LoRaShark for LoRa/LoRaWAN Packet Sniffing and Dissection

Daniel Petikian, Mykhailo Ivaniuk, Charles Zhou
danielpetikian2023@u.northwestern.edu,mykhailoivaniuk2023@u.northwestern.edu,
charleszhou2023@u.northwestern.edu

ABSTRACT

LoRa is one of the latest IoT technologies that is being used in a wide range of applications. The novelty of LoRa makes it difficult for academics and developers to inspect its packets. This can be overcome by introducing a low-cost, easy to use packet sniffer. In this study, we used low-cost ESP32 development boards to build **LoRaShark**, a LoRa sniffer that is compatible with the popular packet analysis program Wireshark, enabling for easy packet inspection.

1 INTRODUCTION

Wireless technology connected over 26 billion devices in 2022 [18]. Developing cognitively efficient, adaptable, and cost-effective Internet of Things (IoT) systems is becoming a difficult issue as a result of the rising number of IoT connectivity demands and different application requirements [2]. In response to the rapid growth of the IoT, Low Power Wide Area Network (LPWAN) technologies have gained prominence. One of the most current LPWAN technologies, LoRa™, has increased in popularity and is now one of the most promising technologies for wide-area IoT [15]. LoRaWAN is the MAC layer of the LoRa stack that employs a star topology to facilitate communication between many End Devices (EDs) and the network Gateway (GW) [9]. Despite LoRa's quick adoption and extensive use, its testing technology is novel and has a number of unresolved issues and difficulties. Some of these long-standing challenges include hardware limits, computational complexity, and a lack of standardization. In comparison to Wi-Fi and Bluetooth, very few systems have a low-cost and reliable capability to examine LoRa packets.

Packet analysis is included in the baselines of everything important to a network because it allows users to check the network's state before problems arise [17]. Wireshark is one of the best software tools for analyzing network traffic among all network traffic analyzers, and it is the most popular option for network professionals, security experts, developers, and researchers[12].

This study describes the development of a low-cost, general-purpose LoRa sniffer that can interact with standard network testing tools like Wireshark. This would include creating a system that can sniff packets and send them to a PC via USB on a LoRa-enabled module. To show LoRaWAN fields using Wireshark, the module would need a dissector as well as a Wireshark interface, as specified in the paper.

2 BACKGROUND

This section covers the LoRa technology and Wireshark specifications, as well as the PCAP Capture File Format that Wireshark uses for packet capturing and recording.

The two main components of the LoRa technology are LoRa and LoRaWAN [8]. LoRa is the physical layer protocol, whereas LoRaWAN is the MAC layer protocol. In the parts that follow, the PHY and MAC layers are discussed individually.

2.1 LoRa Physical Layer

LoRa (short for Long Range) is a wireless physical protocol used in the Internet of Things for devices that require a long-range, low-power wireless network to communicate [8]. Its versatility, low cost, and low power operation, as well as features like GPS-free position tracking, make it a powerful tool. Semtech, a

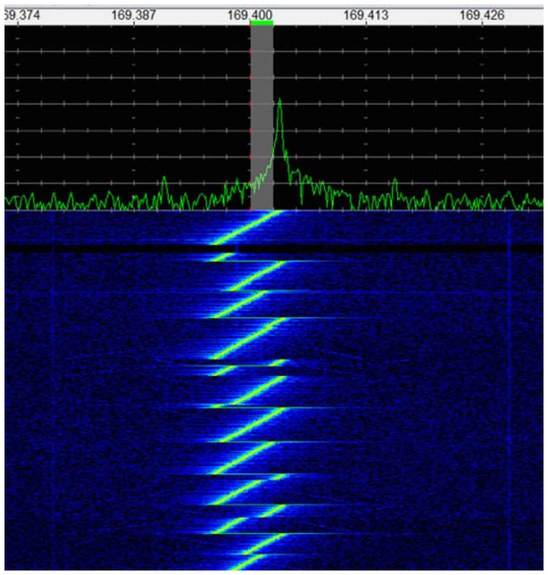


Figure 1: LoRa chrips shown on spectrogram.

