JamSense: Interference and Jamming Classification for Low-power Wireless Networks

John Kanwar[1], Niclas Finne[1], Nicolas Tsiftes[1], Joakim Eriksson[1], Thiemo Voigt[1,2]
Zhitao He[3], Christer Åhlund[4], Saguna Saguna[4]
[1] Research Institutes of Sweden (RISE), Sweden
[2] Uppsala University, Sweden
[3] ASSA ABLOY, Sweden
[4] Luleå University of Technology, Sweden

***Abstract*—Low-power wireless networks transmit at low output power and are hence susceptible to cross-technology interference. The latter may cause packet loss which may waste scarce energy resources by requiring the retransmission of packets. Jamming attacks are even more harmful than cross-technology interference in that they may totally prevent packet reception and hence disturb or even disrupt applications. Therefore, it is important to recognize such jamming attacks. In this paper, we present JamSense. JamSense extends SpeckSense, a system that is able to detect multiple sources of interference, with the ability to classify jamming attacks. As SpeckSense, JamSense runs on resource-constrained nodes. Our experimental evaluation on real hardware shows that JamSense is able to identify jamming attacks with high accuracy while not classifying Bluetooth or WiFi interference as jamming attacks.**

I. Introduction

The Internet of Things (IoT) is expected to enable applications of utmost societal value, such as energy-efficient buildings, smart cities, intelligent grids, and next-generation healthcare. To fulfil this promise, IoT networks need to be able to support applications that require sensor data and actuation commands to be delivered in a timely fashion with high reliability. Wireless networks are, however, exposed to cross-technology interference and different jamming attacks. This is a particular challenge for IoT networks that typically operate wirelessly on low power. Hence, they only can transmit with low output power, which makes them susceptible to such interference and attacks.

While there is a whole body of work that has been dedicated to detecting interference, the detection of different jamming attacks has received much less attention. Both heavy interference and jamming attacks lead to high packet loss and hence degrade the performance of the application if not being dealt with. The difference between the two is that cross-technology interference stems from other operational networks while jamming is usually caused by an attacker that deliberately aims to disrupt or disturb an application.

ISBN 978-3-903176-42-3

One of the challenges is that the battery-powered IoT devices must often last for years. In addition to operating with low transmit output power, they cannot afford power-hungry algorithms for interference and jamming detection. Therefore, in this paper, we opt for a tool that can detect and classify both heavy interference and different jamming attacks using simple algorithms that can be deployed on resource-constrained IoT devices.

Towards this end, we extend SpeckSense [8] with the ability to differentiate between different jamming attacks such as proactive and reactive jamming [20]. SpeckSense features an interference detector that distinguishes between WiFi and Bluetooth interference using an unsupervised learning technique. In addition, it distinguishes between moderate and heavy traffic and identifies WiFi beacons.

As SpeckSense and other approaches to interference classification [3], [22], JamSense is based on fast RSSI sampling. After RSSI sampling, JamSense detects jamming using clustering techniques. Finally, the created clusters are examined to identify the type of jamming in particular based on the jamming's temporal properties.

We implement JamSense on Nordic's nRF52840 platform, a resource-constrained IoT platform with an IEEE 802.15.4 radio [15]. We perform experiments in a small-scale testbed where we expose JamSense to both jamming attacks and external interference. Our results show that we can identify jamming attacks with high accuracy. In addition, we show that we do not misclassify external interference from Bluetooth and WiFi as jamming

attacks. We also show that when JamSense runs in an environment where we do not expect any attacks, it only rarely identifies attacks keeping the number of false positives very low.

Contributions. We make the following contributions in this paper:

- We provide mechanisms to detect and differentiate the most common jamming attacks in low-power wireless networks.
- We integrate these mechanisms into an existing tool, SpeckSense, for interference detection that runs on resource-constrained IoT nodes.
- We demonstrate the capabilities of JamSense via experiments on real hardware. In particular, we show that JamSense identifies jamming attacks with high accuracy while not classifying Bluetooth or WiFi interference as jamming attacks.

