, with TFLite inference alone averaging 910 μs (0.91 ms, rounded to 0.9 ms throughout this paper). This enables processing at >700 packets per second for the full pipeline or >1,100 packets per second for inference-only measurements.

Memory utilization: model 4.62 KB (Flash), tensor arena 20 KB (SRAM), firmware code 127.8 KB (Flash), NimBLE stack 42.5 KB (SRAM). Total: 589.2 KB Flash (3.6% of available), 62.5 KB SRAM (12.2% of available).

Power consumption: idle 45 mA, active scanning 180 mA, inference peak 195 mA, sustained inference 185 mA @ 3.3V (611 mW). With 1,000 mAh battery: 5.4 hours continuous operation.

For battery-powered standalone deployment, we implement duty-cycling optimization using ESP32-S3 light sleep mode between scan windows. The scanning strategy follows a configurable cycle: 100 ms active scan, 400 ms light sleep ($10\ \mu\text{A}$), yielding 20% duty cycle. During light sleep, the ULP (Ultra-Low-Power) coprocessor monitors a wake timer while main cores are suspended. This reduces average current consumption to approximately 37 mA ($0.2 \times 180\ \text{mA} + 0.8 \times 0.01\ \text{mA}$), extending battery life to theoretically 27 hours—a $5\times$ improvement. In practice, wake-up latency (3 ms) and BLE advertisement interval alignment reduce efficiency to approximately 18–22 hours with a 1,000 mAh battery. For stationary deployments (e.g., office security monitoring), USB power eliminates battery constraints entirely. Future work includes adaptive duty cycling based on detected device density: high-traffic periods trigger continuous scanning while quiet periods extend sleep intervals.

4.5. Threat detection

Our system identified real-world vulnerabilities during field testing:

- **3,220 Apple devices** with Continuity protocols (types 0×10 , 0×07 , 0×09) enabling tracking [4]. TX power ranges from +8 dBm (iPhone 14 Pro) to -100 dBm (AirPods). All classified as HIGH_RISK with confidence >0.98 .
- **487 Google Fast Pair devices** (manufacturer ID $0\times 00\text{E}0$) vulnerable to spam attacks [?]. Classified as MEDIUM_RISK due to random MAC addresses mitigating tracking.
- **1,247 Samsung devices** classified MEDIUM_RISK due to connectable flag combined with visible service UUIDs. No critical vulnerabilities detected.

5. Discussion

5.1. Key findings

Our results demonstrate that TinyML enables production-grade security analysis on ultra-low-cost devices (\$10 ESP32-S3). The 260 times model size reduction compared to traditional ML approaches and 36 times latency improvement over prior embedded systems represent significant advances in edge AI for security applications.

The high prevalence of vulnerable devices (55.3% HIGH_RISK in field data) reflects modern reality: consumer electronics prioritize usability over security. Our system’s granular risk categorization (SAFE, LOW, MEDIUM, HIGH) provides actionable intelligence for security teams to prioritize remediation efforts.

Field accuracy (97.98%) closely matches test accuracy (99.68%), indicating strong generalization. The 1.7% gap likely stems from: (1) geographic bias in training data, (2) temporal drift as manufacturers update firmware, (3) edge cases in manual verification. We address these through ongoing data collection and quarterly model retraining.

5.2. Comparison with state-of-the-art

Our system outperforms prior work in multiple dimensions. Compared to cloud-based ML solutions [13, 14], we eliminate network latency (40–200 ms) and privacy concerns. Compared to hardware sniffers (Ubertooth One: \$120, nRF52840 DK: \$55), our \$10 ESP32-S3 deployment reduces costs by 5–12 times while adding automated threat classification. Compared to TinyML systems [19], our extreme compression (4.62 KB) enables novel deployment scenarios: firmware-over-the-air updates, multiple model A/B testing, integration as reusable firmware library.