California-based analog and mixed-signal semiconductor company, manufactures LoRa radios as proprietary products[9].

2.1.1 Chirp Spread Spectrum (CSS) Modulation. While OFDM and FSK modulations are more often used in current wireless systems, LoRa uses a unique Chirp Spread Spectrum modulation to provide superior link budget and low power performance in the noisy and crowded ISM channels[8]. FSK (Frequency Shift Keying) modulation is another option for LoRa radios. Despite it achieves low power consumption, FSK's communication range falls short of the CSS[19]. CSS entails modulating the data-carrying signal using chirp pulses and symbols are represented as frequency modulated chirps[9]. An example of LoRa chirps is shown in Figure 1. The Spreading Factor (SF) refers to the number of bits encoded for a symbol, which can range from 7 to 12[16].

Apart from modulation, additional PHY layer configuration settings like as Carrier Frequency (CF), Transmission Power (TP), Bandwidth (BW), and Coding Rate (CR) can be changed to improve LoRa communication performance[8]. Bandwidth can range between 125, 250, and 500 kHz[16]; a larger number indicates that a packet with a high data rate can be delivered across a shorter distance due to poor receiver sensitivity. A lower signal-to-noise ratio (SNR) is necessary to provide reliable communication with a higher SF. When a result, as the distance between the transmitter and receiver rises, BW must be reduced and SF must be raised in order to transmit longer packets at a larger energy cost.

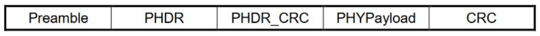


Figure 2: LoRa physical layer packet format

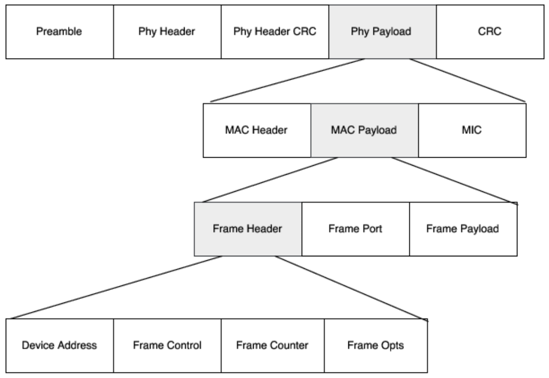


Figure 3: LoRaWAN packet format [3]

2.1.2 Physical Layer Packet Format. The LoRa physical layer packet format is shown in Figure 2. The preamble in the LoRa physical layer packet is used to keep the receiver synced with the transmitter. Physical Layer Headers (PHDR) are divided into two categories: explicit and implicit. PHDR includes the length of the data information, the error correction coding rate, and whether the data load's Cyclic Redundancy Check (CRC) is carried at the end of the packet when communicating with the explicit header. Furthermore, because PHDR has its own CRC, the receiver can verify the packet's integrity by checking the PHDR CRC first[9]. The implicit header mode can be used if the payload length, CR, and CRC are known or fixed, which improves efficiency and reduces transmission time and power consumption, and is therefore employed in LoRaWAN implementations[16].

2.2 LoRaWAN MAC Layer

The LoRa Alliance introduced LoRaWAN, an open-source MAC layer protocol built on top of LoRa PHY. It's made to control a large number of IoT devices that use LoRa radios. It's ideal for smart cities, agricultural, industrial environments, and smart supply chains.

2.2.1 LoRaWAN architecture. The LoRaWAN network is designed for battery-powered EDs in both mobile and stationary configurations [16]. It uses ALOHA-style random access, which allows for energy-efficient communication while also reducing network complexity. The network is structured in a star-of-stars topology, with GWs passing communications between EDs and a backend Network Servers (NS) [5].

2.2.2 MAC Layer Packet Format. A PHY payload is carried by all LoRa uplink and downlink messages, beginning with a single-octet MAC header (MHDR), followed by a MAC payload, and finishing with a 4-octet message integrity code (MIC) [16]. The structure of a LoRaWAN packet is shown in Figure 3.

