A software-defined-radio replay attack on Bluetooth Low Energy devices*

Oleksandr Romaniuk[1,†], Volodymyr Sokolov[1,*,†], Andrii Anosov[1,†], Hennadii Hulak[1,2,3,†], and Nataliia Korshun[1,†]

[1] *Borys Grinchenko Kyiv Metropolitan University, 18/2 Bulvarno-Kudriavska str., 04053 Kyiv, Ukraine*

[2] *Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine, 42 Ac. Glushkov ave., 03680 Kyiv, Ukraine*

[3] *National Academy of the Security Service of Ukraine, 22 Mykhaila Maksymovycha str., 03022 Kyiv, Ukraine*

Abstract

The growth of wireless communication and the expansion of Bluetooth Low Energy devices increase the attack surface. Then, any attack, in particular replay attack, becomes a big problem, because successful implementation of this attack can lead to the compromise of data, its modification, and the transfer of already illegitimate data. Since software-defined radio technology is very accessible today, the process of data compromise is made easier for attackers. The article discusses the process of implementing the replay attack and provides recommendations for its prevention in wireless sensor networks. The main recommendation is data encryption, which makes the data unintelligible to all except the master BLE device and slave, which forms the connection. Also, to effectively counter this type of attack, it is necessary to primarily adhere to Bluetooth Core Specification version 4.2 and above, which will provide a significant level of security against both replay attacks and passive eavesdropping.

Keywords

SDR, GNU Radio, HackRF One, GFSK, replay attack, man-in-the-middle attack, wireless sensor network.

1. Introduction

In the realm of wireless communication, the ubiquity of Bluetooth Low Energy (BLE) devices has ushered in an era of seamless connectivity and efficient data exchange.

In wireless sensor networks, with the growing trend of cyberattacks, cybersecurity is becoming an important aspect on which the reliability of infrastructure and the preservation of confidentiality and integrity of information are based. In this context, a replay attack becomes a serious threat, as its successful implementation can lead to critical consequences, including data interception and modification, respectively, violation of information confidentiality, as well as interference with the normal operation of the system. Leveraging the capabilities of Software Defined Radio (SDR) this type of attack becomes even easier for hackers [1, 2].

BLE technology has found wide application in industrial devices and devices of everyday human life like smartwatches and bracelets, wireless headphones, gaming devices, consumer electronics, etc. [3]. This requires a comprehensive and sophisticated approach to defining and implementing methods to prevent replay attacks [4, 5].

BLE devices can most often be classified as Internet of Things devices. So in 2021, there were 1.5 billion cyberattacks on IoT devices [6]. Annual Bluetooth device shipments worldwide stood at 4.9 billion units in 2022. Yearly shipments are forecast to reach 7.6 billion units in 2027, with a projected compound annual growth rate of nine percent from 2023 to 2027 [7]. This means that more and more devices will need to be protected.

* *CPITS 2026: Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2026, Kyiv, Ukraine*

* Corresponding author.

† These authors contributed equally.

ombabych.fitu18@kubg.edu.ua (O. Romaniuk); v.sokolov@kubg.edu.ua (V. Sokolov); a.anosov@kubg.edu.ua (A. Anosov); h.hulak@kubg.edu.ua (H. Hulak); n.korshun@kubg.edu.ua (N. Korshun)

0000-0002-1214-9406 (O. Romaniuk); 0000-0002-9349-7946 (V. Sokolov); 0000-0002-2973-6033 (A. Anosov); 0000-0001-9131-9233 (H. Hulak); 0000-0003-2908-970X (N. Korshun)

2. Source Review

Papers [8] and [9] describe a similar man-in-the-middle attack on Bluetooth devices. Work [10] discusses the mitigation of this type of attack in the Internet of Medical Things. Article [11] presents an overview of security mechanisms for the Bluetooth mesh network. Also, the development of a new security framework for BLE devices was reviewed [12].

3. Selection of SDR for Experiments

Today, there are a large number of SDR models that are used in various fields such as military, mobile, satellite communications, etc. A comparison of the above-mentioned SDR models is shown in the Table 1.

