Discovery privacy threats via device de-anonymization in LoRaWAN

Pietro Spadaccino, Domenico Garlisi, Francesca Cuomo,
Giorgio Pillon, Patrizio Pisani

PII: S0140-3664(22)00060-3
DOI: https://doi.org/10.1016/j.comcom.2022.02.017
Reference: COMCOM 7047

To appear in: *Computer Communications*

Received date : 30 October 2021
Revised date : 13 February 2022
Accepted date : 19 February 2022

Discovery privacy threats via device de-anonymization in LoRaWAN

Pietro Spadaccino[a], Domenico Garlisi[b,d], Francesca Cuomo[a,d], Giorgio Pillon[c] and Patrizio Pisani[c]

[a]DIET department, Sapienza University of Rome, Rome, Italy
[b]Department of Engineering, University of Palermo, Palermo, Italy
[c]UNIDATA S.p.A. Rome, Italy
[d]CNIT, Consorzio Nazionale Interuniversitario per le Telecomunicazioni, Parma, Italy

A B S T R A C T

LoRaWAN (Long Range WAN) is one of the well-known emerging technologies for the Internet of Things (IoT). Many IoT applications involve simple devices that transmit their data toward network gateways or access points that, in their turn, redirect data to application servers. While several security issues have been addressed in the LoRaWAN specification v1.1, there are still some aspects that may undermine privacy and security of the interconnected IoT devices. In this paper, we tackle a privacy aspect related to LoRaWAN device identity. The proposed approach, by monitoring the network traffic in LoRaWAN, is able to derive, in a probabilistic way, the unique identifier of the IoT device from the temporal address assigned by the network. In other words, the method identifies the relationship between the LoRaWAN DevAddress and the device manufacturer DevEUI. The proposed approach, named DEVIL (DEVice Identification and privacy Leakage), is based on temporal patterns arising in the packets transmissions. The paper presents also a detailed study of two real datasets: i) one derived by IoT devices interconnected to a prominent network operator in Italy; ii) one taken from the literature (the LoED dataset in [1]). DEVIL is evaluated on the first dataset while the second is analyzed to support the hypothesis under the DEVIL operation. The results of our analysis, compared with other literature approaches, show how device identification through DEVIL can expose IoT devices to privacy leakage. Finally, the paper also provides some guidelines to mitigate the user re-identification threats.

1. Introduction

The Internet of Things (IoT) paradigm is nowadays adopted by many applications and it involves several communication technologies. The IoT is revolutionizing the IT sector, and the Low Power Wide Area Network (LPWAN) technology is attracting several operators and service providers thus expecting to have a significant impact.
According to the IoT Analytics forecast, the number of connected IoT devices is growing from the 7B in 2018 to 22B in 2025 [2]. Moreover, in Europe, the ETSI TR 103 526 document [3] foresees that, in 2023, the density will reach 5582 devices per square kilometer and 3.5 gateways/access points for square kilometer. Beside, the IoT Analytics forecast shows the LPWAN will be the fastest growing IoT connectivity technology over the next years. It shows that four technologies account for about 92% of the global installed base of LPWAN-connected devices, namely LoRaWAN, Sigfox, NB-IoT and LTE-M.
In IoT context, LPWAN networks are often used for monitoring applications like metering services related to energy, water and gas. In general these applications are referred to information collected at home or in private spaces and as a consequence are sensible data. For instance, they might be proprietary business information that competitors could use to take advantage, or personal information regarding an organization's employees, customers or simply citizens. Here, the confidentiality of these data must be protected to maintain customers' privacy.
In this paper, we consider LoRaWAN, a well know technology for the IoT, and we look at the procedures to support device anonymization: no identification between device packets and associated user should be made available to third parties that monitor or eavesdrop the wireless channel where these packets are transmitted to.
LoRaWAN presents a basic architecture that involves simple devices transmitting their data towards network gateways (GWs). In LoRaWAN, several applications can be managed from a central network server. Security protection is provided through symmetric encryption at network and application level. Different security issues, arose in the first standard version, have been faced in the new LoRaWAN v1.1 release specification [4], like authentication, integrity protection, replay protection and encryption. However, there are still some weaknesses in the device anonymization and consequently have an impact on the privacy of the devices. In this paper, we tackle the privacy threats in LoRaWAN related to user re-identification, indeed, users identification can disclose information about their behaviors, such as presence of users and private practices.
We propose DEVIL (DEVice Identification and privacy Leakage) and show how it can detect user identity; we also provide some mitigations that can solve these issues. We refer to LoRaWAN Specification v1.1, where, as for the architecture, three main components are present (in Figure 1, from left to right):

1. End Device (ED): it is the simplest low-power device typically deployed to sense and act in the environment; it uses LoRa modulation to communicate with the

pietro.spadaccino@uniroma1.it (P. Spadaccino); domenico.garlisi@unipa.it (D. Garlisi); francesca.cuomo@uniroma1.it (F. Cuomo); g.pillon@unidata.it (G. Pillon); p.pisani@unidata.it (P. Pisani)
ORCID(s): 0000-0001-6256-2752 (D. Garlisi)

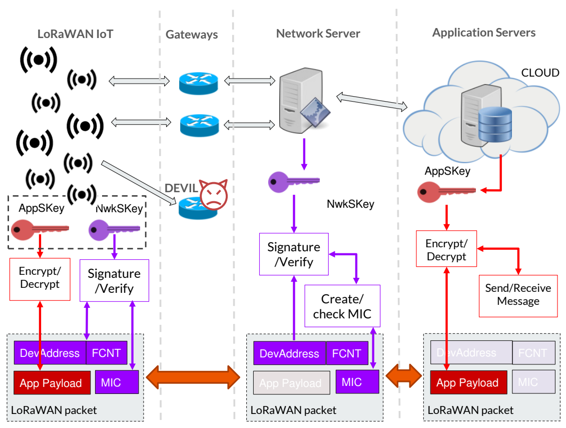

Figure 1: LoRaWAN V1.1 network elements and DEVIL position. Representation of the security protection through signed/symmetric encryption at network and application level.

gateway. It collects measurements from sensors and enforce commands to actuators;

2. Gateway (GW): it is the component that receives packets from the EDs and forwards them to the Network Server and vice-versa; it is connected to the network server through an IP backhaul. GWs can receive the same ED packet when they are deployed in the same geographical area and are in radio visibility of the transmitting ED.
3. Network Server (NS): it is the component that receives ED packets, it is responsible for de-duplicating and decoding packets and to forward to the application servers the application data. The same LoRaWAN network can support different applications server to implement their specific services.

The LoRaWAN network has a star-of-stars topology and the EDs are not associated with a specific GW.
GWs collect and forward to the NS any packet in the coverage area (including packets belonging to different operators), therefore any generic GW can monitor EDs packets to get user information and, in this case, privacy threats may occur. In other words, the incoming and outgoing traffic to and from the GW can be monitored in order to acquire ED information including user activities of the entities/people that are related to the ED. One of these aspects is the identification, by simply eavesdropping on the traffic, of the DeVEUI that is a globally unique identifier assigned by the manufacturer, or by the owner, to the ED. During the communications, for security and privacy reasons, the LoRaWAN protocol is designed to use a temporary address, named DevAddress. Hence the possibility to interpret this parameter is quite critical. Association between DevAddress and DevEUI produces user identification, and the consequent monitoring of the traffic activity may allow to derive users sensible information. For example, in case of IoT devices in an home, any movement or presence of people can be associated to the device generating them and hence to the home itself [5]. The DevEUI can be leveraged to infer information on the ED (manufacturer, type, or the associated application and sensor) that is only exposed during the association process. By linking a DevAddress with a DevEUI, an adversary could combine the information brought by each element and thus increase its knowledge on the ED, including its activities and the associated applications. In other words, an attacker will be able to infer sensitive information such as presence or absence of personnel within a building, etc. For example, the forwarded consumption data can be analyzed using load monitoring techniques to infer activities of the consumers [6]. These are sensitive data that must be protected for preserving the consumers' privacy [7].
DEVIL is based on periodic sequence matching, where, in a probabilistic way, it is possible to associate the DevAddress of the on-air packets to the relevant ED. In Figure 1, DEVIL is represented as a threat at GW level (adversary network node that eavesdrop on the ISM band). DEVIL can have also a positive use since its output can be used as a trusted service that provides information to mitigate re-identification threats or to optimize the resource utilization taking into account external traffic.
The contributions of this work are the following:

- it shows how, by eavesdropping on the LoRaWAN packets, it is possible to derive the DevEUI of the transmitting ED, and thus undermine the related privacy issue;
- it analyses different datasets from real application scenarios to validate the proposed approach; moreover a synthetic dataset has been built to test the robustness of the proposed approach;
- it compares the proposed approach with literature solutions;
- it provides a solution to mitigate the re-identification threat.