2.3 Wireshark Packet Analyzer

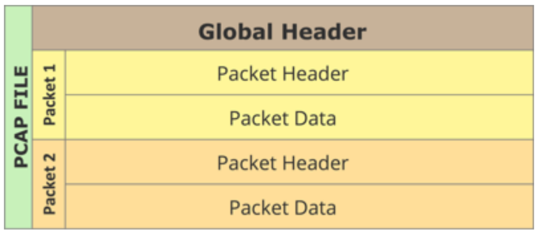


Figure 4: PCAP File Structure

```
typedef struct pcap_hdr_s {
        guint32 magic_number;
        guint16 version_major;
        guint16 version_minor;
        gint32  thiszone;
        guint32 sigfigs;
        guint32 snaplen;
        guint32 network;
} pcap_hdr_t;
```

Figure 5: PCAP Global Header Structure [22]

Wireshark is a commonly used open-source packet capture tool in the industry. Because packets are essentially binary data, Wireshark employs dissectors, which are C/C++/Lua programs that decode messages after WireShark has determined which protocol was used based on file formats.

*2.3.1 PCAP: **P**acket **CAP**tures files.* PCAP (also known as libpcap) is an application programming interface (API) that collects live network packet data from Layers 2 through 7 of the OSI model [10]. Since raw LoRa packets are not directly recognizable by Wireshark, we need to encapsulate LoRa packets into the PCAP format decipherable by Wireshark. The structure of a standard PCAP file is shown in Figure 4. The global header structure is shown in Figure 5: The purpose and format of some fields may not be evident from their name:

- **magic_number** field is used to detect file format and byte ordering. If magic number is 0xa1b2c3d4 then Wireshark knows that all byte ordering is in order. If it's 0xd4c3b2a1 the byte ordering will be reversed.
- **network** field specifies which link-layer header type the Wireshark should expect in order to apply the correct dissector. Since we are trying to capture LoRa packets we used 270 for LoRaTap [11].

WireShark (Version $\geq$ 2.5.0) provides a packet header named LoRaTap for its builtin LoRa dissector and hence can be used for LoRaWan [7, 21]. After we formulate the global header in our PCAP file we need to append each packet to the file with PCAP packet header and the LoRa protocol header prefixes [22].

3 DESIGN AND IMPLEMENTATION

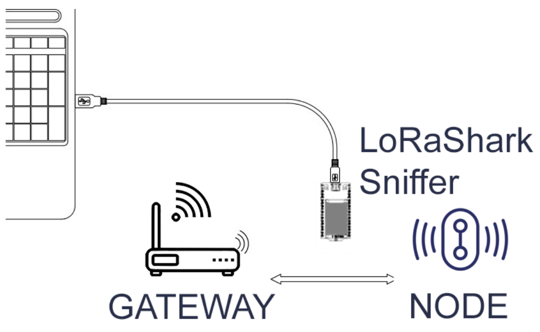


Figure 6: Hardware Setup

Our system took inspiration from existing nRF Bluetooth LE Wireshark sniffer developed by Nordic Semiconductor [14]. A Heltec ESP32 with LoRa Radio was attached to a Linux machine that has Wireshark installed.

The sniffer application on the ESP32 board prints outputs to the computer through the serial interface. A python script functioned as a bridge, parsing serial output from the ESP32 and sending it to Wireshark in the PCAP format. This system can be visualized in Figure 6 and 7.

