FLoRa+: Energy-efficient, Reliable, Beamforming-assisted, and Secure Over-the-air Firmware Update in LoRa Networks

ZEHUA SUN, TAO NI, HUANQI YANG, and KAI LIU, City University of Hong Kong Shenzhen Research Institute and City University of Hong Kong, Hong Kong, China
YU ZHANG and TAO GU, Macquarie University, Sydney, Australia
WEITAO XU, City University of Hong Kong Shenzhen Research Institute and City University of Hong Kong, Kowloon, China

The widespread deployment of unattended LoRa networks poses a growing need to perform Firmware Updates Over-The-Air (FUOTA). However, the FUOTA specifications dedicated by LoRa Alliance fall short of several deficiencies with respect to energy efficiency, transmission reliability, multicast fairness, and security. This article proposes *FLoRa+*, energy-efficient, reliable, beamforming-assisted, and secure FUOTA for LoRa networks, which is featured with several techniques, including delta scripting, channel coding, beamforming, and securing mechanisms. Specifically, we first propose a joint differencing and compression algorithm to generate the delta script for processing gain, which unlocks the potential of incremental FUOTA in LoRa networks. Then, we design a concatenated channel coding scheme with outer rateless code and inner error detection to enable reliable transmission for coding gain. Afterward, we develop a beamforming strategy to avoid biased multicast and compromised throughput for power gain. Finally, we present a securing mechanism incorporating progressive hash chain and packet arrival time pattern verification to countermeasure firmware integrity and availability attacks for security gain. Experimental results on a 20-node testbed demonstrate that *FLoRa+* improves transmission reliability and energy efficiency by up to 1.51× and 2.65× compared with LoRaWAN. Additionally, *FLoRa+* can defend against 100% and 85.4% of spoofing and Denial-of-Service (DoS) attacks.

CCS Concepts: • **Networks** → **Network protocols**; • **Computer systems organization** → **Embedded systems**;

Additional Key Words and Phrases: LoRa, firmware update over-the-air

The work described in this article was substantially sponsored by the project 62101471 supported by NSFC and was partially supported by the Shenzhen Research Institute, City University of Hong Kong. The work was also partially supported by the Research Grants Council of the Hong Kong Special Administrative Region (Project No. CityU 21201420 and CityU 11201422), and NSF of Shandong Province (Project No. ZR2021LZH010) and Guangdong Province (Project No. 2414050001974). The work was also partially supported by CityU APRC grant 9610471, CityU MFPRC grant 9680333, CityU SIRG grant 7020057, CityU SRG-Fd grant 7005666 and 7005984.
Authors' addresses: Z. Sun, T. Ni, H. Yang, K. Liu, and W. Xu (Corresponding author), City University of Hong Kong Shenzhen Research Institute and City University of Hong Kong, China; e-mails: zehua.sun@my.cityu.edu.hk, taoni2-c@my.cityu.edu.hk, huanqi.yang@my.cityu.edu.hk, kailiu@cityu.edu.hk, weitaoxu@cityu.edu.hk; Y. Zhang and T. Gu, Macquarie University, Australia; e-mails: y.zhang@mq.edu.au, tao.gu@mq.edu.au.

ACM 1550-4859/2024/02-ART54
https://doi.org/10.1145/3641548

ACM Reference Format:
Zehua Sun, Tao Ni, Huanqi Yang, Kai Liu, Yu Zhang, Tao Gu, and Weitao Xu. 2024. FLoRa+: Energy-efficient, Reliable, Beamforming-assisted, and Secure Over-the-air Firmware Update in LoRa Networks. *ACM Trans. Sensor Netw.* 20, 3, Article 54 (February 2024), 28 pages. https://doi.org/10.1145/3641548

1 INTRODUCTION

Over the past decades, the prosperous development of diverse **Internet of Things (IoT)** applications has raised demands for diverse networking paradigms, such as short-range wireless standards (e.g., Zigbee, Bluetooth), cellular technologies (e.g., 4G, 5G), and **Low Power Wide Area Networks (LPWANs)** protocols (e.g., LoRa, NB-IoT). The emergence of LPWANs complements the gaps of legacy wireless communication technologies with respect to long-range and ultra-low power-consumption transmission. Among LPWANs, LoRa is one of the most prevalent choices in industry and research communities, which has been dedicated to 163 network operators across 177 countries globally [53].

The widespread deployment of LoRa networks demands software upgrading with the latest standards, preventing security vulnerabilities, and device customization for specific purposes throughout their life cycle. Additionally, the traditional manual services required for upgrading and maintenance of such networks might significantly impede the large-scale deployment and compromise long-term sustainability. Thus, the ability of **Firmware Update Over-The-Air (FUOTA)** is becoming a necessity for LoRa networks. FUOTA, also known as **Over-The-Air Programming (OTAP)**, refers to remotely updating firmware images for embedded devices in an over-the-air manner without tinkering with the underlying hardware. Figure 1 illustrates a typical FUOTA process in LoRa networks. Instructed by the server, the gateway first issues a command packet to create a multicast group for FUOTA and subsequently starts transmitting firmware image fragments in the lossy channels to these LoRa nodes at the agreed time. The nodes open the receiving window accordingly, then reboot after the completion of reception and verification of the firmware. Finally, the gateway performs the update summary by collecting the status reported by these LoRa nodes.

LoRa Alliance has dedicated FUOTA specifications to standardize and refine this task [3, 9]. However, our study shows that the current specification suffers from the following limitations: **(i) Low energy efficiency.** The size of a LoRa firmware image is typically in the order of tens of kB. LoRa nodes are required to keep listening during the FUOTA process until the entire firmware is received, hence resulting in huge energy consumption for power-constrained LoRa nodes. **(ii) Poor transmission reliability.** It has been experimented that the packet loss ratio can be up to 30% at a distance of 4 km outdoors and 0.3 km indoors [35, 47]. Since FUOTA is a critical task that requires large-scale and error-free transmission, any packet loss and symbol error will bring difficulties to firmware reconstruction. **(iii) Biased multicast grouping.** In a multicast group, users with poor link quality may fail in the FUOTA task, and continuous transmission may bring down the throughput of the multicast group. **(iv) Insecure risks.** The FUOTA task is a security-critical task, which may be vulnerable to various active attacks [4], such as firmware integrity and availability attacks [25, 30]. These severe attacks have the potential to disrupt the FUOTA process and even distribute malware. Even though recent studies [1, 22] have attempted to enhance the security of FUOTA by modifying the LoRaWAN specification, they only focused on the general security issues using simulation tools. Therefore, energy-efficient, reliable, unbiased, and secure FUOTA for the LoRa network remains under investigation.

To this end, this article makes the first step toward filling this gap. Specifically, we present a novel FUOTA system named *FLoRa+* and propose delta scripting, channel coding, beamforming,

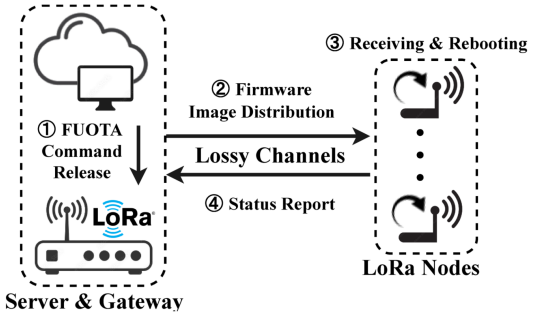


Fig. 1. FUOTA in LoRa networks. A server initiates a FUOTA task, and the LoRa gateway distributes the firmware image in the lossy channels to LoRa nodes. When the reception is completed, LoRa nodes reboot and report their FUOTA status to the gateway.

and securing mechanisms to achieve energy-efficient, reliable, and secure over-the-air firmware updates in LoRa networks.

(1) Delta Scripting. To improve energy efficiency, we propose a joint differencing and compression algorithm to generate the delta script as the patch for processing gain, which unlocks the potential of incremental FUOTA. Specifically, delta scripting refers to generating the patch in the form of differences between the new and old firmware images used for incremental update. We first have a closer look at the firmware image format, then define that of the delta script assisted by several commands. Afterward, we leverage the suffix array [46], a popular data structure in text indexing and data compression, for inter-firmware differencing to generate the delta script through injecting our pre-defined commands. The new firmware is usually a version with a small portion of modifications based on the old one; however, the modified code is relocated and shifted to other memory addresses, which may result in a disproportionate increase in the size of the generated delta script. To address this issue, we design an auxiliary intra-firmware compression algorithm to further reduce the size of the delta script. The constructed delta script, which possesses a relatively small size and low time complexity for firmware reconstruction, is well catered to the high energy efficiency of LoRa.

(2) Channel Coding. To enable reliable and robust transmission, we propose a concatenated channel coding scheme for coding gain, where a rateless code serves as outer code and an error detection code as inner code. **Automatic Repeat-reQuest (ARQ)** is commonly used for poor transmission reliability, but it is not suitable for LoRa transceivers of half-duplex mode due to additional listening overhead and **Acknowledgment (ACK)** implosion. To resolve the problem of packet loss during large-scale multicast transmission, we adopt rateless code as the outer code to generate potentially an infinite number of encoded blocks from the source firmware records in the delta script until the firmware image is reconstructed. To avoid the issues of each firmware record falling short of balancing encoding probability, we design a priority matrix with a dynamic updating strategy that can adjust their weights for fair encoding. However, the decoding of rateless code requires no symbol error of the received packet; otherwise, severe overhead may be suffered. With the property of infinite numbers of transmission of rateless code, the inner error detection code is incorporated. The lightweight channel coding scheme can ensure transmission reliability without compromising much energy consumption.

(3) Beamforming. To overcome biased multicast and compromised throughput, we propose a beamforming strategy for power gain. The multicast mode has shown in References [31, 51] that it can be complemented and augmented by the additional unicast delivery of content. We thus

assign the nodes that fail to reboot with new unicast groups and use beamforming for offering narrower beamwidth and stronger signal strength to distribute the firmware image. A key factor to beamforming is azimuth angle; however, the existing **Angle of Arrival (AoA)** methods have a limited range (e.g., 500 m for OwLL [7] and 100 m for Seirios [36]) and they rely on multiple probing packets from LoRa nodes, which is not suitable for FUOTA tasks. To resolve this issue, we design an explicit beam scanning-based azimuth estimation method by sending probing packets every steering beamwidth to these candidate LoRa nodes, in addition to a policy to set up optimal unicast groups. Such unicast delivery with beamforming can be traded for the link budget for the biased LoRa nodes and achieve optimal network performance.

(4) Securing. To prevent the FUOTA process from various active attacks, we propose a securing mechanism for security gain. These severe active attacks particularly target firmware image integrity (i.e., spoofing attacks) and availability (i.e., DoS attacks), which may disrupt the FUOTA process and deplete the battery service of LoRa nodes. Intuitive approaches involve signing the entire firmware image or each firmware record individually [4]. However, these methods may fail to identify forged packets when the verification step fails or require intensive computation for verification. To this end, we adopt a progressive hash chain verification method [25]. Specifically, we compute the hash value for each firmware record, then integrate it into the payload of the subsequent firmware record to form a hash chain, and finally sign the last firmware record. However, this countermeasure is vulnerable to **Denial-of-Service (DoS)** attacks, as the legitimate LoRa nodes require intensive computation when the attacker sends multiple forged signature packets. To resolve this problem, we leverage the fact of cyclical patterns of legitimate FUOTA packet arrival times. This arrival time pattern verification step can distinguish the forged signature packets from legitimate ones before proceeding with the signature verification process. The proposed securing mechanism can effectively defend against attacks to maintain the firmware integrity and availability during the FUOTA process.

We implement *FLoRa+* in a LoRa testbed consisting of 1 gateway and 20 nodes and conduct extensive evaluations in both indoor and outdoor environments. Experimental results show that *FLoRa+* improves network transmission reliability by up to 1.51× and energy efficiency by up to 2.65× compared with the existing solution in LoRaWAN. Additionally, security analysis shows that *FLoRa+* can defend against 100% of spoofing attacks and mitigate 85.4% of DoS attacks. A demo video is shown at https://youtu.be/dfptVznw5O0. In summary, this article makes the following contributions:

— To the best of our knowledge, *FLoRa+* is the first work significantly improving FUOTA in LoRa networks in terms of energy efficiency, transmission reliability, multicast fairness, and security. By manipulating the aforementioned methods tailored to LoRa, *FLoRa+* possesses the advantages of being energy-efficient, reliable, unbiased, and secure.
— *FLoRa+* has dedicated fourfold key components in a general manner, including delta scripting, channel coding, beamforming, and securing mechanisms. The delta scripting algorithm unlocks the ability of incremental update, the channel coding scheme enhances the reliability and robustness of large-scale firmware image distribution, the beamforming strategy further serves unicast users, and the securing mechanism can effectively defend against attacks to maintain security.
— We evaluate the performance of *FLoRa+* by conducting comprehensive benchmark experiments in a 20-node LoRa testbed. The results show that *FLoRa+* improves network transmission reliability by up to 1.51× and energy efficiency by up to 2.65× compared with the existing solution. Additionally, security analysis shows that *FLoRa+* can defend against 100% of spoofing attacks and mitigate 85.4% of DoS attacks.