The paper presents the following structure. In Section 2 we briefly overview the LoRaWAN standard and its security aspects. The related work is in Section 3 while Section 4 describes real datasets used in our analysis. The model of the considered attacker and the features used to implement it are in Section 5. The core of the proposed approach is in Section 6 and the relevant numerical results in Section 7. Conclusions are reported in Section 9.

2. Security aspects in LoRaWAN

LoRaWAN is a Low Power WAN technology which enables power-efficient wireless communications over very long distances thanks to the LoRa modulation, originally specified by Semtech [8]. It operates on scientific and medical (ISM) radio bands that in the European region is EU863-870 in the frequency plan, providing 8 channels. LoRa defines the physical layer and it is a proprietary technology

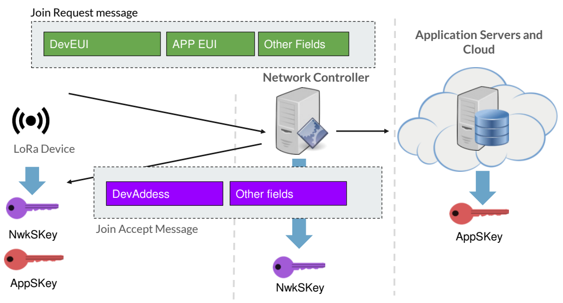

Figure 2: LoRaWAN join-procedure representation according to the Over-the-Air Activation (OTAA).

[8]. In LoRa, EDs support multi-channel and multi-rate communication. Seven different Spreading Factors (SFs) are used which are in principle orthogonal [9]. The combination of a high SF and a small bandwidth produces a more robust link communication, where link distances can be very large, in the order of tens of km. LoRaWAN specification defines the network layer: this specification is publicly available and it is promoted by the open-source LoRa Alliance [10]. LoRaWAN also provides optimization by Adaptive Data Rate (ADR) module. Starting, from the Signal to Noise Ratio (SNR) perceived by GW, the ADR works to set the best SF and transmission power values for each ED that optimize network performance in terms of capacity and energy efficiency.
LoRaWAN specification defines two different addresses. The DevEUI, is 64 bits unique during the ED's life and stored on nonvolatile memory. As described in [11] and summarized in Figure 1 (low part), symmetric keys known by EDs, NS and application server are used to encrypt and sign packets sent over the air. These keys are: i) the Application Session Key (AppSKey), used for encryption and decryption of the payload (the payload is fully encrypted between the ED and the Application Server); ii) the Network Session Key (NwkSKey), used by the NS to validate the integrity of each message by its Message Integrity Code (MIC check). These keys can be distributed in two different ways, based on the two available activation methods. They are: i) Over-the-Air Activation (OTAA) and ii) Activation by Personalization (ABP).
In OTAA, during the Join-procedure, security keys are negotiated with an ED, which receives a dynamic DevAddr. The join-procedure is shown in Figure 2. Each ED uses an Application Key (AppKey) to sign the message. The Join request message contains the JoinEUI and DevEUI of the ED followed by a randomly generated nonce (DevNonce). The NS is then able to trust the ED through the sign and the DevNonce to proceed to the next steps. The AppKey are AES-128 root keys specific to the ED that are assigned to the ED during fabrication. Whenever an ED joins a network via OTAA, the AppKey is used to derive the AppSKey session key at NS and application server. Afterward, the

Table 1
Key elements for the LoRaWAN security.

ELEMENT	DESCRIPTION
AppSKey (128 bit)	Application Session Key, representing the application session key that is used to encrypt all payloads via AES 128-bit algorithm.
NwkSKey (128 bit)	Network session key, representing the network session key that is used to compute a 32-bit cryptographic Message Integrity Check (MIC) signature via AES 128-bit algorithm.
JoinEUI (64 bit)	Unique ID of the Application server (named "AppEUI" in v1.0).
AppKey (128 bit)	is the encryption key between the source of the message (behind the DevEUI) and the destination of the message. This key must be unique for each ED.
DevEUI (64 bit)	Unique ID of the end ED. Announced only during the join procedure
DevAddress (32 bit)	Temporary address assigned by network

NS concludes the Join procedure by sending the Join accept message to the ED and notify the procedure to the application server. Join accept message provides the second temporary identifier, the device address (DevAddress), a 32 bits identifier generated by the Network.

The second method is the Activation by Personalization (ABP), in this case, each ED is already provided with DevAddr, NwkSKey and AppSKey, hence, the join-procedure is not necessary. This strategy, although simpler, has some drawbacks related to security. Table 1 lists the addresses and keys with the relative description.

3. Related works

As reported in [12], [13] and [14], LoRaWAN v1.0 suffers from several security vulnerabilities. These security breaches are partially solved in the LoRaWAN v1.1 specification.

Both ways of activation, presented before, provide a strong level of security using symmetric encryption in message exchanges between ED and NS. But potential attackers can steal the identity of the ED and enforce two different activities, ED impersonation and privacy leakage.

In the case of ED impersonation, an attacker may inject malicious EDs into the network in an attempt to take advantage of the network. In LoRaWAN, this translates into the possibility for an attacker to emulate the identity of an ED, resulting in the possibility of execution of typical actions of an ED (for example sending false measurements and generating false alerts).

A similar issue has been analyzed in [15] where a technique for identifying the DevAddress associated to a DevEUI is proposed. The approach is based on the time sequence of the Join request emitted by an ED and the first message emitted by the same ED after the Join request. This is a temporal linking method. Differently from that paper, we provide here a pattern-based approach that after a Join characterizes an ED by the whole sequence of transmitted packets and their pattern. Moreover, the temporal linking between the Join and the first uplink is not always constant. There could be the case that the NS asks the ED to immediately transmit a ReJoin request, via the MAC command ForceRejoinReq. Since this is a MAC command, it is not visible by the application, being instead handled by the LoRaWAN stack, thus the ReJoin message and the first application uplink may occur in different temporal periods. To demonstrate this consideration, we compare DEVIL with the algorithm presented in [15] and we show that in the best case the accuracy with our approach is improved of 58%. In the literature several papers were oriented to derive traffic analysis. Traffic classification plays an important role in network security and traffic attacks. [16] and [17] present schemes of IoT traffic classification features extraction, also for encrypted traffic. [18] and [19] propose a methodology to process LoRaWAN packets and perform profiling of EDs and clustering them according to their behavior. They analyze clusters' contents and alerts on malfunctioning EDs, remarking the reliability of the proposed approach. In this work we focus on LoRaWAN scenario and to the user re-identification threat.

Moreover, as stated in the introduction, DEVIL can be used also in a positive manner, as an additional module that supports the NS in the network mitigation and optimization operations. DEVIL may be an external engine that monitors the network traffic (by eavesdropping) and generates possible mitigation strategy, as well as network optimization performance based on the externally EDs profiling.

