From Demodulation to Decoding: Toward Complete LoRa PHY Understanding and Implementation

ZHENQIANG XU, SHUAI TONG, PENGJIN XIE, and JILIANG WANG, School of Software, Tsinghua University, China

As an example of Low Power Wide Area Network technology, LoRa has garnered a lot of interest from the academic and business communities. Nevertheless, there is still much to learn about LoRa, and there is a significant performance difference between implementations in terms of SNR and packet reception rate. This article provides a thorough explanation of the LoRa physical layer protocol (PHY) and highlights the underlying causes of the performance disparity. We introduce the first verifiable performance guaranteed full-stack LoRa PHY implementation. In contrast to many previous efforts that need SNR > 0, we improve the demodulation to operate at extremely low SNR (20 dB) and analytically validate the performance. Using LoRa characteristics and packet manipulation, we determine the sequence and parameters of decoding operations, such as dewhitening, error correction, deinterleaving, and so forth. We build a complete real-time LoRa solution using the GNU Radio platform and do extensive testing. Our method can achieve (1) a 100% decoding success rate, whereas existing methods can only support 66.7%, (2) 142 dBm sensitivity, which is the limiting sensitivity of commodity LoRa, and (3) a 3,600-m communication range in the urban area, which is superior to commodity LoRa under the same conditions.

CCS Concepts: • **Networks** → **Network protocols**; **Network performance analysis**; **Wide area networks;**

Additional Key Words and Phrases: Low-power wide-area network, LoRa, physical layer

ACM Reference format:
Zhenqiang Xu, Shuai Tong, Pengjin Xie, and Jiliang Wang. 2022. From Demodulation to Decoding: Toward Complete LoRa PHY Understanding and Implementation. *ACM Trans. Sensor Netw.* 18, 4, Article 64 (December 2022), 27 pages.
https://doi.org/10.1145/3546869

1 INTRODUCTION

Due to its ability to provide long-distance, low-power communication under extremely low SNR conditions, Low Power Wide Area Network (LPWAN) technology has proven to be particularly promising for linking millions of devices in the Internet of Things (IoT). Lately, both academia and business have embraced LoRa, a representative LPWAN technology [1-4]. Smart cities, smart agriculture, and smart logistics are just a few of the IoT applications that have made use of it [5-8]. Many studies have been done in this field. In the research conducted by Xue Geng et al [54], a method for designing preambles for random access and a timing advance estimation method for OTFS systems in high-mobility scenarios are examined. One of the significant findings of this study is that these methods can substantially enhance the performance of communication systems in high-mobility conditions, including applications such as Internet of Things (IoT) networks and intelligent communications.In Ayushi Upadhyay's research [55], the performance analysis of preamble detection at the maximum throughput level for OFDM systems is investigated. One of the key findings of this study is that optimizing the preamble detection process can lead to overall performance improvements in OFDM systems, particularly in aspects such as detection accuracy and error rates.In another study by Yadan Zheng et al [56]., a precise method for positioning using millimeter waves and MLP-Mixer neural networks is introduced. One of the main achievements of this research is that this method can effectively and accurately provide positioning in highly dynamic environments, which can be significantly beneficial in applications such as wireless sensor networks and the Internet of Things (IoT).A study by Vukan Ninkovic et al [57]. presents a deep learning approach for preamble-based packet detection in Wi-Fi networks. One of the main achievements of this research is that utilizing deep networks can significantly enhance the accuracy and efficiency of packet detection under various Wi-Fi network conditions. This improvement can be beneficial I optimizing the performance and security of wireless networks.The paper

by Ruilong Yao et al [58] . examines a power line communication method with robust timing and carrier recovery against narrowband interference, particularly in smart grid applications. One of the main achievements of this research is that this method can significantly improve the stability and performance of communications in highly interfered environments. This improvement can enhance the efficiency and reliability of data transmission networks over power lines.

In the following, we will mention research innovation

Finding the preamble in communication signals is usually a critical step in the synchronization and detection of received signals. Combining time-domain, spectral-domain, and frequency-domain methods can enhance the accuracy and reliability of this process. Below is an overview of these combinations:

Time Domain Method:

In this method, the signal is analyzed in the time domain. Algorithms that look for repeating patterns or specific characteristics in the signal can be employed. For example, using correlation to find similarities between the received signal and a known preamble pattern.

Spectral Domain Method:

In this method, the signal is analyzed in the frequency domain. Fast Fourier Transform (FFT) is usually used for this purpose.Spectral analysis helps identify the main frequencies present in the signal and its specific spectral characteristics.

Frequency Domain Method:

This method examines frequency changes over time. Techniques like Short-Time Fourier Transform (STFT) or Wavelet Transform are suitable for this purpose.This method is particularly useful for identifying minor frequency changes and noise in the signal.

Combining the Methods

Combining these three methods can be done as follows:

Pre-processing in the Time Domain: Initially, filter the signal in the time domain to remove primary noise.

Spectral Analysis: Convert the signal to the frequency domain (e.g., using FFT) to identify its main frequencies.Time-Frequency Analysis: Use techniques like STFT or Wavelet Transform to examine frequency changes over time.Combining Results: Integrate the results of each analysis and look for common patterns or consistent results indicating the presence of the preamble.

Benefits of Combining Methods

Increased Accuracy: Combining multiple methods enhances the accuracy of preamble detection as each method can cover the weaknesses of the others.Reduced Errors: Utilizing several techniques helps in identifying and eliminating noise, reducing the chances of incorrect preamble detection. Improved Noise Resistance: Combined methods are generally more resistant to environmental noise and frequency changes in the signal. In conclusion, using a combination of time-domain, spectral-domain, and frequency-domain methods can help in more accurately and reliably detecting the preamble in communication signals.

This work is in part supported by NSFC No. 62172250, No. 61932013, Tsinghua University Initiative Scientific Research Program.
Authors' address: Z. Xu, S. Tong, P. Xie, and J. Wang (corresponding author), School of Software, Tsinghua University, Beijing, China; emails: xu-zq17@mails.tsinghua.edu.cn, tl19@mails.tsinghua.edu.cn, xiepengjin@tsinghua.edu.cn, jiliangwang@tsinghua.edu.cn.

1550-4859/2022/12-ART64
https://doi.org/10.1145/3546869

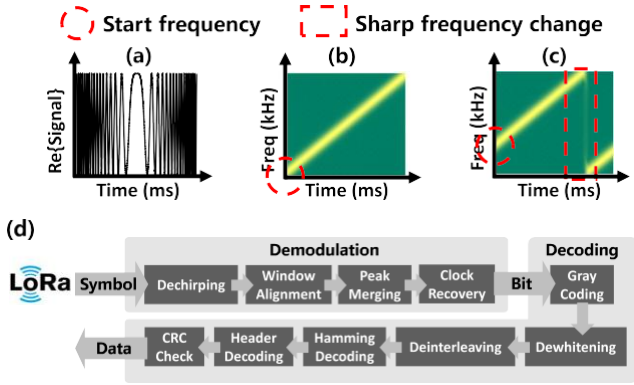

Fig. 1. (a) Real part of a base up-chirp symbol. (b) Base up-chirp symbol. (c) Shifted symbol. (d) Complete procedure of LoRa PHY.

The CSS modulation mechanism is applied by LoRa, which makes it anti-interference and able to realize long-range communication. Collision resolution [9, 10], LoRa backscatter [11–13], weak signal decoding [14, 15], etc., were some of the research works done in that period. A base symbol in the physical layer protocol of LoRa (PHY) is a linearly increasing time-frequency chirp, as shown in Figure 1(a,b). The start frequency (fstart) of a symbol contains the information being encoded. A LoRa symbol has two portions with steep drop in frequency, see Fig. 1(c). Although LoRa has drawn a lot of attention from both academia and industry, the details of LoRa PHY, i.e., how LoRa demodulates and decodes the received signal, are not yet known to us, because LoRa PHY is a closed protocol owned by Semtech Corporation. In this work, we aim at arriving at the complete understanding of the LoRa physical layer and at developing a better tool to analyze the LoRa network. We reveal the fact that most of the existing understandings are incomplete and even incorrect. Consequently, the most widely used LoRa implementation rpp0/gr-lora (RPP0) [16] exhibits a large performance gap with respect to the real hardware. E.g., Packet Reception Rate (PRR) is only 20.0% and it requires SNR much higher than LoRa. Understanding LoRa PHY encompasses two main steps: demodulation and decoding. 1Demodulation translates symbols to raw bits, and decoding translates these raw bits to meaningful data bytes. The aim of LoRa demodulation is to determine fstart of each symbol. In fact, demodulation directly determines the performance of receiver. Currently, a demodulation method that is mainly adopted translates the signal to a single tone by multiplying each symbol with a linearly decreasing chirp and extracts the frequency of the single tone as the start frequency. This operation might incur a high SNR loss in signaling losing in the translation and failure to leverage LoRa features, which prohibits LoRa from long-range low-SNR communication. We explain the exact reasons for signal losing in Section 3.1. For the decoding, existing works [16–21] can not derive the order and parameters for the decoding operations. Thus, they have a very low PRR even for high SNR, for example, under 66.7%, due to the wrong perception of LoRa decoding process. These methods are also often incomplete and over-claimed; for example, methods in References [16–18] claim they can support all spreading factors (SF), though our experiments show they failed to do so.

[1]For the transmitter, modulation, and encoding. Here we only discuss the behavior of the receiver for simplicity.

Table 1. Comparison of BastilleResearch/gr-lora (BR), RPP0 and tapparelj/gr-lora_sdr (TAPP) and our implementation

	BR [17]	RPP0 [16]	TAPP [18]	Ours
All SFs	✗	✗	✗	✓
All CRs	✓	✓	✓	✓
CRC	✗	✗	✓	✓
Explicit Mode	✗	✓	✓	✓
Implicit Mode	✓	✗	✓	✓
Clock Recovery	✗	✗	✗	✓
Low SNR Support	✗	✗	✗	✓
Real Time	✓	✓	✗	✓
Packet Coverage	4.2%	20.0%	66.7%	100%

This paper presents a full comprehension of and realization of a real-time full-stack LoRa PHY with provable performance guarantees. In the demodulation part, we reveal the fundamental reason of SNR loss to be phase misalignment due to window misalignment and internal symbol phase offset. We first compensate the window misalignment by accurate measurement of the window offset. Internal symbol phase offset is addressed by oversampling the signal with 2B, as opposed to existing works where the sampling frequency is B (Bandwidth), in order to calculate the peaks for each of the segments. The paper concludes by providing a way in which to merge those two peaks with minimal sensitivity losses. Sensitivity can be theoretically proved to be very close to "perfect" demodulation in our method. We also suggest a dynamic clock drift compensation algorithm for low-cost LoRa devices, which have relatively long LoRa packets. For decoding, we present the way LoRa PHY translates the bits from the demodulation part to meaningful packets. From the number of symbols in a LoRa packet, calculated using the official formula [22], we further deduce and check the structure of the packet coming from LoRa by doing some manipulations to the content of the packet. We will use the structure of the packet to analyze the requirements of the decoding processes and derive the order of the Gray coding, deinterleaving, Hamming decoding, and dewhitening. We can then use the deduced order of decoding and packet format to spoof transmitting packets into recovering the configuration parameters of the decoding process, the header structure, and the CRC polynomial. Finally, we realize a real-time SDR LoRa PHY, illustrated in Figure 1(d). A comparison with states of the art is shown in Table 1. Our implementation of decoding beats theirs by a significant margin in several aspects: being real-time, requiring very low SNR to work, having broad support for all modes and parameters for LoRa, and working robustly with devices that are inexpensive, having clock drift and long packets. We theoretically show that our implementation could work under SNR = 20 dB. We also prove that the order of the decoding operations is provable and it can be verified with the random sequence generated, as we discuss in Section 4.3. As a result, in all our experiments, 100% of the decoded packets were correct. Therefore we consider the configuration as reliable. Contributions. We show the performance gap between existing LoRa implementations and commodity LoRa, and reveal the fundamental reasons. We present a comprehensive understanding of the demodulation and decoding of LoRa. We implement a full-stack real-time LoRa PHY on the GNU Radio SDR platform. Based on our analysis, we can even analyze and improve the performance of the commodity.

[2]https://github.com/jkadbear/LoRaPHY.