II. Jamming Attacks on Low-power Wireless Networks

Jamming attacks can disturb or disrupt communication between nodes by preventing the sender from transmitting packets, or by preventing the reception of packets [20]. Although there exists a large number of radio jamming attacks, the most common jammers are the *constant*, *deceptive*, *random*, and *reactive* jammers [4], [11], [12], [16], [17], [20]. All of these jammers have shown to be effective in that they prevent transmission or reception of most, if not all packets [20]. These jammers are based on the same low power radios used by IoT devices. They are attractive to hackers because of their affordability and power efficiency, in comparison to bulky, high-power radio transmitters.

A. Proactive jammers

A proactive jammer sends out interference signals regardless of the data communication that happens in the network. It transmits random or specific chosen bits on a chosen channel to occupy it, thereby preventing other nodes from transmitting or receiving on this channel. There are three main types of proactive jammers.

1) Constant jammer: A constant jammer sends out a continuous radio signal in the wireless medium. It transmits random bits on a chosen channel and does not obey the rules of the MAC protocol in operation [19].

2) Deceptive jammer: A deceptive jammer is similar to a constant jammer in that it sends out a continuous flow of radio signals. In contrast to a constant jammer, it transmits packets instead of random bits. Its goal is to trap the victim into a receiving mode where it is unable to do anything else.

3) Random jammer: A random jammer alternates between sleeping and transmits states. It emits radio signals for a certain time t_j, and then sleeps for a certain time t_s. By doing so, it saves energy and becomes harder to detect than a constant jammer. The variables t_j and t_s can be fixed random values. The jamming itself can be conducted using either bits or packets [19].

B. Reactive jammer

A reactive jammer is a jammer that is reactive in that it reacts to its environment before it takes action. This type of jammer starts transmitting radio signals only when it senses activity on a channel that the victims are using. This way, the jammer targets the reception of packets, which makes it harder to detect [9].

A common implementation of a reactive jammer is the *start-of-frame delimiter jammer* [5] that we call SFD-jammer. The beginning of an 802.15.4 frame consists of a preamble and a start-of-frame delimiter (SFD). The purpose of the preamble and SFD is to alert a receiver that there is a packet to be received and to keep synchronization between devices that are communicating. When the SFD-jammer notices a preamble and an SFD on the channel, it quickly turns from reception to transmission mode to transmit an interference packet in order to corrupt the payload of the packet that is being transmitted to the receiving node. This results in the packet getting dropped by the receiving node. By doing so, the SFD-jammer not only keeps itself more concealed to the environment by only attacking packets that are being transmitted but it also saves energy by not sending jamming signals on random noise.

III. Design and Implementation

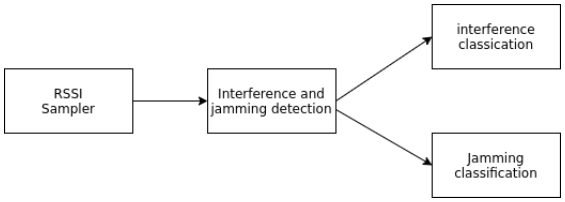

Fig. 1: High-level architecture of JamSense.

The high-level architecture of JamSense follows the one of SpeckSense [8]. It consists of several modules shown in Figure 1. First, JamSense samples the RSSI values. After the RSSI sampling, JamSense detects interference and jamming using clustering methods. Finally, JamSense classifies both interference and jamming.

A. RSSI Sampler

Like many other approaches for interference detection, including SpeckSense [8], we employ an RSSI sampler. The RSSI sampler captures the energy in the channel, which will typically increase during a period of interference. The RSSI-sampler reads RSSI values at a frequency of 25.6 kHz. As in SpeckSense, we quantize the RSSI to avoid accounting for smaller variations in the RSSI, and to save memory. In order to further save memory space, we run-length encode the quantized RSSI readings. The RSSI sampler stores a power level and its duration into a 2D vector. The power level depends on the strength of the RSSI. We define 20 different power levels. For example, the RSSI sequence: -93, -92, -56, -57, -55, -29, -28, -70, -70, -70, -94. would lead to the following sequence of 2D vectors: (1, 2), (10, 3), (17, 2), (7, 3), (1, 1) since, e.g., -93 and -92 are mapped to the same power level (power level 1).

To avoid saturation, e.g., in the case of a constant jammer getting stuck sampling, we set a maximum number for the duration with an empirically chosen value. After having produced a certain amount of 2D vectors, these are sent to the detection module.