4. Analysis of datasets from real application scenarios

In this section we provide an analysis of different datasets from real application scenarios demonstrating that typical IoT application services, such as metering applications, produce traffic with periodic behavior.
In our study, we consider four datasets, two real LoRaWAN datasets collected by the UNIDATA S.p.A. IoT operator (it manages several application services in Italy), a real open dataset named LoRaWAN at the Edge Dataset (LoED) [1], and a synthetic dataset.
The two real datasets are referred to two IoT applications, an energy meter application service with 130 EDs, and a water meter application service with 300 EDs. Both the services are locate in Italy, and for both, we consider 18 months of operation in the period ranging from January 2019 to April 2020. Energy metering EDs use tick-counters connected to the distributor energy meters to detect energy consumption, water metering EDs have instead an embedded LoRaWAN transceiver. The energy meter application is served from one GW, and each ED forwards the measurement 1 time per hour, on average. The complete dataset contains 1,131,685

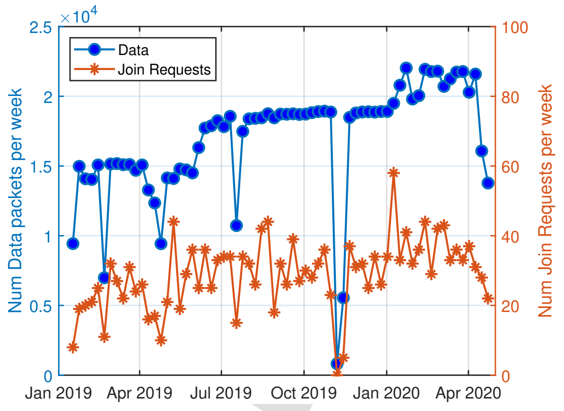

Figure 3: Number of received packets per week (blue line and left y axis), and number of Join request packets per week (red line and right y axis), in a period of 18 months for the energy metering applications service.

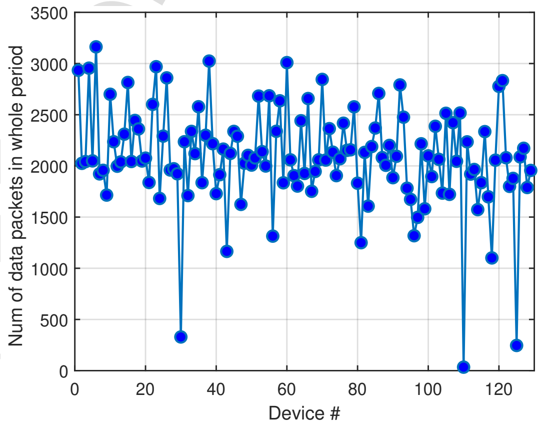

Figure 4: Number of received packets in a period of 18 months for the water meter application service.

packets. Figure 3 depicts (blue line and left y axis) the number of received packets per week of the whole network. Figure 3 also shows the number of Join request packets received per week (red line and right y axis). Regarding the water meter dataset, each ED generates about 18 packets per week. Two GWs cover the whole area where the 300 water meters are placed. Figure 4 shows the number of received packets for the first 130 EDs in the considered period (sorted by SNR). The complete dataset contains 929,587 packets. For our analysis, we consider EDs with a high average SNR to exclude devices with error rate more than 5% that may result in a wrong timing interpretation.

From the Figure 5 we can notice that the regular part has a cycle of 9 packets, where the first seven present an inter-arrival times of 12 hours. While the last 2 packets present a smaller temporal distance, but their inter-arrival times distance sum is equal to 12 hours (4 and 8 in the case of the

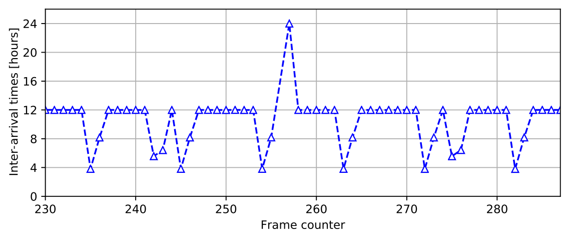

Figure 5: Time evolution of the inter-arrival times for the LoRaWAN water metering devices.

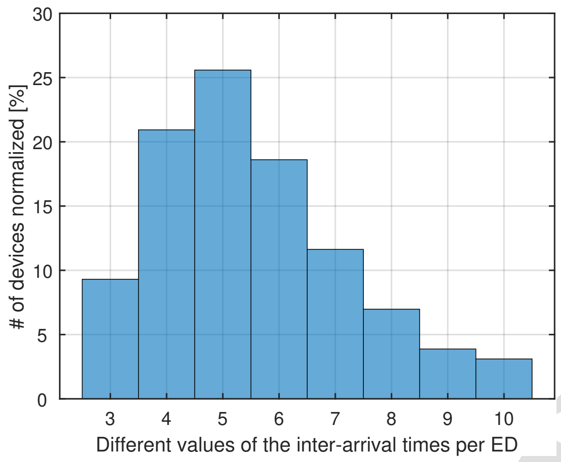

Figure 6: Distribution of different values of the inter-arrival times per ED, in a period of 18 months (we consider 95% of the total occurrences). We remark that the average number of packets for each ED in the reference period is 2000.

figure). The inter-arrival times of the eighth and ninth packet remain constant over time for the same device. The figure also shows the lack of a packet that has not been received by the NS (at 256 frame counter value). The transmission pattern also has a non regular part, the study carried out has allowed us to find the occasional existence of irregularities in the patterns, which however have a certain characteristic. Randomly one of the packets of the regular pattern can be replaced by two or more packets. The Figure 5 shows one of these situations (at 275th and 276th frame counter values).

From the above analysis, we deduce that the total number of different inter-arrival times is limited. In Figure 5 we show 57 inter-arrival times (56 packets, more than 3 weeks of operation) and only 5 different inter-arrival times values.

To better show this inter-arrival behavior, we selected the first 130 EDs (sorted by SNR). We isolated the packets sent by a device and constructed a sequence, named inter-arrival sequence, where the element i is the time elapsed between packet i and $i + 1$, accounting for lost packets as described in Section 5. Using this sequence, we can plot in Figure 6 the distribution of the number of distinct values of inter-arrival times per ED (we consider 95% of the total

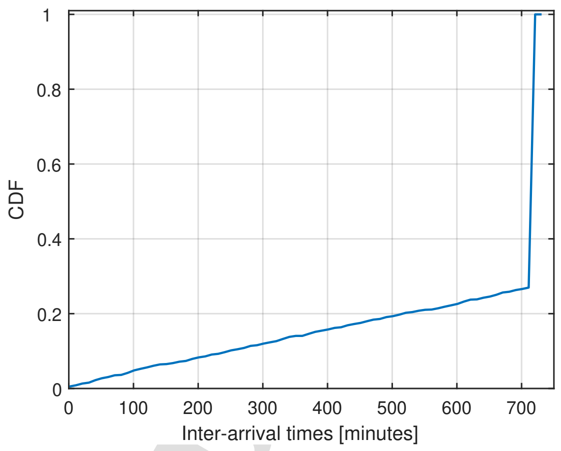

Figure 7: CDF of the inter-arrival times in a period of 18 months for the water metering application.

occurrences). From the figure, we note that the maximum number of different values of the inter-arrival times per ED is 10, moreover, more than 85% the occurrence is in the range between 3 and 7. Again, the higher bar is related to 5 different values of the inter-arrival times per ED that happens for the 25% of EDs, this value confirms the above analysis.

Finally, Figure 7 shows how the inter-arrival times values are distributed. According to the CDF, the 27% of the inter-arrival times are uniformed distributed in the range 10 minutes - 12 hours (720 minutes), while the remain 73% is represented by the 12 hours value. This result remarks the analysis of the pattern plotted in Figure 5. Same consideration can be provided for the energy meter dataset, but we avoid it for sake of space.