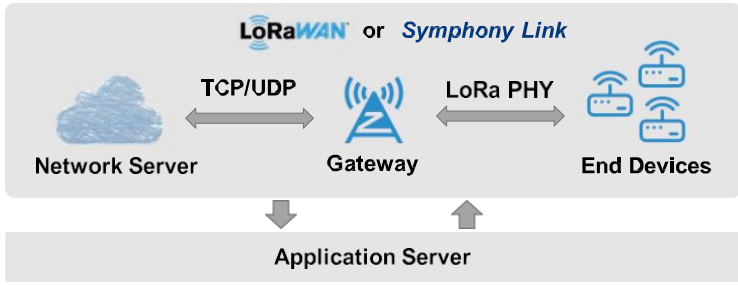

Fig. 2. LoRa network architecture. LoRaWAN or Symphony Link protocol controls how a network server interacts with end devices.

LoRa. For instance, we demonstrate that commodity LoRa chip is vulnerable to "phase misalignment attack" (Section 6.5) and demonstrate how this vulnerability can be mitigated. We widely evaluate our solution in an experimental framework with results presented. Results show sensitivity of our implementation reaching 142 dBm, which is the limiting sensitivity of commodity LoRa. Our implementation has an effective communication range of 3,600 m in the urban environment, while commodity RPI gateways (Section 5) only reach the maximum of 2,800 m. Our LoRa implementation has much higher decoding success rate (100%) than existing works (≤66.7%).

BACKGROUND 2.1 Known and Unknown Facts of LoRa

Figure 2 shows mainstream LoRa network architecture. The LoRa network operates under the control of the LoRaWAN [23] or Symphony Link [24] protocol, which presents an interface to the application server's data. There exist three main communication entities in the LoRa network, i.e., network server, gateway, and end device. The network server communicates with the gateway through a normal internet connection; on the other hand, the end devices interchange with the gateway through using LoRa PHY wireless protocol. Full open source implementations are available for nearly all components and protocols used in Figure 2, except LoRa PHY. While some reverse-engineering works have been done, the complete understanding of LoRa PHY has not been finalized yet. In the following, we show some basic concepts of LoRa PHY. For details on demodulation and decoding, refer to Sections 3 and 4. LoRa PHY uses CSS to obtain long-range transmissions. The basic communication unit in LoRa PHY is a chirp symbol. We denote this unshifted chirp symbol by base chirp, as shown in Figure 1(b). When the frequency increases with time we call it up-chirp and otherwise down-chirp. The novel part of the LoRa modulation is to use the start frequency of a cyclically shifted symbol to encode data. Figure 1(c) shows a LoRa symbol with two chirp segments. In LoRa specifications [22, 25, 26]

the number of bits encoded by a symbol is SF. The SF satisfies $2^{SF} = B \quad T$, where B is bandwidth and T is chirp period. There are at most 2^{SF} cyclically shifted up-chirp symbols given a specific SF. An up-chirp can be represented as

$$chirp(t; f_0) = Ae^{j2\pi (f_0 + \frac{B}{2T} t)t}, \tag{1}$$

where A is amplitude and f_0 is start frequency ($t = 0$).

Figure 3 shows the basic understanding of a standard LoRa packet. Overall, it has three parts: preamble, **start frame delimiter (SFD)**, and data symbols. The preamble is a series of base

Fig. 3. A LoRa packet in the time-frequency plane.

up-chirps followed by two up-chirp symbols indicating network ID.[3] The SFD is 2.25 down-chirps indicating the start of data symbols. Data symbols include PHY header, payload, and payload CRC (header and CRC are optional).

LoRa supports the following configurations. **Code Rate (CR)**: LoRa applies Hamming code of coding rate $\frac{4}{5}$, $\frac{4}{6}$, $\frac{4}{7}$ or $\frac{4}{8}$ [22]. **Low Data Rate Optimization (LDRO)**: When sending long packets,

LoRa enables better stability for LDRO mode at the cost of data rate. Implicit/explicit mode: In explicit mode, there is a PHY header in the packet, and in implicit mode, there is no PHY header. Unknown: (1) The ideal demodulation of LoRa, i.e., how a symbol is translated to bits, is still unknown, which is the key for LoRa to achieve low SNR and long-distance communication. (2) None of the existing works proposes a full understanding of LoRa packet structure. (3) Details of LoRa specific modes are unknown, e.g., LDRO mode.

1.1 State-of-the-Art Open Source Implementations The space of prior research in LoRa communication is quite rich, including LoRa collision decoding [1, 2, 4, 9, 10, 27], LoRa chirp based backscatter [11–13, 28], LoRa network optimization [15, 29], and so on. Unfortunately most of them do not release public accessible implementations. Considering that the LoRa network architecture is brand new and not maturely understood by the community, any new open source efforts deserve appreciation. Hence, we first summarize the publicly available LoRa-related open source projects.

Upper Layer Implementations The upper layer of LoRa network has rich open source implementations. There are embedded implementations from Semtech corporation for official LoRa gateway and end device [30, 31]. ChirpStack [32] is a common implementation used for network servers. A light implementation that combines the network server with the application server is provided by LoRaWAN-Server [33]. The lowest level code here is the driver for LoRa chips. Therefore it does not include the physical layer mechanism of LoRa communication.

Physical Layer Implementations Since LoRa PHY is a proprietary protocol belonging to Semtech corporation, there does not exist any public document explaining all the details of LoRa PHY. Existing researches and systems mainly rely on three open source SDR implementations, i.e., BR [17], RPP0 [16], and TAPP [18]. BR focuses on the implicit header mode of the LoRa packet. Dechirping-based demodulation is applied by BR. It completely ignores some important details of dechirping and is unreliable in low SNR. We will discuss the details in Section 3. Besides, BR decodes only LoRa packets with SF = 8. Our tests showed that BR can decode only the first 4 bytes of a packet and fails to decode any complete packet with more than four bytes. RPP0 targets explicit header mode LoRa packet decoding. LoRa symbols are demodulated by RPP0, as in Figure 1(c). It searches for sharp frequency changes and then uses them to estimate the starting frequencies of the

[3]The two symbols are not base up-chirps typically. They are used like network masks to separate different networks. For example, they are set to 0x0304 for public network in LoRaWAN.

[4]Though existing works have revealed many details about LoRa PHY, the non-ideal demodulation performance and non-100% decoding success rate mean their results are incomplete.

symbols. RPP0 depends on time-domain information, and it cannot recover LoRa packets for SNR<0. TAPP is the recent implementation for LoRa. It's not optimized for LoRa demodulation and has introduced high computation overhead. PRR of TAPP goes down with the decoder unable to consume the input data in time. Based on our evaluation, it gets to a packet loss of 70% at a rate of one packet per second. The average PRR of TAPP is only 66.7%. For decoding, none of those existing works can provide full decoding capability, and thus they can only decode a portion of LoRa packets. Besides, there are other LoRa PHY related works that provide open source implementations. Charm [14] focuses on weak signal decoding by leveraging the multi-gateway structure of the LoRa network. The Charm decoder [34] is implemented offline with MATLAB only. The author does not provide how to decode and solve the phase alignment problem (explained in Section 3). TinySDR [35] is an SDR platform developed for internet-of-things networks. A LoRa modulator [36] is implemented in an FPGA, leading to an easy hardware chirp generation method. But the demodulation process is usually much more complex than the modulation part, and there's currently no open-source FPGA-based LoRa demodulator. More importantly, the decoding process is not even touched in TinySDR. We can see that current works fail to provide a fully working implementation and complete understanding of LoRa PHY.

Fundamental Limitations of Current Works

1. The existing works cannot work at low SNR, so they lack this important advantage of LoRa.

2. The existing works are not able to provide full decoding capability for LoRa, and their performance is far lower than the commodity LoRa hardware. They are not able to support full LoRa functionalities. That is, LDRO mode, all SFs, and so on.

Our goal is to devise a full LoRa PHY implementation, including demodulation and decoding, to make extremely low SNR signals receivable and all parameters/modes of LoRa available.

DEMODULATION In this section, we disclose the demodulation process of LoRa and show how to break the fundamental limitations of existing LoRa implementations.

Phase-aligned Dechirping The LoRa demodulator converts chirp symbols to bit streams by first removing chirps from the received signal. This process can basically be divided into two main steps of dechirping. The first is that the received chirp is multiplied by a base down-chirp, and then the Fourier transformation is carried out on the multiplication results, which is translated to energy peaks in the frequency domain, with each chirp signal related to a multiple-reflection point. In principle, such multiplication will give a single-tone signal with continuous phase and fully defined group delay difference and whose energy may add up to only one peak, as shown in Figure 4(a3) (labeled IDEAL). However, in practice, there is an inevitable phase misalignment when the chirp frequency drops from maximum to minimum. This phase misalignment is basically brought by hardware instabilities of cheap LoRa devices and is random-distributed within 0 to 2π. Through phase misalignment, energy peaks by Fourier transformation are badly distorted, as shown in Figure 4(a4), hence limiting LoRa demodulation performance, especially under low SNR. While the misalignment issue between phases does not seriously hurt dechirping results, as has been proved in previous works, their performances degrade rapidly as the SNR condition becomes worse, well below the ideal LoRa demodulation sensitivity. The objective is to approach the ideal sensitivity of LoRa demodulation under phase misalignment. The design of our LoRa demodulator mainly comprises the following components.

Window Alignment So we need to align the demodulation window, sample points level, exactly on each symbol in order to keep all the energy from the symbol for demodulation. Window

alignment in conventional works is done by correlation. However, CFO will not make an accurate window alignment only with the help of preamble correlation. We utilize the SFD part in the LoRa packet for window alignment, while eliminating the impact of

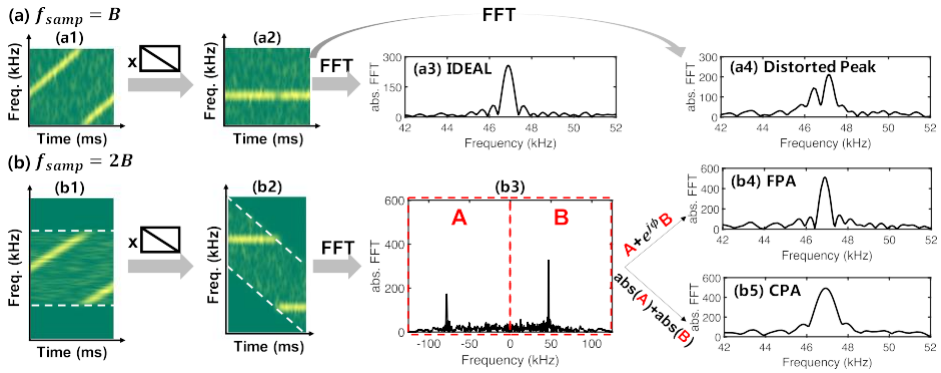

Fig. 4. The principal procedure of dechirping is multiplication of down-chirp and then FFT. (a) Conventional dechirping with SNR loss. For an ideal case, we get a peak like (a3). But for a non-ideal case, the frequency peak would be distorted like (a4), that incurs wrong demodulation results. (b) Proposed phase-aligned dechirping. We separate the two chirp segments within one LoRa symbol by using a low-pass filter and oversampling and then combine them. (b4) and (b5) give, respectively, the fine and coarse phase-aligned dechirping results that are all-conditions robust.

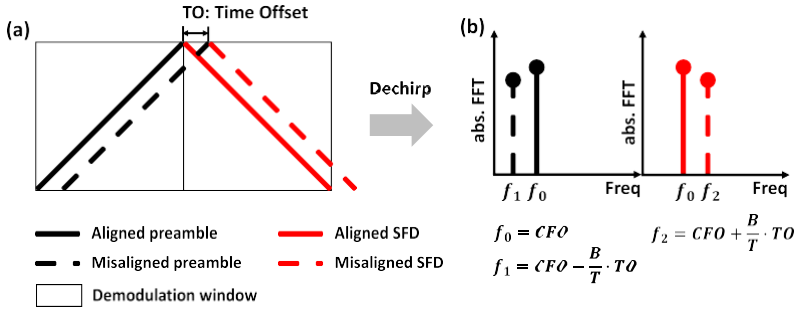

Fig. 5. Window alignment using base up-chirps and down-chirps.

CFO. Combining the up-chirps in the preamble with the down-chirps in SFD is the crucial operation. For both of these, we apply dechirping (multiplying the down-chirp by a base up-chirp). The up- and down-chirp peaks will emerge at the same frequency, independent of the CFO, if the demodulation window is exactly aligned with the signal, as illustrated in Figure 5. Otherwise, depending on the temporal offset between the chirp and the demodulation window, there is a noticeable change in the frequency of the two peaks. Consequently, by utilizing the frequency of peaks that correspond to the preamble and SFDs, we can attain precise alignment.