B. Jamming detection

In order to identify the different jamming attacks, we use a K-means clustering algorithm. The K-means clustering algorithm takes the 2D vectors created by the RSSI-sampler and randomly initializes centroids from the 2D vectors. Each 2D vector will be assigned to the nearest cluster centroid, which results in k clusters. The K-means clustering algorithm will recalculate the positions of the centroids until a stop condition is met.

For the assignment of 2D vectors to clusters, the algorithm uses the Euclidean distance. The cluster centres are updated by recalculating the average of the samples. The Euclidean distance algorithm is weighted higher towards the power level in order to emphasize and give more weight to 2D vectors that have a short duration and high power level.

Cluster assignment and the update of the centroids are repeated until a termination condition is met. The cluster assignment terminates when distance is below a threshold of 0.001 which we empirically determined by observing when the algorithm does not further increase the accuracy of the clusters. The other termination condition is that the number of clusters exceed the maximum k. The k in this algorithm is not known in advance as there can be a different number of interference sources. k is therefore calculated in real-time and is tested with different values of k. k starts at a value of 1 and is increased by one as long as the cost of difference of the algorithm is less than the termination threshold (0.001).

C. Jamming Classification

To classify the different jamming attacks, we execute a *cluster sampler* process. After the K-means clustering algorithm has been executed and has created several clusters, we sort the clusters after duration and compare with the jamming attacks' pre-determined thresholds, which are duration and power level. If a cluster's attributes are inside the pre-determined threshold, we save that cluster as a cluster sample.

When five cluster samples have been collected within a certain time duration, their average and each cluster's difference to the average is calculated. If the difference is smaller than a certain threshold, meaning that the sample values are close enough to each other, JamSense concludes that the cluster sample is not a random attack but either a constant or deceptive jamming attack as these attacks are consistent. If the difference is larger than the threshold, JamSense classifies the attack as a random jammer, as random jammers are inconsistent. If five samples are not collected during a certain time, and the characteristics of an SFD jamming attack has been detected once, it is most likely a reactive SFD jamming attack.

Figure 2 and 3 visualize how JamSense classifies the constant, deceptive and SFD-jammer. The power level 11 is selected as the minimum power level where interference prevents the reception of a packet. The constant and deceptive jamming attacks have the same characteristics regarding power level and duration. They differ in how they are transmitting bits: Constant jammers send random bits while deceptive jammers transmit packets containing a preamble and an SFD. In order to differentiate between these two jamming attacks, we check whether we receive a preamble and an SFD or not. We determine the thresholds experimentally. They have been shown to work well in our experiments. Future work could use more advanced machine-learning methods to define the thresholds.

D. Interference detection

The detection of unintentional interference closely follows SpeckSense [8]. If there are no alerts for a jamming attack, JamSense runs its interference detection to detect WiFi or Bluetooth interference. The RSSI values, which have been quantized and run-length encoded, are extracted into bursts. A burst is a consecutive sequence

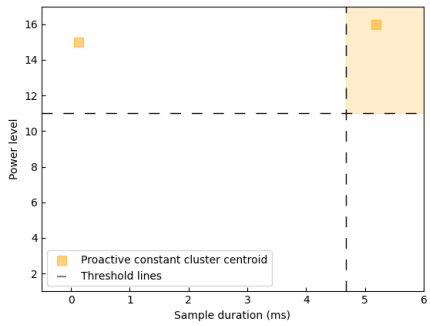

Fig. 2: JamSense clustering a constant and deceptive jamming attack with the help of the K-means algorithm. Constant and deceptive jamming attacks show a long duration because of the continuous radio signal they create.

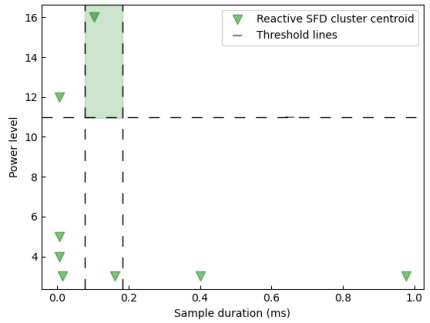

Fig. 3: JamSense clustering a reactive jamming attack with the help of its K-means algorithm. The duration of the SFD-jamming attack is very short (note the different x-axis compared to the previous figure) because it is only active for a short time when it senses activity on a channel.

where the channel is active, i.e., the power level is greater than one. A burst is created when the channel stops being idle by taking the weighted mean power level and the total duration of the sequence. For example, from the 2d-vectors (3,3) (4,1) (2,1) we compute the following burst: $(\frac{3\times3+4\times1+2\times1)}{(3+1+1)}, (3+1+1)) = (3, 5)$. The bursts are then handed over to the interference classification module.