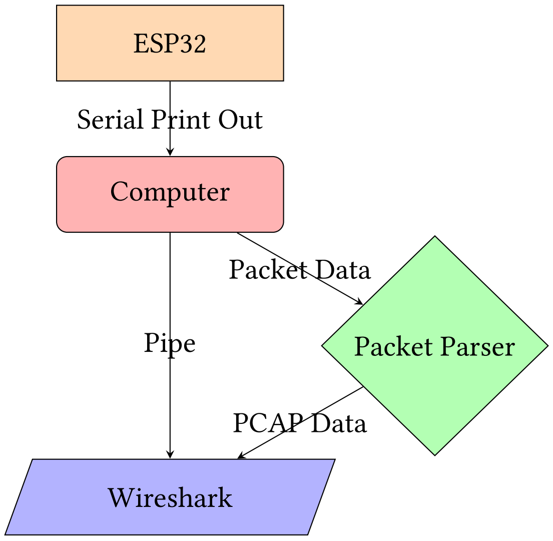


Figure 7: LoRaShark Flow Chart

3.1 Hardware

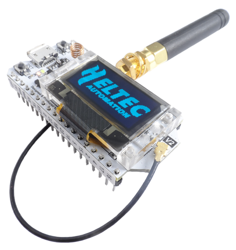


Figure 8: Heltec ESP32 with a LoRa Radio

The hardware chosen is the ESP32 LoRa32 V2 by Heltec. It contains a dual-core 32-bit ESP32 microprocessor along with a SX1276 LoRa radio by Semtech [1]. It also has an OLED screen which can be programmed as well. We chose this board because of its inexpensive price (≤ $25), compact size (2"x1"), and widespread availability (Amazon). It's also Arduino compatible, which simplifies our development. The ESP32 board is shown in Figure 8.

3.2 Single Channel LoRaWAN Gateway and Node

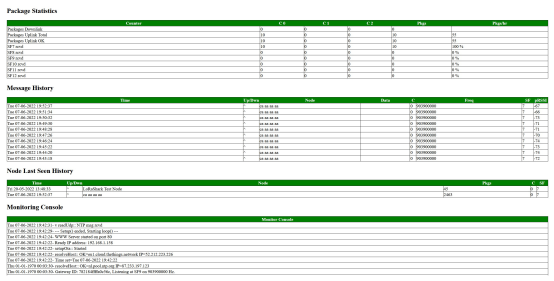
We need to create a LoRaWAN network to test our sniffers. Due to the necessity to monitor a wide variety of frequencies, a commercial LoRaWAN gateway is prohibitively expensive. We opted to mimic a LoRaWAN Gateway using the ESP32 boards we had on hand to keep the cost within the scope of this study.

The Single Channel LoRaWAN Gateway software was updated to fit our hardware and requirements [20]. Due to the hardware restrictions of the SX1276 radio, we can only mimic a single channel LoRa gateway. However, it is sufficient for testing purposes. It's a single-channel LoRaWAN gateway with a web interface and the ability to connect to The Things Network, one of the most popular LoRaWAN NS [6]. As a proof of concept for testing, a LoRaWAN node with the same network configuration as the gateway was set up to transmit a short message. The LoRaWAN Gateway's web interface is shown in Figure 9.

3.3 Packet Sniffers

Initially, we contemplated sniffing packets using the Single Channel LoRaWAN Gateway. We later learned that we could achieve the same functionality with the manufacturer's basic LoRa library.

3.3.1 Sniffer Based on Gateway. Every detail about the packets it receives is recorded by the LoRa gateway. We've decided to use this information and send it through the serial port. For this reason, we modified the current LoRaWAN gateway firmware to add this additional capability to the gateway. Then, on the receiving end, we enabled debug statements using the gateway web control panel, allowing data output through serial for the gateway. Because a LoRaWAN network requires



Package Statistics

Counter	C 0	C 1	C 2	Pkgs	Pkgs/hr
Packages Downlink	0	0	0	0	
Packages Uplink Total	10	0	0	10	55
Packages Uplink OK	10	0	0	10	55
SF7 rcvd	10	0	0	10	100 %
SF8 rcvd	0	0	0	0	0 %
SF9 rcvd	0	0	0	0	0 %
SF10 rcvd	0	0	0	0	0 %
SF11 rcvd	0	0	0	0	0 %
SF12 rcvd	0	0	0	0	0 %