The third dataset is the LoED open dataset [1]. The dataset collected packets generated by smart city application and research deployment in a period time of 4 months. The packets were collected by 9 getaways in central London. During the acquisition time were collected 11,263,001 packets from devices with 145,023 different DevAddr. LoED is an open dataset that exposes insights into how LoRaWAN operates in real-world urban deployments. For our analysis, we filter the total number of device based on the SNR values, we consider EDs with high average SNR to avoid device with error rate more than 5%. According to this characteristic, we extract 500 EDs from the total number of EDs. In this subgroup, we find out two types of behaviors, EDs that have continuously transmissions, for example inter-arrival times in the range 1-10 minutes, 60% EDs in the subgroup belong to this type, and EDs with periodic behavior, 40% EDs in the subgroup belong to this type. Figure 8 shows the CDF of the inter-arrival times in the whole period for the EDs with the periodic behavior, from the figure we can notice that 40%

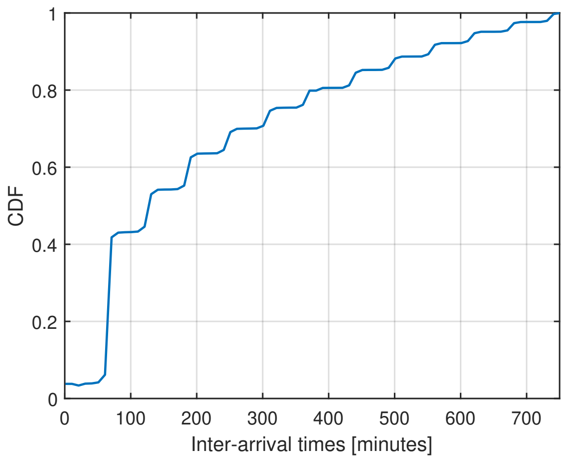

Figure 8: CDF of the inter-arrival times in the whole period of 4 months for 40% EDs in the LoED dataset.

of the inter-arrival times is equal to 60 minutes, other inter-arrival times are multiples of 60 minutes and confirm the periodical behavior of the packets transmission.

According to the previous analysis we confirm that the typical traffic generated from LoRaWAN EDs perform a periodic behavior. Based on this result and to better validate the proposed approach, we build a synthetic dataset, the scope of it is to produce a suitable dataset that enables parameters modification. We use the synthetic dataset to stress DEVIL in different conditions, where we extract its performance. We use the synthetic dataset to provide the result reported in Section 7.

The dataset is built by specifying the number of devices, denoted as N, and the maximum length of a periodic pattern, denoted as S. Each synthetic dataset generated is composed by $\approx 10^6$ packets. For each one of the N devices, we generate the inter-arrival time between each packet i and $i+1$ following this procedure: first we draw a time $\hat{t}_i$ from the inter-arrival time distribution extracted from the LoED dataset. Then we draw a random jitter uniformly distributed between 0 and $\alpha \cdot \hat{t}_i$ where α is a scalar. Finally, the inter-arrival time is:

$$t_i = \hat{t}_i + \text{uniform}\left(0, \alpha \cdot \hat{t}_i\right) \tag{1}$$

The scalar α is a noise parameter and regulates how much the generated inter-arrival times follow the underlying distribution extracted from the LoED dataset.

5. Attacker model and features identification

The relevant LoRaWAN fields used in DEVIL and the relative description, are reported in Table 2. These fields have been extracted and pre-processed and represent the key features considered in our analysis.

The targeted EDs should be LoRaWAN v1.1 compliant. In this scenario, an ED may sometimes send a Join or ReJoin request to the NS, which replies with a Join accept containing a new DevAddr for the ED.

Table 2
Key fields present in the datasets. Note that the field DEV_EUI is not used by DEVIL algorithm. Its purpose is to use it for cross checking and accuracy computation of the predictions of DEVIL. All the other fields are sent in cleartext in the LoRaWAN packet header and can be observed by passively listening the traffic.

PARAMETER	DESCRIPTION
DEV_ADDR	Unique identifier of the ED in the network
DEV_EUI	Unique identifier of the physical ED (None if the ED is unknown)
FCnt	Frame counter: counter (increased by 1 for each packet sent from an ED)
TMST	Time of arrival of the packet
TYPE	Type of packet, specifies if the packet is data or Join request

The goal of the attacker is to associate a LoRaWAN DevAddr to the real DevEUI of an ED, by passively monitoring the traffic. More accurately, let e be a DevEUI, a be a DevAddr and $A(e) = \{a_1, a_2, \dots\}$ be the set of DevAddr addresses assigned to the DevEUI e in a time span. The goal of the attacker is to find which DevAddr addresses were assigned to an ED DevEUI, i.e., to infer $A(e)$ for a target DevEUI e.

We started by analyzing the available data described in Section 4. We isolated the packets sent by a device and constructed a sequence, named inter-arrival sequence, where the element i is the time elapsed between two consecutive packets i and $i+1$. In order for two packets to be consecutive it is necessary that $\text{FCnt}_{i+1} = 1{+}\text{FCnt}_i$, where FCnt_i denotes the value of the FCnt field of packet i. This ensures that packet $i+1$ is the next packet sent after i by the same device. In this way, the inter-arrival sequence does not contain inter-arrival times related to packets that are not sent one after the other, which could happen in case of lost packets. Moreover, the attacker is able to only passively listen to the LoRaWAN traffic, without accessing to encrypted information at NS-level.

We found that many EDs send their packets following periodic temporal patterns i.e. the inter-arrival sequence has some periodicity. One example of such a temporal pattern, could be sending one packet every hour during the day and zero packets during the night. In the energy metering data, we have found that the majority of the EDs sends 24 packets, once every hour, then a 25-th packet after about 30 seconds, and then this pattern repeats.

For the scope of this work, we will treat the application as a black box, not giving the algorithm any application-specific information. An attacker carrying out the described de-anonymization in this scenario, is forced to handle the cumulative traffic composed by also non-targeted EDs, whose packets are listened on the air and which do not have to follow any temporal behavior. Such EDs create noise in the data and the attacker should be able to deal with it, since it

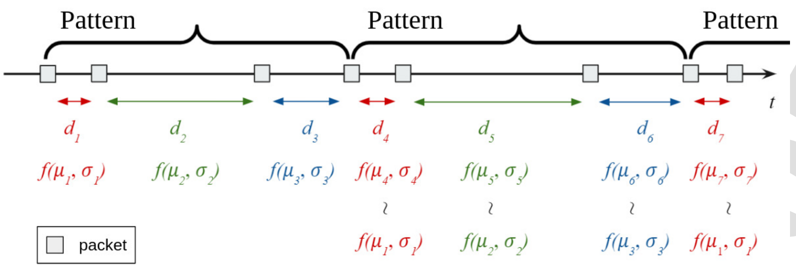

Figure 9: Example of modeling of the periodic sequence for the packet inter-arrival times. The line is the temporal axis, and the gray squares are packets belonging to a single ED. The time d_i that occurs between two consecutive packets p_i, p_{i+1} is distributed following a probability distribution $f(\mu_i, \sigma_i)$. The chain has period $\tau = 3$, with $\mu_i = \mu_{i+\tau}$ and $\sigma_i = \sigma_{i+\tau}$. The inter-arrival times highlighted in red d_1, d_4, d_7 follow the same distribution $f(\mu_1, \sigma_1)$, the ones in green d_2, d_5 follow another distribution $f(\mu_2, \sigma_2)$ and the ones in blue d_3, d_6 follow $f(\mu_3, \sigma_3)$.

has no means *a priori* to know whether a packet belongs to a targeted ED or not.

The temporal analysis of packets is therefore an important feature to be considered when characterizing the traffic. More specifically, our analysis is focused on the time elapsing between a packet and the next one from the same ED (in the uplink or in the downlink). Let t_i be the timestamp of a packet p with frame counter (FCnt) $p_{\text{FCnt}} = i$, relative to a given DevAddr. Timestamp and FCnt are two packet fields used in our approach and reported in Table 2. The first is the timestamp added from the GW at the exact moment in which the packet has been received.

Since in LoRaWAN v1.1 the FCnt is a counter of the sent frames, the time $d_i = t_{i+1} - t_i$ is the time between two consecutive packets, the inter-arrival time. Moreover, the FCnt field is sent as unencrypted both in uplink or downlink and therefore could be observed by a third party. Our assumption is that the values d_i have some periodic behavior. More precisely, we model each d_i as a random variable having distribution $d_i \sim f(\mu_i, \sigma_i)$. Let the integer value τ be the period when the inter-arrival time has the same distribution, that is $d_i, d_{\tau+i}, d_{2\tau+i}, \cdots \sim f(\mu_i, \sigma_i)$, i.e. the inter-arrival time distribution repeats after τ inter-arrivals. Figure 9 illustrates an example of the model where periodic d_i are identified (with period $\tau = 3$).