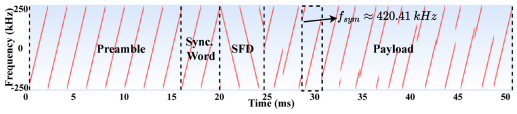


Fig. 2. LoRa signal spectrogram with the settings of SF=10 and BW=500 kHz. The chirp symbol enclosed by the dotted box represents the payload of $f_{sym} \times 2^{SF}/BW$ =‘1101011101’/‘861’.

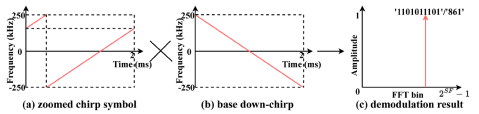


Fig. 3. LoRa packet demodulation process. The received symbol is multiplied with a base down-chirp, where the index of the resulting one FFT peak indicates the demodulated result.

This journal article is an extension of our previous work, *FLoRa* [52]. *FLoRa* incorporates threefold key components dedicated to FUOTA in LoRa networks, including delta scripting, channel coding, and beamforming. Building upon *FLoRa*, *FLoRa+* introduces an effective securing mechanism (Section 4.4) as the countermeasure for firmware integrity and availability attacks and performs security analysis (Section 5.3.6) to evaluate its effectiveness.

2 PRELIMINARY

In this section, we give a brief preliminary of LoRa and FUOTA.

2.1 LoRa

Broadly speaking, LoRa is one of the representative LPWAN technologies. Narrowly, LoRa is a **physical (PHY)** layer modulation technique derived from **Chirp Spread Spectrum (CSS)**.

Chirp Spread Spectrum (De)Modulation. Figure 2 illustrates a typical LoRa signal spectrogram. In LoRa modulation, the frequency of chirp signals sweeps linearly within the pre-defined bandwidth BW from its shifted initial frequency f_{sym} at a rate k over time t, denoted as

$$S\left(t, f_{sym}\right) = e^{j2\pi\left(f_{sym} - \frac{BW}{2} + \frac{k}{2}t\right)t}. \tag{1}$$

Figure 3 illustrates the demodulation process of a LoRa packet. The receiver performs the “dechirp” operation, where each received symbol is multiplied with a base down-chirp, denoted as

$$S\left(t, f_{sym}\right) \cdot S^{-1}\left(t, 0\right) = e^{j2\pi f_{sym} t}. \tag{2}$$

Then, the receiver applies the **Fast Fourier Transform (FFT)** on the multiplication result, where the index of the resulting one FFT peak indicates the demodulated LoRa symbol.

Data Rate. LoRa signal is mainly configured by three parameters, namely, **Spreading Factor (SF)**, **BandWidth (BW)**, and **Code Rate (CR)** [15]. SF indicates the number of bits per chirp symbol can represent, ranging from 7 to 12. BW is typically 125 kHz, 250 kHz, and 500 kHz. CR, denoting the coding rate of the **Forward Error Correction (FEC)** mechanism, can be set to 1 to 4. Thus, the LoRa data rate DR is specified by

$$DR = SF \times \frac{BW}{2^{SF}} \times \frac{CR}{CR + 4}. \tag{3}$$

2.2 Firmware Update Over-the-air

LoRaWAN is a data link layer specification built on top of LoRa, defining the typical star-topology network architecture and corresponding service specifications inclusive of FUOTA.

Network Architecture. Figure 4 illustrates a typical LoRaWAN network architecture, rendering a star topology inclusive of end devices, gateways, network servers, and application servers.

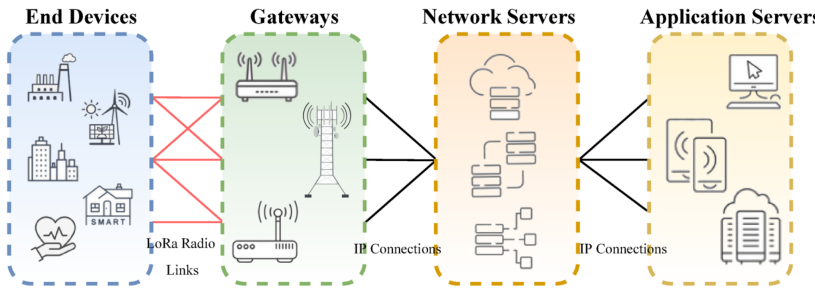


Fig. 4. LoRaWAN network architecture [53], including end device, gateway, network server, and application server.

FUOTA. To enable FUOTA tasks in LoRa networks, LoRa Alliance has dedicated three FUOTA specifications [9], i.e., remote multicast setup, fragmentation data block transport, and clock synchronization. In brief, regarding uplink-oriented LoRa radios and duty cycle limitation, the remote multicast setup specification enables FUOTA downlink transmission on a group of LoRa devices. It contains two commands specifying device addresses and session keys in the multicast group and the communication class (B or C) of LoRa devices. Concerning the large-size firmware image, the fragmentation data block transport specification defines the fragmentation session setup mechanisms, specifying parameters of firmware image size, fragment size, and so on. It recommends that a **Low-Density Parity-Check (LDPC)** code is used for erasure correction; however, the LDPC code falls short of flexibility due to its fixed coding rate and requirement for pre-defining. Another requirement is clock synchronization, serving for agreeing on the start time to further minimize energy consumption during the process of FUOTA.

However, we reveal four fundamental problems when performing FUOTA tasks in practice. Due to the characteristics of low data rate, LoRa networks cannot afford the high energy overhead to receive the entire firmware image. Then, the deployment scenario of LoRa networks makes them suffer from transmission loss due to the dynamic and poor link quality. Additionally, multicast cannot ensure that all nodes complete FUOTA tasks. Finally, the broadcast nature of the wireless medium makes FUOTA tasks vulnerable to severe active attacks. Thus, the energy efficiency, success rate, multicast fairness, and security issues make the FUOTA task in LoRa networks challenging.

3 SYSTEM OVERVIEW

In this section, we present a brief overview of *FLoRa+*, including the design goals and the design overview.

Design Goals. The design of *FLoRa+* is driven by the following considerations:

- **Energy Efficiency.** FUOTA is an energy-intensive task, while LoRa is a power-constrained technology. Regarding the large size of the firmware image, we aim to unlock the potential of incremental FUOTA in LoRa networks and design lightweight firmware decoding and reconstruction algorithms.
- **Transmission Reliability.** LoRa nodes need to construct the firmware image losslessly to reboot themselves. However, packet loss and symbol error are unavoidable to a varying degree in large-scale FUOTA downlink transmission, due to signal fading and interference. To solve this problem, channel coding is leveraged to detect or correct errors for reliable transmission.
- **Multicast Fairness.** Several multicast users with poor link quality may bring down the throughput of the multicast group. Therefore, we aim to assign these nodes in additionally

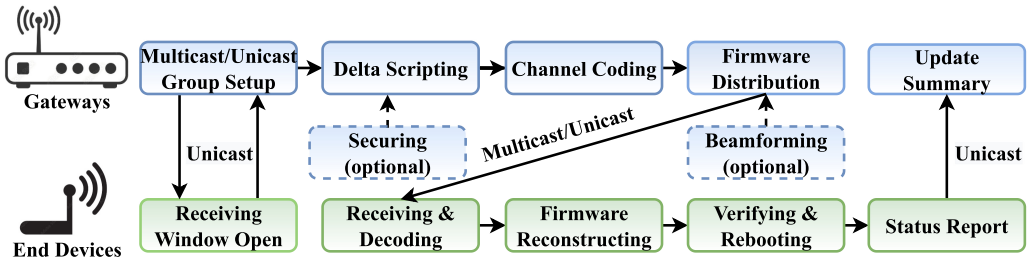


Fig. 5. System overview of *FLoRa+*. The gateway and end devices perform the corresponding operations according to the workflow procedure of FUOTA.

created unicast groups and trade beamforming for a better link budget and strong signal strength.

— **Attack Resistance.** The FUOTA task is susceptible to various active attacks, potentially posing a risk of disrupting the FUOTA process and even distributing malicious code. Thus, the securing mechanisms are necessary to effectively and efficiently identify the forged packets to defend against various attacks.

— **System Compatibility.** The design of *FLoRa+* should not be conflicted with the current LoRaWAN specification. Additionally, the hardware diversity of LoRa transceivers and the indeterminacy of the LoRa network also make it necessary for *FLoRa+* to have the property of compatibility.

System Overview. Figure 5 illustrates the overview of *FLoRa+*. When preparing a FUOTA task for the LoRa network, it is assumed that the gateway has been informed of the information of the corresponding LoRa nodes (e.g., identifiers and firmware versions). First, the gateway creates a multicast/unicast group by issuing a command packet specifying FUOTA parameters (e.g., firmware image size, fragment number, start time, and security key) and waits for the feedback from LoRa nods. Afterward, the gateway constructs the delta script of the new and old firmware images by leveraging the suffix array for inter-firmware differencing and intra-firmware compression. Then, the gateway distributes the delta script to LoRa nodes after performing a concatenated coding scheme, where the fragments in the delta script are for outer rateless encoding followed by inner error detection encoding. Among them, a degree distribution function and a priority matrix are designed for reasonable and fair fragment choosing during the encoding process. The LoRa nodes receive and decode these packets, which are subsequently used to reconstruct the firmware image. After the firmware integrity check and security authentication, the nodes reboot themselves and then report the FUOTA status to the gateway for an update summary. For nodes that fail to reboot, *FLoRa+* utilizes an explicit beam scanning method by sending probing packets every steering beamwidth to acquire azimuth information for the unicast group setup. The delta script is then distributed to the unicast groups accordingly. As for security, *FLoRa+* provides optional solutions to counter active firmware image integrity (i.e., spoofing attacks) and availability (i.e., DoS attacks) attacks, ensuring the security of the FUOTA task. The countermeasure for spoofing attacks requires the gateway to compute the hash chain for the delta script and sign the corresponding packets, which allows the LoRa nodes to identify the forged packets and perform firmware integrity check. To mitigate DoS attacks, the countermeasure involves the LoRa nodes to verify the arrival time patterns of received packets for security authentication.

4 SYSTEM DESIGN

We now describe the system design of *FLoRa+*, including delta scripting (Section 4.1), channel coding (Section 4.2), beamforming (Section 4.3), and securing (Section 4.4).

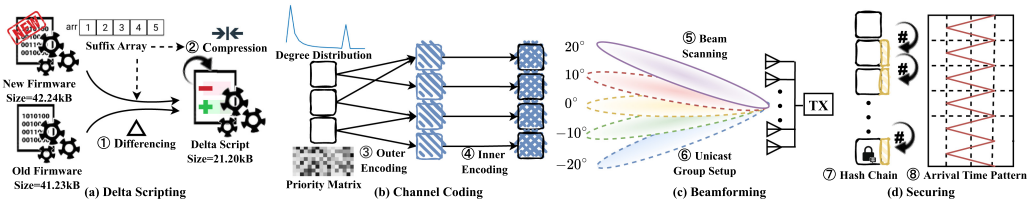


Fig. 6. System design of *FLoRa+*. The joint differencing and compression algorithm generates the delta script to enable incremental FUOTA. The concatenated channel coding scheme includes the outer rateless code and inner error detection code. The beamforming strategy is composed of beam scanning-based azimuth estimation method and unicast group setup policy, while the securing mechanism incorporates progressive hash chain and arrival time pattern verification steps.

4.1 Joint Delta Scripting Algorithm

As mentioned previously, the large-size firmware image needs to be distributed entirely to LoRa nodes during the FUOTA process. Thus, the energy-intensive and error-prone FUOTA task is challenging to LoRa technology due to its low data rate and constrained power requirement. In the field of reprogramming, incremental update has been widely used as an option to enhance energy efficiency [4], which mainly relies on generating the delta script between the new and old firmware images to be transmitted to IoT nodes. Then, nodes reconstruct the new firmware image using the stored old one. Unfortunately, although differences between the old and new firmware are generally not large, the modified code is relocated and shifted to other memory addresses, which may result in a disproportionate increase in the size of the generated delta script. To solve this problem, we first adopt an inter-firmware differencing algorithm [39] between the new and old firmware images to generate the delta script, then propose an intra-firmware compression algorithm to further minimize the size of the delta script (see Figure 6(a)).