The HackRF One is a half-duplex 8-bit SDR that covers the frequency range from 1 to 6,000 MHz, with a maximum bandwidth of 20 MHz.

Table 1.
Comparison of SDR Devices

Model	Chipset	Freq. range, MHz	Freq. band, MHz	ADC bit rate, bit	Price, $
HackRF One	MAX5864, MAX2837, RFFC5072	1-6000	20.0	8	300
SDRplay RSPduo	Mirics MSi2500	0-2000	8.0	14	300
Airspy R2 SDR	Rafael Micro R820T2	24-1700	10.0	12	220
Airspy Mini SDR	Rafael Micro R820T2	24-1700	6.0	12	120
LimeSDR	LMS7002M	0.1-3800	61.4	12	800
LimeSDR Mini	LMS7002M	10-3500	40.0	12	300
bladeRF x40	LMS6002D	300-3800	40.0	12	520
bladeRF 2.0 micro xA4	Analog Devices AD9361	47-6000	56.0	12	540
ETTUS USRP B200	Analog Devices AD9364	70-6000	61.4	12	1290
ADALM-Pluto SDR	Analog Devices AD9363	325-3800	20.0	12	230
RTL-SDR Blog V3	RTL2832U	0.5-1766	8.0	8	30

Also, HackRF One is an open-source device and compatible with various software such as GNU Radio, SDR#, etc., which makes it popular among radio amateurs [13]. Accordingly, the HackRF One SDR device was chosen for further research and experimentation, because it is a very popular model among radio amateurs, it is in the middle price range, and it has a wide frequency range compared to competitors, which makes it versatile for the implementation of various scenarios. HackRF One has nine revisions for 2023 [14], their description is shown in Table 2.

HackRF One is an SDR SoC that is an open-source project, so its source code is publicly available. With this device, you can research radio signals, carry out a thorough analysis of the radio frequency spectrum, and create your programs.

Table 2.
HackRF One SDR Revisions

Rev.	Description	Years
r1-r4	Great Scott Gadgets released the first revision of HackRF One of 2014 called r1. In subsequent production cycles up to and including r4 revision, there were no changes in the hardware design.	2014-20
r5	An experimental version that has not been commercially produced for sale.	2014-20
r6	SKY13350 RF switches have been replaced with SKY13453. Although the SKY13453 uses simplified control logic, it does not require any firmware modification.	2020
r7	SKY13453 radio frequency switches are replaced with SKY13350. Updated USB VBUS detection resistor values.	2021
r8	SKY13350 radio switches have been replaced with SKY13453.	2021-22
r9	MAX2837 is replaced by MAX2839. Si5351C is replaced by Si5351A with additional clock frequency distribution. Added a series diode to the antenna port power supply.	2023

4. Comparison of Technologies

A comparison of the technical characteristics of Bluetooth and BLE [15] and ZigBee [16] technologies is presented in Table 3 [17].

Table 3.
Comparison of Bluetooth, BLE, and ZigBee

Parameter	Bluetooth	BLE	ZigBee
Frequency range, MHz	2400	2400	868, 915, 2400
Bit rate, Kbit/s	700-2100	1000	20, 40, 250
Signal modulation type	GFSK, π/4 DQPSK, 8DPSK	GFSK	BPSK, ASK, O-QPSK
The method of spectrum expansion	FHSS (1 MHz)	FHSS (2 MHz)	DSSS
Receiver sensitivity, dBm	-70	-70	-92 (868/915 MHz)
Transmitter output power, dBm	20, 4, 0 (class 1, 2, 3)	-20..10	-85 (2400 MHz)
Maximum data transfer rate in the channel, Mbit/s	1	1	-32..0
Packet data size, bytes	< 358	8-47	0.02-0.25

As for security, all three technologies use the symmetric 128-bit Advanced Encryption Standard encryption algorithm. Considering the features of both technologies, and their not significant difference in characteristics, BLE was chosen for the further experiment, based on its popularity, that is, wider use, and better compatibility of this technology with devices of different manufacturers, compared to ZigBee.