Since we do not know the value of τ a priori, let us denote as $\hat{\tau}$ the prediction of the real value of τ. We now describe a subroutine of DEVIL that extracts these temporal patterns from the inter-arrival sequence, estimating the period of the sequence $\hat{\tau}$. First, the procedure takes into account the presence of lost packets, i.e. packets that were transmitted by a device but were not received by the NS. To identify the lost packets, we observe which FCnt values are not present in the data. Let us define as $L(a)$ the set of FCnt values not observed for a DevAddr a, due to packet losses:

$$L(a) = \{i < z < j \text{ s.t. } \exists\, p_{\text{FCnt}} = i, \exists\, p_{\text{FCnt}} = j, \nexists\, p_{\text{FCnt}} = z\} \tag{2}$$

The first step of the subroutine yielding the period of the temporal sequence is to calculate inter-arrival times from packets:

$$d_i = \begin{cases} t_{i+1} - t_i & \text{if } i, i+1 \notin L(a) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The value d_i is the time between the reception of two packets p_1, p_2 with FCnt respectively $p_{1,\text{FCnt}} = i, p_{2,\text{FCnt}} = i + 1$, if p_1, p_2 have been received. If either one of such packets is not observed, $d_i = 0$. Notice that the value of zero is a placeholder for the next step, instead of a real inter-arrival time.

A period $\hat{\tau}$ is fixed and the subroutine yields an estimation metric which is low if $\hat{\tau}$ is likely to be close to the true value of τ, and it is high otherwise. This is done by first extracting the sequences:

$$s_j = [d_j, d_{\hat{\tau}+j}, d_{2\hat{\tau}+j}, d_{3\hat{\tau}+j}, ...],\ \forall j\{\in 1, \dots, \hat{\tau} - 1\} \tag{4}$$

Then it computes the standard deviation σ_{s_j} for each of the sequences s_j. In the computation, the components $d_i = 0$ are not considered and are discarded: in this way we are ignoring the inter-arrival times of non-consecutive packets in case of a packet loss. The values of the standard deviations are then summed up:

$$\xi_{\hat{\tau}} = \sum_{j \in \{1,\dots,\hat{\tau}-1\}} \sigma_{s_j} \tag{5}$$

The quantity $\xi_{\hat{\tau}}$ is an estimation error for the prediction $\hat{\tau}$. The lower $\xi_{\hat{\tau}}$ is, the more likely $\hat{\tau} = \tau$. More accurate aggregation and error functions could be used, for example one that takes into account the standard deviation of a time d_i, but for our purposes the sum of the standard deviations expressed in Eq. 4 worked well. In Figure 10, the orange line shows an example of the values $\xi_{\hat{\tau}}$ with different values of $\hat{\tau}$.

The algorithm, which will be described in the next section, will make its decision based on the estimation error in Eq. (5). Since this error is the aggregation of standard deviations, it requires some sort of periodicity of the inter-arrival time of the packets. As hinted in Fig. 10, non-periodic

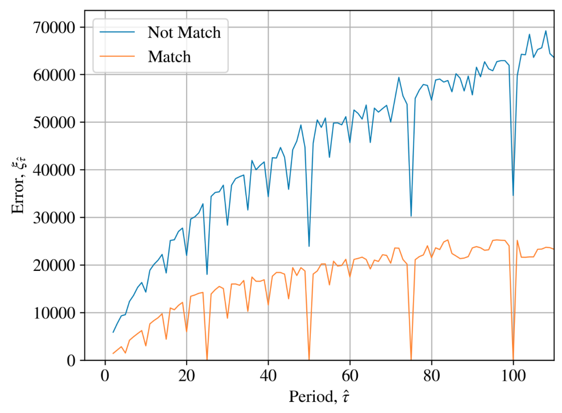

Figure 10: Estimation error $\xi_{\hat{\tau}}$ obtained on a sequence of packet inter-arrival times with different values of period $\hat{\tau}$. We observe the error spiking to almost zero when the predicted period $\hat{\tau}$ is equal to the true period τ. Moreover, we have multiple spikes in correspondence to the integer multiples of the period, since if τ is the period of the sequence also $k\tau$ is a valid period of the sequence, for any integer k. When associating a_1, a_2 DevAddr, their packets are concatenated and the sequence of inter-arrival times is extracted. The plot shows the error $\xi_{\hat{\tau}}$ as a function of $\hat{\tau}$ in case a_1, a_2 are a correct matching, i.e. they belong to the same device, and in case they are an incorrect matching, represented by the orange and blue lines, respectively.

arrivals will have a high estimation error, while in periodic situations this error will be close to zero. In other words, the matching operated by DEVIL exploits the periodicity in the inter-arrival sequence of the devices. A possible countermeasure of this attack could indeed be to add a random component, such a random jitter, to the time in which an uplink packet is scheduled, as it is described in Section 8.

6. Procedure

In this section we describe the DEVIL algorithm that, given a DevAddr, derives the associated LoRaWAN DevEUI. The procedure is structured in two steps. The first step produces a mapping M where $M(a_1) \to a_2$ if a device that had assigned the address a_1 gets assigned the address a_2. In the second step, it uses this information to infer the final set of addresses assigned to a DevEUI e, namely $A_p(e)$, where we indicate as $A_p(e)$ the output of the algorithm and as $A_t(e)$ the ground truth. Notice how the two tasks carried out by the two steps are different: in the first step we identify that two addresses belong to the same device, in the second step we identify precisely the DevEUI of that device.

We first introduce some notation. Let a be a LoRaWAN DevAddr. Let us denote by $S(a)$ and $E(a)$ respectively the timestamp in which we receive the first packet by a and the last packet by a. We state that two DevAddrs a_1, a_2 are *consecutive* if the following holds:

$$S(a_2) > E(a_1) \ \wedge \ S(a_2) < \left[E(a_1) + T_w\right] \qquad (6)$$

Algorithm 1 First step of DEVIL algorithm. It produces the mapping $M(a_1) \to a_2$, where a_1 is a DevAddr assigned to a device and a_2 is the next DevAddr used by the same device after a Join or ReJoin procedure.

```
1:  for $a_1 \in$ all DevAddr do
2:      $a^* \leftarrow$ NaN
3:      $\langle \hat{\xi}_{n,\min} \rangle \leftarrow \langle \infty \rangle$
4:      for $a_2 \in C(a_1)$ do
5:          concatenate packets having DevAddr $a_1, a_2$
6:          extract $d_i$ sequence
7:          compute $\langle \hat{\xi}_n \rangle$
8:          if $|\{i \text{ s.t. } \langle \hat{\xi}_n \rangle_i < \langle \hat{\xi}_{n,\min} \rangle_i\}|$ then
9:              $\hat{\xi}_{n,\min} \leftarrow \hat{\xi}_n$
10:             $a^* \leftarrow a_2$
11:         end if
12:     end for
13:     $M(a_1) \leftarrow a^*$
14: end for
15: return $M$
```

where the time window T_w a parameter of DEVIL. With the previous equation, we imply that the first packet of a_2 should occur inside a window which starts at the time of the last packet of a_1 and has length of T_w seconds. We denote by $C(a)$ the set of consecutive addresses of a. In other words, $C(a)$ is the set of Device Addresses which have sent their first packet within a time window T_w after the last packet of a is observed.

In the first step of DEVIL, the procedure iterates over all DevAddr a_1. For each a_1, it iterates over all its consecutive addresses $a_2 \in C(a_1)$. The algorithm concatenates the packets of a_1, a_2, considering them as they were generated by a single device. The inter-arrival times d_i of this packet sequence is extracted, following Eq. 3. The subroutine described in Section 5 is called, returning the values of $[\sigma_{\hat{\tau}_1}, \sigma_{\hat{\tau}_2}, \dots]$ for all periods $\hat{\tau}_i$ of interest. Let n be an integer and let $\langle \hat{\xi}_n \rangle$ the n smallest values obtained of $\xi_{\hat{\tau}_i}$. Let a_2^* the address that generated the lower values of $\langle \hat{\xi}_n \rangle$, the final mapping is updated by setting $M(a_1) \to a_2^*$. Figure 10 shows values of $\xi_{\hat{\tau}}$ in case $\langle a_1, a_2 \rangle$ is a correct or incorrect association. Notice that due to the periodic traffic of the devices, a DevAddr a_1 is always associated to a DevAddr $a_2 \in C(a_1)$. If this is not the case, a threshold on $\langle \hat{\xi}_n \rangle$ could be applied, in this case the algorithm would be aware that a_1 stopped transmitting, i.e. there is no $a_2 \in C(a_1)$ that is associated to the same device. Moreover, other radio-space features could be leveraged for refining the prediction, like the Received Signal Strength Indicator (RSSI) experienced at the gateways, physical layer information are always retrieved at GW. The pseudocode of the first step is reported in Alg. 1.