E. Interference classification

For classifying the WiFi and Bluetooth interference, we use the same K-means cluster algorithm for identifying the jamming attacks. There are, however, a few differences: the Euclidean distance algorithm is not weighted as we are not focusing on emphasizing short durations and high power levels since these are relevant only for jamming attacks. A non-weighted Euclidean distance algorithm yields better results for identifying WiFi and Bluetooth.

Furthermore, the K-means clustering algorithm clusters a group of bursts. The interference classification is done by comparing the average burst size and burst power level to pre-determined thresholds that have been profiled earlier. SpeckSense also identifies different types of channel activity by calculating and comparing inter-burst separation and average clusters created by the interference. We do not include this feature as our aim is to identify jamming attacks and heavy interference—not whether it is WiFi from web browsing or WiFi from video streaming.

IV. Experimental Evaluation

In this section, we evaluate several important properties of JamSense. Towards this end, we perform experiments on real hardware in a small-scale testbed. We implement JamSense on Nordic's nRF52840 platform, a resource-constrained IoT platform that features an IEEE 802.15.4 radio. This low-power radio is also used by other studies dealing with interference and jamming [3], [8], [22]. JamSense is implemented using the Contiki-NG operating system [2]. Our unoptimized implementation fits within 74 KB of program memory. The overall RAM usage is contained within 20 KB out of the 256 KB that are available on the platform.

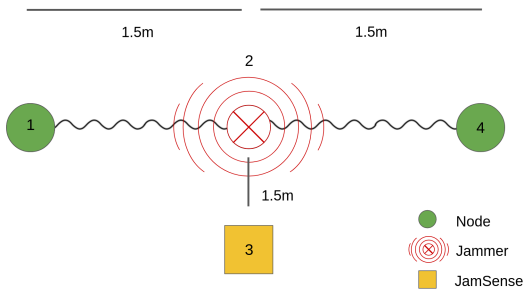

Fig. 4: Setup of the classification experiment. Two nodes communicate with UDP while the jammer executes the different jamming attacks.

A. Classification of Jamming attacks

Motivation. This experiment is carried in order to test the capability of JamSense to correctly identify the different jamming attacks.

Setup. The setup consists of four different nodes as shown in Figure 4. Node 1 and Node 4 communicate with each other by sending UDP packets. Node 2 performs the different jamming attacks while Node 3 runs JamSense to identify the attacks. We perform each jamming attack 1000 times.

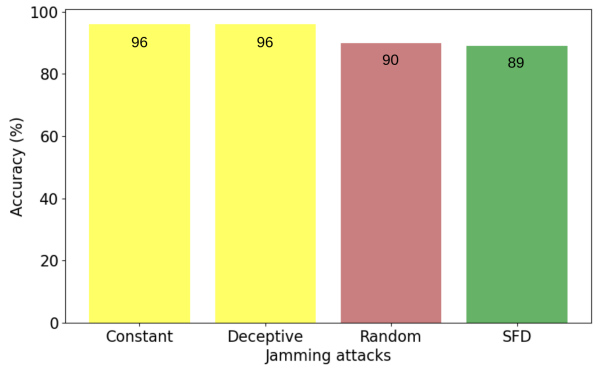

Fig. 5: Accuracy of JamSense for detecting and identifying the different jamming attacks.

Result. Figure 5 shows the accuracy with which JamSense identifies the different attacks. The figure shows that JamSense is able to identify all attacks with high accuracy. In particular the identification of the constant and deceptive jamming attacks has very high accuracy. The reason for that is that their characteristics differ substantially from the other jamming attacks in that they have a longer and consistent duration compared to the SFD and random jammer. The SFD-jammer is harder to detect because it only jams for a very short duration. Hence, JamSense is sometimes not able to detect it. While the other jamming attacks have a consistent duration, the proactive random jammer does not. However, in the worst case, since the jamming duration is random, it may have the same features as the constant jammer or the SFD-jammer. In that case, JamSense is not able to identify this jammer correctly.