5. GFSK Simulation with IEEE 802.15.1 Hardware

During the experimental part, modulation and demodulation will be implemented in two stages. This is necessary to first intercept the stream of BLE packets, demodulate it, and then send the already edited packets (the man-in-the-middle or replay attack) to the recipient, modulating this data [8-10]. The HackRF One SDR device acts as a receiver and transmitter. Data to be intercepted, namely iBeacon and AltBeacon, will be generated by various mobile applications. Identification of

the successful return of bacon will also be carried out by the aforementioned applications (Figure 1).

Let's analyze this circuit of the block diagram in order. In the upper part of the image we see many Variable blocks, these are the variables used. The freq variable means the frequency at which all manipulations are performed, it is 2.426 GHz, so channel 38 (ble_channel variable), the announcement channel, is actively listened to by all devices, and that is why it was chosen. The ble_channel_spacing variable has a value of 2 MHz, this is the bandwidth, and it is the same for all BLE technology channels.

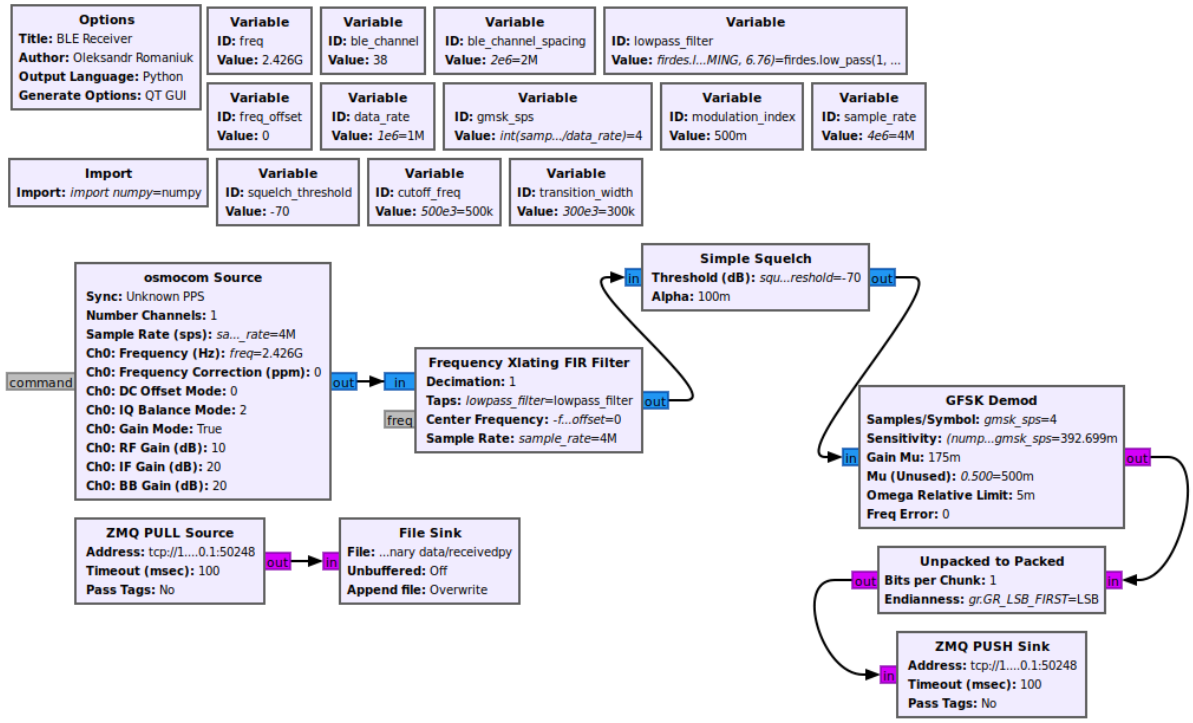


Figure 1: Block diagram of interception of BLE packets.

The lowpass_filter variable is used in the Frequency Xlating FIR Filter block and consists of the formula described in the following link [18]. The freq_offset value is zero, that is, there is no frequency shift from the center of the channel; data_rate is the maximum data transfer rate in the channel. The variable gmsk_sps is used in the Gaussian Frequency Shift Keying (GFSK) Demod block and is calculated as a fraction of the sampling rate (sample_rate) and the data rate in the channel. The value of modulation_index for BLE packets can be from 0.45 to 0.55, but 0.5 was chosen as the average value — this is the modulation index of the signal. The squelch_threshold variable is minus 70 dBm, and means the signal muting threshold; cutoff_freq and transition_width are the signal cutoff and transition width values respectively, both variables used in the Frequency Xlating FIR Filter. The Import block with the value "import numpy" is required to import the library and then use it to calculate the sensitivity in the simulation.