4.1.1 Differencing Algorithm. When a FUOTA task is released, both the new and old firmware images can be acquired. The key idea of delta scripting is to use the differencing algorithm to find the common segments between these two firmware images and inject the corresponding commands to encode the delta script. Specifically, we use a classical suffix array-based differencing algorithm from Reference [39] to generate the delta script. Suffix arrays are an efficient data structure obtained by sorting all suffixes of a string. It can be used to minimize the size of the delta script, enable efficient reconstruction of the original firmware image, and further serve for the proposed compression algorithm. To make this article more self-contained, we first succinctly describe the suffix array, then present the differencing algorithm.

Suffix Array. Suppose there is a string $S = c_0c_1 \dots c_{n-1}$ of length n, $suffix[i]$ refers to the suffix starting from the ith character c_i to the end character c_{n-1}, namely, $suffix[i] = c_ic_{i+1} \dots c_{n-1}$. Two arrays, suffix array $sa[n]$ and rank array $rk[n]$, are used as auxiliary tools. The suffix array $sa[i]$ is defined as the index character of the suffix ranked ith after sorting all suffixes of S by dictionary order. The rank array $rk[i]$ denotes the rank of $suffix[i]$. Thus, it possesses the mathematical properties of $sa[rk[i]] = rk[sa[i]] = i$.

To construct the suffix array $sa[n]$ and rank array $rk[n]$ for the given string S, common solutions include plain, doubling [10], and DC3 algorithms [27], possessing the time complexity of $O(n^2 \log n)$, $O(n \log n)$, and $O(n)$, respectively. Thus, we adopt the DC3 algorithm [27] for the suffix array construction. The idea of the algorithm is as follows: (1) dividing all suffixes of S into two parts based on the result of their index mod 3, i.e., $A = \{suffix[i] \mid i \mod 3 \neq 0\}$ and $B = \{suffix[i] \mid i \mod 3 = 0\}$; (2) radix sorting for the first three characters of A; (3) radix sorting for B; (4) merging the results to construct the suffix array.

Afterward, the suffix array and rank array are used to construct the height array $height[n]$ of S, with which we can find the **Longest Common Prefix (LCP)** between the new and old firmware images to generate the delta script. We first define $lcp(i, j)$ $(i < j)$ as the length of the LCP of $suffix[i]$ and $suffix[j]$. The height array $height[i]$ is then defined as the LCP between $suffix[sa[i]]$ and its predecessor $suffix[sa[i-1]]$, denoted as

$$height[i] = \begin{cases} height[0] = 0, i = 0 \\ height[i] = lcp(sa[i], sa[i-1]), i = 1, 2, \ldots, n-1. \end{cases} \tag{4}$$

In simple terms, the height array represents the LCP of the two suffixes ranked next to each other. With the height array, $lcp(i, j)$ with arbitrary i and j can be calculated. It is revealed that if the values of the height array with varying indexes from $i + 1$ to j are always greater than a certain number, then this number is the length of the LCP, denoted as

$$lcp(i, j) = \min(height[i + 1], \ldots, height[j]) \, (i < j). \tag{5}$$

Delta Script Format. After finding LCPs by constructing the suffix arrays, we need to understand the firmware image format and then define the delta script. Generally, a firmware image is composed of many firmware records, where each firmware record is a binary/hexadecimal string with finite length, including start code, byte count, address, record type, data, and checksum [65].

To encode the delta script, we design two commands for inter-firmware differencing: *KEEP* and *UPDATE*. Specifically, the former is used to maintain the common segments between two firmware images, while the latter is used for recording different ones. Two commands would not conflict with the contents of the firmware image, defined as

$$\begin{aligned} &Command\ KEEP : K < entry, addr, offset >, \\ &Command\ UPDATE : U < str > . \end{aligned} \tag{6}$$

For command *KEEP*, the $entry$ indicates the index of the firmware record, $addr$ and $offset$ represent the starting address and length of the common segments, respectively. For command $UPDATE$, str indicates the different segments. For size saving and simplicity, this command can be replaced directly with str.

Differencing. Given the new and old firmware images composed of F_{new}_FR and F_{old}_FR firmware records, they are denoted as $F_{new}[F_{new}\_FR]$ and $F_{old}[F_{old}\_FR]$, respectively. For ith firmware records having the same address (i.e., entry) of two firmware images $F_{new}[i]$ and $F_{old}[i]$, a connection character "-" is utilized to merge the two records, i.e., $S = F_{old}[i] + \text{'-'} + F_{new}[i]$. Then, we use the aforementioned suffix array data structure to find the LCP of S. For those LCPs whose lengths are greater than the pre-defined threshold, we utilize the *KEEP* command to encode them in the delta script; otherwise, the *UPDATE* command is used. The threshold is set to 11, which is the minimum length of the replaced *KEEP* command.

4.1.2 Compression Algorithm. Due to the disproportionate increase in the size of the generated delta script resulting from firmware modification, we need to further decrease its size for the power-constrained LoRa networks. Most previous work on compression algorithms can reach a considerable result, but at the cost of a huge computational overhead and memory usage [28]. To meet the requirements of high energy efficiency of LoRa networks, we propose a novel suffix array-based compression algorithm, with which we can achieve a tradeoff between compression performance and computational overhead.

Similar to the differencing algorithm, we add one more command, *LABEL*, for intra-firmware compression. In command *LABEL*, the reoccurring segments are added to the dictionary and

subsequently replaced with the dictionary index, denoted as

$$Command\ LABEL : L < index, str > / L < index >, \tag{7}$$

where str appears only for the first time.

Given the generated delta script $F_\Delta[F_\Delta\_FR]$ composed of F_Δ_FR (i.e., the larger one between F_{new}_FR and F_{old}_FR) firmware records, the common segments appear within and between firmware records. Thus, we define a window containing a source buffer s_buffer and a compression buffer c_buffer, denoted as $W = s\_buffer$+'-'+c_buffer. For ith firmware record $F_\Delta[i]$, we first feed it into the compression buffer to find LCP in intra-record level. Then, $F_\Delta[i]$ slides into the source buffer, and the rest of firmware records $F_\Delta[j \neq i]$ are fed into the compression buffer to find LCP in inter-record level. We repeat the above operations until all records are traversed. Similar to the differencing algorithm, for those LCPs whose lengths are greater than the pre-defined threshold, we utilize the *LABEL* command to encode them in the delta script; otherwise, the *UPDATE* command is used. We set the threshold to 5, which is the minimum length of the replaced *LABEL* command. The whole delta scripting process is summarized in Algorithm 1.

4.2 Concatenated Channel Coding Scheme

After preparing the delta script, the gateway distributes it to LoRa nodes in fragments. However, as discussed above, although LoRa signals can provide high sensitivity and be resilient to in-band/out-of-band interference, the long-range deployment scenarios make packet loss and symbol error inevitable. Additionally, **Cyclic Redundancy Check (CRC)** is not enabled in the downlink transmission according to the LoRaWAN specification. Thus, the unreliable link quality problem needs to be resolved before firmware reconstruction during the error-prone FUOTA task. To this end, we propose a concatenated channel coding scheme (see Figure 6(b)), where a rateless code [37] serves as outer code to combat packet loss and an error detection code as inner code to avoid symbol error.

4.2.1 Concatenated Encoding. We introduce the concatenated encoding process for the firmware image, including outer rateless encoding followed by inner error detection encoding.

Outer Encoding. Regarding the packet loss issues in multicast scenarios, rateless code has been proven as an effective method [51], which demonstrates a strong capability of resilience to dynamic channel quality. The key idea of conventional rateless codes is to continuously select an appropriate number according to the degree distribution function and generate the encoded blocks containing this number of source symbols. Unfortunately, this may lead to a biased choice of the encoding source symbols. To this end, *FLoRa+* designs a priority matrix with a dynamic updating strategy to adjust the weights of all firmware records for fair choosing while maintaining historical transmission information during multicast. Below, we introduce the outer encoding process, along with the degree distribution function and priority matrix.

Given the aforementioned delta script with multiple firmware records $F_\Delta[F_\Delta\_FR]$, we suppose the number of firmware records is $n = F_\Delta\_FR$ and the largest length of the firmware record is a. Thus, ith record is denoted as $F_\Delta[i] = c_{i_0}c_{i_1}\ldots c_{i_{a-1}}$ with a symbols. For the sake of completeness of the expression, those records whose lengths are less than a are filled by the character "#" for the rest of the digits. The encoder selects a set of d original firmware records to generate an encoded block $EB[j]$ at the jth times. The degree d conforms to a special degree distribution, while d records are chosen based on the priority matrix. The encoder performs a bit-wise XOR operation for those d original firmware records to dynamically generate the encoded blocks. The outer encoding process

ALGORITHM 1: Delta script construction algorithm.

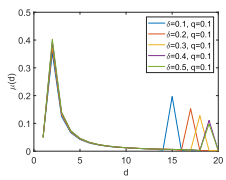
Input: The new and old firmware image, $F_{new}[F_{new}\_FR]$ and $F_{old}[F_{old}\_FR]$.
Output: The delta script, $F_\Delta[F_\Delta\_FR]$.

Function *calc_sa*(S)**:**
Suffix array sorting based on the DC3 algorithm;
Return $sa[S.length()]$;
Function *calc_height*(S)**:**
for $i \leftarrow 1$ **to** $S.length() - 1$ **do**
  $height[i] = lcp(sa[i], sa[i-1])$;
end
Return $height[S.length()]$;
Function *calc_lcp*(S)**:**
$calc\_sa(S)$; $calc\_height(S)$; $< addr, offset > \leftarrow max(height[S.length()])$;
Return $addr, offset$;
Differencing Algorithm:
for $entry \leftarrow 0$ **to** $max([F_{new}\_FR, F_{old}\_FR) - 1$ **do**
  $calc\_lcp(S \leftarrow F_{old}[entry]$+'-'+$F_{new}[entry])$;
  if $offset \geq threshold$ **then**
    $F_\Delta \leftarrow K < entry, addr, offset >$; $F_\Delta \leftarrow U < S.substr() >$;
  else
    $F_\Delta \leftarrow U < S >$;
  end
end
Compression Algorithm:
for $entry \leftarrow 0$ **to** $F_\Delta\_FR - 1$ **do**
  $c\_buffer \leftarrow F_\Delta[entry]$;
  for $entry2 \leftarrow 0$ **to** $F_\Delta\_FR - 1$ **do**
    if $entry \neq entry2$ **then**
      $s\_buffer \leftarrow F_\Delta[entry]$; $c\_buffer \leftarrow F_\Delta[entry2]$;
    end
    $calc\_lcp(W \leftarrow s\_buffer$+'-'+$c\_buffer)$;
    if $offset \geq threshold$ **then**
      $F_\Delta \leftarrow L < index, str > /L < index >$; $F_\Delta \leftarrow U < W.substr() >$;
    else
      $F_\Delta \leftarrow U < c\_buffer >$;
    end
  end
end
Return the delta script, $F_\Delta[F_\Delta\_FR]$.

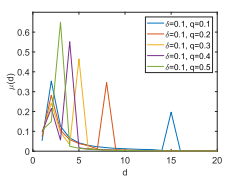
is represented by

$$EB[j] = \begin{array}{l} \left(c_{d_{1_0}} \oplus c_{d_{2_0}} \oplus \cdots \oplus c_{d_{d_0}}\right)\left(c_{d_{1_1}} \oplus c_{d_{2_1}} \oplus \cdots \oplus c_{d_{d_1}}\right) \\ \ldots \left(c_{d_{1_{a-1}}} \oplus c_{d_{2_{a-1}}} \oplus \cdots \oplus c_{d_{d_{a-1}}}\right), \end{array} \tag{8}$$

where $d_1, d_2, \ldots, d_d$ represent the selected d entries. After constructing the blocks, two kinds of information are appended as the header: degree d and the entries of these selected firmware records.

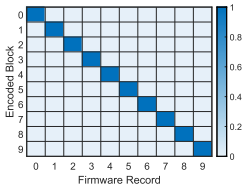


(a) RSD with different δ.

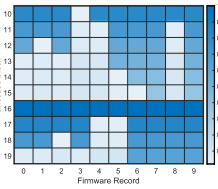


(b) RSD with different q.

Fig. 7. Degree distribution function with $n = 100$ (only the first 20 indices shown).



(a) Source submatrix Γ_S.



(b) Redundancy submatrix Γ_R.

Fig. 8. Priority matrix Γ of size $m \times n$ (an example of $m = 20, n = 10$).

The encoded block is then denoted as $EB[j] = c_{j_0} c_{j_1} \ldots c_{j_{b-1}}$, where b is the updated length. The encoder can continuously select a degree d and generate corresponding encoded blocks.