Peak merging and oversampling. As we previously disclosed, the primary cause of signal-to-noise ratio (SNR) loss in demodulation is the phase misalignment leading to peak distortion when the chirp frequency decreases from its maximum to minimum. We suggest an oversampling-based method that approaches the optimal sensitivity of LoRa demodulation and recovers the peak distortion in the presence of phase misalignment as a solution to this issue. LoRa modulates signals by cyclically altering the frequency of a base up-chirp, as seen in Section 2. Two chirp segments make up each modulated chirp: one with an initial frequency of f0 and the other with f0 B. The received chirp symbols are demodulated by existing works using a sampling frequency equivalent to the chirp bandwidth B.Thus, after multiplying by the base

Due to frequency aliasing, both down-chirp segments are converted into single tone signals of the same frequency. However, there is an unavoidable phase misalignment between the two chirp segments. When the Fourier transformation is applied to the entire symbol, the energy of these two chirp segments adds up destructively, resulting in peak distortion and SNR loss during LoRa demodulation. Existing efforts have no understanding of the importance of phase misalignment. As a result, they are unable to discern between the signals from these two chirp segments and fail to correct the peak distortion.

We present an oversampling-based technique for efficiently distinguishing the two chirp segments in the frequency domain. Specifically, because the beginning frequencies of the two chirp segments are f0 and f0 B, when the signal is oversampled with a frequency greater than 2B (i.e., Fs 2B), the dechirping produces two different peaks at f0 and Fs B + f0, as illustrated in Figure 4(b3). There is no phase mismatch within each chirp segment, hence both of these peaks are distortion-free. This allows us to precisely focus the energy of the entire chirp by coherently adding these two peaks of chirp segments, eliminating the influence of phase misalignment.

For the remainder of this section, we demonstrate how to efficiently integrate these two energy peaks in the frequency domain with little sensitivity loss for LoRa demodulation. Because of phase misalignment, the phases of the two peaks in Figure 4(b3) are out of sync. Because of the phase mismatch between these two complicated peaks, just putting them together will not raise the peak height or even cancel each other out. We present Fine-grained Phase Alignment (FPA), which searches for all conceivable values of phase misalignment (i×2π) and compensates it before adding two peaks. We repeat potential phase okffset and select the Δφ that produces the highest peak. The result of FPA is shown inFigure 4(b4). Compared with Figure 4(a3), FPA can approach the performance of IDEAL as thephase difference is compensated.

For the remainder of this section, we demonstrate how to efficiently integrate these two energy peaks in the frequency domain with little sensitivity loss for LoRa demodulation.
Because of phase misalignment, the phases of the two peaks in Figure 4(b3) are out of sync. Because of the phase mismatch between these two complicated peaks, just putting them together will not raise the peak height or even cancel each other out. We present Fine-grained Phase Alignment (FPA), which searches for all conceivable values of phase misalignment (i×2π) and compensates it before adding two peaks. We repeat potential phase okffset and select the Δφ that produces the highest peak. We observe that CPA dramatically alters the peak's shape, increasing the major lobe's width. Additionally, while adding up, CPA increases noise levels, which somewhat reduces demodulation efficacy as compared to FPA. We can alternate between FPA and CPA methods based on the receiver's processing capacity. The following presents a theoretical analysis of the symbol error rate (SER) for IDEAL, FPA, and CPA with regard to various SNRs. Assuming an additive white Gaussian noise (AWGN) channel, we simulate the noise as having a complex Gaussian distribution and broadcast the LoRa signal over it. Both the noise and the chirp symbols become energy peaks in the frequency domain during the dechirping process. Any time one of the noise peaks is higher than the intended symbol peak, a symbol error occurs. Assuming that the noise maximum peak is hn and the target symbol's peak height is hd, we can calculate the expected symbol error rate as

$$SER = P(h_d < h_n). \tag{2}$$

SER is a function of SNR theoretically. The calculation details of h_d and h_n for IDEAL, FPA, and CPA

are listed in Section 8. We plot the theoretical performance of IDEAL, FPA, and CPA under SF8/12 in

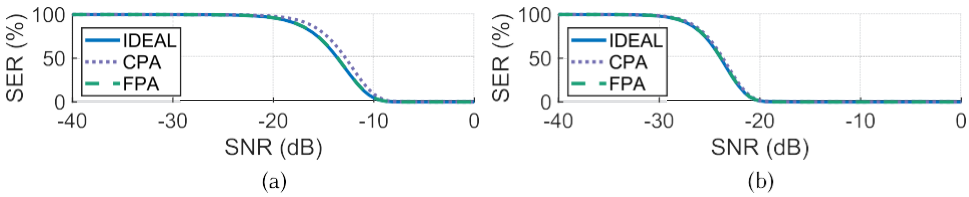

Fig. 6. Theoretical SER-SNR curve for (a) SF8 and (b) SF12.

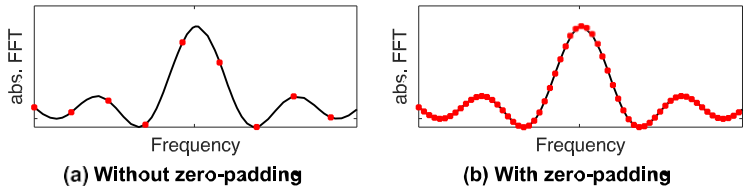

Fig. 7. Zero-padding effects. Red dots represent the DFT results after dechirping. Black lines represent the DTFT result after dechirping.

Figure 6.5 Empirically, the searching step length in FPA (i.e., 1/k) is set at 1/16 (a tradeoff between computing overhead and performance). While the theoretical SER of CPA in SF8 is marginally bigger than that of the other two approaches, we see that both FPA and CPA have performance very near to IDEAL for SF12. In conclusion, the theoretical analysis indicates that FPA and CPA are two useful and efficient techniques for LoRa demodulation; these conclusions are further supported by the findings of the experiment presented in Section 6. As Figure 6(b) illustrates, our theoretical research also demonstrates that LoRa could function at very low SNR (i.e., as low as 20 dB).

Maximal Development. We have already shown how to translate received chirps into frequency domain peaks. To retrieve the modulated data bits, we must then determine the maximum peak frequency. In actuality, though, peak height estimate is strongly tied to the frequency resolution following the Fourier transform. The ideal Discrete-Time Fourier Transform (DTFT) result on the target signal, which is a continuous function of frequency, is displayed as a black line in Figure 7(a). In actuality, the target signal is solely processed by the receiver using the Discrete Fourier Transform (DFT), a sampled version of DTFT in the frequency domain. Figure 7 shows the DFT output as red dots.

The resolution of the naïve DFT is low frequency. As a result, its outputs are unevenly distributed throughout the optimal DTFT curve, which affects peak estimation and lowers the resulting peak height. To achieve precise peak estimate, which is comparable to interpolation in the frequency domain, we zero-pad the signal in the time domain. As seen in Figure 7(b), zero-padding greatly enhances the DFT's frequency resolution, which is advantageous for our peak estimation. In Section 6.3, we demonstrate that fourfold zero-padding strikes a compromise between computation expense and demodulation sensitivity. Additional zero-padding has a negligible advantage.

1.1 Clock Recovery

Time Recuperation Because LoRa is frequently used on inexpensive devices, the oscillators that are equipped with it could not be very precise, leading to erroneous bin values. The first eight

dechirped peak bins of two packets from a LoRa device with a CFO of around 16 kHz are displayed in Table 2. We see that the peak bins' fractional component continues to wander over time. The fractional bins will accumulate and the integer bin will experience mistakes if we do not counteract that drift. The primary source of these bin drifts is the Sampling Frequency Offset (SFO). The real LoRa symbol is τ shorter (or longer) than the conventional symbol period T because of the variation in sampling frequency. Consequently, there is extra drift at the start frequency of the subsequent symbol as

[5]Bit-level parameters such as CR and CRC are irrelevant to SER in symbol level. It should be noted that bandwidth does not appear in the above SER formula, because SNR is a function of bandwidth (if we increase the bandwidth without raising the transmitting power, then the in-band SNR will drop).

Table 2. Peak Bin Drift under SF10 and SF12 (with Bandwidth 125 kHz)

		First 8 Peak Bins							
SF10 (LDRO off)	Expected	677.0	333.0	861.0	93.0	853.0	149.0	789.0	597.0
	Received	677.2	333.2	861.3	93.3	853.4	149.4	789.4	597.5
SF12 (LDRO on)	Expected	1757.0	3453.0	673.0	3757.0	2409.0	809.0	2981.0	2545.0
		=	=	=	=	=	=	=	=
		011011	110101	001010	111010	100101	001100	101110	100111
		011101	111101	100001	101101	101001	101001	100101	110001
	Received	1757.8	3453.9	674.0	3758.2	2410.3	810.4	2982.5	2546.6
		=	=	=	=	=	=	=	=
		011011	110101	001010	111010	100101	001100	101110	100111
		011101	111101	100010	101110	101010	101010	100110	110010
		+0.8	+0.9	+0.0	+0.2	+0.3	+0.4	+0.5	+0.6

Expected: the ideal bins without clock error; received: the actual received bins.

$$\Delta f = \frac{B}{T} \cdot \tau. \tag{3}$$

Because the carrier frequency and sampling frequency are generated by the same oscillator, we have

$$\frac{\tau}{T} = \frac{SFO}{f_{samp}} = \frac{\Delta f_{osc}}{f_{osc}} = \frac{CFO}{f_{RF}}, \tag{4}$$

where f_{RF} is reference carrier frequency, f_{osc} is reference oscillator frequency, and Δf_{osc} is oscillator frequency bias. After translating to bin values, the estimated bin drift between two consecutive symbols is[6]

$$\Delta bin = \frac{\Delta f}{B/2^{SF}} = \frac{CFO}{f_{RF}} \cdot 2^{SF}. \tag{5}$$

For example, when CFO= 16 kHz, f_{RF} = 470 MHz, and SF = 10, the estimated bin drift $\Delta bin \approx 0.035$, which matches the tendency in Table 2. We subtract accumulated Δbin from the peak bins before decoding.

Here, the LDRO mode is an additional factor to take into account. Generally, LoRa chips automatically enable LDRO when a chirp period is too long (i.e., T > 16 ms) [22]. In LDRO mode, as seen in the bottom of Table 2, the predicted bin values always assume the form of 4n + 1, which means the two Least Significant Bits (LSBs) do not encode data. Given that longer packets include more susceptible lower bits, this approach seeks to better safeguard the contents. The two LSBs are stable enough to encode data (e.g., SF10 and bandwidth 125 kHz) for short-duration LoRa transmissions. However, we discover that the first eight symbols in the experiment are always in LDRO mode in order to safeguard the header data.(Section 4).

[6]Without zero-padding, one integer bin in LoRa always represents a frequency of $-\frac{B}{2^{SF}}$ regardless of the number of FFT points or sampling frequency.

2 DECODING

2.1 Overview

This decoding part is not a rehash of earlier research. Our goal is to offer a verifiable inference on the configurations, decoding order, and packet structure of LoRa. Equation (6) for determining the number of symbols in a packet is known to us from official LoRa datasheets [22, 25, 37] and patents [26]. The packet structure, on the other hand, is unknown. Additionally, we are aware that there are four primary steps involved in decoding: de-whitening (W), deinterleaving (I), gray coding (G), and Hamming decoding (H). On the other hand, neither the process sequence nor the configuration parameters of individual processes are known. We investigate the LoRa decoding by three phases to demonstrate how LoRa PHY transforms the bits from the demodulation module into meaningful packets. First, we use the number of symbols formula to deduce the packet structure from the black-box LoRa. We then disclose the hierarchy of the four primary decoding procedures. Lastly, we deduce each decoding process' setup settings.

2.2 Packet Structure Inference

The official document [22] provides a formula to calculate the number of symbols in a packet:

$$n_{sym} = 8 + \max\left\{\left\lceil \frac{2PL + 4CRC - 5IH - SF + 7}{SF - 2DE} \right\rceil \cdot \frac{4}{CR}, 0\right\}, \tag{6}$$

where PL is the number of payload bytes, SF is spreading factor, CRC is 1 if CRC check is enabled and otherwise 0, IH is 1 if implicit header mode (without header) is enabled and 0 if explicit mode (with header) is enabled, and DE is 1 if LDRO is enabled and otherwise 0.

We can derive the following *properties* from Equation (6):

- A LoRa packet has at least eight data symbols as $n_{sym} \geq 8$.
- $2PL + 4CRC$: The n_{sym} of a packet with PL payload bytes and CRC check enabled is equal to the n_{sym} of a packet with $PL + 2$ payload bytes and CRC check disabled. We can infer that the CRC check takes 2 bytes in the packet.
- $2PL + 4CRC - 5IH$: Similarly to CRC, we can infer the header takes 2.5 bytes.
- $SF - 2DE$: SF is the number of bits in a data symbol. Enabling LDRO causes the reduction of two bits per data symbol.
- $\frac{4}{CR}$: The number of data symbols increase with the unit of $\frac{4}{CR}$ (CR=$\frac{4}{5}$, $\frac{4}{6}$, $\frac{4}{7}$, or $\frac{4}{8}$). For example, $CR = \frac{4}{7}$ means the packet applies (7,4) Hamming code, and n_{sym} has the form $\frac{4}{7}n + 8 = 7n + 8 (n = 0, 1, 2, \dots)$.