Message History

Time	Up/Dwn	Node	Data	C	Freq	SF	pRSSI
Tue 07-06-2022 19:52:37	^	ca aa aa aa		0	903900000	7	-67
Tue 07-06-2022 19:51:34	^	ca aa aa aa		0	903900000	7	-66
Tue 07-06-2022 19:50:32	^	ca aa aa aa		0	903900000	7	-73
Tue 07-06-2022 19:49:30	^	ca aa aa aa		0	903900000	7	-71
Tue 07-06-2022 19:48:28	^	ca aa aa aa		0	903900000	7	-71
Tue 07-06-2022 19:47:26	^	ca aa aa aa		0	903900000	7	-70
Tue 07-06-2022 19:46:24	^	ca aa aa aa		0	903900000	7	-74
Tue 07-06-2022 19:45:22	^	ca aa aa aa		0	903900000	7	-73
Tue 07-06-2022 19:44:20	^	ca aa aa aa		0	903900000	7	-74
Tue 07-06-2022 19:43:18	^	ca aa aa aa		0	903900000	7	-72

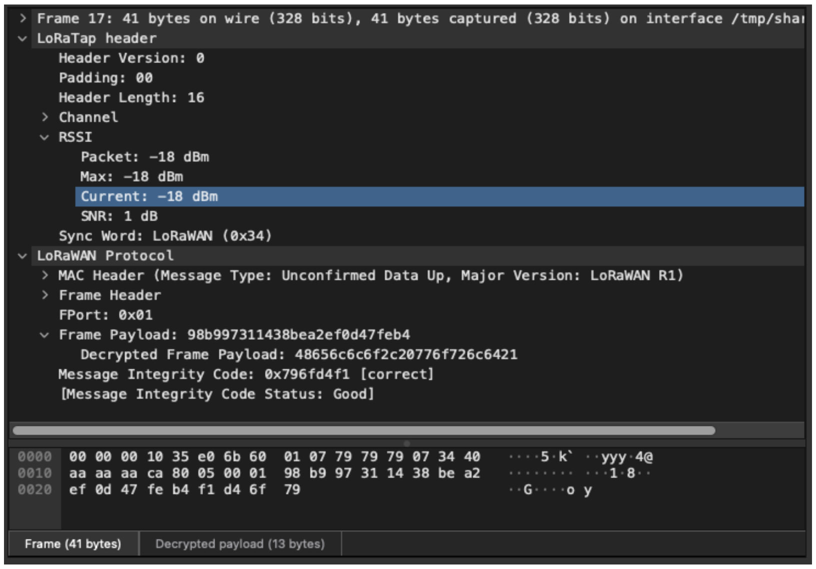
Node Last Seen History

Time	Up/Dwn	Node	Pkgs	C	SF
Fri 20-05-2022 13:48:33	^	LoRaShark Test Node	45	0	7
Tue 07-06-2022 19:52:37	^	ca aa aa aa	2463	0	7

Monitoring Console

```
Monitor Console
Tue 07-06-2022 19:42:31- v readUdp:: NTP msg rcvd
Tue 07-06-2022 19:42:29- --- Setup() ended, Starting loop() ---
Tue 07-06-2022 19:42:24- WWW Server started on port 80
Tue 07-06-2022 19:42:22- Ready IP address: 192.168.1.158
Tue 07-06-2022 19:42:22- setupOta:: Started
Tue 07-06-2022 19:42:22- resolveHost:: OK=eu1.cloud.thethings.network IP=52.212.223.226
Tue 07-06-2022 19:42:22- Time set=Tue 07-06-2022 19:42:22
Thu 01-01-1970 00:03:30- resolveHost:: OK=nl.pool.ntp.org IP=87.233.197.123
Thu 01-01-1970 00:03:30- Gateway ID: 782184ffffa0c56c, Listening at SF9 on 903900000 Hz.
```

Figure 9: LoRaWAN Gateway web interface



```
> Frame 17: 41 bytes on wire (328 bits), 41 bytes captured (328 bits) on interface /tmp/shar
v LoRaTap header
    Header Version: 0
    Padding: 00
    Header Length: 16
  > Channel
  v RSSI
      Packet: -18 dBm
      Max: -18 dBm
      Current: -18 dBm
      SNR: 1 dB
    Sync Word: LoRaWAN (0x34)
v LoRaWAN Protocol
  > MAC Header (Message Type: Unconfirmed Data Up, Major Version: LoRaWAN R1)
  > Frame Header
    FPort: 0x01
  v Frame Payload: 98b997311438bea2ef0d47feb4
      Decrypted Frame Payload: 48656c6c6f2c20776f726c6421
    Message Integrity Code: 0x796fd4f1 [correct]
    [Message Integrity Code Status: Good]

0000  00 00 00 10 35 e0 6b 60  01 07 79 79 79 07 34 40   ····5·k` ··yyy·4@
0010  aa aa aa ca 80 05 00 01  98 b9 97 31 14 38 be a2   ········ ···1·8··
0020  ef 0d 47 fe b4 f1 d4 6f  79                        ··G····o y
```