Osmocom Source is the unit responsible for the signal source, it receives the digital data forming the signal from the physical receiver and transmits it to the GNU Radio environment; has only the output connector, to the right of the block. Frequency Xlating FIR Filter performs signal frequency transformation and at the same time reduces signal sampling, it is necessary to highlight the narrowband part of a wideband signal without the need to center this narrowband part by frequency. This is appropriate in this case, because the HackRF One captures a wide frequency band with a very high sampling rate, but the desired signal occupies only a narrow part of the frequency band [19]. Simple Squelch is a squelch block based on average signal power and threshold in dBm. It works in such a way that the output signal is equal to the input signal if the average input signal is greater than or equal to the threshold, or zero otherwise. The mean value is

calculated using the quadratic value both for averaging and for comparison with the threshold value [20].

The GFSK Demod block was described earlier, and the Unpacked to Packed block converts the unpacked byte stream into the packed stream after demodulation. The least significant bits are extracted from each input byte, then these bits are tightly packed into the output bytes so that all 8 or 16 bits of the output bytes are filled with the correct input bits [21]. The ZMQ PUSH Sink, ZMQ PULL Source, and File Sink blocks are used to save all processed data to a file for further analysis and editing.

When the data is received and edited, it must be forwarded to the recipient. Thus, the second step is shown in Figure 2.

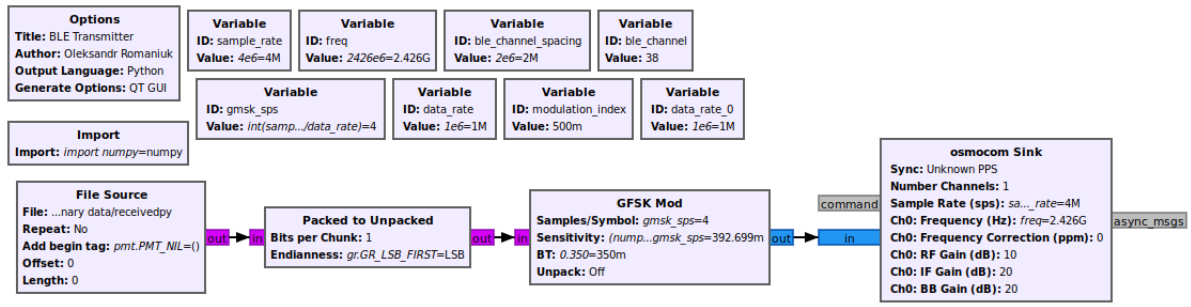


Figure 2: Block diagram of broadcasting BLE packets.

Let's consider this circuit of the block diagram in order. All actions are performed in the reverse direction, that is, everything starts with the File Source block, where the path to the file already edited by the attacker, in raw binary format, is indicated. The variables remain the same as those used in the previous step.

The packed-to-unpacked block carries out the reverse conversion to the unpacked-to-packed block, i.e., a stream of packed bytes is converted into a stream of unpacked bytes [22]. The GFSK Mod modulation block was described earlier, the osmocom Sink is responsible for transmitting data in the form of a signal, according to the specified parameters.

6. Preparation of the Experiment

In the previous section, we set up the GNU Radio environment. The following experiment will use the HackRF One physical SDR device, so it needs to be configured first, and also choose the necessary accessories and the auxiliary software. First of all, we will update the packages of the Ubuntu operating system, where SDR will be used. Next steps: connect HackRF to the USB port, and download only the necessary HackRF utilities from the repository: sudo apt-get install hackrf gqrx-sdr gr-osmosdr hackrf-doc hackrf-firmware inspectrum libhackrf-dev libhackrf0. This version is the most recent of all available for this revision of the device, and for this, a firmware update was carried out according to the following documentation [23].