Based on the properties, we then manipulate the data packets to infer the packet structure. We first gradually increase packet size and observe a stairlike increase of the number of data symbols in real signals. If $CR = \frac{4}{7}$, then the number of symbols n_{sym} would increase following an arithmetic sequence (i.e., $8, 15, 22, \dots$). We also observe that when continually increasing the packet size, the newly added bytes are encoded in the last $\frac{4}{CR}$ symbols, and the first $\frac{4}{CR}(n-1) + 8$ symbols do not change. It reveals that data bytes are encoded by a $\frac{4}{CR}$ symbols block.

Further, we find that the first eight symbols are specially treated. Whether in explicit or implicit header mode, the first eight symbols' values always appear in the form of $4k + 1 (k = 0, 1, 2, \dots)$, which means the last two bits of data are discarded, and there are $SF - 2$ bits data in each symbol. We can infer that the first eight symbols are in LDRO mode, i.e., $DE = 1$. LDRO gives better protection, and the header may be in the first eight symbols. We know that LoRa uses coding rate CR to protect the payload, and the payload bytes are encoded by a $\frac{4}{CR}$ symbols block. Similarly, the first eight symbols form an eight symbols block, and we guess that it uses a coding rate $\frac{4}{8}$ to give the header the highest protection (4 parity bits for a nibble). We also observe that the first eight

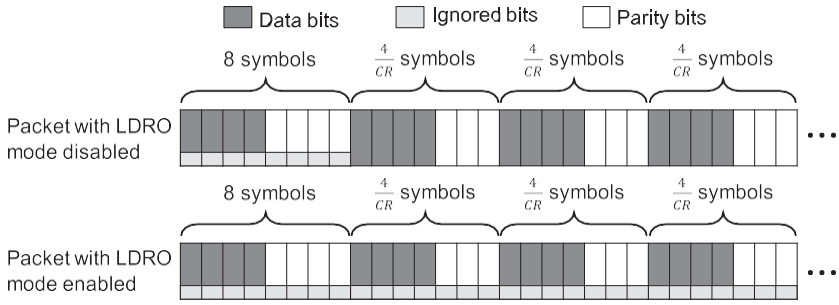

Fig. 8. The inferred LoRa packet structure for LDRO mode disabled/enabled. When LDRO mode is disabled, only the first eight symbols have ignored bits (protecting header information). When LDRO mode is enabled, the least significant two bits are ignored for all the symbols.

symbols encode payload bits in implicit header mode. Such a design can reduce the complexity of the decoding hardware complexity, because the explicit and implicit header mode can now be processed with the same procedure.

Figure 8 shows the inferred packet structure of a packet with LDRO mode disabled/enabled. In LDRO mode, the last two bits in each symbol are discarded. No matter whether the LDRO mode is enabled or disabled, the first eight symbols are in LDRO mode and use coding rate $\frac{4}{8}$. The following symbols are in blocks containing $\frac{4}{CR}$ symbols. The first four symbols encode data bits while the rest symbols encode parity bits.

Packet Structure Verification. We verify the above packet structure by comparing Equation (6) and the calculated symbol number using the inferred packet structure. For $SF \geq 7$, the first eight symbols contain $8 \cdot (SF - 2) \cdot \frac{4}{8} = 4SF - 8 \geq 20$ data bits. Therefore, the first eight symbols could always include all header information. Except the header, the first eight symbols can contain $4(SF - 2) - 20(1 - IH) = 4SF + 20IH - 28$ bits payload. Suppose there are PL bytes ($8PL$ bits) payload in total. If CRC is enabled, then additional 16 bits are needed. Except the first eight symbols, the total number of required data bits for the rest symbols is $8PL+16CRC-20IH-4SF+28$. Whitening and Gray decoding do not affect the total number of bits and the total number of symbols. The Interleaving operation only requires us to pad redundant bytes when the rest payload bytes cannot fill a block, thus not affecting the total number of symbol blocks. Each block has $\frac{4}{CR}$ symbols, and each symbol in the block could encode $SF - 2DE$ bits. Due to the existence of parity bits, only $4(SF - 2DE)$ bits in a block are actual data bits. Then the total number of symbols in the packet is

$$8 + \left\lceil \frac{8PL + 16CRC - 20IH - 4SF + 28}{4(SF - 2DE)} \right\rceil \cdot \frac{4}{CR}. \tag{7}$$

It is obvious that formula (7) is the same as Equation (6), which means the inferred packet structure is correct.

2.3 Order of Decoding Operations

In this part, we attempt to determine the sequence of the four major LoRa decoding processes: gray coding (G), deinterleaving (I), Hamming decoding (H), and dewhitening (W). It is clear that gray coding should be the initial step because it aims to tackle the symbol neighboring drift problem. Hamming decoding uses bytes, but LoRa symbols include SF - 2DE bits. As a result, it relies on the modified bytes stream obtained after deinterleaving. Therefore, the order of "G, I, H" should be "G→I→H." As a result, dewhitening has three alternative places, namely: after

"g," "i," or "h." The implementations of BR, RPP0, and TAPP take three distinct dewhitening positions.

We show that the position of the dewhitening does not effect the decoding outcomes. Because the encoding process order is the inverse of the decoding order, we utilize the operation in encoding to demonstrate the impact of the location of "W." Consider the data as a bit vector D; interleaving is an operation that rearranges the location of bits. We may see interleaving as a matrix I, with each row or column holding only one "1." Hamming coding, a linear coding technique, may be represented as a matrix H. H D = [P D D], where P represents the parity matrix. Whitening is defined as W ⊕ D, where W is a random bit vector that is exclusive-or (XOR). Following are the three possible orders of decoding and the corresponding encoding operations:

$$\text{“W}\rightarrow\text{I}\rightarrow\text{H”:}\quad W_1 \oplus (I \cdot [P \cdot D\ D]), \tag{8a}$$

$$\text{“I}\rightarrow\text{W}\rightarrow\text{H”:}\quad I \cdot (W_2 \oplus [P \cdot D\ D]), \tag{8b}$$

$$\text{“I}\rightarrow\text{H}\rightarrow\text{W”:}\quad I \cdot [P \cdot (W_3 \oplus D)\ (W_3 \oplus D)], \tag{8c}$$

where W_1, W_2, and W_3 are three different random bit vectors. Formula (8b) can be rewritten as

$$(I \cdot W_2) \oplus (I \cdot [P \cdot D\ D]). \tag{9}$$

Therefore, formula (8a) and formula (9) are equivalent (let $W_1 = I \cdot W_2$). Formula (8c) can be rewritten as

$$I \cdot ([P \cdot W_3\ W_3] \oplus [P \cdot D\ D]). \tag{10}$$

Formula (10) is a special case of formula (8b) (let $W_2 = [P.\ W_3\ W_3]$). Therefore, both “I→W→H” and “I→H→W” can be represented by “W→I→H,” which means the position of “W” does not affect the final decoding results. We will show the correct position for “W” in Section 4.4.

For ease of analysis, we assume the decoding order is "G W I H." Because the even parity of all-zeros is zero, and interleaving in LoRa is simply a diagonal realignment of all bits, the output codewords following "H" and "I" are zeros when the input bits are all zero. We assume that LoRa uses the generally used even-parity check, and the end findings confirm that this assumption is valid. "W" represents the XOR of data values with a pseudo-random sequence. When the decoding order is "G→W→I→H," as x 0 = x, and we set all transmission bits to zeros, the output values following "G" are the dewhitening sequence. Then, we could use the derived dewhitening sequence to recover the temporal results before “I” and “H” for further analysis.

2.4 Configuration of Each Operation

This section describes the critical settings for the four processes in LoRa decoding: gray coding, deinterleaving, Hamming decoding, and dewhitening. We demonstrate how to uncover the header structure and CRC polynomial.

The use of gray codes. In many wireless communication systems, gray coding—a translation from a bit vector to a binary representation—is extensively utilized. The Gray coding representations next to each other differ by just one bit. It is more likely to occur in wireless communication networks that we will mistake a symbol for its neighboring symbol than for a different, random sign. For instance, as discussed in Section 3.2, the neighboring bin drift is typical in LoRa modulation. Gray coding reduces the bit error brought on by neighboring misidentification to one bit per symbol, which increases the likelihood that the error correction process will fix it. The standard Gray coding can be expressed as

$$v = v_0 \oplus (v_0 >> 1), \tag{11}$$

where v_0 represents the raw demodulated bin value, >> is the bitwise right shift operation, and v is the Gray coding output. However, when we continue our analysis based on natural symbol

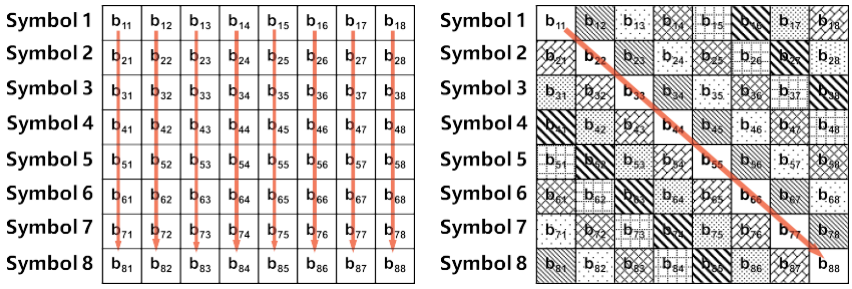

Fig. 9. An example deinterleaving block for 8 SF8 symbols ($CR = \frac{4}{8}$). Each row in the block is an 8-bit representation of a LoRa symbol after Gray coding. $b_{i,j}$ represents the jth bit of the ith symbol. For example, $b_{i,1}$ is the LSB of the symbol i. (a) Column-major order row-line interleaving, the nth codeword is composed of bits $b_{i,n}(i = 1, \ldots, 8)$. (b) Diagonal interleaving used in LoRa, the nth codeword is composed of bits $b_{i,(i+n-1)\%8+1}$ $(i = 1, \ldots, 8)$.

mapping and conventional Gray coding, we are unable to successfully decode packets 100% of the time. This issue affects Project RPP0 and there is still a defect that has to be solved.7. To get the rationale, let us revisit LoRa's LDRO mode. In order to lessen the impact of bin drift, the encoder adds "1" after placing data bytes into high SF 2 bits of a symbol when LDRO is enabled. From a hardware point of view, adding "1" in any scenario makes sense since it allows us to avoid using circuit resources to determine whether LDRO is enabled.

We assume that the encoder produces symbols with a single bin shift. If this is the case, we should deduct "1" from the demodulation findings before applying Gray coding. Thankfully, the decoding findings confirm our predictions. The "G" in LoRa decoding procedure might be changed as

$$v = (v_0 - 1) \oplus ((v_0 - 1) >> 1). \tag{12}$$

Observe that TAPP performs comparable techniques to Gray coding as we do, however they describe it as a distinct Gray coding process and generate the additional "one" via a brute-force algorithm. Nevertheless, using a non-standard Gray coding appears out of place. Our explanation, however, makes more sense.

disentanglement. Burst mistakes are lessened by the use of interleaving. Errors might be spread across several bit groups by interleaving and fixed via forward error correction (FEC). According to [26], LoRa uses diagonal interleaving rather than the more common row-line interleaving. The eight symbols' column-major order row-line interleaving is seen in Figure 9(a). The LSBs (bi,1) of the eight symbols are put together into a byte for row-line interleaving. We know from Section 3 that a symbol's least significant bits (LSBs) are more brittle than its most significant bits (MSBs). It is a poor design from the FEC's point of view to combine delicate parts. LoRa uses diagonal interleaving, as seen in Figure 9(b). The more reliable diagonal interleaving divides the brittle LSBs into separate bytes. After that, we work with the sent packets to extract the intricate diagonal mapping. As an illustration, we transmit packets in implicit header mode with SF = 8 and CR = 4. As seen in Figure 9(b), the interleaving block is therefore an 8 8 block. First, we use the usual (7, 4) Hamming code with a single bit extension to be the FEC used in CR = 4. As a result, the codeword for nibble "0000" is "00000000," while the codeword for "1111" is "11111111," following "G" and "W."8 Assuming that the four transmitting bytes are 0x0F and the other bytes are all zeros, we note

[7]https://github.com/rpp0/gr-lora/issues/99.