Degree Distribution Function. To enable reasonable degree choosing, we apply the **Robust Soliton Distribution (RSD)** $\mu(d)$ as the degree distribution function. RSD $\mu(d)$ is originally designed for **Luby Transformer (LT)** code [37], one of the representative rateless codes:

$$\mu(d) = (\rho(d) + \tau(d))/\sum_{i=1}^{n}(\rho(d) + \tau(d)), d = 1, 2 \ldots, n. \tag{9}$$

It is derived based on the **Ideal Soliton Distribution (ISD)** $\rho(d)$ by adding an additional component $\tau(d)$, denoted as

$$\rho(d) = \begin{cases} \frac{1}{n}, d = 1 \\ \frac{1}{d(d-1)}, d = 2, 3, \ldots, n, \end{cases} \quad \tau(d) = \begin{cases} P/(nd), d = 1, 2, \ldots, \lceil n/P \rceil - 1 \\ P(\ln(P/\delta))/n, d = \lceil n/P \rceil \\ 0, d = \lceil n/P \rceil + 1, \ldots, n. \end{cases} \tag{10}$$

Among these, $\rho(d)$ is the probability distribution over the integers varying from 1 to n, while $\tau(d)$ relies on a set of parameters $P = q \ln(n/\delta)\sqrt{n}$, where $q > 0$ is a constant and $\delta \in (0, 1]$ represents the upper bound for the decoding failure probability. As shown in Figure 7, $\rho(d)$ has a spike at 2, but the extra component $\tau(d)$ in RSD adds an extra spike at n/P. Thus, when receiving a certain number of blocks, the lower bound for the success probability of decoding is $1 - \delta$. In our experiments, we utilize $\delta = 0.1$ and $q = 0.4$ to enable high decoding success probability and moderate size of encoded blocks.

Priority Matrix. To avoid biased firmware record choosing, we design a priority matrix Γ of size $m \times n$ ($m > n$) for outer encoding. Among them, n is the number of firmware records, while m represents the number of encoded blocks, i.e., the number of transmission iterations. Each element in Γ, denoted as $\gamma_{j,i} \in [0, 1]$, represents its priority of the firmware record $F_\Delta[i]$ during the transmission of encoded blocks $EB[j]$. As shown in Figure 8, we split Γ into two parts: the fixed source submatrix Γ_S containing the first n rows of Γ and the varying redundancy submatrix Γ_R containing the rest of $m - n$ rows of Γ. For the source submatrix Γ_S, the values of the diagonal positions are set to 1 while the others are 0. This makes sure that only one source firmware record is encoded into the block for transmission in turn, regardless of the degree selected. For redundancy submatrix Γ_R, the initial elements are all set to 1, which are updated whenever an encoded block is sent. We adopt the exponential smoothing method [16] for updating $\gamma(t + 1) = \alpha \times d/n + (1 - \alpha) \cdot \gamma(t) - flag$, where α is the learning rate, and $flag$ denotes whether this firmware record is chosen. When all elements in Γ_R become extremely low, they are set to the initial values to start a new round of redundancy encoding. The above learning rate updating strategy is used to adjust the weights of all firmware records for adapting to fair source firmware

record choosing while maintaining historical transmission information during multicast. We set α to 0.2 to ensure that higher importance is placed on the previously un-selected firmware records.

Inner Encoding. After outer encoding, the encoded block is fed for inner encoding to avoid symbol error. *FLoRa+* designs an error detection code rather than an error correction code, since error correction typically requires longer redundancy bytes, which may degrade transmission efficiency. On the contrary, with the feature of infinite transmission of rateless code, the packet can be simply discarded when a symbol error is found.

Inspired by the idea of linear block code [57], we design a novel error detection code. In particular, we define an error detection code $EB[j,k]$ for the encoded block $EB[j]$ by dividing it into $\lceil b/k \rceil$ segments. For each segment, we use the XOR operation for these k characters to generate the check byte, which is appended at the end of each segment. The inner encoding process is represented by

$$EB[j,k] = \begin{array}{l} c_{j_0} c_{j_1} \ldots c_{j_{k-1}} \left(c_{j_0} \oplus c_{j_1} \oplus \cdots \oplus c_{j_{k-1}}\right) \\ c_{j_k} c_{j_{k+1}} \ldots c_{j_{2k-1}} \left(c_{j_k} \oplus c_{j_{k+1}} \oplus \cdots \oplus c_{j_{2k-1}}\right) \ldots \\ c_{j_{(\lceil b/k \rceil - 1)k}} c_{j_{(\lceil b/k \rceil - 1)k+1}} \ldots c_{j_{\lceil b/k \rceil k-1}} \left(c_{j_{(\lceil b/k \rceil - 1)k}} \oplus c_{j_{(\lceil b/k \rceil - 1)k+1}} \oplus \cdots \oplus c_{j_{\lceil b/k \rceil k-1}}\right). \end{array} \tag{11}$$

In our experiment, we set k to 8 for a tradeoff between accuracy and energy overhead. After inner encoding, the encode block is denoted as $EB[j,k] = c_{j_0} c_{j_1} \ldots c_{j_{c-1}}$, where c is the updated length. It is noted that we set a threshold-enabled stop condition that the number of redundancy packets does not exceed 25% of the firmware records if not enough update reports are received from LoRa nodes.

4.2.2 Concatenated Decoding. After the firmware records of the delta script are encoded and then distributed, LoRa nodes receive these packets to decode them. Specifically, the concatenated decoding process includes the inner error detection decoding followed by the outer rateless decoding to reconstruct the delta script.

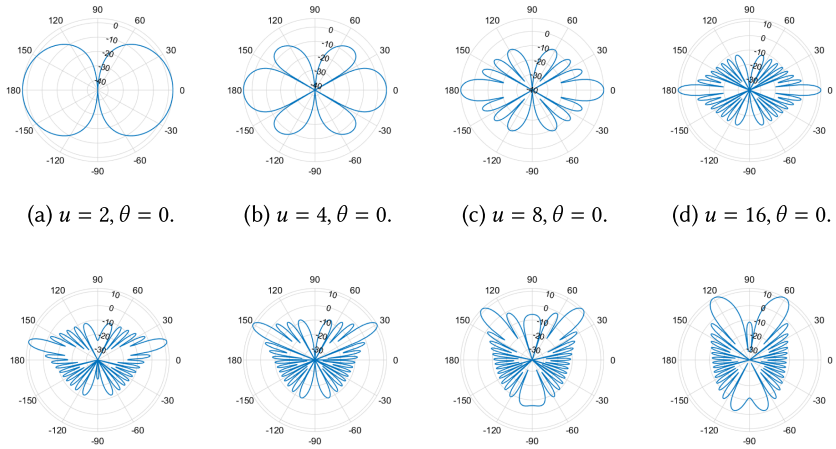
Inner Decoding. For a received packet $EB[j,k]$, the LoRa nodes utilize the XOR operation to compute the check codes for each segment and then compare these computed ones with the appended ones. The received packet can then be fed to the outer decoder for further decoding if the check codes of all segments are correct; otherwise, LoRa nodes discard this packet.

Outer Decoding. For outer decoding, LoRa nodes first find those encoding blocks of degree 1. They perform XOR operations between these blocks of degree 1 and the rest of the blocks. In such a process, more encoding blocks of degree 1 are generated. LoRa nodes repeat the above process until all firmware records of $F_\Delta[F_\Delta\_FR]$ are reconstructed. The decoding process terminates with failure if there is no such block of degree 1. Since the whole decoding process involves the XOR operations only, the decoding scheme is lightweight to power-constrained LoRa networks.

4.2.3 Firmware Image Reconstruction. After receiving all packets during the FUOTA process, the firmware image reconstruction can be performed. Specifically, the firmware image reconstruction requires the LoRa nodes to first perform decoding algorithms to construct the delta script. Afterward, recall Section 4.1, by simply traversing each firmware record of delta script and performing corresponding command recovery operations with an order of *LABEL*, *KEEP*, and *UPDATE*, LoRa nodes can losslessly recover the firmware image. At this stage, the majority of LoRa nodes have successfully acquired the firmware image to reboot themselves.

4.3 Beamforming Strategy

Several multicast users with poor link quality may fail the FUOTA task. To solve this problem, we additionally group these LoRa nodes into several unicast deliveries and utilize beamforming



(a) $u = 2, \theta = 0$. (b) $u = 4, \theta = 0$. (c) $u = 8, \theta = 0$. (d) $u = 16, \theta = 0$.

(e) $u = 16, \theta = \pi/12$. (f) $u = 16, \theta = \pi/6$. (g) $u = 16, \theta = \pi/4$. (h) $u = 16, \theta = \pi/3$.

Fig. 9. Beamforming patterns with different u and θ.

to achieve a better link budget and stronger signal strength [29]. However, to acquire the key information to beamforming, such as azimuth angles of candidate LoRa nodes, the existing LoRa AoA methods [7, 36] are not desirable due to their limited resolution and range. To overcome this limitation, we design an explicit beam scanning-based azimuth estimation method, along with an optimal unicast group setup policy (see Figure 6(c)).

4.3.1 Beam Scanning-based Azimuth Estimation Method. Before presenting the beam scanning-based azimuth estimation method, it is essential to discuss the impact of beamforming. By definition, beamforming is a spatial filtering technique to radiate or capture energy in a specific direction based on an array of radiators over its aperture [29]. Thus, there are two primary factors to beamforming: types of antenna arrays and beam-steering azimuth angles. For instance, suppose a LoRa transmitter has an antenna array of **Uniform Linear Array (ULA)**, a commonly used type in beamforming, in which antennas are arranged in a straight line with a distance of half the wavelength between every two adjacent ones. Figure 9 illustrates several beamforming patterns with different numbers of antennas u and beam-steering azimuth angles θ. We can observe that u and θ render the angle and the main lobe of the beam, respectively. However, the main lobe of the beam is too abstract a performance metric to quantize, thus, we utilize **Half Power Beam Width (HPBW)** [21] to indicate the beamwidth ψ. Specifically, HPBW refers to the angular separation width, in which the magnitude of the radiation pattern decreases by -3 dB (typically 50%) from the peak of the main lobe of the beam.

To efficiently and dependably acquire the azimuth angles of the candidate LoRa nodes, *FLoRa+* implements an explicit beam scanning method. The idea of this method involves sending probing packets in the directions with the highest gain of beamforming and relying on the returned packets from nodes to collaboratively determine the azimuth angle. With beamforming, the LoRa transmitter sends p probing packets every beamwidth ψ to all candidate LoRa nodes that wait for the unicast delivery, thus resulting in $p \times \lceil 2\pi/\psi \rceil$ probing packets in total. Each probing packet contains the beam-steering azimuth angle information θ (i.e., the direction with the highest gain of beamforming) during the current transmission. Afterward, LoRa nodes may receive some of the probing packets and then respond to the LoRa transmitter the ones with top-l largest **Received Signal Strength Indicator (RSSI)** values. In our experiments, the values of p and l

are both empirically set as 3 to ensure stable and accurate estimation of the azimuth angle while maintaining high energy efficiency. It is noted that the time duration and energy consumption during beam scanning are trivial compared to FUOTA firmware distribution.

4.3.2 Optimal Unicast Group Setup Policy. Given the candidate LoRa nodes that wait for the unicast delivery and the information of beamforming with respect to steering azimuth angle and beamwidth, we need to set up the unicast groups elaborately to maximize the performance of FUOTA tasks. To this end, we design an optimal unicast group setup policy based on the hierarchical clustering method [40] to acquire the number of unicast groups. In brief, *FLoRa+* computes the azimuth angle differences between every two of these nodes, where Manhattan distance is utilized as the similarity metric, denoted as

$$\|\theta_i - \theta_j\| = \sum_l \left|\theta_{i_l} - \theta_{j_l}\right|, \tag{12}$$

where θ_i and θ_j represent the azimuth angle of LoRa node i and j. With the computed similarity matrix, the beamwidth ψ is utilized as the constraint to assign different unicast groups for these candidate LoRa nodes. Afterward, the firmware image can be distributed accordingly until all unicast deliveries are complete to finish the FUOTA task.

4.4 Securing Mechanism

The FUOTA task is a security-critical task, which can potentially be susceptible to a variety of attacks [4], since legitimate nodes within a multicast group are set in the mode of continuously listening. Specifically, these severe attacks can be divided into either active or passive attacks [2]. An active attack directly damages or modifies data to disrupt network operations [23, 41, 56], while a passive attack eavesdrops and analyzes communications to intercept network data [42–44, 67]. This article focuses on active attacks with more destructive effects, particularly those targeting firmware image integrity (i.e., spoofing attacks) and availability (i.e., DoS attacks). These active attacks are represented by spoofing attacks to send forged firmware records and DoS attacks to send multiple signature packets requiring much computational resources. As such, a security mechanism is of utmost importance. We now present several attack models along with corresponding countermeasures to address these potential vulnerabilities.