The next step is to consider the selected software for generating and receiving BLE packets, namely iBeacon and AltBeacon beacons. For this, several applications installed on a mobile phone with the Android operating system were considered, but only two were chosen: BeaconScope and Locate. These two applications have flexible packet creation and editing capabilities and had the clearest BLE signals during the SDR Console v. 3.2 software scan (Figure 3) [24].

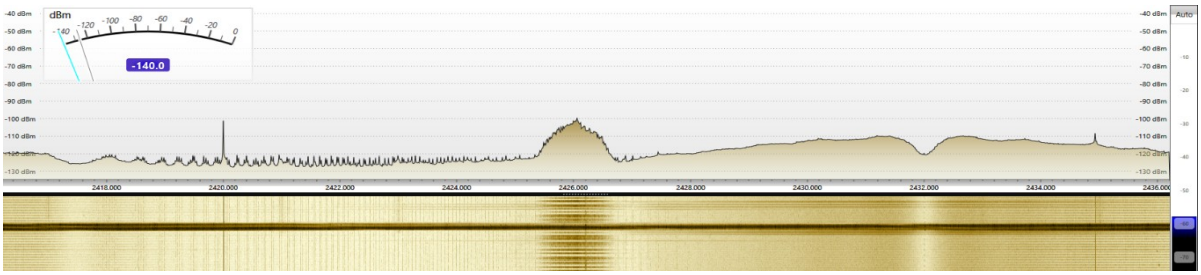


Figure 3: Scanning of BLE packets by the SDR Console software.

HackRF One was about two meters away from the phone during the scan, as you can see in both images the small, thin lines are the iBeacon packets, and the bolder ones are the phone's scan of the Bluetooth frequency set to soak.

7. Attack Emulation on a Sensor Network

During the emulation of a replay attack for the IEEE 802.15.1 standard, i.e. BLE technology, the following tasks are set: generation of BLE packets using BeaconScope and Locate mobile applications; their interception by the HackRF One SDR device; return packets to the recipient (to the same mobile application). It is also necessary to analyze the packets and consider the possibility of their compromise. We turn on the generation of packets one by one (Figure 4) and start the first block diagram of the generation (Figure 1).

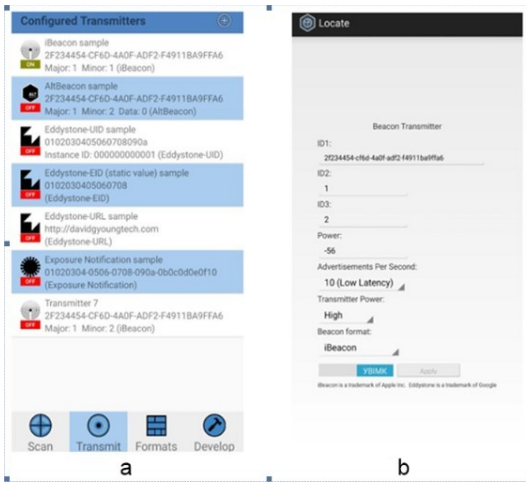
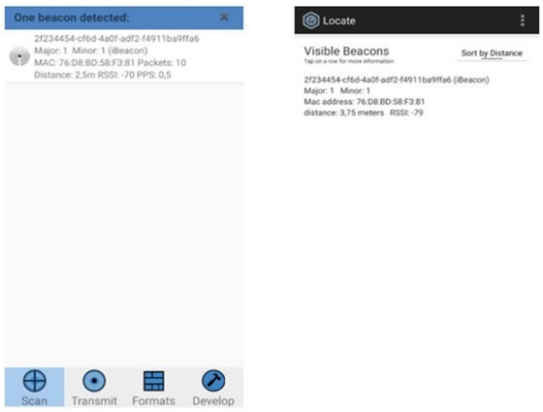


Figure 4: Enabling the generation of BLE packets by (a) BeaconScope and (b) Locate mobile applications.