Figure 10: Sniffed LoRa Packet dissection shown in Wireshark

an NS, this sniffer has the drawback of needing to be connected to the internet in order to function.

3.3.2 Standalone Sniffer. We realized that we were looking at all the packets on the ESP32 boards only at the physical layer. We gained a sense that this should be possible with the simple LoRa library provided by Heltec[4]. We analyzed the library and discovered that other than RSSI and SNR, the other values (frequency, channel, etc.) had to be manually set up. We then wrote the LoRa-sniffer.ino file to upload to the ESP32s. After seeing the print out statements with the complete PHY layer packet, we realized we can use this code and not have to be dependent on being connected to the internet. Also, because it is not dependent on a LoRaWAN gateway, it can dissect other types of LoRa packets as long as the sync word is correct.

3.4 Python Script

The Python script is in charge of decoding serial printouts from the ESP32 board and sending them to Wireshark through piping.

3.4.1 Wireshark Piping. UNIX pipes can be used by Wireshark to receive data from another program. A pipe is opened and Wireshark is launched by the python script. The script then write the PCAP global header as described in Section 2.3.1, and continues to parse data transmission from the ESP32, which receives LoRAWAN packets, and then appending them to the WireShark pipe in LoRaTap format.

3.4.2 Parsing. Parsers are similar in both types of sniffers. One line containing the whole packet contents, followed by the meta data, will be serially transmitted for each packet received by ESP32. Following that, the metadata is written to a Wireshark pipe in the appropriate format. We're using Python's struct package to properly encapsulate data for PCAP and LoRaTap.

4 EVALUATION

Our project was successful in its goal to be able to send and dissect LoRaWAN packets to Wireshark. We were able to look at metadata and data from PHY, MAC, and frame. An example can be seen in Figure 10. Furthermore it can used as an abstraction for capturing protocols that requires specialized hardware for sensing and later transmitting captured data through serial. The source code can be found at our GitHub repository.

4.1 Metadata

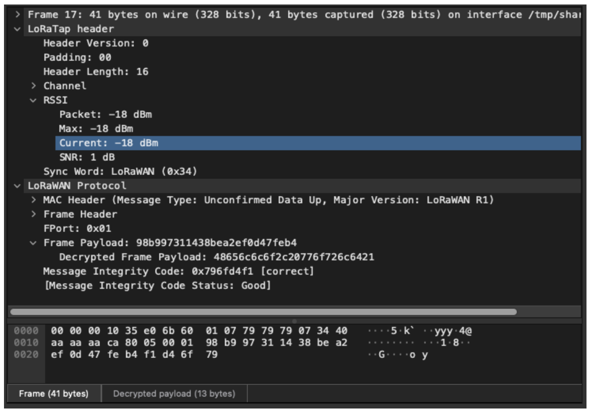
The metadata includes parameters from the LoRa PHY layer header, including packet signal information.

4.1.1 Packet Header. Each packet that comes into Wireshark comes in with a live timestamp and the correct data length. Without the correct data length, the system crashes due to misplaced bytes.

4.1.2 Channel Information. The frequency, bandwidth, and spreading factor of the received packet may all be seen in Wireshark. When configuring the Heltec device, this is done on the hardware level. This data is hard-coded, and if it were incorrect, the user would be unable to view any packets.

4.1.3 RSSI. Within the RSSI section, Wireshark can recognize RSSI and SNR. RSSI (Received Signal Strength Indicator) is a measurement used to determine whether the signal strength is good or not for a wireless connection. SNR (signal to noise ratio) is the ratio of the signal power the device received to the noise floor. It is symbolic to the quality of the received signal [13].