4.4.1 Attack Model. Similar to previous security-oriented works in LoRa networks [56, 66], we assume that an attacker has comprehensive knowledge of the FUOTA system of *FLoRa+*, including system design, firmware structure, and packet format. The attacker has the capability to eavesdrop and transmit packets over public channels. Initially, the attacker performs traffic detection to identify a surge of downlink packets within a short time frame, which indicates the initiation of a FUOTA process. Afterward, the attacker can launch active attacks by sending specific packets to legitimate LoRa nodes in the multicast group to disrupt the FUOTA process. We specifically consider the following active attacks:

- **Spoofing Attack.** This type of attack involves impersonating LoRa gateway to manipulate the legitimate LoRa nodes to receive packets that has not undergone proper validation or filtering. Specifically, the attacker aims to intercept legitimate packets, modify their contents randomly or purposely, then forward these forged packets to the LoRa nodes within the multicast group. As a result, legitimate nodes receive incorrect firmware records, preventing them from correctly recovering the firmware image.
- **DoS Attack.** This type of attack involves causing a network crash or node computation overload. In this attack scenario, the attacker has the ability to send multiple forged signature packets into the LoRa network to launch the DoS attack. LoRa nodes that receive these

packets are forced to perform resource-intensive signature verification computations, which can rapidly deplete their limited battery power.

— **Other Attacks.** The FUOTA process may also be vulnerable to other attacks, such as injection and encoding attacks. In particular, injection attacks can fraudulently issue a command packet to initiate a FUOTA process, distributing malicious firmware images and even gaining control of the LoRa network. Encoding attacks exploit vulnerabilities in the encoding/decoding process (especially in the rateless code) by sending forged encoding redundancy packets, where the decoded incorrect firmware records disrupt the FUOTA process.

4.4.2 Countermeasure. We introduce the countermeasures against the aforementioned attacks, including progressive hash chain and arrival time pattern verification steps.

Progressive Hash Chain Verification. To maintain firmware image integrity, an intuitive method is to compute the digital signature for the entire firmware image. However, the firmware image must be entirely received before it can be verified, and forged packets sent by the attacker cannot be identified specifically. An alternative method is incorporating packet-level verification; however, it is too resource-intensive for the power-constrained LoRa network. Inspired by Sluice [30], we adopt a lightweight progressive hash chain verification method. As shown in Figure 10, the rationale is creating a hash chain by computing the hash value for each firmware record and integrating that value into the payload of the subsequent firmware record. We then digitally sign the final firmware record to ensure its authentication and leverage the hash chain value to verify the rest of the firmware records progressively. This method allows for maintaining firmware image integrity and preserving computing resources for LoRa networks.

Specifically, recall in Section 4.2.1, the delta script is defined as $F_\Delta[F_\Delta\_FR]$, while ith record is represented as $F_\Delta[i] = c_{i_0}c_{i_1}\dots c_{i_{a-1}}$ where the largest length of the firmware record is a. We calculate the hash value of each firmware record and append it to the payload of the next one. Specifically, our method adopts BLAKE2s [5] as the hash function, which is an efficient and secure hash algorithm, making it suitable for resource-constrained mobile applications. The length of the generated hash from BLAKE2s is variable, ranging from 1 to 32 bytes. In this case, we set the hash length to 8 bytes for a tradeoff between security and efficiency. By appending the hash value of $F_\Delta[i]$ to the payload of $F_\Delta[i+1]$, each firmware record (except the first one) contains the hash value of its previous one, which forms a hash chain. Then, we sign the last firmware record using a classical asymmetric encrypt scheme, i.e., **RivestShamirAdleman (RSA)** [48]. Combining the hash chain and digital signature ensures that any forged packets from the attacker can be detected efficiently.

We theoretically validate the effectiveness of the progressive hash chain verification scheme. BLAKE2s generates an 8-byte hash value, resulting in 2^{64} possible hash values. According to the principle of the birthday attack [8], a collision can be found via approximately $\sqrt{2^{64}} = 2^{32}$ hash computations. Thus, the attacker is required to perform nearly 4 trillion hash computations to launch an attack with a probability of 50%. Overall, our method is secure and reliable in terms of firmware image integrity.

Arrival Time Pattern Verification. Firmware image availability is also a critical consideration during the FUOTA process. FUOTA is vulnerable to DoS attacks when the attacker sends multiple forged signature packets, overloading LoRa nodes performing intensive computation for signature verification. To mitigate this vulnerability, we leverage the cyclical patterns in legitimate FUOTA packet arrival times. As shown in Figure 11, legitimate LoRa gateway packets exhibit cyclical arrival patterns, while the forged signature packets from the attacker have disorderly patterns. To this end, we introduce an arrival time pattern verification step before proceeding

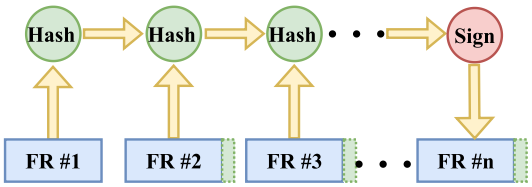


Fig. 10. Progressive hash chain verification step. The hash value for each firmware record is integrated into the subsequent one to form a hash chain, and the last firmware record is signed.

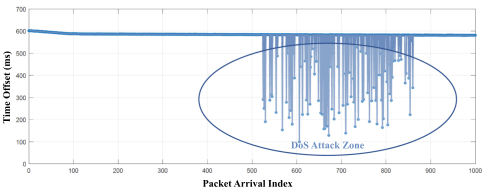


Fig. 11. Arrival time pattern verification step. The points in the DoS attack zone exhibit random arrival time patterns compared with those legitimate ones.

with the signature verification process. Incorporating this step can serve as a weak authentication approach to maintain firmware image availability, which effectively mitigates DoS attacks and prevents attackers from overwhelming the system with forged packets.

Specifically, upon receiving the FUOTA command packet, LoRa nodes record the arrival time of the packets during the early period of the FUOTA process. Afterward, they select a reference time point t_{ref} and calculate the time cycle T_{ref}. The attacker is considered unable to launch an attack during this early period, as having to perform traffic detection first. To verify arrival time patterns, LoRa nodes first compute the time offset of each packet with reference time point t_{ref}, then take the modulo operation with the time cycle T_{ref}, and finally compare it with the pre-defined threshold. We set the threshold to $0.1 \times T_{ref}$. Such an arrival time pattern verification step can effectively and efficiently identify these forged signature packets without verifying complex signatures, which ensures the firmware image availability.

In addition to the aforementioned injection and encoding attacks, certain measures can be incorporated. Specifically, one effective measure against injection attacks is to sign the command packet, making it computationally infeasible for an attacker to forge or tamper with it, provided they do not possess the required private key for verification. To mitigate encoding attacks, it is required to sign all redundancy packets to prevent from failing to reconstruct the source firmware records. Overall, the proposed method employed in the FUOTA process contributes to maintaining firmware image integrity and availability.

5 EVALUATION

In this section, we evaluate the performance of *FLoRa+*, including the experimental setup (Section 5.1), overall performance (Section 5.2), and microbenchmarking (Section 5.3).

5.1 Experimental Setup

We now move to evaluate *FLoRa+*. We first describe a LoRa testbed we build, then present the benchmarks of *FLoRa+* and the existing solution in LoRaWAN, followed by a set of performance metrics.

Testbed. We implement *FLoRa+* in a LoRa testbed with **Commercial Off-The-Shelf (COTS)** LoRa devices and deploy the testbed on our campus to evaluate the performance of *FLoRa+* for a period of 1 month. Figures 12 and 13 show our experimental setup. The testbed is composed of 1 LoRa transmitter (gateway) and 20 LoRa receivers (end devices), deployed in both outdoor and indoor scenarios. Each LoRa transceiver is assembled by integrating an Arduino ATmega328P **microcontroller unit (MCU)**, an SD card shield with 8 GB capacity, a Dragino LoRa SX1276 Shield operating at 868 MHz, and an omnidirectional antenna with 2 dBi gain. Then, each transceiver is connected to a Raspberry Pi 4b single-board computer supplied by the battery via a USB port.

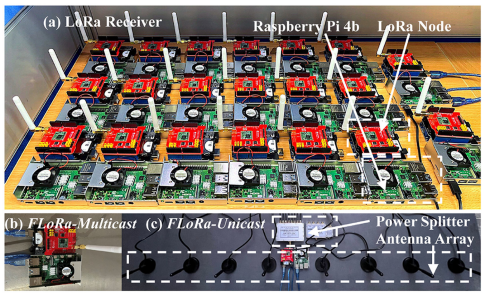
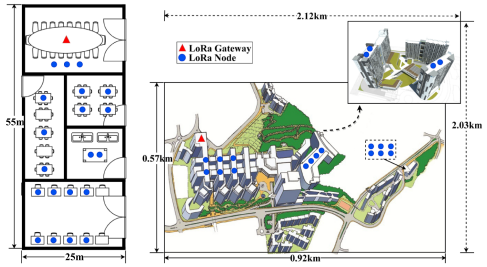


Fig. 12. Hardware used in our testbed. (a) illustrates the LoRa receiver, (b) and (c) illustrate the LoRa transmitter (i.e., *FLoRa+-Multicast* and *FLoRa+-Unicast* represent the FUOTA process in the multicast and unicast manner, respectively).



(a) Indoor layout. (b) Outdoor layout.

Fig. 13. Experimental layout of the LoRa testbed. (a) illustrates the indoor layout with an area of 25 m × 55 m, and (b) illustrates the outdoor layout with an area of 2.12 km × 2.03 km, where most nodes are placed in a region with an area of 0.92 km × 0.57 km.

All the Raspberry Pis are connected to campus Wi-Fi for experiment control and result visualization. It is noted that the size of the testbed is comparable to several state-of-the-art works, such as CurvingLoRa [34] (12 nodes, 1.45 km × 0.35 km outdoors) and FTrack [61] (20 nodes, 47 m × 20 m indoors).

Benchmarks. We compare *FLoRa+* with the existing solution in LoRaWAN, i.e., *Baseline*. *Baseline* adopts LDPC as the error erasure code with its coding rate set to 5%–15%, which means *Baseline* can transmit 5%–15% redundancy packets for error recovery. Specifically, we utilize *FLoRa+-Multicast* and *FLoRa+-Unicast* to represent the FUOTA process in the multicast and unicast manner, respectively. Additionally, *FLoRa+-Security* represents securing the FUOTA process.

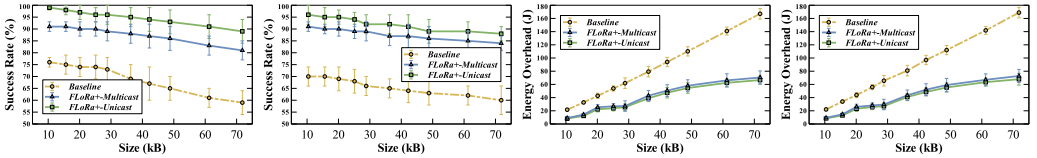
Performance Metrics. We use the following performance metrics:
(1) Success Rate (SR) is defined as the ratio of successfully rebooted LoRa nodes to the total number of ones, which measures the transmission reliability during FUOTA tasks.
(2) Energy Overhead (EO) is defined as the average energy consumption of all LoRa nodes in the testbed for packet receiving and inner decoding, which measures the energy efficiency during FUOTA tasks. EO is measured by the Monsoon Power Monitor [50].
For each metric, we report the average results and the standard deviations by conducting sufficient times.

5.2 Overall Performance

We evaluate the performance of *FLoRa+* with different firmware image sizes and LoRa configuration parameters.

5.2.1 Evaluation of Firmware Image Size. **Setup.** We compare the performances of *FLoRa+* with *Baseline* in 10 firmware image sizes, increasing from 10.40 kB to 71.83 kB. By default, our experiments are conducted with LoRa parameter settings of $SF = 9$, $BW = 125$ kHz, and $TP = 17$ dBm, where SF represents the spreading factor, BW is the bandwidth, and TP is the transmission power. Each firmware record in the firmware image is modulated as one LoRa packet to be sent at intervals. The interval is set to 0.5 s to meet the requirements for greater than the LoRa packet air-time.

Result. Figures 14(a) and 14(b) show the evaluation results of different firmware image sizes on SR. We observe that the SR-Size curves of *Baseline*, *FLoRa+-Multicast*, and *FLoRa+-Unicast* show a similar decreasing trend as size increases. The performance of *Baseline* shows a significant decrease at



(a) SR-Size curves outdoors. (b) SR-Size curves indoors. (c) EO-Size curves outdoors. (d) EO-Size curves indoors.

Fig. 14. Overall performance of *FLoRa+* under different firmware image sizes. (a) and (b) illustrate the SR-Size curves outdoors and indoors, respectively. Likewise, (c) and (d) illustrate the EO-Size curves. The solid line and dashed line represent the performances of *FLoRa+* and *Baseline*, respectively.