After generating several hundreds of packets and capturing as much data as possible with the HackRF One SDR device, we stop this process and proceed to the second block diagram (Figure 2). In the File Source block, specify the path to the file with the intercepted data, and launch this prepared project. Now mobile applications are reconfigured to scan BLE beacons, that is, their search and identification; and HackRF One acts as a transmitter. Figure 5 shows the real number of packets received back. The universally unique identifier Universally Unique Identifier (UUID) is the same in sent and received packets.



a b

Figure 5: Number of iBeacon packets found by (a) BeaconScope and (b) Locate mobile applications.

In the first case, 10 packets were received, and in the second, 4, but several hundred were sent as a priority. It was at this stage that difficulties appeared because as it turned out later, the HackRF One SDR device had certain features that became an obstacle to the full completion of the experiment.

Then the previous technical solutions of other users who also used this SDR were analyzed. [25] presents an experiment on modulation and demodulation of a Bluetooth signal, and searching for BLE packets in a binary file with intercepted data. A user encountered an issue with the HackRF One's SDR selectivity being narrower than necessary to receive packets. An estimate of 75% of the packets that the SDR ingress loop passes is provided.

According to this source [26], the author considers the following reasons for the failed experiment: involuntary change of some signal bits (which affects the design of the iBeacon, which should not be disturbed to successfully receive packets from the SDR to the receiver, that is, a mobile device with a special application); bitwise shift of the received signal, which further affects the structure of the iBeacon package, with which further operations will be unsuccessful.

According to this discussion [27], the author also, after analyzing the received packets, notices that he has random bits (proposes that it is noise), i.e. Frequency Xlating FIR Filter blocks (converts the frequency of the signal and simultaneously downsamples the signal using a thinning filter with a finite impulse response, because the desired signal occupies only a narrow part of the bandwidth, and SDRs cover a wide bandwidth due to a very high sampling frequency) and Power Squelch (noise suppression) do not work correctly.

Thus, based on the above technical solutions, it can be concluded that to solve the problem of capturing and playing BLE packets, the hardware capabilities of a regular SDR, software debugging to implement digital signal conversion, and the influence of interference in the ISM range, the characteristics of HackRF One may not be enough; as well as possible incorrect functioning of some blocks in the GNU Radio environment.

However, a certain number of packets were still intercepted and successfully returned. Let's analyze such BLE packets and the possibility of their compromise. The data is received in binary form (see Figure 6), so it cannot be viewed with an ordinary text editor.

For this, the GHex software was chosen, which was installed on the Ubuntu operating system without problems. GHex is an editor that can process "raw" data from binary files and display it for editing in a traditional hex editor window.

Thus, by translating the UUID into the hexadecimal system, it is possible to implement the search for captured BLE packets, with the subsequent compromise of the data and sending it to the recipient. For the receiving party to successfully receive compromised packets, they may be "bombarded" in the channel to prevent legitimate packets from being received.

Figure 6: GHex editor with captured data displayed in binary and hexadecimal.

8. Requirements for Protecting Wireless Sensor Networks from Replay Attack

To protect BLE devices in wireless sensor networks from replay attacks, consider the following requirements and best practices:

1. Data encryption makes the data unreadable by all except the master BLE device and slave device.
2. Break UUIDs into pieces as they are broadcast over the BLE radio frequency channel.
3. Generation and use of one-time dynamic UUIDs.

Observance of Bluetooth Core Specification version 4.2 and above, which will provide a significant level of security against various types of attacks.

Conclusions Future Work

From the experiment, it can be seen that packets were intercepted and duplicated, so a replay attack was successfully simulated. A small number of packets were received due to insufficient SDR HackRF One characteristics or problems with GNU Radio software blocks.

During the emulation of the replay attack, the following tasks were implemented: iBeacon and AltBeacon-type BLE packets were generated using selected mobile applications; their interception by the HackRF One SDR device; reverse transfer of packets to mobile applications. As a result, ten iBeacon packets and four AltBeacon packets were successfully delivered, which were analyzed using GHex software. Accordingly, conclusions were made regarding the possibility of compromising BLE packets.

In future studies, it is planned to emulate a replay attack using other SDRs and other technologies such as ZigBee, and development of recommendations for mitigating replay attacks.