The first step of DEVIL is concluded and it has yielded the mapping $M(a_1) \to a_2$, where a_2 is the DevAddr that a device gets assigned after a_1.

Algorithm 2 Second step of DEVIL. The output is A_p where $A_p(e) = \{a_1, a_2, \dots\}$ is the predicted set of DevAddr values which were assigned to the device having DevEUI e.

```
1:  for $a \in$ all DevAddr do
2:      $\bar{E} \leftarrow$ set of all DevEUI
3:      $\bar{A} \leftarrow \{a\}$
4:      while $a \in M$ and $|\bar{E} \cap J(a, M(a))| > 0$ do
5:          $\bar{E} \leftarrow \bar{E} \cap J(a, M(a))$
6:          $a \leftarrow M(a)$
7:          $\bar{A} \leftarrow \bar{A} \cup a$
8:      end while
9:      for $e \in \bar{E}$ do
10:         $A_p(e) \leftarrow A_p(e) \cup \bar{A}$
11:     end for
12: end for
13: return $A_p$
```

For the second step of the algorithm, we denote as $J(a_1, a_2) = \{e_1, e_2, \dots\}$ the set of DevEUI values that performed a Join or ReJoin request in a time window spanning from $E(a_1)$ to $S(a_2)$ i.e. from the time of the last received packet of a_1 to the first received packet of a_2. The second step of DEVIL iterates on all DevAddr a_i. It gets a_{i+1} following the update rule:

$$a_{i+1} \leftarrow M(a_i) \tag{7}$$

and constructs the set $A_{p,i+1} = A_{p,i} \cap J(a_i, a_{i+1})$. The algorithm iterates following the update rule in Eq. 7. If for some value of $i = i'$ we have that $a_{i'} \notin M$ or $|A_{p,i'+1}| = 0$, the algorithm stops updating and for all DevEUI $e \in A_{p,i'}$ it updates the final solution:

$$A_p(e) = A_p(e) \cup \{a_i, a_{i+1}, \dots, a_{i'}\},\ \forall e \in A_{p,i'} \tag{8}$$

In words, the second step of DEVIL checks for the DevEUI of the EDs performing Join requests in a time frame $[E(a_1), S(a_2)]$, for any DevAddr addresses a_1, a_2 such that $M(a_1) \to a_2$. The pseudocode for the second step of the algorithm is reported in Alg. 2.

6.1. DEVIL computational complexity

We analyzed the computational complexity of both the first and the second step of DEVIL. In the first step, the outer loop iterates over any observed Device Address a_1. The inner loop iterates over the set of all the consecutive Device Addresses $a_2 \in C(a_1)$. The number of iterations therefore depends on the size of the consecutive device addresses, which depends on the time window T_w and on the available data. In general, the number of iterations of the inner for-loop is equal to the average size of the consecutive addresses $|C|$ defined as follows:

$$|C| = \frac{1}{|A|} \sum_a |C(a)|$$

where A is the set of all Device Addresses. On our dataset. with $N = 150$ devices and a time window $T_w = 6$ hours, the average value of the consecutive addresses was $|C| \approx 90$. Then, for every pair $\langle a_1, a_2 \rangle$ the algorithm computes cost values for each possible length of the inter-arrival sequence, from 1 to S, where S is the maximum length of a periodic pattern in the inter-arrival sequence. Therefore, the final computational cost of the first step is $O(|A| \cdot |C| \cdot S)$.

In the second step, the algorithm goes through all device addresses a and examines the Join Request messages observed within a time frame from the last packet of the DevAddr a to the first packet of the DevAddr $M(a)$. Iteratively, the algorithm follows the change of DevAddr by updating $a \leftarrow M(a)$ until the M chain is finished or there are no Join Request messages between a and $M(a)$. This procedure is executed for each address a, making the final computational cost of the second step $O(|A|^2)$. The reason of the squared term is that an address is analyzed multiple times: once in the outer loop as a and possibly several times while following the change of DevAddr M. We also tried a variation of the algorithm in which we analyzed a DevAddr only once, bringing the computational cost down to $O(|A|)$. However, we observed a reduction in terms of accuracy, since errors made in the first step for the computation of the mapping M had more influence on the final result.

Table 3
Parameters of the algorithm and characterization of the data.

Variable	Value
T_w	3700 s
n	5
β_{lost}	17.5%

7. Results evaluation

In this section, we report the results of the application of DEVIL to data generated by a real LoRaWAN application. Table 3 reports the numerical value of parameters used in our experiments. The time T_w is the length of the time window in which two DevAddr can possibly be consecutive, following the definition given in Sec. 6. The integer n is a threshold used in the first step of the algorithm. The ratio β_{lost} is the fraction of packets that were lost during the communication session. The algorithm is aware of which packets were lost, following the definition of d_i in Eq. 3 and its further processing. The high value of β_{lost} is not negligible and justifies the processing of the missed packets. If this is not done, phase-shifts would have certainly been verified in the temporal analysis of the traffic, which would have invalidated the results. Moreover, the application does not always respect the periodicity assumption and causes noise to the system.

We evaluated the accuracy of the two steps of DEVIL separately. The final accuracy of DEVIL is the one in the second step, inferring the DevAddr assigned to a DevEUI. Nonetheless, the first step of DEVIL is a powerful tool to know whether two consecutive DevAddr were assigned to

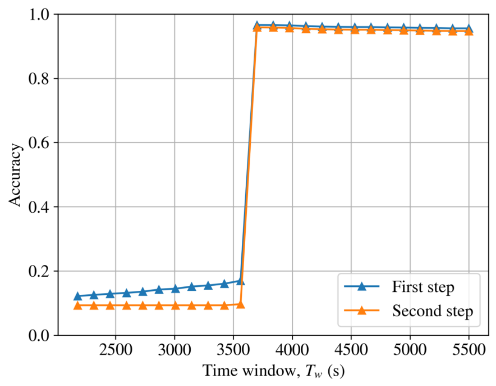

Figure 11: Accuracy of the first and second step when varying the parameter T_w. For low values of T_w, the algorithm does not find the correct matching of a DevAddr and the accuracy of the whole algorithm is low. This happens since, given a DevAddr a, the short window causes the set of consecutive addresses $C(a)$ to not contain the good match of a. For $T_w \geq$ 3600s we observe high values of accuracy. This is due to the majority of devices sending an uplink packet roughly every hour. Moreover, when T_w is greater than 3600s we observe a slight decrease in the accuracy. This happens since the unnecessarily wide window causes the set of consecutive addresses $C(a)$ to include an increasing number of addresses that are not a match for a anymore, thus increasing the possibility of a wrong match.

the same ED, even if this step does not derive the exact DevEUI. Moreover, generally speaking we can expect to know whether or not two consecutive DevAddrs were assigned to the same ED with higher accuracy than the exact DevEUI of the ED.

For evaluating the accuracy of the first step, constructing the mapping M, we simply computed all the pairs $\langle a_1, a_2\rangle$ such that $M(a_1) \rightarrow a_2$, and considered the ratio of the number of correct predictions against the number of total predictions. For evaluating the accuracy of the second step, we accounted for the case that a DevAddr a could be associated to one or more DevEUIs, i.e. it could be that $a \in A_p(e_1)$ and $a \in A_p(e_2)$. This occurs when the algorithm is uncertain about the DevEUI to be associated to the address a. We therefore considered as indicator the variable i_a, with value 1 if the prediction is correct and 0 otherwise, and the number of times a appears in the solution $n_a = |\{e \text{ s.t. } a \in A_p(e)\}|$. Then the accuracy is expressed as the quantity $\frac{i_a}{n_a}$ averaged for all a.