5.3. Limitations

Dataset collection was performed in Kyiv, Ukraine, potentially introducing geographic bias. Our manufacturer distribution (Apple 24.8%, Google 9.1%, Samsung 6.8%, Microsoft 5.2%) reflects the Eastern

European device market, where Apple and Samsung dominate the premium segment while budget Chinese manufacturers constitute a significant portion of the “unknown” category (18.2%). This regional composition may affect model generalization: markets with different brand dominance (e.g., Xiaomi prevalence in Asia, or higher Apple concentration in North America) could exhibit distinct vulnerability patterns due to manufacturer-specific BLE implementations. The model’s strong performance on Apple Continuity detection (3,220 devices, confidence >0.98) may not transfer equally to regions where alternative ecosystems dominate. We plan multi-city, multi-country collection campaigns to quantify this geographic bias and improve model universality through transfer learning techniques.

The BLE security landscape evolves continuously. Our model captures a late snapshot of threats for 2024, requiring periodic retraining as new vulnerabilities emerge. We are investigating online learning techniques for concept drift adaptation.

Current system identifies known vulnerability patterns from rule-based labeling. Zero-day threats with novel attack vectors may evade detection. To address this limitation, we are developing a complementary anomaly detection module using variational autoencoders (VAE). The proposed architecture compresses the 15-dimensional feature vector through encoder layers ($15 \rightarrow 8 \rightarrow 4$ latent dimensions) and reconstructs via symmetric decoder layers. During training on “normal” traffic (SAFE and LOW_RISK samples), the VAE learns the distribution of benign BLE advertisements. At inference, packets with reconstruction error exceeding two standard deviations from the training mean are flagged as potential zero-day anomalies. Preliminary experiments on synthetic anomalies (injected malformed packets, unusual manufacturer-payload combinations) show 89% detection accuracy with 4.2% false positive rate. The VAE module requires approximately 2.1 KB additional Flash and 8 KB SRAM, remaining within ESP32-S3 constraints. Integration would operate in parallel with the classification model: high-confidence classifications proceed normally, while low-confidence predictions trigger VAE anomaly scoring for secondary verification.

ESP32-S3 supports dual-band Wi-Fi. Future versions could perform joint BLE/Wi-Fi analysis for comprehensive wireless security monitoring with target latency <5 ms using parallel processing on dual cores.

5.4. Implications

The proposed system addresses practical deployment needs across multiple domains. In enterprise environments, continuous BLE auditing at 10 times lower cost than traditional solutions allows one scanner per 500 m² to provide 95% office coverage, with SIEM integration via MQTT/REST API enabling automated incident response workflows. For medical IoT, real-time monitoring of BLE-enabled devices (insulin pumps, pacemakers, patient monitors) is achieved entirely on-device, ensuring HIPAA/GDPR compliance by avoiding cloud transmission of sensitive health data. At the municipal scale, the low cost and energy efficiency of the system enable citywide deployment for detecting rogue beacons and tracking threats in smart city infrastructure. Finally, Android/iOS mobile apps connecting to ESP32 via Bluetooth Serial can democratize security awareness, providing users with real-time alerts about tracking-capable devices in their vicinity.

5.5. Ethical considerations

BLE scanning raises privacy concerns that must be carefully addressed. We implement ethical safeguards: (1) data minimization collecting only technical parameters, not user content, (2) responsible disclosure with 90-day vendor notification windows before public release, (3) dual-use prevention through open-source release promoting defensive applications, (4) regulatory compliance as passive BLE reception is legal in most jurisdictions. We advocate for privacy-by-design in BLE implementations: mandatory MAC randomization, reduced TX power defaults, encrypted advertisement payloads.

6. Conclusions

We presented a TinyML-based system for real-time BLE vulnerability detection on ultra-low-cost microcontrollers (\$10 ESP32-S3). The system achieves 99.68% test accuracy and 97.98% field accuracy with 0.9 ms average inference latency and 4.62 KB model size.

Key contributions include: (1) neural network architecture fitting in 4.62 KB (260 times smaller than traditional ML approaches), (2) comprehensive dataset of 23,481 packets from 144 manufacturers across 13 diverse locations, (3) extensive field validation on 6,804 devices detecting real-world vulnerabilities in Apple Continuity and Google Fast Pair, (4) complete open-source implementation enabling reproducibility and community extension.

Our hybrid approach demonstrates that effective security monitoring does not require expensive hardware or cloud connectivity. The system operates entirely on-device, preserving privacy while providing real-time threat detection for enterprise, medical, smart city, and consumer applications.