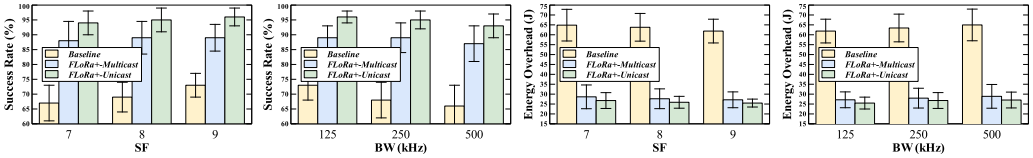
a large size, but *FLoRa+-Multicast* and *FLoRa+-Unicast* demonstrate robustness. The SR of *Baseline* is 0.76 at a size of 10.40 kB and 0.59 at a size increased to 71.83 kB in an outdoor environment, while the SR of *FLoRa+-Multicast* is ranged from 0.91 to 0.81. The mean SR of *FLoRa+-Unicast* reaches 0.95 due to the use of beamforming for the unicast group. The performance boosted by *FLoRa+-Unicast* based on *FLoRa+-Multicast* indoors is slightly lower than that outdoors. This may be due to multipath effects in an indoor environment. Thus, both *FLoRa+-Multicast* and *FLoRa+-Unicast* have obtained consistently higher SR than *Baseline* for all firmware sizes. Overall, the results verify the reliability of *FLoRa+*, which improves 1.30–1.51× outdoors and 1.36–1.47× indoors than *Baseline*.

Similarly, Figures 14(c) and 14(d) show the evaluation results of different firmware image sizes on EO. The EO-Size curves of *Baseline*, *FLoRa+-Multicast*, and *FLoRa+-Unicast* show a similar increasing trend as size increases. The EO of *Baseline* is 21.67 J at a size of 10.40 kB and 167.02 J at a size increased to 71.83 kB, while that of *FLoRa+-Multicast* is ranged between 9.06 J and 70.45 J outdoors. Both *FLoRa+-Multicast* and *FLoRa+-Unicast* have obtained consistently lower EO than *Baseline*, especially at a large size. For example, the EO of *Baseline* at a size of 25.12 kB is close to that of *FLoRa+-Multicast* at a size of 42.24 kB outdoors. The EO of *FLoRa+-Unicast* is slightly lower than that of *FLoRa+-Multicast*, since beamforming offers better link quality and a smaller number of redundancy packets are sent. The results verify the energy efficiency of *FLoRa+*, which improves 1.91–2.65× outdoors and 1.90–2.60× indoors than *Baseline*.

In addition, we provide an intuitive EO comparison using battery life as follows: The EO of *Baseline* and *FLoRa+-Unicast* is 61.87 J and 25.49 J when the firmware image is 28.82 kB, respectively. Taking a 5V battery with a capacity of 500 mAh (i.e., 9,000 J) as the example, the energy costs of *Baseline* and *FLoRa+-Unicast* amount to 0.69% and 0.28% of the total energy supply of the battery. In this case, a LoRa node can complete 145 times FUOTA tasks under *Baseline* and 353 times under *FLoRa+*, respectively. These results demonstrate that *FLoRa+* incurs a low EO and is more efficient than the existing solution in LoRaWAN.

5.2.2 Evaluation of LoRa Configuration Parameters. **Setup.** We evaluate the performance of *Baseline* and *FLoRa+* with various LoRa configuration parameters, including 3 SFs (i.e., 7, 8, and 9) and 3 BWs (i.e., 125 kHz, 250 kHz, and 500 kHz). By default, our experiments are conducted using a firmware image with a size of 28.82 kB outdoors.

Result. Figures 15(a) and 15(b) show the evaluation results of different LoRa configuration parameters on SR. For SF, we observe that the SR of *Baseline*, *FLoRa+-Multicast*, and *FLoRa+-Unicast* show a slight increase as SF increases. The SR of *Baseline* is 0.67 at SF=7 and 0.73 at SF=9, while that of *FLoRa+-Multicast* is ranged between 0.88 and 0.89. This is because a larger SF incurs a lower data rate but provides better resilience for interference and fading. Additionally, the characteristic of robustness to packet loss allows *FLoRa+* to show resilience at low SF configuration. For BW, we observe that *Baseline*, *FLoRa+-Multicast*, and *FLoRa+-Unicast* show a slight decrease as BW increases. The SR of *Baseline* is 0.73 at BW=125 kHz and 0.66 at BW=500 kHz, while that of *FLoRa+-Multicast*



(a) SR-SF histograms. (b) SR-BW histograms. (c) EO-SF histograms. (d) EO-BW histograms.

Fig. 15. Overall performance of *FLoRa+* under different LoRa configuration parameters. (a) and (b) illustrate the SR under different SFs and BWs, respectively. Likewise, (c) and (d) illustrate the EO under the same settings.

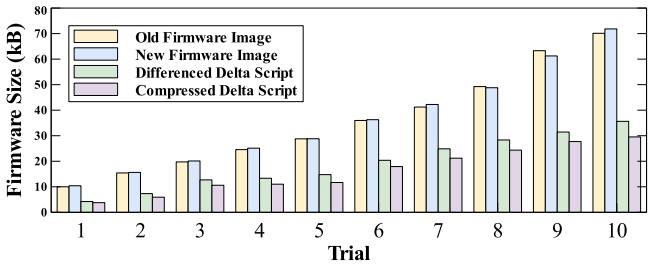


Fig. 16. Performance of delta scripting under different firmware sets.

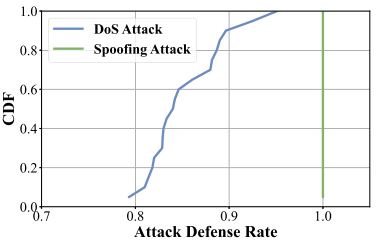


Fig. 17. CDF of for the attack defense rate.

is ranged from 0.89 to 0.87. This is because a higher BW provides a higher data rate but lower sensitivity due to the additional noise integrated. Thus, both *FLoRa+-Multicast* and *FLoRa+-Unicast* have obtained consistently higher SR than *Baseline* under different configuration parameters.

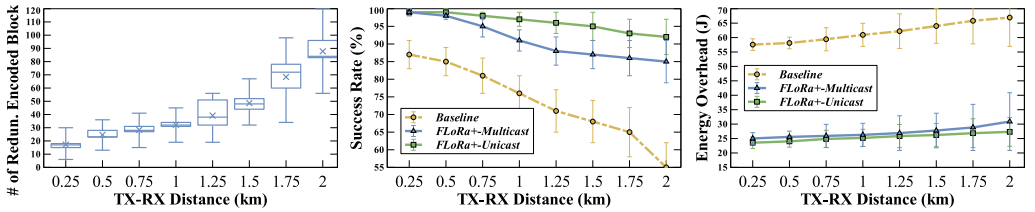
Figures 15(c) and 15(d) show the evaluation results of different LoRa configuration parameters on EO. For SF, we observe that *Baseline*, *FLoRa+-Multicast*, and *FLoRa+-Unicast* all show a slight decrease as SF increases. This is because a higher SF incurs less packet loss, thus fewer redundancy packets are sent for *FLoRa+*. The EO of *Baseline* is 64.86 J at SF=7 and 61.87 J at SF=9, while that of *FLoRa+-Multicast* is ranged between 28.64 J and 27.14 J. For BW, we observe that *Baseline*, *FLoRa+-Multicast*, and *FLoRa+-Unicast* all show a slight increase as BW increases. Likewise, a higher BW incurs more redundancy packets for *FLoRa+*. The EO of *Baseline* is 61.87 J at BW=125 kHz and 64.97 J at BW=500 kHz, while that of *FLoRa+-Multicast* is ranged between 27.14 J and 28.89 J. For different LoRa configuration parameters, both *FLoRa+-Multicast* and *FLoRa+-Unicast* have obtained consistently lower EO than *Baseline*.

5.3 Microbenchmarking

We now evaluate *FLoRa+* with respect to delta scripting (Section 5.3.1), channel coding (Section 5.3.2), beamforming (Section 5.3.3), ablation study (Section 5.3.4), computational overhead (Section 5.3.5), and security analysis (Section 5.3.6). Specified, the experiments will use the default settings in Section 5.2.

5.3.1 Evaluation on Delta Scripting. **Setup.** We evaluate the performance of delta scripting on 10 trials of firmware image sets with a size varying from 10.40 kB to 71.83 kB. By default, the new firmware image is a random modification of the old firmware, thus possessing a similar size to the old one.

Result. Figure 16 shows the evaluation results of delta scripting with 10 trials of firmware image sets. We observe that *FLoRa+* consistently achieves a much smaller size of differenced and compressed delta script compared with the new firmware image. For example, in trial 5



(a) Link quality. (b) SR-Distance curves. (c) EO-Distance curves.

Fig. 18. Performance of channel coding under different TX-RX distances, including link quality, SR, and EO, respectively.

with the new firmware at a size of 28.82 kB and the old one at a size of 28.78 kB, the sizes of the differenced and the compressed delta script are 14.76 kB and 11.67 kB, respectively. For all firmware image sets, the size of the differenced delta script decreases by 36.8%–59.4% on the basis of the new firmware, while that of the compressed one further decreases by 10.4%–20.9% based on the differenced one. This verifies the effectiveness of our delta scripting algorithm.

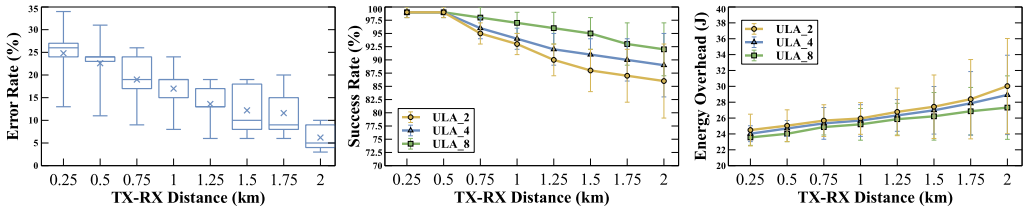
5.3.2 Evaluation on Channel Coding. **Setup.** We evaluate the performance of channel coding under eight different deployed TX-RX distances that vary from 0.25 km to 2 km.

Result. Figure 18(a) shows the results of link quality under eight TX-RX distances, represented by the number of redundancy encoded blocks sent of *FLoRa+-Multicast*. We observe that the box plot shows an increasing trend as distance increases, which means link quality becomes worse. The average number of redundancy encoded blocks sent is 17.2 at a distance of 0.25 km, while becomes 87.8 at a distance of 2 km; the reason is that the occurrence of packet loss and symbol error tends to increase the number of redundancy blocks as TX-RX distance increases. This also suggests the necessity of reliable channel coding during FUOTA tasks.

Figure 18(b) shows the evaluation results of eight TX-RX distances on SR. We observe that the SR-Distance curves of *Baseline*, *FLoRa+-Multicast*, and *FLoRa+-Unicast* all show a decreasing trend as distance increases. For TX-RX distances ranging between 0.25 km and 2 km, the SR of *Baseline* is ranged from 0.87 to 0.55, which encounters a significant decrease with a higher distance. While that of *FLoRa+-Multicast* and *FLoRa+-Unicast* are ranged from 0.99 to 0.85 and 0.99 to 0.92, which are consistently higher than *Baseline* due to the proposed reliable channel coding scheme. For example, the SR of *Baseline* at a distance of 0.25 km is close to that of *FLoRa+-Multicast* at a distance of 1.25 km. The SR of *FLoRa+-Unicast* approaches an average value of 95.1% due to the use of beamforming.

Figure 18(c) shows the evaluation results of eight TX-RX distances on EO. The EO-Distance curves of *Baseline*, *FLoRa+-Multicast*, and *FLoRa+-Unicast* also show a similar increasing trend as distance increases. The EO of *Baseline* is 57.56 J at a distance of 0.25 km, while it becomes 66.94 J at a distance of 2 km. Both *FLoRa+-Multicast* and *FLoRa+-Unicast* have obtained consistently lower EO than *Baseline*, especially at a longer TX-RX distance. The EO of *FLoRa+-Multicast* and *FLoRa+-Unicast* are ranged from 25.02 J to 30.88 J and 23.56 J to 27.32 J, respectively. This verifies the effectiveness of our concatenated channel coding scheme.

5.3.3 Evaluation on Beamforming. **Setup.** We evaluate the performance of beamforming under different numbers of ULA antennas. Specifically, we deploy three ULAs with 2, 4, and 8 antennas, denoted as ULA_2, ULA_4, and ULA_8, respectively. Each antenna is calibrated to be placed half a wavelength (i.e., 17.28 cm) away from its adjacent one. LoRa transmitting signal is fed to the ULA through an equal power splitter.



(a) Error rate of beamforming strategy for ULA_8.

(b) SR-Distance curves.

(c) EO-Distance curves.

Fig. 19. Performance of beamforming under different numbers of ULA antennas. (a) illustrates the error rate of the beamforming strategy. (b) and (c) illustrate the SR and EO, respectively.