In Figure 11 the accuracy of the first and second step of DEVIL is reported as a function of the T_w parameter. From this analysis we extract the best value of the T_w parameter (3700s), reported in Table 3.

Finally, Table 4 reports the accuracy of the two steps of the algorithm, separately. The first step achieves an accuracy of more than 95% in the construction of the mapping M.

Table 4
Accuracy of the two separate steps of DEVIL algorithm using the water metering dataset.

Algorithm	Description	Accuracy
First step	Construct mapping $M(a_1) \rightarrow a_2$ from a DevAddr a_1 to the next DevAddr a_2 utilized by the same device	0.959
Second step	Produce $A_p(e) \rightarrow \{a_1, a_2, \dots\}$ reconstructing the set of addresses which were assigned to DevEUI e	0.936
Final Accuracy		**0.936**

This demonstrates that the algorithm is able to process the data of a real LoRaWAN application and to understand, with high confidence, the change of DevAddr happening for an ED. The second step of the algorithm scores an accuracy of 93% in finding all the LoRaWAN addresses assigned to a specific DevEUI, successfully reconstructing the hidden mapping between a DevAddr and its DevEUI.

In Table 5, we have reported the accuracy of DEVIL compared with the algorithm presented in [15]. The comparison is done at the second step of DEVIL, since it is this step of DEVIL that gives as output the final de-anonymization. In all scenarios, DEVIL obtains a higher accuracy, with an increase of 58% in case of water and energy metering devices and 26% in case of the synthetic dataset. Note that we did not evaluate the algorithms on the LoED dataset, since it does not contain the information on the DevEUI of the packets which is used as ground truth to compute the accuracy.

We have analyzed the running time of the procedures, both DEVIL and the algorithm proposed in [15]. The total execution times are reported in Table 5, reporting one entry per dataset and algorithm. The algorithms were executed on a laptop equipped with an AMD Ryzen 7 4800U CPU and were implemented with 14 parallel execution threads. The reported time is the effective CPU time. From the results, it is clear that DEVIL requires more time for its execution when compared to the comparison algorithm, which does not consider the timings of the inter-arrival times: its computational complexity could be approximated as O($|A| \cdot |C|$) which is lower than DEVIL's one, as shown in Sec. 6.1.

We remark that, in comparison with the algorithm in [15], where 94% of the DevEUI are matched, DEVIL reaches about the same performance but, differently from [15], it can work with a more flexible network monitoring without relying on the presence of messages (Join requests) sent by the EDs. In other words, if some Join requests messages are not observed the approach in [15] fails, while DEVIL continues to work successfully deriving the correct matching. The first step of DEVIL can work with different not overlapping T_w, that not necessarily include Join request packets. Finally in the second step it is sufficient to capture only once a Join request in one of the T_w to map the correct DevEUI to the ED.

Dataset	# packets	Period	DEVIL time	Ancian et al. time	Step	DEVIL	Ancian et al.
Water metering	0.9×10^6	15 months	43.97s	16.01s	First	0.959	0.592
					Second	0.936	
Energy metering	1.1×10^6	20 months	266.62s	5.66s	First	0.946	0.576
					Second	0.913	
Synthetic (N=1500)	1.1×10^6	10 months	3137.75s	64.34s	First	0.965	0.596
					Second	0.824	

Table 5
Accuracy of DEVIL in the water metering, energy metering and synthetic datasets. We reported the accuracy of both the first and the second step of DEVIL. The accuracy of the first step is always higher than the accuracy of the second step, due to the fact that the second step uses as input the output of the first step. In this way the accuracy of the first step serves as upper bound of the accuracy of DEVIL. In the rightmost column of the table we reported the accuracy of the de-anonymization algorithm presented in [15]. Also, we have reported the CPU time of the procedures.

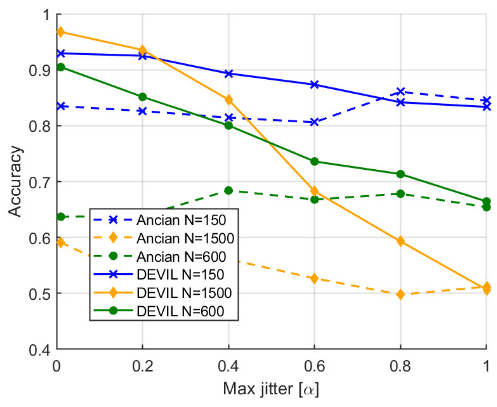

Figure 12: Accuracy of DEVIL and [15] with different values of the jitter α in the synthetic dataset and with different number of devices related to the evaluation of the countermeasure.

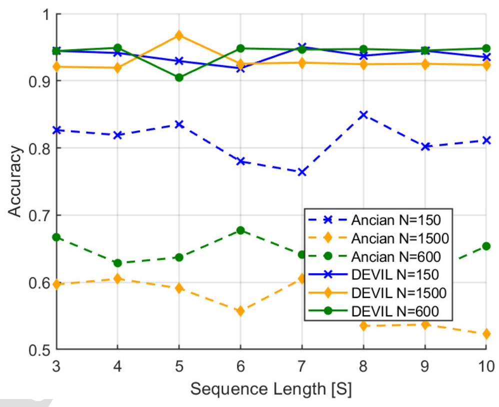

Figure 13: Accuracy of DEVIL with different values of pattern length S in the synthetic dataset and with different number of devices.

The choice of the T_w parameter should be related with the data on which the algorithm is applied. A low duration of the T_w window, the lower the accuracy of the algorithm. This is due to the lower number of devices being in the considered time frame: with a small T_w the it could be the case that the right device is discarded by the algorithm since its first packet after the change of DevAddress happens only after a time T_w starting from its last packet with the old DevAddress. In theory, selecting T_w to be as large as the maximum inter-arrival time possible for an application should be enough to ensure that the correct DevAddress is considered by the algorithm. In reality, however, packet losses may occur, therefore an observed inter-arrival time of a device could be higher than the maximum estimated inter-arrival time and the value of T_w should be increased accordingly. On the other hand, the lower T_w the faster the algorithm will run. Indeed, T_w influences the number of considered devices in the selection of next DevAddress in the first step of the algorithm: since the algorithm has to check fewer candidates it is able to run in less time.

8. Countermeasures

As hinted in Section 5, DEVIL requires some periodic behavior in the sequence of the inter-arrival times. Therefore, one possible countermeasure could be to add a random component disrupting the periodicity. In our experiments, we added a jitter which is uniformly distributed in $[0, \alpha\,\hat{t}]$ as described in Eq. (1) to randomize, to some extent, the scheduling of the uplink packets.

However, the added jitter could cause the a disturbance of the overlying application. For example, if our application is composed by smart sensors logging the temperature every hour, with the addition of this jitter the ED must be able to bufferize and/or delay packets, even if this should impact only few packets at most.

We express the random jitter in terms of α, and not in absolute terms, to remark that this jitter should be related to the current value of the non-jittered interarrival time. This gives a measure on how much the jitter impacts the ED behavior and ultimately the application.

Figure 12, we have reported the accuracy of DEVIL using different values of α and with different numbers of devices. We have used the synthetic dataset to stress the proposed approach and evaluate the robustness of it as a function of the jitter value. As expected, the accuracy of DEVIL decreases when increasing α, since the higher it is this parameter the higher the disruption of the DEVIL time analysis. We also plot the accuracy of the algorithm in [15] on the same dataset; since this algorithm does not depend on the temporal analysis of the inter-arrival times, its accuracy remains constant when varying α.

In Figure 13 we report the accuracy of DEVIL with different values of maximum periodic pattern length S in the synthetic dataset. We can observe that the performance of DEVIL is unchanged when varying the length of the patterns of the known inter-arrival times. Moreover, we observe that the accuracy of DEVIL is constant when increasing the number of devices N, scoring an accuracy of more than 90% with all values $N = 150$, $N = 600$ and $N = 1500$.

Summarizing, the introduction of a specific jitter value to the system can be considered as a countermeasure technique against the device de-anonymization operated by DEVIL. Suitable jitter values can be evaluated by DEVIL and suggested to the NS to make more robust some devices in terms of privacy.