B. Identifying Bluetooth Interference

Fig. 6: Setup for JamSense identifying Bluetooth interference created by wireless headphones

Motivation. We build on SpeckSense [8] that is able to detect Bluetooth interference. The purpose of this experiment is to show that JamSense does not confuse Bluetooth interference with any jamming attacks.

Setup. Figure 6 shows the setup of the experiment. In this experiment, a node running JamSense is placed 1 meter apart from a Bluetooth wireless headset. The wireless headset is turned on and Bluetooth interference is created. We evaluate if JamSense correctly identifies Bluetooth interference by running it 4000 times.

Results. Figure 8 shows that JamSense is able to identify the Bluetooth interference in 87% of the cases. In the other cases, JamSense is just not able to identify the Bluetooth interference but it does not misclassify it as a jamming attack. This is important to avoid false alarms. We receive the same result when we position the Bluetooth interferer 30 centimetres or 2 meters away from JamSense.

C. Identifying WiFi Interference

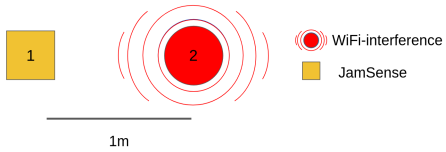

Fig. 7: JamSense identifying deterministic WiFi interference created by Jamlab-NG.

Motivation. The goal of this experiment is similar to the previous one. In this experiment, however, we investigate whether JamSense identifies WiFi interference without confusing it with a jamming attack.

Setup. As shown in Figure 7, one node running JamSense is placed 1 meter from a Raspberry Pi 3 Model B that runs JamLab-NG [13]. JamLab-NG is a widely used tool that creates deterministic WiFi interference. For JamLab-NG we use the default values for the periodic interval that is 13 ms and the frame length of 600 bytes. We use the standard settings of 802.11g and a bit rate of 1 Mbps.

Result. The results in Figure 9 show a successful classification rate of 86 %. In the remaining cases, JamSense does not classify the WiFi interference correctly. This happens since JamSense does not detect the WiFi interference and not because it misclassifies the interference as a jamming attack.

D. Identifying Simultaneous Interference Sources

Motivation. JamSense extends SpeckSense by identifying jamming attacks in addition to identifying Bluetooth and WiFi interference. This experiment evaluates if JamSense is able to identify and distinguish between multiple interference sources simultaneously as SpeckSense does while making sure that WiFi and Bluetooth interference do not get classified as jamming attacks.

Setup. Figure 8 shows the setup of the experiment. We place a Bluetooth wireless headset and a node running

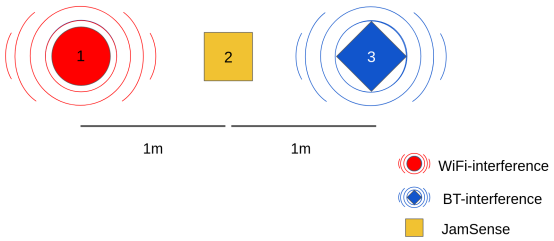

Fig. 8: JamSense identifying Bluetooth and WiFi interference simultaneously.

JamLab-NG 1 meter from JamSense [14]. The JamLab-NG settings are set to the same settings as in the previous experiment: a Tx power of 15 mW, a periodic interval of 13 ms, and a frame length of 600 bytes.

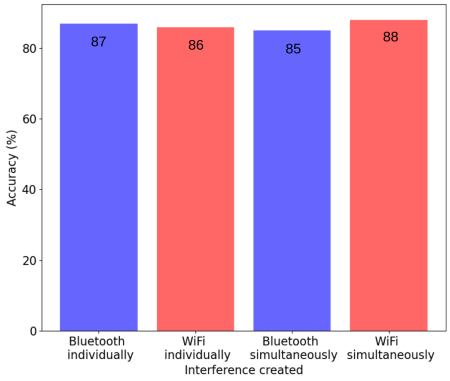

Fig. 9: Accuracy of JamSense identifying Bluetooth and WiFi interference individually and simultaneously.

Result. Figure 9 shows a successful classification rate of 85 % in detecting Bluetooth and an 88 % success rate in identifying WiFi, similar to the individual experiments described above. When JamSense is unsuccessful in identifying the interference, it is because of failure to detect WiFi and Bluetooth interference. This means that also when several interferers are active at the same time, JamSense does not confuse them with a jamming attack.