Table 1. Ablation Study of *FLoRa+*

Module	**Delta Scripting**	✗	✓	✗	✗	✓	✓
	Channel Coding	✗	✗	✓	✗	✓	✓
	Beamforming	✗	✗	✗	✓	✗	✓
Metric	**SR (%)**	73	75	86	82	89	96
	EO (J)	61.87	26.31	68.8	54.76	27.14	25.49

Result. Figure 19(a) shows the evaluation results of the beamforming strategy of ULA_8 under eight TX-RX distances. Specifically, we utilize the error rate, the ratio of the absolute error to the maximum measurement range (i.e., 360°), as the metric to indicate the accuracy of the proposed beamforming strategy. We observe that the error rates show a decreasing trend as distance increases. The beamforming strategy has an average error rate of 24.80% when the TX-RX distance is 0.25 km. As TX-RX distance increases, the average error rate reaches 6.20% at a distance of 2 km. This means that the accuracy of the azimuth estimation method is susceptible to interference by other probing packets at a close distance. The low error rate level verifies the effectiveness of our beamforming strategy.

Figure 19(b) shows the evaluation results of eight TX-RX distances on SR. We observe that the SR-Distance curves of ULA_2, ULA_4, and ULA_8 all show a decreasing trend as distance increases. For ULA_2, the SR is 0.99 at a distance of 0.25 km, but it decreases to 0.86 at a distance of 2 km; while that of ULA_4 and ULA_8 are ranged from 0.99 to 0.89 and 0.99 to 0.92, respectively, which are consistently higher than ULA_2 due to better link quality offered. For example, the SR of ULA_2 at a distance of 0.75 km is close to that of ULA_8 at a distance of 1.5 km.

Figure 19(c) shows the evaluation results of eight TX-RX distances on EO. The EO-Distance curves of ULA_2, ULA_4, and ULA_8 show a similar increasing trend as distance increases. The EO of ULA_2 is 24.49 J at a distance of 0.25 km, while it becomes 30.03 J at a distance of 2 km. Both ULA_4 and ULA_8 have obtained consistently lower EO than *Baseline*, especially at a longer TX-RX distance. The EO of ULA_4 and ULA_8 are ranged from 24.03 J to 28.92 J and 23.56 J to 27.32 J, respectively; this is because a larger number of ULA antennas can result in a higher power gain.

5.3.4 Ablation Study. **Setup.** We add an ablation study to show the impact of independent modules of *FLoRa+* on the effectiveness of FUOTA tasks.

Result. Table 1 shows the results of the ablation study. We observe that these three key components of *FLoRa+* contribute to the performance gain of FUOTA tasks. Specifically, the delta scripting algorithm can significantly reduce energy consumption, from 61.87 J to 26.31 J of EO. The channel coding scheme is mainly beneficial to increase transmission reliability, from 0.73 to 0.86

Table 2. Computational Overhead of *FLoRa+*, Indicated by the Running Time and Energy Consumption

Trial	New Firmware (kB)	Delta Script (kB)	Running Time (s)	Energy Consumption (J)
1	10.40	3.78	0.125	1.376
2	15.63	5.92	0.128	1.514
3	20.11	10.60	0.136	1.553
4	25.12	11.02	0.140	1.763
5	28.82	11.67	0.145	1.821
6	36.28	17.92	0.166	1.993
7	42.24	21.20	0.172	2.081
8	48.80	24.36	0.178	2.113
9	61.25	27.72	0.184	2.146
10	71.83	29.53	0.195	2.237

of SR; while beamforming results in enhanced performances with respect to both SR and EO due to better link quality offered.

5.3.5 Computational Overhead. **Setup.** We now evaluate the computational overhead by running the firmware reconstruction algorithm on the online server of the Raspberry Pi single-board computer. The computational overhead is indicated by the running time and energy consumption, both of which are averaged by measuring 20 times.

Result. Table 2 illustrates the results of the computational overhead of *FLoRa+* under different firmware sizes. We observe that the running time increases as firmware size goes up. The running time is 0.125 s at a size of 3.78 kB, while 0.195 s at a size of 29.53 kB. Similarly, energy consumption increases as running time rises. The energy consumption is 1.376 J at a size of 3.78 kB, while 2.237 J at a size of 29.53 kB. The results show that firmware reconstruction can be finished in 0.2 s at a low energy consumption level. The low level of computational overhead verifies the efficiency of *FLoRa+*.

5.3.6 Security Analysis. **Setup.** We now perform the security analysis of *FLoRa+-Security*. We evaluate the effectiveness of the proposed securing mechanism by launching spoofing and DoS attacks. For spoofing attacks, the attacker randomly revises the default firmware records, computes the corresponding hash functions, then transmits these packets to legitimate LoRa nodes. For DoS attacks, the attacker transmits 200 forged signature packets in a random time delay during the transmission of 1,000 legitimate signature ones.

Result. Figure 17 illustrates the results of the **Cumulative Distribution Function (CDF)** for the attack defense rate. Regarding spoofing attacks, the attack defense rate of *FLoRa+-Security* reaches 100% for those successfully rebooted LoRa nodes, as the attacker cannot generate the forged packets that possess the hash values recorded in the hash chain. It is noted that for those LoRa nodes that are not rebooted, their attack defense rate is not recorded, since they cannot obtain the complete hash chain for verification. In the case of DoS attacks, the *FLoRa+-Security* can mitigate an average of 85.4% of the DoS attack packets. These findings highlight the effectiveness of the proposed securing mechanisms in protecting the FUOTA process against different types of attacks.

6 DISCUSSION

We discuss a few issues and future work here.

Beamforming. Antenna arrays and beamforming techniques can offer better link quality, thus improving the success rate and energy efficiency of FUOTA tasks in LoRa networks. In our

experiments, we use ULAs with manually steering the azimuth angle to strengthen the signal in specific directions from a low-cost perspective. Thus, the experimental results are sensitive to the power loss of intermediate hardware elements and the non-uniformity of each antenna channel. To solve this problem, we plan to utilize the phased array antennas [29] in our further work to achieve precise beamforming by flexibly modifying the feed-in phase and amplitude of signals in each antenna.

In the meantime, additional hardware updates and maintenance costs should also be key considerations for commercial use. There appears to be a tradeoff between the requirement for multiple antennas in beamforming and the fact that commercial LoRa gateways are typically equipped with only two antennas.

Experimental Setting. The experimental setting is flexible and can be tailored to different scenarios. For example, *FLoRa+-Multicast* and *FLoRa+-Unicast* can serve as tradeoff options. *FLoRa+-Multicast* is suitable for large-scale LoRa networks in a short FUOTA time demand. However, *FLoRa+-Unicast* can be employed under link quality detection in advance, serving for small-scale nodes with subpar link quality, albeit over a longer time duration. Additionally, *FLoRa+-Security* is intended for applications with high security requirements.

Extensive Applications. With the growing development of LoRa technology in various application scenarios [53], FUOTA is becoming an indispensable functionality to facilitate the deployment of LoRa networks in a manpower-saving and cost-effective manner. However, different scenarios present unique requirements and challenges. For example, in transportation and robotics applications [66], node mobility is a critical factor to consider in FUOTA implementations. When a node is in motion, the link quality may significantly degrade, resulting in frequent transmission loss. As such, adapting FUOTA to different use cases can further promote their adoption in real-world applications.

7 RELATED WORK

In this section, we briefly review several aspects of related work.

LoRa. LoRa technology has been attracting significant efforts due to its promising prospects, with respect to PHY demodulation [33, 58, 59], **Media Access Control (MAC)** protocols [14, 18], security [23, 49], and applications [19, 26, 68]. For PHY demodulation, NELoRa [33] adopts deep learning networks for the extracted fine-grained LoRa chirp to achieve a high upper limit of the SNR gain. Besides, LoRa parallel demodulation capability also receives much attention [54, 60]. For example, NScale [54] leverages LoRa packet structure and chirp feature in time and frequency domains for collision disambiguation. For MAC protocol, LMAC [14] and LoRaCP [18] empower LoRa with **Carrier Sense Multiple Access (CSMA)** and **Time Division Multiple Access (TDMA)** capabilities, respectively. For instance, LMAC designs three versions to implement basic CSMA capability and even acquire channels' crookedness information locally and broadly. Against various attacks, a large number of countermeasures, such as key generation [64, 66], authentication [17, 56], and packet recovery [23], were proposed. SLoRa [56] leverages **Carrier Frequency Offset (CFO)** and link signature features for node authentication. Additionally, recent attempts have built various LoRa-enabled applications, such as backscatter [20, 45], sensing [62, 68], and localization [7, 11]. These works are in parallel to *FLoRa+*.

OTA Programming. The existing OTAP protocols mainly target for communication technologies and network topology. Mature OTAP specifications and platforms are developed for legacy wireless communication technologies, including Wi-Fi, Zigbee, and Bluetooth [4]. Additionally, plenty of firmware dissemination protocols have been proposed for **Wireless Sensor Networks**

(WSNs), such as Trickle [32], Deluge [24], and Seluge [25]. However, these methods are designed for multi-hop WSN networks, which are hence not suitable for single-hop LoRaWAN networks. Compared with these works, *FLoRa+* fills the gap of OTAP in LoRa technology.

Incremental Update. As an energy-efficient option, incremental update mainly relies on the differencing algorithm to generate the delta script, including block-level [55] and byte-level [12, 39] ones based on the matching granularity [4]. Rsync [55] relies on a sliding window to find LCP in the fixed blocks, while R3diff [12] computes the hash values for every three continuous bytes. Unlike these methods, *FLoRa+* proposes a joint differencing and compression algorithm to minimize the size of the delta script.

Channel Coding. Channel coding is widely adopted for robust transmission. For example, DaRe [38] integrates convolution and fountain code for data recovery in LoRaWAN. OPR [6] exploits gateway spatial diversity with packet RSSI value for error detection while leveraging the **Message Integrity Check (MIC)** field for error correction in LoRa networks. Differently, *FLoRa+* designs a concatenated channel code for FUOTA downlink transmission, including outer rateless code and inner error detection code.

Beamforming. Beamforming can enhance the signal power in interested directions and enable **Space-Division Multiple Access (SDMA)**. Beamforming potential has been unlocked in LoRa sensing with respect to long-range and multi-target respiration monitoring [63, 68]. Differently, *FLoRa+* is the first work adopting beamforming to boost the performance of FUOTA tasks in LoRa networks.

Securing. Securing OTA programming to defend against various attacks is an essential task. In terms of firmware integrity attacks, hash chains incorporating digital signatures is a common solution to maintain the firmware integrity, as proposed in Seluge [25], Sluice [30], and Securing Deluge [13]. When addressing firmware availability attacks, Seluge [25] employs a weak authentication method based on message-specific puzzles to mitigate DoS attacks. Apart from progressive hash chain verification, *FLoRa+* designs an arrival time pattern verification step, tailored to single-hop and resource-constrained LoRa networks.

8 CONCLUSION

In this article, we propose *FLoRa+*, an energy-efficient, reliable, beamforming-assisted, and secure FUOTA system for LoRa networks. *FLoRa+* presents four essential techniques. First, we propose a joint delta scripting algorithm to unlock the potential of incremental FUOTA in LoRa networks. Then, we present a concatenated channel coding scheme to resolve the link quality indeterminacy. Afterward, we design a beamforming strategy to prevent biased multicast and compromised throughput. Last, we propose a securing mechanism to counter active firmware integrity and availability attacks. We conduct extensive experiments on a 20-node LoRa testbed to evaluate the performance of *FLoRa+*. The results illustrate that *FLoRa+* improves network transmission reliability by up to 1.51× and energy efficiency by up to 2.65× compared with the existing solution in LoRaWAN. Additionally, security analysis shows that *FLoRa+* can defend against 100% of spoofing attacks and mitigate 85.4% of DoS attacks.

REFERENCES

[1] Khaled Abdelfadeel, Tom Farrell, David McDonald, and Dirk Pesch. 2020. How to make firmware updates over LoRaWAN possible. In *IEEE WoWMoM*.

[2] Frederic Amiel, Karine Villegas, Benoit Feix, and Louis Marcel. 2007. Passive and active combined attacks: Combining fault attacks and side channel analysis. In *IEEE FDTC Workshop*.