[8]Note that here we have removed the influence of whitening.

In Figure 9, b11 = b22 = b88 = 1. As a result, the fourth byte, 0x0F, is represented by the major diagonal. We cannot always obtain eight ones in a block if we use the normal Gray coding directly, as demonstrated by the one bin shift problem discussed in Gray coding. This issue may be resolved by shifting the mapping by one, which also exactly aligns with our subsequent decoding procedure. The complete mapping for interleaving may be obtained by altering the all-1 data bits in the sent packet, as seen in Figure 9(b). The deinterleaving technique is identical for other factors. The sole distinction is that

the block size becomes $\frac{4}{CR} \times (SF - 2DE)$. As a result, we summarize the deinterleaving process as

$$c_{i,j} = b_{j,(i+j-1)\%(SF-2DE)+1}, \tag{13}$$

where $c_{i,j}$ is the jth bit of the ith codeword after deinterleaving, $b_{j,i}$ is the ith bit of the jth symbol after Gray coding, and $i \in \{1, 2, \ldots, \frac{4}{CR}\}$, $j \in \{1, 2, \ldots, SF-2DE\}$.

Hamming Decoding. After deinterleaving, we get the encoded data in the form of codewords, but the position of data bits and parity bits in a codeword are still unknown. LoRa provides four valid CR values ($\frac{4}{5}$, $\frac{4}{6}$, $\frac{4}{7}$, and $\frac{4}{8}$), which determine the codeword length and thus FEC strength. When CR is set to $\frac{4}{7}$ or $\frac{4}{8}$, LoRa applies Hamming(7, 4) code or extended Hamming code with one additional parity bit. When CR is set to $\frac{4}{5}$ or $\frac{4}{6}$, LoRa can detect bit errors but cannot correct them. First, we assume that a nibble with code rate $\frac{4}{8}$ is protected by the standard Hamming(8, 4) code. However, the findings of our final study indicate that certain adjustments are needed to this assumption. We could change the transmitting bytes while keeping one particular bit constant in order to ascertain the location of each data/parity bit in the codeword. For instance, we set the transmitting bytes to 0x01, 0x03, 0x05, and 0x0F.9 in order to examine the location of the LSB of the fourth byte. Next, our target, or in this case, the LSB, is the bit in the interleaving block that is always set to 1. In contrast to the typical Hamming(8, 4) code p1p2d1p3d2d3d4p4, where di is the ith data bit and pi is the ith parity bit, we discover that the four LSBs in the codeword are data bits and the four MSBs are parity bits. The relationship between data bits and parity bits in a conventional Hamming(8, 4) coding is represented by equation set (14)

,

$$\begin{aligned} p_1 &= d_1 \oplus d_2 \oplus d_4 \\ p_2 &= d_1 \oplus d_3 \oplus d_4 \\ p_3 &= d_2 \oplus d_3 \oplus d_4 \\ p_4 &= d_1 \oplus d_2 \oplus d_3 \oplus d_4. \end{aligned} \tag{14}$$

The four LSBs are referred to as d1, d2, d3, and d4 in that order. We change the data bits to maintain pi = 1 in order to determine which bit in the MSBs is pi. When choosing codewords with d1 d2 d4 = 1, for instance, the parity bit p1 is always equal to "1." In a similar manner, p2 and p3's locations are determined. But the definition of p4 does not suit the remaining parity bit. Upon close examination, we discover that it is a parity encompassing d1, d2, and d3, that is to say

.,

$$p_5 = d_1 \oplus d_2 \oplus d_3. \tag{15}$$

For other code rates, we can derive the bit position similarly. When $CR = \frac{4}{7}$ is used, parity bit p_1 is abandoned. When $CR = \frac{4}{6}$ is used, parity bits p_1 and p_2 are abandoned. When $CR = \frac{4}{5}$ is used, there is only one parity bit and it is natural to use p_4 to cover all bits. The conclusion does not change when SF varies. Figure 10 shows the bit positions for different code rates. Note that LoRa applies non-standard Hamming code, the naming number of parity is arbitrary and our naming is just one kind of them.

Dewhitening. In Section 4.3, we assume the dewhitening operation happens after "G" and we prove that the dewhitening position does not affect the final results. We here discuss the "correct" position for dewhitening. The process of deinterleaving and Hamming decoding has been

[9]Note that we cannot directly control parity bits.

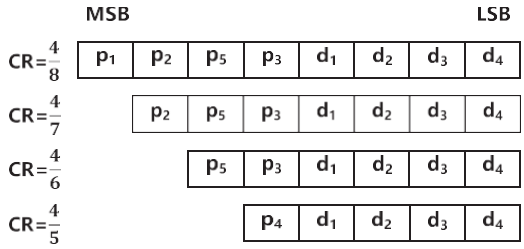

Fig. 10. Parity bits and data bits position in a codeword with CR = $\frac{4}{8}, \frac{4}{7}, \frac{4}{6}, \frac{4}{5}$.

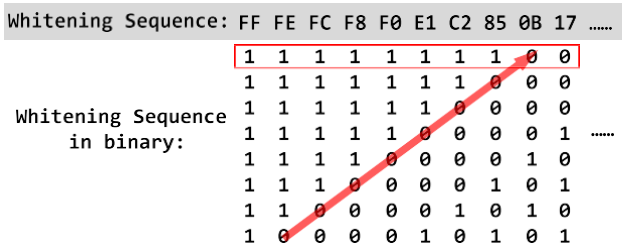

Fig. 11. The whitening sequence of "G→I→H→W."

understood as stated above. The "W" location is moved to the other two positions, and all-zero bytes are sent in order to determine the whitening sequence for each order option. We discover that while the whitening sequences of "G I W H" and "G I H W" remain the same, "G W I H" provides distinct whitening sequences for varied combinations of SF and CR. It makes more sense to use the same sequence for every packet. So, we start by excluding "G W I H." Next, we must select the appropriate sequence from "G I W H" and "G I H W." According to the LoRa chip datasheet [22], a Linear Feedback Shift Register (LFSR) generates the whitening sequence in FSK mode. We assume that an LFSR that generates the whitening sequence for LoRa mode also exists. on get the matching LFSR, we use the Berlekamp–Massey algorithm [38] on the whitening sequences of the two decoding orders. No matter whatever order we place the bits in for "G I W H" (LSB first, MSB first, etc.), the smallest LFSR size that the Berlekamp–Massey algorithm yields is at least 64. However, we are aware that an n-bit LFSR can always be used to generate a 2n-bit sequence. An output of a 64-bit LFSR might represent any sequence with 128 bits. The "G I W H" sequence's LFSR is too lengthy. Examining the "G I H W" whitening sequence in detail in Figure 11 reveals that it differs from ordinary random bits in that each byte appears to represent a state of an LFSR. In order to execute the Berlekamp–Massey algorithm, we gather the MSBs of the sequence bytes, as indicated by the red box in Figure 11. For deriving such a sequence, the LFSR polynomial is x8 + x6 + x5 + x4 + 1. The literature [39] shows that it is the maximal-length polynomial for the shift-register with eight bits. The LFSR's structure is seen in Figure 12, where we can observe that each of the register's eight bits is utilized to whiten a single bit of data. These bits together make up a byte of the whitening sequence. We believe that the whitening sequence of "G I H W" is in the right order since it may be produced by an LFSR that is substantially shorter.

.

2.5 CRC

The raw LoRa PHY packets are displayed following four stages of processing: dewhitening, deinterleaving, Hamming coding, and Gray coding. In earlier sections, we turned off CRC checking and sent packets in implicit header mode. Next, we attempt to examine the LoRa

payload's CRC method. According to what we saw in Section 4.2, a packet's CRC checksum takes up 16 bits at the end. In order to verify that CRC is covered, we first send packets with the identical data content that have both explicit and implicit headers. The findings indicate that the CRC is only carried out on the payload.

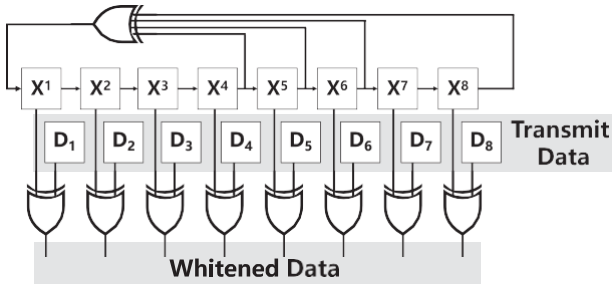

Fig. 12. LFSR $x^8 + x^6 + x^5 + x^4 + 1$ used in LoRa.

In theory, computing the CRC checksum of a set of bits involves treating the set as a big binary integer and figuring out the residual in relation to a "divisor." Typically, the divisor is expressed as a polynomial based on the Galois field of two elements, GF (2), such as x3 + x + 1. Determine the polynomial that is utilized by analyzing the CRC portion of the LoRa PHY. One characteristic of CRC is that in the case of a degree n polynomial, the remainder equals the polynomial itself if the initial dividend was 1. The CRC checksum in the final two bytes of a packet that is constructed solely with the last bit equal to 1 indicates the precise polynomial that is being utilized. Meanwhile, it is impossible to choose the CRC polynomial at random; It needs to adhere to certain guidelines in order to meet certain standards. Consequently, selecting the polynomial from the usual CRC polynomials makes sense. It requires minimal work to identify the actual polynomial utilized by comparing the polynomial we derive with the standard polynomial sets. Based on the analysis above, we set the final two bytes of the payload to 0x0001, 0x0080, 0x0100, and 0x8000, respectively, and activate the payload CRC in implicit header mode (other bytes are set to zero). We evaluate the four potential combinations since we do not know the endian and LSB-MSB order in CRC hardware implementations. Surprisingly, though, the outcome exceeds our expectations. The CRC checksum and the final two payload bytes match identically, regardless of what we set in those final two bytes. In a typical CRC implementation, n zero bits are padded after data to guarantee that the final n bits are also fully protected by CRC before performing the remainder computation.10 The situation we saw indicates that LoRa PHY does not use the zero-padding step. In order to obtain the polynomial, we transmit 0x0001, 0x0080, 0x0100, and 0x8000 at the third and fourth bytes from last. The CRC bytes that were received are 0x1021, 0x9188, 0x3331, and 0x1B98, in that order. The polynomial CCITT-16, which is x16 + x12 + x5 + 1, is denoted by the symbol 0x1021. Using this polynomial, we apply CRC check and test packets with random bytes. All of the samples' CRC bytes match the CRC checksums that our implementation computes. Owing to the nature of CRC, even with a tiny sample size, it is difficult for an incorrect CRC implementation to have the same checksum as the right one, proving the accuracy of our conclusion.

.

2.6 Header

The header of the LoRa PHY is the leftmost unknown component. We already know that the header in explicit mode consists of 20 bits from Section 4.2. Finding the 20 bits' purpose and arrangement is our aim. Because the maximum payload length is 255, Payload Length (PL), which is supposed to be encoded in the header, occupies at least 8 bits. We attempt to decode the packet in implicit header mode by varying the payload length and assuming that the header has

[10]Though the code level implementation may not explicitly contain the zero-padding step [40].

likewise been whitened. Sadly, in implicit header mode, we are unable to get the data bytes.

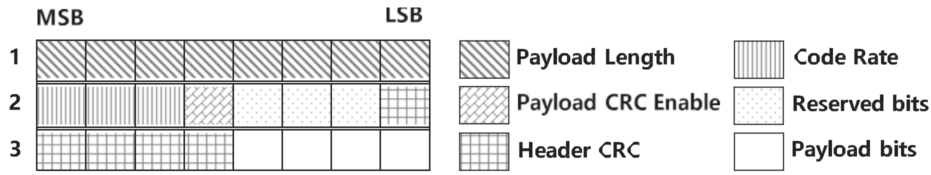

Fig. 13. Header structure of LoRa in the first three bytes.