E. Exploring False Positives

Fig. 10: JamSense listening to the environment when no jamming attacks are active in order to evaluate if JamSense returns false positives.

Motivation. We perform this experiment to evaluate if JamSense returns false positives when there are no jamming attacks. From an applicant point of view, it is important to avoid false positives in order to prevent unnecessary alarms.

Setup. JamSense runs in an apartment building consisting of four stories with eight households for 10 hours. We do not expect any jamming attacks. There are of course other private WiFi access points that create WiFi traffic and there might be Bluetooth traffic from devices we are not aware of.

Result. During the 10 hours of operating, JamSense returned one false positive while it was active 27740 times. The false positive was the identification of an SFD-jammer. We have not fine-tuned JamSense to avoid such false positives. We believe that with some more tuning or using more advanced methods to set thresholds, we could avoid such false positives. We also note that this is a false positive from a single resource-constrained node. An alarm system would likely be based on input from several nodes over a longer duration than one single identification of an attack.

V. Related Work

In this paper, we extend SpeckSense [8], a tool for interference detection for low-power wireless networks to also classify different jamming attacks. Like many other approaches, SpeckSense is based on active sampling of RSSI values.

Zacharias et al. aim at identifying the main interference source (rather than multiple interferers as SpeckSense) using fast RSSI sampling [22]. Based on a fixed RSSI threshold, they extract temporal features such as channel idle and busy times that they use to identify the interferer.

Grimaldi et al. propose an SVM-based approach to classify between 802.11 and microwave interference as well as channels without interference [3]. Also, their approach is based on RSSI sampling, albeit with a lower sampling rate. Yi et al. use deep learning to differentiate between different interferers [21]. Also, their method is based on raw RSSI samples that constitute the input to their deep neural network. Their approach uses a more powerful Raspberry Pi rather than a resource-constrained platform.

In contrast to active RSSI sampling, SoNIC samples the RSSI only during the reception of a packet [6]. The authors perform this sampling to determine the symbols of the received payload that suffered transmission errors. They identify the source of interference based on those corrupted symbols. Hithnawi et al. [7] do not only detect but also mitigate interference using a larger number of

features that include corrupted symbols. Similar, Barac et al. also look at symbol error density to infer the wireless link conditions [1]. Towards this end, they make use of additional symbols for forward error correction.

In addition to providing interference detection, JamSense also differentiates between different jamming attacks. Xu et al. propose to identify jammers based on signal strength, carrier sensing time and packet delivery ratio [20]. They also propose strategies to defend against jammers but do not evaluate their jammer identification strategies. Kasturi et al. aim at using machine learning techniques for classifying three types of jammers, namely constant, reactive and random jammers [10]. They train their classifiers with data collected in the ns-3 simulator and use the same simulator and data for the evaluation of their algorithms. While we also opt for the classification of jamming attacks, we implement our algorithms on real, resource-constrained hardware. Also, Wang and Wyglinski try to identify jammers [18] by combining statistical data of packets send and delivery ratio. As Kasturi et al. they evaluate their work in simulation only.

VI. Conclusions

Low-power wireless networks are exposed to interference and jamming attacks which lead to packet loss, higher energy consumption and even disrupted applications. In this paper, we have presented JamSense, a novel tool that is able to detect and identify the four most popular jamming attacks with high accuracy. JamSense is implemented on resource-constrained devices. Our evaluation on a small-scale testbed has shown that JamSense identifies constant and deceptive jammers with an accuracy above 95% and deceptive and SFD-jammers with an accuracy of around 90%. In addition, JamSense maintains the capability of classifying cross-technology interference from WiFi and Bluetooth that it inherits from SpeckSense with an accuracy above 85% without classifying such interference as jamming attacks.

Acknowledgements

This work was supported by the ITEA 3 project STACK funded by the Swedish Innovation Agency VINNOVA.

References

[1] Filip Barac, Stefano Caiola, Mikael Gidlund, Emiliano Sisinni, and Tingting Zhang. Channel diagnostics for wireless sensor networks in harsh industrial environments. *IEEE Sensors Journal*, 14(11):3983–3995, 2014.

[2] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *29th annual IEEE international conference on local computer networks*, pages 455–462. IEEE, 2004.

[3] Simone Grimaldi, Aamir Mahmood, and Mikael Gidlund. An svm-based method for classification of external interference in industrial wireless sensor and actuator networks. *Journal of Sensor and Actuator Networks*, 6(2):9, 2017.