Acknowledgment

The hardware for the experiments was sponsored by Jakob Löwen.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] M. TajDini, V. Sokolov, P. Skladannyi, Performing Sniffing and Spoofing Attack Against ADS-B and Mode S using Software Define Radio, In: 2021 IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo). IEEE, Nov. 29, 2021. doi:10.1109/ukrmico52950.2021.9716665

[2] M. TajDini, V. Sokolov, V. Buriachok, Men-in-the-Middle Attack Simulation on Low Energy Wireless Devices using Software Define Radio, In: 8th International Conference on "Mathematics. Information Technologies. Education:" Modern Machine Learning Technologies and Data Science, vol. 2386, pp. 287-296, 2019.

[3] V. Sokolov, P. Skladannyi, V. Astapenya, Bluetooth Low-Energy Beacon Resistance to Jamming Attack, In: 13th International Conference on Electronics and Information Technologies (ELIT). IEEE, Sep. 26, 2023. doi:10.1109/elit61488.2023.10310815

[4] V. Sokolov, P. Skladannyi, N. Korshun, ZigBee Network Resistance to Jamming Attacks, In: International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo). IEEE, Nov. 13, 2023. doi:10.1109/ukrmico61577.2023.10380360

[5] V. Sokolov, P. Skladannyi, A. Platonenko, Jump-Stay Jamming Attack on Wi-Fi Systems, In: 18th International Conference on Computer Science and Information Technologies (CSIT). IEEE, Oct. 19, 2023. doi:10.1109/csit61576.2023.10324031

[6] A. Oladele, Top IoT Market Demands: Internet of Things Forecasts 2023, 2022.

[7] Statista, Bluetooth Device Shipments Worldwide from 2015 to 2027, 2024.

[8] K. Saravanan, L. Vijayanand, R. K. Negesh, A Novel Bluetooth Man-in-the-Middle Attack based on SSP using OOB Association Model, arXiv, 2012. doi:10.48550/arXiv.1203.4649

[9] T. Melamed, An Active Man-in-the-Middle Attack on Bluetooth Smart Devices, Int. J. Saf. Secur. Eng., vol. 8, no. 2. International Information and Engineering Technology Association, pp. 200-211, Feb. 01, 2018. doi:10.2495/safe-v8-n2-200-211

[10] O. Salem, et al., Man-in-the-Middle Attack Mitigation in Internet of Medical Things, In: IEEE Transactions on Industrial Informatics, vol. 18, no. 3, pp. 2053-2062, Mar. 2022, doi:10.1109/tii.2021.3089462

[11] E. Kalinin, et al., IoT Security Mechanisms in the Example of BLE, Computers, vol. 10, no. 12, p. 162, Nov. 2021, doi:10.3390/computers10120162

[12] Q. Zhang, Z. Liang, Z. Cai, Developing a New Security Framework for Bluetooth Low Energy Devices, Computers, Materials & Continua, vol. 59, no. 2, pp. 457-471, 2019, doi:10.32604/cmc.2019.03758

[13] Great Scott Gadgets, HackRF One, 2023.

[14] Great Scott Gadgets, List of Hardware Revisions, 2021.

[15] Bluetooth SIG, Bluetooth® Wireless Technology, 2024.

[16] Sanchhaya Education Private Ltd., Introduction of IEEE 802.15.4 Technology, 2024.

[17] F. Kuan, Short-Range Wireless Technology vs. Long-Range Wireless Technology, 2022.

[18] GNU Radio, GNU Radio Manual and C++ API Reference, ver. 3.9.4.0, 2023.

[19] GNU Radio, Frequency Xlating FIR Filter, 2023.

[20] GNU Radio, Simple Squelch, 2022.

[21] GNU Radio, Unpacked to Packed, 2020.

[22] GNU Radio, Packed to Unpacked, 2022.

[23] Great Scott Gadgets, Updating Firmware, 2021.

[24] SDR-radio, SDR Console, 2024.

[25] Vick, Decoding Bluetooth Signal and Packets using GnuRadio, 2018.

[26] M. Mwakyanjala, GFSK Modulation/Demodulation with GNU Radio and USRP, 2016.

[27] M. Müller, GFSK Demodulation with Xlating Filter in GNU Radio, 2016.