The whitening sequence utilized for the header is the source of the issue. Nevertheless, we may determine PL's location by looking at the bits that vary with payload length. Keep in mind that when the packet payload length varies, just the header CRC and PL bits in the first 2.5 bytes will change. We note that there may be PL bits in places 1–8 and 16–20. The intermediate findings also demonstrate that the header is not whitened and that the first byte of a header is precisely PL before to dewhitening. There are only 255 potential whitening bytes that our adapted LFSR may produce. For header dewhitening, there are no more whitening bytes available. Therefore, it makes sense to leave the packet header unwhitened. Our subsequent experiments on identifying additional bits validate this hypothesis. As a result, in the following, we do not apply the dewhitening procedure on the header. The coding rate and payload-CRC-enable bit positions are computed in a similar way when one parameter is changed and others are fixed. For CR = 4, 4, 4, 4, 4, the coding rate bits are 001, 010, 011, 100, respectively. If payload CRC is enabled, the payload-CRC-enable bit is 5 6 7 8 set to 1. Five bits quickly changed in our trials when we varied the conditions. These five bits are assumed to be the header CRC. We refer to the three bits that remain 0 regardless of the parameter value as reserved bits. The header structure in the first three bytes is shown in Figure 13. Using the procedure described in Section 4.5, we were unable to discover a typical CRC5 polynomial that would meet the header CRC findings. However, because the CRC process is linear, we might describe the CRC computation equivalently by using a CRC matrix.11 We represent a bit vector v1 with 12 parameter bits (PL, CR, and payload-CRC-enable bit). The five header CRC bits are represented by a bit vector v2. Finding a matrix M that satisfies v2 = M v1 is our goal. Given that we have control over v1, we may create a sequence of v1, such as v (1), v (2),..., v (12). The result is a matrix V1. The corresponding v2 series 1^1~1 is made up of v (1), v (2),..., v (12). They create matrix V2. Consequently, V2 = M ·V1. 2^2^2 M = V2 if V1 is an identity matrix. How can one create a V1 identification matrix? Although the content of the header is under our control, we are unable to precisely regulate the status of individual bits. Sending a packet with v1 that contains only a single "1" appears to be impossible in our situation. To our astonishment, though, the LoRa chip can also send a packet with zero payload length. At that point, all eight PL bits are zeros. The single coding rate bit in v1 is 1 if we deactivate payload CRC and set CR = 4 or CR = 4 or CR = 4 [illegible]

packet B does not. Then the corresponding bit vectors are $v_1^A = 000000000011, v_2^A = 01100, v_1^B = 000000000010, v_2^B = 00111$. Therefore,

$$
\begin{aligned}
M \cdot (000000000001) &= M \cdot (v_1^A \oplus v_1^B) \\
&= M \cdot v_1^A \oplus M \cdot v_1^B \\
&= v_2^A \oplus v_2^B \\
&= 01011.
\end{aligned} \tag{16}
$$

[11]Here we ignore the reserved bits.

[10]Though the code level implementation may not explicitly contain the zero-padding step [40].

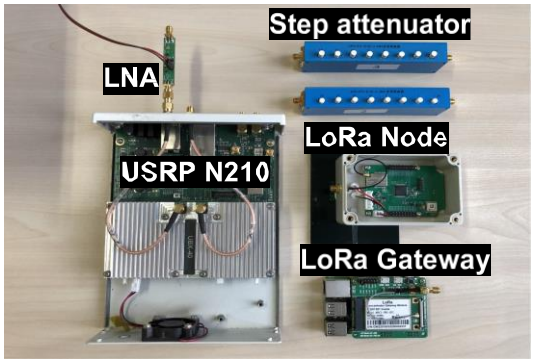

Fig. 14. Devices in our experiments: a USRP N210 SDR device with additional LNA, step aNenuators for measuring sensitivity in wired environments, commodity LoRa nodes, and a commodity LoRa gateway.

A similar process can be applied for PL bits. Finally, we summarize the header CRC calculation as $v_2 = M \cdot v_1$, where

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \quad (17)$$

3 IMPLEMENTATION

We use the GNU Radio software-defined radio platform to implement the full LoRa PHY in C++. Additionally, we offer a MATLAB version code for simulation needs. For the receiver, we use two distinct blocks: the demodulator and the decoder (for the transmitter, we use the modulator and the encoder). Therefore, our solution may be extended in the future to accommodate other demodulation algorithms without requiring modifications to the decoder block. For instance, by substituting a collision demodulator for the LoRa demodulator, we may decode collision packets straight away without altering the decoding block logic. Every experiment in Section 6 is carried out in real time on the USRP N210; the raw baseband signals are not saved for offline processing. A laptop running Ubuntu 19.10 with an i5-7200U CPU and 8 G of RAM is linked to the USRP. The performance of our built LoRa PHY is assessed using commodity LoRa devices, such as Raspberry-Pi 3B+ LoRa gateways (RPI) with SX1301 [37] and LoRa end nodes with SX1268 [25]/SX1278 [22]. The tools we employed in our research are depicted in Figure 14. In wired experiments, the step attenuator is used to test sensitivity.

.

4 EVALUATION

4.1 Decoding Success Rate

We evaluate if our constructed LoRa PHY is capable of decoding the LoRa packets sent by the commercial LoRa devices in order to confirm its efficacy. We set up a generic LoRa device to continuously send sporadic packets to our LoRa receiver. Through the serial interface, the transferred bytes' ground truth is recorded. We configure the LoRa transmitter with many parameters for a thorough assessment, such as six SFs (7 ∼ 12), four CRs (4, 4, 4, 4), two header modes (explicit/implicit), and with/without CRC, i.e., six ×124 × 2 × 2 = 96 choices. We employ a bandwidth of 250 kHz and a carrier frequency of 475 MHz

. The payload length is set to

[12]Carrier frequency and bandwidth do not affect the decoding logic.

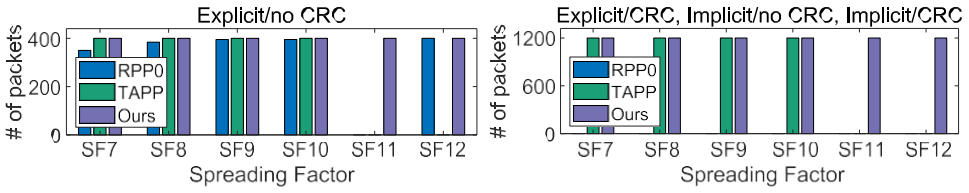

Fig. 15. Decoding success rate test. A total of 9,600 packets are sent, and all code rates ($\frac{4}{5}, \frac{4}{6}, \frac{4}{7}, \frac{4}{8}$) are tested. RPP0 cannot decode any SF11 packets. TAPP cannot decode any SF11/12 packets. Our decoder could decode all packets. (a) Number of correctly received packets of three implementations in explicit header mode with CRC off. (b) RPP0 does not support the configuration of {explicit header mode with CRC on, implicit header mode with CRC off, implicit header mode with CRC on} and therefore receives none.

sixteen bytes. The broadcast is repeated one hundred times for every arrangement. Since the LoRa transmitter is positioned near to the USRP receiver, all signals are picked up with high signal-to-noise ratios. Since everything is perfect for communication, every missing packet indicates that the implementation is not complete.Figure 15 displays the experiment's outcome. RPP0 only supports the explicit header, while BR only functions at SF8 with the implicit header among all configurations. These two approaches don't support CRC. Because BR and TAPP's decoding procedures are computationally costly, they are unable to handle huge packet lengths at high transmission rates. In particular, TAPP begins to lose packets when the transmitter duty cycle exceeds 0.5 and BR is unable to decode any packets larger than 5 bytes. Owing to all of these restrictions, BR's total PRR is just 4.2%. Furthermore, we observe that RPP0 is unable to resolve SF11 packets and that TAPP is unable to decode any SF11/SF12 packets. In conclusion, TAPP covers 6,400/9,600 66.7% of the packets, whereas RPP0 covers 1,924/9,600 20.0% of the packets. Quite the opposite—our decoder is capable of decoding every packet in every LoRa setup.

Sensitivity

We check our LoRa decoder's sensitivity in this step. We use a 20-meter RF wire with two step attenuators to link a commercial LoRa transmitter to our decoder. Therefore, by varying the values of these two step attenuators, we can precisely regulate the attenuation of the signal.Because wired connections, such as RF cables and connectors, can impart attenuation and create mistakes in sensitivity calculation, using the step attenuators' values as the channel attenuation is not accurate. Therefore, we carefully measure and calibrate the link attenuation before to the experiment using a spectrum analyzer, Keysight N9322C. Then, in order to prevent wireless channel leakage, we lowered the transmitting power to its lowest setting, 7.9 dBm.

We test the SF8 arrangement with a 250 kHz bandwidth. For the remainder of this section, we do not compare BR's performance because it is unable to decode packets larger than four bytes. The sensitivity findings of four SDR decoders are displayed in Figure 16(a). When the payload is 32 bytes, the least RSSI that achieves PRR > 90% is the definition of the sensitivity condition in LoRa. RPP0 and TAPP's PRR rapidly decreases when the received signal's RSSI decreases since they do not use the LoRa characteristics for demodulation. RPP0 and TAPP have sensitivity values of 106 and 108 dBm, respectively. On the other hand, our decoder's performance stays consistent and it may reach a sensitivity of just 126 dBm.

We also do an experiment with a bigger SF, or SF12. The outcome is displayed in Figure 16(b). Our measured sensitivity of 142 dBm is close to LoRa's ideal sensitivity. It's interesting to note that the RPI gateway's sensitivity is just 139.5 dBm. This is a result of the RPI gateway not using the LoRa gateway's ideal design. Consequently, our decoder performs better than it.

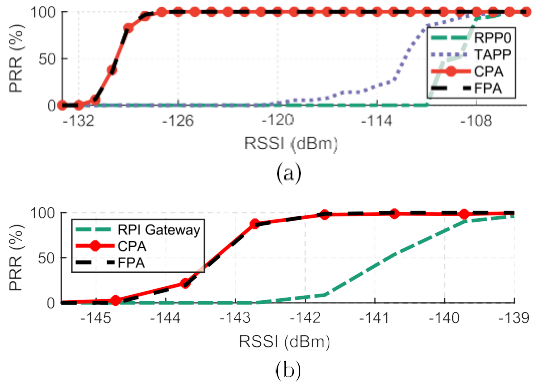

Fig. 16. Sensitivity tests of different decoders with (a) 10-byte payload, SF8, BW = 250 kHz and (b) 32-byte payload, SF12, BW = 125 kHz.

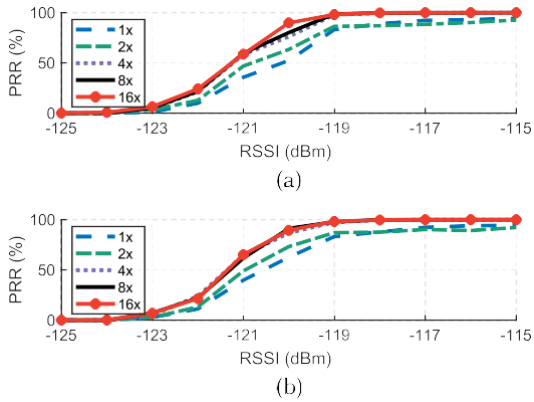

Fig. 17. Zero-padding ratio influence on (a) CPA and (b) FPA with $k = 8$.

4.2 Influence of Zero-Padding

Next, we confirm the effectiveness of our suggested peak refining approach, which enhances the frequency resolution by zero-padding. We evaluate both the FPA and CPA approaches using the identical experimental setup as described in Section 6.2. The receiver performance at various RSSI and zero-padding ratios is displayed in Figure 17. We see that PRR rises with increasing zero-padding ratio r for both FPA and CPA. There is a discernible increase in performance for r ≥ from 1 to 4, however this gain becomes minimal for r 4.

4.3 Encoder Test

Here, we report the results of our implementation of a LoRa encoder—a LoRa decoder turned upside-down. When comparing the encoder and decoder implementations, the superfluous bytes used to complete the final four symbols are the sole differences. Despite appearing to be corrected in the LoRa chip implementation, these bytes have no value on the receiver side. We pad zeros in our implementation to fill the left bytes. Upon activation, the encoder establishes a UDP port and proceeds to await data. Using a Python script, we pass 16 bytes to the encoder process. The produced LoRa signal is then sent from USRP to a LoRa end device. To verify that the data are being sent appropriately, we examine the device's serial port outputs. We transfer data using various parameters (i.e., different SFs, CRs, and bandwidth). In

addition, the broadcast is repeated 100 times for every parameter value. The final findings demonstrate that all 9,600 packets transmitted by our encoder could be correctly decoded, indicating the validity of our study and implementation from a different angle.

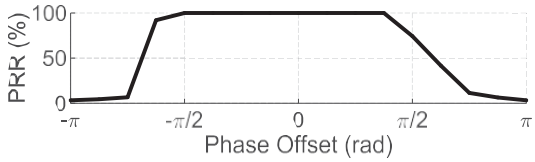

Fig. 18. LoRa chip's PRR when receiving specially constructed LoRa signals with different values of phase misalignment.