4.1.4 Sync Word. The sniffer has the sync word set by default as 0x34 for LoRaWAN 3. Wireshark is able to further dissect the packet with a built-in LoRaWAN dissector by recognizing the sync word.



```
> Frame 17: 41 bytes on wire (328 bits), 41 bytes captured (328 bits) on interface /tmp/shar
v LoRaTap header
    Header Version: 0
    Padding: 00
    Header Length: 16
  > Channel
  v RSSI
      Packet: -18 dBm
      Max: -18 dBm
      Current: -18 dBm
      SNR: 1 dB
    Sync Word: LoRaWAN (0x34)
v LoRaWAN Protocol
  > MAC Header (Message Type: Unconfirmed Data Up, Major Version: LoRaWAN R1)
  > Frame Header
    FPort: 0x01
  v Frame Payload: 98b997311438bea2ef0d47feb4
      Decrypted Frame Payload: 48656c6c6f2c20776f726c6421
    Message Integrity Code: 0x796fd4f1 [correct]
    [Message Integrity Code Status: Good]

0000  00 00 00 10 35 e0 6b 60  01 07 79 79 79 07 34 40   ····5·k` ··yyy·4@
0010  aa aa aa ca 80 05 00 01  98 b9 97 31 14 38 be a2   ········ ···1·8··
0020  ef 0d 47 fe b4 f1 d4 6f  79                        ··G····o y
```

Figure 11: Encrypted Frame Payload shown in Wireshark

4.2 MAC Header

The message type and major version are all available for view. The message type tells what kind of LoRaWAN packet was sent [23].

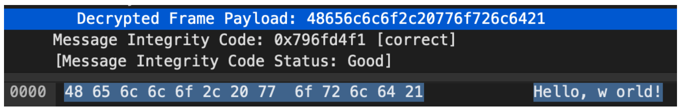
4.3 Frame Header

Within the LoRaWAN frame header we can see the device address and other flags within the frame control. Wireshark also correctly dissects the Frame Port shown in Figure 11.

4.4 Frame Payload

The frame payloads are encrypted if keys are not present. Keys can be inputted to the Wireshark under Preferences->Protocols->LoRaWAN. The parameters should be from the target LoRaWAN network (Figure 12):

- Device Address[1]
- Network Key
- Application Key
- Application EUI



```
  Decrypted Frame Payload: 48656c6c6f2c20776f726c6421
Message Integrity Code: 0x796fd4f1 [correct]
[Message Integrity Code Status: Good]
0000  48 65 6c 6c 6f 2c 20 77  6f 72 6c 64 21      Hello, w orld!
```

Figure 12: Decrypted payload data after supplying keys described in Section 4.4.

4.5 Message Integrity Code

The last four bytes of the payload inform the receiver and Wireshark about the message's integrity. Wireshark will recalculate the code based on the packet data and compare it to the code from the packet and see if they matches.

5 CONCLUSION

LoRa is a transformative technology in the IoT space, but it needs a cost-effective and user-friendly network monitoring and development solution. To address this, we proposed LoRaShark, a low-cost solution for real-time sniffing of LoRa/LoRaWAN packets that is compatible with Wireshark, an industry standard tool for packet analyzation. The sniffer's performance and accuracy were evaluated on a LoRaWAN network. In the future, a more complicated solution based on Software Defined Radios (SDR) might be utilized to build an advanced sniffer capable of monitoring LoRa packets across a broad frequency range.

[1]The ordering of bytes has to be in reverse order. (i.e. `0xa1b2c3` should be `0xc3b2a1`)

REFERENCES

[1] [n. d.]. WiFi LoRa 32 (V2.1). https://heltec.org/project/wifi-lora-32/

[2] Mays Alshaikhli, Tarek Elfouly, Omar Elharrouss, Amr Mohamed, and Najmath Ottakath. 2022. Evolution of Internet of Things From Blockchain to IOTA: A Survey. *IEEE Access* 10 (2022), 844–866. https://doi.org/10.1109/access.2021.3138353

[3] Emekcan Aras. [n. d.]. Exploring the Security Vulnerabilities of LoRa. https://www.researchgate.net/figure/LoraWAN-Packet-Structure_fig2_318575428