[3] A. Anastasiou, Panayiotis Christodoulou, Klitos Christodoulou, Vasos Vassiliou, and Zinon Zinonos. 2020. IoT device firmware update over LoRa: The blockchain solution. In *IEEE DCOSS*.
[4] Konstantinos Arakadakis, Pavlos Charalampidis, Antonis Makrogiannakis, and Alexandros Fragkiadakis. 2021. Firmware over-the-air programming techniques for IoT networks—A survey. *ACM CComput. Surv.* 54, 9 (2021), 1–36.
[5] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. 2013. BLAKE2: Simpler, smaller, fast as MD5. In *Springer ACNS*.
[6] Artur Balanuta, Nuno Pereira, Swarun Kumar, and Anthony Rowe. 2020. A cloud-optimized link layer for low-power wide-area networks. In *ACM MobiSys*.
[7] Atul Bansal, Akshay Gadre, Vaibhav Singh, Anthony Rowe, Bob Iannucci, and Swarun Kumar. 2021. OwLL: Accurate LoRa localization using the TV whitespaces. In *ACM/IEEE IPSN*.
[8] Mihir Bellare and Tadayoshi Kohno. 2004. Hash function balance and its impact on birthday attacks. In *Springer EUROCRYPT*.
[9] Julien Catalano. 2021. LoRaWAN firmware update over-the-air (FUOTA). *J. ICT Stand.* (2021), 21–34.
[10] Andreas Crauser and Paolo Ferragina. 2002. A theoretical and experimental study on the construction of suffix arrays in external memory. *Algorithmica* 32 (2022), 1–35.
[11] Zhuangzhuang Dai, Muhamad Risqi U. Saputra, Chris Xiaoxuan Lu, Vu Tran, L. N. S. Wijayasingha, M. Arif Rahman, John A. Stankovic, Andrew Markham, and Niki Trigoni. 2022. Deep odometry systems on edge with EKF-LoRa backend for real-time indoor positioning. In *IEEE CPS-ER Workshop*.
[12] Wei Dong, Biyuan Mo, Chao Huang, Yunhao Liu, and Chun Chen. 2013. R3: Optimizing relocatable code for efficient reprogramming in networked embedded systems. In *IEEE INFOCOM*.
[13] Prabal K. Dutta, Jonathan W. Hui, David C. Chu, and David E. Culler. 2006. Securing the deluge network programming system. In *ACM/IEEE IPSN*.
[14] Amalinda Gamage, Jansen Christian Liando, Chaojie Gu, Rui Tan, and Mo Li. 2020. Lmac: Efficient carrier-sense multiple access for LoRa. In *ACM MobiCom*.
[15] Weifeng Gao, Wan Du, Zhiwei Zhao, Geyong Min, and Mukesh Singhal. 2019. Towards energy-fairness in LoRa networks. In *IEEE ICDCS*.
[16] Everette S. Gardner Jr. 1985. Exponential smoothing: The state of the art. *Int. J. Forecast.* 4, 1 (1985), 1–28.
[17] Chaojie Gu, Linshan Jiang, Rui Tan, Mo Li, and Jun Huang. 2021. Attack-aware synchronization-free data timestamping in LoRaWAN. *ACM Trans. Sensor Netw.* 18, 1 (2021), 1–31.
[18] Chaojie Gu, Rui Tan, and Xin Lou. 2019. One-hop out-of-band control planes for multi-hop wireless sensor networks. *ACM Trans. Sensor Netw.* 15, 4 (2019), 1–29.
[19] Xiuzhen Guo, Longfei Shangguan, Yuan He, Nan Jing, Jiacheng Zhang, Haotian Jiang, and Yunhao Liu. 2022. Saiyan: Design and implementation of a low-power demodulator for LoRa backscatter systems. In *USENIX NSDI*.
[20] Xiuzhen Guo, Longfei Shangguan, Yuan He, Jia Zhang, Haotian Jiang, Awais Ahmad Siddiqi, and Yunhao Liu. 2020. Aloba: Rethinking ON-OFF keying modulation for ambient LoRa backscatter. In *ACM SenSys*.
[21] Han Han, Yihong Liu, and Lei Zhang. 2020. On half-power beamwidth of intelligent reflecting surface. *IEEE Commun. Lett.* 25, 4 (2020), 1333–1337.
[22] Derek Heeger, Maeve Garigan, Eirini Eleni Tsiropoulou, and Jim Plusquellic. 2021. Secure LoRa firmware update with adaptive data rate techniques. *Sensors* 21, 7 (2021), 2384.
[23] Ningning Hou, Xianjin Xia, and Yuanqing Zheng. 2021. Jamming of LoRa PHY and countermeasure. In *IEEE INFOCOM*.
[24] Jonathan W. Hui and David Culler. 2004. The dynamic behavior of a data dissemination protocol for network programming at scale. In *ACM SenSys*.
[25] Sangwon Hyun, Peng Ning, An Liu, and Wenliang Du. 2008. Seluge: Secure and DoS-resistant code dissemination in wireless sensor networks. In *ACM/IEEE IPSN*.
[26] Jinyan Jiang, Jiliang Wang, Yijie Chen, Yihao Liu, and Yunhao Liu. 2023. LocRa: Enable practical long-range backscatter localization for low-cost tags. In *ACM MobiSys*.
[27] Juha Kärkkäinen and Peter Sanders. 2003. Simple linear work suffix array construction. In *ICALP*.
[28] Naoto Kimura and Shahram Latifi. 2005. A survey on data compression in wireless sensor networks. In *IEEE ITCC*.
[29] Shajahan Kutty and Debarati Sen. 2015. Beamforming for millimeter wave communications: An inclusive survey. *IEEE Commun. Surv. Tutor.* 18, 2 (2015), 949–973.
[30] Patrick E. Lanigan, Rajeev Gandhi, and Priya Narasimhan. 2006. Sluice: Secure dissemination of code updates in sensor networks. In *IEEE ICDCS*.
[31] David Lecompte and Frédéric Gabin. 2012. Evolved multimedia broadcast/multicast service (eMBMS) in LTE-advanced: Overview and Rel-11 enhancements. *IEEE Commun. Mag.* 50, 11 (2012), 68–74.
[32] Philip Levis, Neil Patel, David Culler, and Scott Shenker. 2004. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *USENIX NSDI*.

[33] 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 *ACM SenSys*.
[34] Chenning Li, Xiuzhen Guo, Longfei Shangguan, Zhichao Cao, and Kyle Jamieson. 2022. CurvingLoRa to boost LoRa network throughput via concurrent transmission. In *USENIX NSDI*.
[35] Jansen C. Liando, Amalinda Gamage, Agustinus W. Tengourtius, and Mo Li. 2019. Known and unknown facts of LoRa: Experiences from a large-scale measurement study. *ACM Trans. Sensor Netw.* 15, 2 (2019), 1–35.
[36] Jun Liu, Jiayao Gao, Sanjay Jha, and Wen Hu. 2021. Seirios: Leveraging multiple channels for LoRaWAN indoor and outdoor localization. In *ACM MobiCom*.
[37] Michael Luby. 2002. LT codes. In *IEEE FOCS*.
[38] Paul J. Marcelis, Vijay Rao, and R. Venkatesha Prasad. 2017. DaRe: Data recovery through application layer coding for LoRaWAN. In *IoTDI*. 97–108.
[39] Biyuan Mo, Wei Dong, Chun Chen, Jiajun Bu, and Qiang Wang. 2012. An efficient differencing algorithm based on suffix array for reprogramming wireless sensor networks. In *IEEE ICC*.
[40] Fionn Murtagh and Pedro Contreras. 2012. Algorithms for hierarchical clustering: An overview. *Wiley Interdiscip. Rev.: Data Min. Knowl. Discov.* 2, 1 (2012), 86–97.
[41] Tao Ni, Yongliang Chen, Weitao Xu, Lei Xue, and Qingchuan Zhao. 2023. XPorter: A study of the multi-port charger security on privacy leakage and voice injection. In *ACM MobiCom*.
[42] Tao Ni, Guohao Lan, Jia Wang, Qingchuan Zhao, and Weitao Xu. 2023. Eavesdropping mobile app activity via radio-frequency energy harvesting. In *USENIX Security*.
[43] Tao Ni, Jianfeng Li, Xiaokuan Zhang, Chaoshun Zuo, Wubing Wang, Weitao Xu, Xiapu Luo, and Qingchuan Zhao. 2023. Exploiting contactless side channels in wireless charging power banks for user privacy inference via few-shot learning. In *ACM MobiCom*.
[44] Tao Ni, Xiaokuan Zhang, Chaoshun Zuo, Jianfeng Li, Zhenyu Yan, Wubing Wang, Weitao Xu, Xiapu Luo, and Qingchuan Zhao. 2023. Uncovering user interactions on smartphones via contactless wireless charging side channels. In *IEEE S&P*.
[45] 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 *ACM SIGCOMM*.
[46] Simon J. Puglisi, William F. Smyth, and Andrew H. Turpin. 2007. A taxonomy of suffix array construction algorithms. *ACM Comput Surv.* 39, 2 (2007), 4–es.
[47] Yidong Ren, Li Liu, Chenning Li, Zhichao Cao, and Shigang Chen. 2022. Is LoRaWAN really wide? Fine-grained LoRa link-level measurement in an urban environment. In *IEEE ICNP*.
[48] Mark Shand and Jean Vuillemin. 1993. Fast implementations of RSA cryptography. In *IEEE ARITH*.
[49] Cheng Shen, Tian Liu, Jun Huang, and Rui Tan. 2021. When LoRa meets EMR: Electromagnetic covert channels can be super resilient. In *IEEE S&P*.
[50] Monsoon Solutions. 2021. High Voltage Power Monitor: Monsoon solutions: Bellevue. Retrievedfrom:https://www.msoon.com/high-voltage-power-monitor
[51] Yin Sun, C. Emre Koksal, Kyu-Han Kim, and Ness B. Shroff. 2014. Scheduling of multicast and unicast services under limited feedback by using rateless codes. In *IEEE INFOCOM*.
[52] Zehua Sun, Tao Ni, Huanqi Yang, Kai Liu, Yu Zhang, Tao Gu, and Weitao Xu. 2023. FLoRa: Energy-efficient, reliable, and beamforming-assisted over-the-air firmware update in LoRa networks. In *ACM/IEEE IPSN*.
[53] Zehua Sun, Huanqi Yang, Kai Liu, Zhimeng Yin, Zhenjiang Li, and Weitao Xu. 2022. Recent advances in LoRa: A comprehensive survey. *ACM Trans. Sensor Netw.* 18, 4 (2022), 1–44.
[54] Shuai Tong, Zhenqiang Xu, and Jiliang Wang. 2020. CoLoRa: Enabling multi-packet reception in LoRa. In *IEEE INFOCOM*.
[55] Andrew Tridgell. 1999. Efficient algorithms for sorting and synchronization. (1999).
[56] Xiong Wang, Linghe Kong, Zucheng Wu, Long Cheng, Chenren Xu, and Guihai Chen. 2020. SLoRa: Towards secure LoRa communications with fine-grained physical layer features. In *ACM SenSys*.
[57] Jack Wolf. 1978. Efficient maximum likelihood decoding of linear block codes using a trellis. *IEEE Trans. Inf. Theor.* 24, 1 (1978), 76–80.
[58] Xianjin Xia, Qianwu Chen, Ningning Hou, and Yuanqing Zheng. 2022. HyLink: Towards high throughput LPWANSs with LoRa compatible communication. In *ACM SenSys*.
[59] Xianjin Xia, Qianwu Chen, Ningning Hou, Yuanqing Zheng, and Mo Li. 2023. XCopy: Boosting weak links for reliable LoRa communication. In *ACM MobiCom*.
[60] Xianjin Xia, Ningning Hou, Yuanqing Zheng, and Tao Gu. 2021. PCube: Scaling LoRa concurrent transmissions with reception diversities. In *ACM MobiCom*.
[61] Xianjin Xia, Yuanqing Zheng, and Tao Gu. 2019. FTrack: Parallel decoding for LoRa transmissions. In *ACM SenSys*.

[62] Binbin Xie and Jie Xiong. 2020. Combating interference for long range LoRa sensing. In *ACM SenSys*.
[63] Binbin Xie, Yuqing Yin, and Jie Xiong. 2021. Pushing the limits of long range wireless sensing with LoRa. *ACM Interact. Mob. Wear. Ubiq. Technol.* 5, 3 (2021), 1–21.
[64] Weitao Xu, Sanjay Jha, and Wen Hu. 2018. LoRa-key: Secure key generation system for LoRa-based network. *IEEE Internet Things J.* 6, 4 (2018), 6404–6416.
[65] Ismail Yabanova, Sezai Taskin, Huseyin Ekiz, and Hasan Cimen. 2012. Bootloader design application for embedded systems by using controller area network. *Computer Science* 1 (2012), 952–957.
[66] Huanqi Yang, Zehua Sun, Hongbo Liu, Xianjin Xia, Yu Zhang, Tao Gu, Gerhard Hancke, and Weitao Xu. 2023. Chirp-Key: A chirp-level information-based key generation scheme for LoRa networks via perturbed compressed sensing. In *IEEE INFOCOM*.
[67] Qiang Yang, Kaiyan Cui, and Yuanqing Zheng. 2023. VoShield: Voice liveness detection with sound field dynamics. In *IEEE INFOCOM*.
[68] Fusang Zhang, Zhaoxin Chang, Jie Xiong, Rong Zheng, Junqi Ma, Kai Niu, Beihong Jin, and Daqing Zhang. 2021. Unlocking the beamforming potential of LoRa for long-range multi-target respiration sensing. *ACM Interact. Mob. Wear. Ubiq. Technol.* 5, 2 (2021), 1–25.

Received 1 November 2023; revised 10 January 2024; accepted 11 January 2024