4.4 Influence of Phase Misalignment

We suggest FPA and CPA for LoRa demodulation in Section 3. We demonstrate that both approaches perform conceptually near to the IDEAL demodulation. This is due to the fact that they are susceptible to phase jitter brought on by multi-path and imperfect hardware. What is the real debugging technique used in the commercial LoRa chip, then? Is FPA and CPA adopted? Remarkably, we discover that the LoRa chip's demodulation algorithm is inferior than either FPA or CPA. We create LoRa packets with five carefully crafted bytes in them. Consequently, the four successive data symbols have the same symbol with fstart = 0, i.e., a symbol with a mid-point abrupt frequency drop. Phase mismatch between the two chirp segments is what most easily compromises such a sign. On the two segments, we manually apply phase offset before sending the packet over SDR. Our receiver's PPR is 100% and it can withstand phase misalignment when using FPA and CPA. However, we see that while receiving packets with large phase misalignment, the LoRa gateway frequently experiences packet loss. PRR in relation to phase misalignment is seen in Figure 18. When the phase offset is between $(-\pi, \pi)$, its PRR is almost 100%. However, when the phase offset increases, it decreases rapidly. As a result, the LoRa chip uses a weaker demodulation technique than what our techniques employ. Put otherwise, the cheap LoRa chip is susceptible to attacks known as "phase misalignment attacks." Even if it is difficult to carry out this sort of "attack," any little flaws in the security sector are important to note.

Assuming that the LoRa chip calculates $\Delta\varphi$ from the preamble and applies $\Delta\varphi$ to the subsequent data symbols, we speculate that the LoRa chip may adjust a fixed phase offset $\Delta\varphi$ throughout the demodulation process. The packet cannot be successfully demodulated by the LoRa device if $\Delta\varphi$ changes. As a result, when the phase offset varies, the PRR decreases. Naturally, until Semtech makes the docs available to the public, we are unaware of the internal implementation of the LoRa chip. Nevertheless, in comparison to our FPA and CPA approach, the LoRa chip demodulation implementation is less effective.

4.5 Outdoor Test

At last, we make a real-world comparison between our decoder and a commodity RPI gateway. We set the receiver outside the second-floor window. In order to ensure that there is no Line-of-Sight path between the transmitter and the receivers, we carefully choose the transmitter sites, which matches the typical scenario of LoRa communication. In explicit header mode, the transmitter continually transmits a 32-byte packet with the following settings: SF12, BW = 125 kHz, CR = 4, and CRC enabled. Using a tripod, we raise the transmitter to a height of 1.8 meters. Figure 19 displays the above-mentioned experiment configuration.

PRR is depicted at various communication distances in Figure 20. Our decoder can handle a communication range of up to 3,600 meters when PRR = 95% is set as a bar, but the commodity RPI gateway can only support a maximum of 2,800 meters. The RPI gateway's PRR variation at 2,400 meters is an intriguing fact. We speculate that the phase offset brought on by the intricate surroundings is the culprit. The commodity LoRa chip is susceptible to phase misalignment issues,

as was indicated in Section 6.5. As a result, our decoder seems to be more trustworthy for outside conversations.

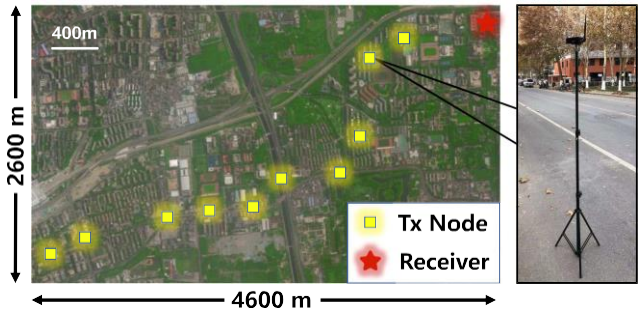

Fig. 19. Deployment of the transmiNer and receiver. The transmiNer is placed on a 1.8-m tripod.

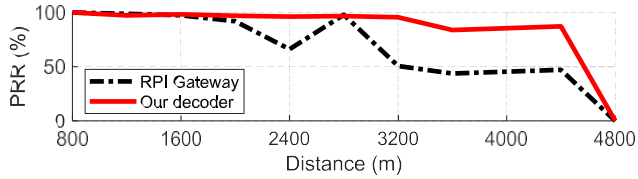

Fig. 20. Outdoor communication range experiments.

5 RELATED WORK

Reverse engineering of wireless protocols. The Internet of Things is developing quickly, and with it, many different proprietary protocols. There are several internal motivators behind the development of closed protocols rather than utilizing pre-existing standard protocols. For instance, a private protocol may spare Internet of Things makers from paying licensing fees, or the new protocol might provide certain capabilities aimed at particular embedded devices and applications. Nevertheless, departing from the norm might occasionally provide attackers with an advantage since the makers may use some unsecure designs. These protocols, such as Apple Wireless Direct Link ad hoc protocol [44, 45], In-Car wireless network [43], Cryptographic RFID [41, 42, 53], and others, are typically undocumented and closed to the public, which draws network researchers to reverse engineer them. By reverse engineering these protocols, we can uncover design defects and gain a deeper understanding of the current network. Our work also intends to improve the tool for evaluating the LoRa network and to assist other researchers with a deeper understanding of the LoRa physical layer.

articles pertaining to LoRa PHY. Many applications find LoRa's distinctive modulation, which offers a long range, intriguing. The communication range of traditional backscatter technology is restricted to meters. More recently, backscatter incorporating LoRa has improved the range to the kilometer level [12, 47, 52]. In the meantime, Netcatter [13] facilitates 256 current backscatter transmissions by utilizing LoRa-like modulation. The network capacity is relatively minimal due to LPWAN's high coverage characteristics and low data rate [46], which leads to critical collision issues. To increase the throughput of the LoRa network, numerous attempts have been tried. Weak signal decoding [50, 51] and collision decoding [1-4, 9, 10, 27, 49] are two further types of representative work. Their primary reliance is on LoRa PHY demodulation's peculiarity to distinct chirps to distinct packets. Our effort will advance research linked to LoRa PHY and provides a greater understanding of LoRa PHY.

.

6 CONCLUSION

This paper provides a thorough explanation of LoRa demodulation and decoding and reveals the underlying causes of the performance discrepancy between commercial LoRa and previous research. This work represents the first full implementation of the LoRa PHY to the black-box commodity LoRa chip with a proven performance guarantee. We optimize the demodulation to attain a theoretical performance guarantee for decoding at a very low SNR of 20 dB. By using officially known data about LoRa and packet manipulation, we also determine the sequence and parameters of decoding operations, such as dewhitening, error correction, deinterleaving, and so forth. Furthermore, we implement the first complete real-time SDR LoRa PHY on the GNU Radio platform. Our method outperforms existing methods by achieving a 100% decoding success rate, 142-dBm sensitivity, and a communication range of 3,600 m in urban areas, comparable to commodity LoRa.

APPENDIX

SER Calculation Model. Suppose the signal is transmitted through an AWGN channel, where the noise follows the complex Gaussian distribution, i.e., $\mathcal{CN}(0, \sigma^2)$. Then, after dechirping, the noise distribution turns to $\mathcal{CN}(0, M\sigma^2)$, where M is the number of samples within the demodulation window. The maximal height of noise has a small variance and can be approximated as $c\sigma$, where c is a constant. Suppose the data peak height h_d follows Gaussian distribution $\mathcal{N}(\mu_d, \sigma_d^2)$. We have

$$SER = P(h_d < c\sigma) = Q\left(\frac{c\sigma - \mu_d}{\sigma_d}\right), \tag{18}$$

where Q is the tail function of the standard normal distribution. Denote SNR as $\Gamma = \frac{A^2}{\sigma^2}$, where A is the signal amplitude.

IDEAL. Since multiplying and FFT are linear, the complex data peak in IDEAL follows complex Gaussian distribution $\mathcal{CN}(h, M\sigma^2)$, where $h = MA$. The peak height follows Rice distribution, which can be approximated as Gaussian distribution $\mathcal{N}(\mu_1, \sigma_1^2)$, where $\mu_1 = \sigma\sqrt{\frac{M\pi}{2}}L_{\frac{1}{2}}\left(-\frac{h^2}{2M\sigma^2}\right)$ and $\sigma_1^2 = 2M\sigma^2 + h^2 - \mu_1^2$. $L_{\frac{1}{2}}(\cdot)$ is Laguerre polynomial. According to Reference [47], the maximal height of other $M-1$ noise height is approximately $\sqrt{2M\sigma^2 H_{M-1}} = c_1\sigma$, where $H_l = \sum_{i=1}^{l}\frac{1}{i}$ depicts the lth harmonic number. Taking μ_1, σ_1, and c_1 into Equation (18), we derive the SER of IDEAL as follows;

$$SER_1 = Q\left(\frac{\sqrt{2H_{M-1}} - \sqrt{\frac{\pi}{2}}L_{\frac{1}{2}}\left(-\frac{M\Gamma}{2}\right)}{\sqrt{2 + M\Gamma - \frac{\pi}{2}L_{\frac{1}{2}}\left(-\frac{M\Gamma}{2}\right)}}\right). \tag{19}$$

FPA. If the phase compensation $\vartheta = 0$, then the frequency domain after adding equals IDEAL. Shifting the noise by ϑ does not affect the zero-mean noise distribution. Therefore, the noise distribution after adding can still be considered as $\mathcal{CN}(0, M\sigma^2)$, which means the maximal noise height is $c_2\sigma$, where $c_2 = c_1$. The data peak in FPA follows $\mathcal{CN}(h_1 + e^{j\theta}h_2, M\sigma^2)$, where $h_1(h_2)$ is the first (second) chirp segment height and $h_1 + h_2 = h$. Let $h_3 = |h_1 + e^{j\theta}h_2|$. The peak height follows Rice distribution, which can be approximated as $\mathcal{N}(\mu_2, \sigma_2^2)$, where $\mu_2 = \sigma\sqrt{\frac{M\pi}{2}}L_{\frac{1}{2}}\left(-\frac{h_3^2}{2M\sigma^2}\right), \sigma_2^2 = 2M\sigma^2 + h_3^2 - \mu_2^2$. SER of FPA is related to the symbol transmitted, and here we simplify the sending symbol with $h_1 = h_2 = \frac{h}{2}$. The best case is that $\vartheta = 0$ and $h_3 = h$. The worst case is that $|\vartheta| = \frac{\pi}{k}$ and

$h_3 = h\cos(\frac{\pi}{2k})$. We consider the average of them $h \cdot \frac{1+\cos(\frac{\pi}{2k})}{2} = h\cos^2(\frac{\pi}{4k})$ as final h_3. Therefore, the SER of FPA can be approximated as

$$SER_2 = Q\left(\sqrt{\frac{2H_{M-1} - \sqrt{\pi}L_{\frac{1}{2}}(-\frac{M\Gamma}{2})\cos^4(\frac{\pi}{4k})}{2 + M\Gamma\cos^4(\frac{\pi}{4k}) - \pi L_{\frac{1}{2}}^2(-\frac{M\Gamma}{2})\cos^4(\frac{\pi}{4k})}}\right). \tag{20}$$

CPA. The two separated data peak heights in CPA follow Rice distribution, respectively. The sum of them have the approximated distribution $W(\mu_3, \sigma^2)$, where $\mu_3 = \mu_3^{\mathrm{I}} + \mu_3^{\mathrm{II}}, \sigma_3^2 = 2M\sigma^2 + h_1^2 + h_2^2 - (\mu_3^{\mathrm{I}})^2 - (\mu_3^{\mathrm{II}})^2, \mu_3^{\mathrm{I}} = \sqrt{\frac{M\pi h_1}{2h}}L_{\frac{1}{2}}(-\frac{h_1 h}{2M\sigma^2}), \mu_3^{\mathrm{II}} = \sqrt{\frac{M\pi h_2}{2h}}L_{\frac{1}{2}}(-\frac{h_2 h}{2M\sigma^2})$. The noise peak height follows Rayleigh distribution. Denote the absolute value summation of two part of noise as $Y_i (i = 1, 2, \ldots, M-1)$. Y_i approximately follows $W(\mu_Y, \sigma_Y^2)$, where

$$\mu_Y = (\sqrt{h_1} + \sqrt{h_2})\sqrt{\frac{M\pi}{2h}}\sigma, \sigma_Y^2 = \frac{4-\pi}{2}M\sigma^2. \tag{21}$$

Let $S = \max_{i=1,2,\ldots,M-1} Y_i$. By Jensen's inequality [48],

$$\begin{aligned} e^{t\mathrm{E}(S)} &\le \mathrm{E}\left[e^{tS}\right] = \mathrm{E}\left[\max_i e^{tY_i}\right] \\ &\le \sum_{i=1}^{M-1} \mathrm{E}\left[e^{tY_i}\right] = (M-1)e^{\mu_Y t + \frac{1}{2}\sigma_Y^2 t^2}, \end{aligned} \tag{22}$$

where the last equality follows from the definition of the Gaussian moment generating function. Taking the logarithm on both sides of inequality (22), we have

$$\mathrm{E}(S) \le \frac{\ln(M-1)}{t} + \mu_Y + \frac{\sigma_Y^2 t}{2}. \tag{23}$$

For $t > 0$, according to the inequality of arithmetic and geometric means, we know

$$\mathrm{E}(S) \le \mu_Y + \sqrt{2\ln(M-1)}\sigma_Y = c_3\sigma, \tag{24}$$

where $c_3 = (\sqrt{h_1} + \sqrt{h_2})\sqrt{\frac{M\pi}{2h}} + \sqrt{(4-\pi)M\ln(M-1)}$. We use upper bound of $\mathrm{E}(S)$ to estimate the max noise peak height in CPA method. Consider the situation $h_1 = h_2 = \frac{h}{2}$; we have

$$SER_3 = Q\left(\sqrt{\frac{\sqrt{\pi} + \sqrt{(4-\pi)\ln(M-1)} - \sqrt{\pi}L_{\frac{1}{2}}(-\frac{M\Gamma}{4})}{2 + M\Gamma - \pi L_{\frac{1}{2}}^2(-\frac{M\Gamma}{4})}}\right). \tag{25}$$

REFERENCES

[1] Shuai Tong, Zhenqiang Xu, and Jiliang Wang. 2020. CoLoRa: Enabling multi-packet reception in LoRa. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'20)*. IEEE, 2303–2311.