Future work includes international dataset expansion, active threat mitigation, multi-protocol support, adversarial robustness testing, and energy-efficient mobile deployment optimization.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

- [1] SNS Insider, Bluetooth Low Energy Market Size, Share & Segmentation, Technical Report SNS/ICT/7016, 2025.
- [2] V. Sokolov, P. Skladannyi, V. Astapenya, Bluetooth low-energy beacon resistance to jamming attack, in: 2023 IEEE 13th International Conference on Electronics and Information Technologies (ELIT), IEEE, 2023, pp. 270–274.
- [3] A. Novytskyi, V. Sokolov, L. Kriuchkova, P. Skladannyi, Determining the error distribution of ble beacons at antenna near and far fields, in: Cyber Hygiene & Conflict Management in Global Information Networks (CH&CMiGIN), volume 4024, 2025, pp. 133–143.
- [4] G. Celosia, M. Cunche, Discontinued privacy: Personal data leaks in apple bluetooth-low-energy continuity protocols, Proceedings on Privacy Enhancing Technologies 2020 (2020) 26–46.
- [5] Flipper Devices, Tech specs of flipper zero, <https://docs.flipper.net/zero/development/hardware/tech-specs>, 2025.
- [6] D. Antonioli, N. O. Tippenhauer, K. Rasmussen, The knob is broken: Exploiting low entropy in the encryption key negotiation of bluetooth br/edr, in: 28th USENIX Security Symposium, 2019, pp. 1047–1061.
- [7] D. Antonioli, N. O. Tippenhauer, K. Rasmussen, Bias: Bluetooth impersonation attacks, in: 2020 IEEE Symposium on Security and Privacy (SP), 2020, pp. 549–562.
- [8] J. S. Al-Azzeh, M. Al Hadidi, R. S. Odarchenko, S. Gnatyuk, Z. Shevchuk, Z. Hu, Analysis of self-similar traffic models in computer networks, International Review on Modelling and Simulations 10 (2017).
- [9] M. Ryan, Bluetooth: With low energy comes low security, in: 7th USENIX Workshop on Offensive Technologies (WOOT 13), 2013.
- [10] M. TajDini, V. Sokolov, P. Skladannyi, Performing sniffing and spoofing attack against ads-b and mode s using software defined radio, in: 2021 IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo), 2021, pp. 1–5.
- [11] O. Solomentsev, M. Zaliskyi, O. Zuiev, Estimation of quality parameters in the radio flight support operational system, Aviation 20(3) (2016) 123–128. doi:10.3846/16487788.2016.1227541.
- [12] V. Larin, et al., Prediction of the final discharge of the uav battery based on fuzzy logic estimation

- of information and influencing parameters, in: 2022 IEEE 3rd KhPI Week on Advanced Technology (KhPIWeek), 2022, pp. 1–6. doi:10.1109/KhPIWeek57572.2022.9916490.
- [13] K. Fawaz, K.-H. Kim, K. G. Shin, Protecting privacy of ble device users, in: 25th USENIX Security Symposium, 2016, pp. 1205–1222.
 - [14] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, P. Burnap, A supervised intrusion detection system for smart home iot devices, IEEE Internet of Things Journal 6 (2019) 9042–9053.
 - [15] S. Shende, S. Thorat, Long short-term memory (lstm) deep learning method for intrusion detection in network security, International Journal of Engineering Research & Technology (IJERT) 9 (2020).
 - [16] P. Warden, D. Situnayake, TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers, O'Reilly Media, Sebastopol, CA, 2019.
 - [17] C. R. Banbury, et al., Benchmarking tinyml systems: Challenges and direction, arXiv preprint arXiv:2003.04821 (2021).
 - [18] R. David, et al., Tensorflow lite micro: Embedded machine learning for tinyml systems, in: Proceedings of Machine Learning and Systems (MLSys), volume 3, 2021, pp. 800–811.
 - [19] N. N. Alajlan, D. M. Ibrahim, Tinyml: Enabling of inference deep learning models on ultra-low-power iot edge devices for ai applications, Micromachines 13 (2022) 851.
 - [20] E. Abdullaiev, Ble vulnerability detection system using tinyml, <https://github.com/elmir132/ble-vulnerability-tinyml>, 2025. GitHub repository.
 - [21] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations (ICLR), 2015.