[4] Heltec Automation. [n. d.]. WiFi LoRa 32 (V2.1). https://github.com/HelTecAutomation/Heltec_ESP32

[5] Roger Pueyo Centelles, Felix Freitag, Roc Meseguer, and Leandro Navarro. 2021. Beyond the Star of Stars: An Introduction to Multihop and Mesh for LoRa and LoRaWAN. *IEEE Pervasive Computing* 20 (04 2021), 63–72. https://doi.org/10.1109/mprv.2021.3063443

[6] Laird Connectivity. 2021. Migrating to The Things Stack v3: Changes Ahead. https://www.lairdconnect.com/resources/blog/migrating-things-stack-v3-changes-ahead

[7] Erik de Jong. 2017. LoRaTap Encapsulation. https://github.com/eriknl/LoRaTap

[8] Shilpa Devalal and A. Karthikeyan. 2018. LoRa Technology - An Overview. , 284–290 pages. https://doi.org/10.1109/ICECA.2018.8474715

[9] Mohammed Jouhari, El Mehdi Amhoud, Nasir Saeed, and Mohamed-Slim Alouini. 2022. A Survey on Scalable LoRaWAN for Massive IoT: Recent Advances, Potentials, and Challenges. *arXiv:2202.11082 [cs, eess]* (02 2022). https://arxiv.org/abs/2202.11082

[10] Tim Keary. 2019. PCAP: Packet Capture, what it is what you need to know. https://www.comparitech.com/net-admin/pcap-guide/

[11] LIBPCAP. [n. d.]. Link-layer header types | TCPDUMP LIBPCAP. https://www.tcpdump.org/linktypes.html

[12] Vivens Ndatinya, Zhifeng Xiao, Vasudeva Rao Manepalli, Ke Meng, and Yang Xiao. 2015. Network forensics analysis using Wireshark. *International Journal of Security and Networks* 10 (2015), 91. https://doi.org/10.1504/ijsn.2015.070421

[13] The Things Network. [n. d.]. RSSI and SNR. https://www.thethingsnetwork.org/docs/lorawan/rssi-and-snr/

[14] Nordic Semiconductor. 2021. nRF Sniffer for Bluetooth LE v4.1.0. https://infocenter.nordicsemi.com/pdf/nRF_Sniffer_BLE_UG_v4.1.0.pdf

[15] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. 2017. A survey on LPWA technology: LoRa and NB-IoT. *ICT Express* 3 (03 2017), 14–21. https://doi.org/10.1016/j.icte.2017.03.004

[16] N Sornin, M Luis, T Eirich, T Kramp, and O Hersent. 2015. The authors reserve the right to change specifications without notice. LoRa Specification 2 NOTICE OF USE AND DISCLOSURE 5. https://lora-alliance.org/wp-content/uploads/2020/11/2015_-_lorawan_specification_1r0_611_1.pdf

[17] Jim Thor. 2009. Why You Need a Network Analyzer. http://www.technewsworld.com/story/67411.html

[18] Lionel Vailshery. 2021. Global number of connected IoT devices 2015-2025. https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/

[19] Lorenzo Vangelista. 2017. Frequency Shift Chirp Modulation: The LoRa Modulation. *IEEE Signal Processing Letters* 24 (12 2017), 1818–1821. https://doi.org/10.1109/lsp.2017.2762960

[20] M Westenberg. 2021. GitHub - things4u/ESP-1ch-Gateway: Version 6 of the single channel gateway. https://github.com/things4u/ESP-1ch-Gateway

[21] WireShark. 2018-02-06. Wireshark 2.5.0 Release Notes. https://www.wireshark.org/update/relnotes/wireshark-2.5.0.html

[22] WireShark. 2020. Libpcap File Format. https://wiki.wireshark.org/Development/LibpcapFileFormat

[23] RF Wireless World. [n. d.]. LoRaWAN MAC Layer Message Formats | LoRaWAN MAC Commands. https://www.rfwireless-world.com/Tutorials/LoRaWAN-MAC-layer-inside.html