[2] Xianjin Xia, Yuanqing Zheng, Tao Gu. 2020. FTrack: Parallel decoding for LoRa transmissions. *IEEE/ACM Trans. Netw.* 28, 6 (2020), 2573–2586. https://doi.org/10.1109/TNET.2020.3018020

[3] Zhenqiang Xu, Shuai Tong, Pengjin Xie, and Jiliang Wang. 2020. FlipLoRa: Resolving collisions with up-down quasi-orthogonality. In *Proceedings of the IEEE International Conference on Sensing, Communication and Networking (SECON'20)*.

[4] Zhe Wang, Linghe Kong, Kangjie Xu, Liang He, Kaishun Wu, and Guihai Chen. 2020. Online concurrent transmissions at LoRa gateway. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'20)*. IEEE, 2331–2340.

[5] Wenju Zhao, Shengwei Lin, Jiwen Han, Rongtao Xu, and Lu Hou. 2017. Design and implementation of smart irrigation system based on LoRa. In *Proceedings of the IEEE Globecom Workshops (GC Wkshps'17)*. IEEE, 1–6.

[6] Irfan Fachrudin Priyanta, Frank Golatowski, Thorsten Schulz, and Dirk Timmermann. 2019. Evaluation of LoRa technology for vehicle and asset tracking in smart harbors. In *Proceedings of the 45th Annual Conference of the IEEE Industrial Electronics Society (IECON'19)*, Vol. 1. IEEE, 4221–4228.
[7] Davide Magrin, Marco Centenaro, and Lorenzo Vangelista. 2017. Performance evaluation of LoRa networks in a smart city scenario. In *Proceedings of the IEEE International Conference on Communications (ICC'17)*. IEEE, 1–7.
[8] Umber Noreen, Ahcène Bounceur, and Laurent Clavier. 2017. A study of LoRa low power and wide area network technology. In *Proceedings of the International Conference on Advanced Technologies for Signal and Image Processing (ATSIP'17)*. IEEE, 1–6.
[9] Rashad Eletreby, Diana Zhang, Swarun Kumar, and Osman Yağan. 2017. Empowering low-power wide area networks in urban settings. In *Proceedings of the ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'17)*.
[10] Shuai Tong, Jiliang Wang, and Yunhao Liu. 2020. Combating packet collisions using non-stationary signal scaling in LPWANs. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'20)*.
[11] Yao Peng, Longfei Shangguan, Yue Hu, Yujie Qian, Xianshang Lin, Xiaojiang Chen, Dingyi Fang, and Kyle Jamieson. 2018. PLoRa: A passive long-range data network from ambient LoRa transmissions. In *Proceedings of the ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'18)*.
[12] Vamsi Talla, Mehrdad Hessar, Bryce Kellogg, Ali Najafi, Joshua R. Smith, and Shyamnath Gollakota. 2017. LoRa backscatter: Enabling the vision of ubiquitous connectivity. *Proc. ACM Interact. Mobile Wear. Ubiq. Technol.* 1, 3 (2017), 1–24.
[13] Mehrdad Hessar, Ali Najafi, and Shyamnath Gollakota. 2019. Netscatter: Enabling large-scale backscatter networks. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'19)*. 271–284.
[14] Adwait Dongare, Revathy Narayanan, Akshay Gadre, Anh Luong, Artur Balanuta, Swarun Kumar, Bob Iannucci, and Anthony Rowe. 2018. Charm: Exploiting geographical diversity through coherent combining in low-power wide-area networks. In *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'18)*. IEEE, 60–71.
[15] Akshay Gadre, Revathy Narayanan, Anh Luong, Anthony Rowe, Bob Iannucci, and Swarun Kumar. 2020. Frequency configuration for low-power wide-area networks in a heartbeat. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'20)*.
[16] Pieter Robyns, Peter Quax, Wim Lamotte, and William Thenaers. 2018. A multi-channel software decoder for the LoRa modulation scheme. In *Proceedings of the International Conference on Internet of Things, Big Data and Security (IoTBDS'18)*.
[17] Matthew Knight and Balint Seeber. 2016. Decoding LoRa: Realizing a modern LPWAN with SDR. In *Proceedings of the GNU Radio Conference*.
[18] Joachim Tapparel, Orion Afisiadis, Paul Mayoraz, Alexios Balatsoukas-Stimming, and Andreas Burg. 2020. An open-source LoRa physical layer prototype on GNU radio. In *Proceedings of the IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC'20)*. 1–5. DOI: http://dx.doi.org/10.1109/SPAWC48557.2020.9154273
[19] Josh Blum. 2016. LoRa modem with LimeSDR. Retrieved from https://myriadrf.org/news/lora-modem-limesdr/.
[20] RevSpace. DecodingLoRa. Retrieved from https://revspace.nl/DecodingLora.
[21] Alexandre Marquet, Nicolas Montavont, and Georgios Z. Papadopoulos. 2019. Investigating theoretical performance and demodulation techniques for LoRa. In *Proceedings of the IEEE IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'19)*. IEEE, 1–6.
[22] Semtech. 2020. Data Sheet SX1276/77/78/79, Rev. 7.
[23] LoRaWAN Specification v1.1. Retrieved from https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/.
[24] Symphony Link. Retrieved from https://www.link-labs.com/symphony.
[25] Semtech. 2019. Data Sheet SX1268, Rev. 1.1.
[26] Olivier Bernard, André Seller, and Nicolas Sornin. 2014. Low power long range transmitter. Patent EP 2 763 321 A1.
[27] Xiong Wang, Linghe Kong, Liang He, and Guihai Chen. 2019. mLoRa: A multi-packet reception protocol in LoRa networks. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP'19)*. IEEE, 1–11.
[28] Rajalakshmi Nandakumar, Vikram Iyer, and Shyamnath Gollakota. 2018. 3D Localization for sub-centimeter sized devices. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys'18)*.
[29] Yinghui Li, Jing Yang, and Jiliang Wang. 2020. DyLoRa: Towards energy efficient dynamic LoRa transmission control. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'20)*. IEEE, 2312–2320.
[30] LoRaMac-node. Retrieved from https://github.com/Lora-net/LoRaMac-node.
[31] SX1302 LoRa Gateway project. Retrieved from https://github.com/Lora-net/sx1302_hal.
[32] ChirpStack. Retrieved from https://www.chirpstack.io/.
[33] LoRaWAN Server. Retrieved from https://github.com/gotthardp/lorawan-server.

[34] Charm Decoder. Retrieved from https://github.com/WiseLabCMU/charm-decoder.
[35] Mehrdad Hessar, Ali Najafi, Vikram Iyer, and Shyamnath Gollakota. 2020. TinySDR: Low-power SDR platform for over-the-air programmable IoT testbeds. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'20)*. 1031–1046.
[36] TinySDR. Retrieved from https://github.com/uw-x/tinysdr.
[37] Semtech. 2017. Data Sheet SX1301, V2.4.
[38] Nadia Ben Atti, Gema M. Diaz-Toca, and Henri Lombardi. 2006. The berlekamp-massey algorithm revisited. *Applic. Algebr. Eng. Commun. Comput.* 17, 1 (2006), 75–82.
[39] Wikipedia. LFSR. https://en.wikipedia.org/wiki/Linear-feedback_shift_register.
[40] Ross Williams et al. 1993. A painless guide to CRC error detection algorithms. https://zlib.net/crc_v3.txt.
[41] P. Fraga-Lamas and T. M. Fernández-Caramés. 2017. Reverse engineering the communications protocol of an RFID public transportation card. In *Proceedings of the IEEE International Conference on RFID (RFID'17)*. 30–35.
[42] Karsten Nohl, David Evans, Starbug Starbug, and Henryk Plötz. 2008. Reverse-engineering a cryptographic RFID tag.. In *Proceedings of the USENIX Security Symposium*, Vol. 28.
[43] Ishtiaq Rouf, Robert D. Miller, Hossen A. Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. 2010. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *Proceedings of the USENIX Security Symposium*, Vol. 10.
[44] Milan Stute, David Kreitschmann, and Matthias Hollick. 2018. One billion apples' secret sauce: Recipe for the Apple wireless direct link ad hoc protocol. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 529–543.
[45] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. 2019. A billion open interfaces for Eve and Mallory: MitM, DoS, and tracking attacks on iOS and macOS through Apple wireless direct link. In *Proceedings of the USENIX Security Symposium*. 37–54.
[46] Branden Ghena, Joshua Adkins, Longfei Shangguan, Kyle Jamieson, Philip Levis, and Prabal Dutta. 2019. Challenge: Unlicensed LPWANs are not yet the path to ubiquitous connectivity. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom'19)*. 1–12.
[47] Tallal Elshabrawy and Joerg Robert. 2018. Closed-form approximation of LoRa modulation BER performance. *IEEE Commun. Lett.* (2018).
[48] Pascal Massart. 2007. *Concentration Inequalities and Model Selection*, Vol. 6. Springer.
[49] Qian Chen and Jiliang Wang. 2021. AlignTrack: Push the limit of LoRa collision decoding. *IEEE 29th International Conference on Network Protocols (ICNP'21)*, 1–11. DOI: 10.1109/ICNP52444.2021.9651985
[50] Chenning Li, Hanqing Guo, Shuai Tong, Xiao Zeng, Zhichao Cao, Mi Zhang, Qiben Yan, Li Xiao, Jiliang Wang, and Yunhao Liu. 2021. NELoRa: Towards ultra-low SNR LoRa communication with neural-enhanced demodulation. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 56–68.
[51] Shuai Tong, Zilin Shen, Yunhao Liu, and Jiliang Wang. 2021. Combating link dynamics for reliable lora connection in urban settings. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 642–655.
[52] Jinyan Jiang, Zhenqiang Xu, Fan Dang, and Jiliang Wang. 2021. Long-range ambient LoRa backscatter with parallel decoding. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 684–696.
[53] Qianwen Miao, Fu Xiao, Haiping Huang, Lijuan Sun, and Ruchuan Wang. 2019. Smart attendance system based on frequency distribution algorithm with passive RFID tags. *Tsinghua Science and Technology* 25, 2 (2019), 217–226.
[54] X. Geng, N. Zhu, Z. Chen, C. Ci, Z. Wang, and H. Wu,2022 "Random Access Preamble Design and Timing Advance Estimation for OTFS Systems in High-Mobility Scenarios," *[Journal/Conference Name]*, vol. , 11. , 34.
[55] A. Upadhyay, 2019"Performance Analysis of Preamble Detection at Maximum Throughput Level for OFDM," *[Journal/Conference Name]*, vol1. , 14. , pp
[56] Y. Zheng, B. Huang, and Z. Lu, "MLP-mmWP: High-Precision Millimeter Wave Positioning Based on MLP-Mixer Neural Networks," *[Journal/Conference Name]*, vol1. , 99. , pp. -,
[57] V. Ninkovic, D. Vukobratovic, A. Valka, and D. Dumic,2021 "Preamble-Based Packet Detection in Wi-Fi: A Deep Learning Approach
[58] S. Liu, F. Yang, D. Li, R. Yao, and J. Song,2022 "Power Line Communication with Robust Timing and Carrier Recovery against Narrowband Interference for Smart Grid,

Received 24 February 2022; revised 19 May 2022; accepted 5 June 2022