[4] T. Hamza, G. Kaddoum, A. Meddeb, and G. Matar. A survey on intelligent mac layer jamming attacks and countermeasures in wsns. In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2016.

[5] Zhitao He and Thiemo Voigt. Precise packet loss pattern generation by intentional interference. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–6. IEEE, 2011.

[6] Frederik Hermans, Olof Rensfelt, Thiemo Voigt, Edith Ngai, Lars-Åke Nordén, and Per Gunningberg. Sonic: Classifying interference in 802.15. 4 sensor networks. In *Proceedings of the 12th international conference on Information processing in sensor networks*, pages 55–66, 2013.

[7] Anwar Hithnawi, Hossein Shafagh, and Simon Duquennoy. Tiim: Technology-independent interference mitigation for low-power wireless networks. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, pages 1–12, 2015.

[8] Venkatraman Iyer, Frederik Hermans, and Thiemo Voigt. Detecting and avoiding multiple sources of interference in the 2.4 ghz spectrum. In *European Conference on Wireless Sensor Networks*, pages 35–51. Springer, 2015.

[9] S. Jaitly, H. Malhotra, and B. Bhushan. Security vulnerabilities and countermeasures against jamming attacks in wireless sensor networks: A survey. In *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, pages 559–564, 2017.

[10] GS Kasturi, Ansh Jain, and Jagdeep Singh. Detection and classification of radio frequency jamming attacks using machine learning. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, 11(4):49–62, 2020.

[11] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou. A survey on jamming attacks and countermeasures in wsns. *IEEE Communications Surveys Tutorials*, 11(4):42–56, 2009.

[12] Opeyemi Osanaiye, Attahiru S Alfa, and Gerhard P Hancke. A statistical approach to detect jamming attacks in wireless sensor networks. *Sensors*, 18(6):1691, 2018.

[13] Markus Schuß, Carlo Alberto Boano, Manuel Weber, Matthias Schulz, Matthias Hollick, and Kay Römer. Jamlab-ng: Benchmarking low-power wireless protocols under controllable and repeatable wi-fi interference. In *EWSN*, pages 83–94, 2019.

[14] Markus Schuß, Carlo Alberto Boano, Manuel Weber, Matthias Schulz, Matthias Hollick, and Kay Römer. Jamlab-ng: Benchmarking low-power wireless protocols under controllable and repeatable wi-fi interference. In *EWSN*, pages 83–94, 2019.

[15] Nordic Semiconductor. About Nordic Semiconductor. https://www.nordicsemi.com/About-us/, (visited on 2021-06-02).

[16] Radosveta Sokullu, Ilker Korkmaz, Orhan Dagdeviren, Anelia Mitseva, and Neeli R Prasad. An investigation on ieee 802.15. 4 mac layer attacks. In *Proc. of WPMC*, volume 41, pages 42–92, 2007.

[17] Satish Vadlamani, Burak Eksioglu, Hugh Medal, and Apurba Nandi. Jamming attacks on wireless networks: A taxonomic

survey. *International Journal of Production Economics*, 172:76 – 94, 2016.

[18] Le Wang and Alexander M Wyglinski. A combined approach for distinguishing different types of jamming attacks against wireless networks. In *Proceedings of 2011 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 809–814. IEEE, 2011.

[19] X. Wei, Q. Wang, T. Wang, and J. Fan. Jammer localization in multi-hop wireless network: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 19(2):765–799, 2017.

[20] Wenyuan Xu, Ke Ma, Wade Trappe, and Yanyong Zhang. Jamming sensor networks: attack and defense strategies. *IEEE network*, 20(3):41–47, 2006.

[21] Su Yi, Hao Wang, Wenqian Xue, Xiaojing Fan, Lefei Wang, Jun Tian, and Ryuichi Matsukura. Interference source identification for ieee 802.15. 4 wireless sensor networks using deep learning. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–7. IEEE, 2018.

[22] Sven Zacharias, Thomas Newe, Sinead O'Keeffe, and Elfed Lewis. A lightweight classification algorithm for external sources of interference in ieee 802.15. 4-based wireless sensor networks operating at the 2.4 ghz. *International Journal of Distributed Sensor Networks*, 10(9):265286, 2014.