9. Conclusions and future work

This work presents an algorithm that reconstructs which temporally varying LoRaWAN DevAddr addresses are assigned to the unique identifier DevEUI. Our algorithm is based on temporal traffic analysis, gathering information about the timings in which uplinks traffic is generated by IoT devices. Our methodology is organized into two steps. In the first step we find a matching between a DevAddr utilized by an ED and the next DevAddr that the same ED gets assigned after a successful Join or ReJoin operation. In the second step instead we utilize the result found previously to map every DevAddr to its DevEUI, observing the sequence of the Join and ReJoin messages.

We perform an analysis of 3 real LoRaWAN datasets: i) two metering services, energy and water, provided by a real LoRaWAN application, supplied by the UNIDATA operator; ii) one open dataset relevant to a mix of application services. We use these datasets to evaluate the accuracy of the proposed algorithm in comparison with other approach present in literature. We evaluate the two steps differently and found that our system reconstructs the assignment of a DevAddr to an ED DevEUI with high accuracy. In comparison with the literature approach, we show that in the best case the accuracy of our approach is improved of 58%. Another potentiality for DEVIL is to support the NS in the network optimization strategy. We use a synthetic dataset to evaluate the robustness of the proposed algorithm as the transmission jitter changes. The jitter values found by DEVIL algorithm, can be introduced to the system to produce a mitigation techniques. Jitters are supplied to the NS.
Future research will be dedicated to improve the algorithm by adding more features to fingerprint and extract EDs behavior. For example, radio parameters such as SNR and RSSI could be leveraged to refine the association. Thus, the output of DEVIL can be used as input to the NS to optimize the network and to apply mitigation strategies [20].

Acknowledgement

We thank UNIDATA S.p.A. who provided insight and expertise that assisted our research greatly. UNIDATA S.p.A. is an Italy IoT operator that manages several application services (e.g. metering, GPS tracking, and smart light). The UNIDATA network currently involves more than 5000 EDs and 200 GWs. In 2020, the total amount of EDs whose transmissions were received by the UNIDATA GWs was over 80K (they include EDs from different operators). Moreover, the network collected approximately 4.83×10^8 packets over the last year (6 % are transmitted by EDs of the UNIDATA domain). Part of the activities is related to the H2020 ELEGANT project (Grant agreement ID: 957286).

References

[1] L. Bhatia, M. Breza, R. Marfievici, J. A. McCann, Loed: The lorawan at the edge dataset: Dataset, in: Proceedings of the Third Workshop on Data: Acquisition To Analysis, DATA '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 7–8. doi:10.1145/3419016.3431491.
URL https://doi.org/10.1145/3419016.3431491

[2] IoT Analytics, LPWAN Market Report 2018-2023, https://iot-analytics.com/product/lpwan-market-report-2018-2023/ (September 2018).

[3] ETSI, Final draft ETSI EN 300 220-1 V2.4.1 (2012-01). Technical Report REN/ERM-TG28-434, https://www.etsi.org/deliver/etsi_tr/103500_103599/103526/01.01.01_60/tr_103526v010101p.pdf (2012).

[4] L. Alliance, Lorawan 1.1 specification, LoRa Alliance 11 (2017) 2018–04.

[5] P. Leu, I. Puddu, A. Ranganathan, S. Capkun, I send, therefore i leak: Information leakage in low-power wide area networks, 2018, pp. 23–33. doi:10.1145/3212480.3212508.

[6] S. Tonyali, K. Akkaya, N. Saputro, A. S. Uluagac, M. Nojoumian, Privacy-preserving protocols for secure and reliable data aggregation in iot-enabled smart metering systems, Future Generation Computer Systems 78 (2018) 547–557. doi:https://doi.org/10.1016/j.future.2017.04.031.
URL https://www.sciencedirect.com/science/article/pii/S0167739X17306945

[7] K. Gajowniczek, T. Zabkowski, Smart metering and data privacy issues, Information Systems in Management 2 (2013) 239–249.

[8] Semtech, LoRa (2015).

[9] D. Croce, M. Gucciardo, I. Tinnirello, D. Garlisi, S. Mangione, Impact of spreading factor imperfect orthogonality in lora communications, in: A. Piva, I. Tinnirello, S. Morosi (Eds.), Digital Communication. Towards a Smart and Secure Future Internet, Springer International Publishing, Cham, 2017, pp. 165–179.

[10] Sornin, N., Yegin, A., et al., LoRaWAN 1.1 Specification, https://lora-alliance.org/resource-hub/lorawantm-specification-v11 (2017).

[11] I. You, S. Kwon, G. Choudhary, V. Sharma, J. T. Seo, An enhanced lorawan security protocol for privacy preservation in iot with a case study on a smart factory-enabled parking system, Sensors 18 (06 2018). doi:10.3390/s18061888.

[12] I. Butun, N. Pereira, M. Gidlund, Security risk analysis of lorawan and future directions, Future Internet 11 (1) (2019). doi:10.3390/fi11010003.
URL https://www.mdpi.com/1999-5903/11/1/3

[13] B. Oniga, V. Dadarlat, E. De Poorter, A. Munteanu, Analysis, design and implementation of secure lorawan sensor networks, in: 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 2017, pp. 421–428. doi:10.1109/ICCP.2017.8117042.

[14] R. Miller, "lora security building a secure lora solution", Vol. Presented at BSides Conference, 2016.

[15] L. Ancian, M. Cunche, Re-identifying addresses in LoRaWAN networks., Tech. rep., Inria Rhône-Alpes; INSA de Lyon (08 2020).
URL https://hal.inria.fr/hal-02926894

[16] K. Lin, X. Xu, H. Gao, TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of iiot, Comput. Networks 190 (2021) 107974. doi:10.1016/j.comnet.2021.107974.
URL https://doi.org/10.1016/j.comnet.2021.107974

[17] H. Yao, P. Gao, J. Wang, P. Zhang, C. Jiang, Z. Han, Capsule network assisted iot traffic classification mechanism for smart cities, IEEE Internet of Things Journal 6 (5) (2019) 7515–7525. doi:10.1109/JIOT.2019.2901348.

[18] J. M. Valtorta, A. Martino, F. Cuomo, D. Garlisi, A clustering approach for profiling lorawan iot devices, in: I. Chatzigiannakis, B. De Ruyter, I. Mavrommati (Eds.), Ambient Intelligence, Springer International Publishing, Cham, 2019, pp. 58–74.

[19] D. Garlisi, A. Martino, J. Zouwayhed, R. Pourrahim, F. Cuomo, Exploratory approach for network behavior clustering in lorawan, Journal of Ambient Intelligence and Humanized Computing (2021). doi:10.1007/s12652-021-03121-z.

[20] D. Garlisi, I. Tinnirello, G. Bianchi, F. Cuomo, Capture aware sequential waterfilling for lorawan adaptive data rate, IEEE Transactions on Wireless Communications 20 (3) (2021) 2019–2033. doi:10.1109/TWC.2020.3038638.

AUTHORSHIP STATEMENT

Manuscript title: Discovery privacy threats via device de-anonymization in LoRaWAN

All persons who meet authorship criteria are listed as authors, and all authors certify that they have participated sufficiently in the work to take public responsibility for the content, including participation in the concept, design, analysis, writing, or revision of the manuscript. Furthermore, each author certifies that this material or similar material has not been and will not be submitted to or published in any other publication before its appearance in the *Hong Kong Journal of Occupational Therapy*.

Authorship contributions

Please indicate the specific contributions made by each author (list the authors' initials followed by their surnames, e.g., Y.L. Cheung). The name of each author must appear at least once in each of the three categories below.

Category 1

Conception and design of study: Pietro Spadaccino, Domenico Garlisi, Francesca Cuomo ____, ____, ____, ____;

acquisition of data: Giorgio Pillon, Patrizio Pisani ____, ____, ____, ____;

analysis and/or interpretation of data: ____, ____, ____, ____.

Category 2

Drafting the manuscript: Pietro Spadaccino, Domenico Garlisi, Francesca Cuomo ____, ____, ____, ____;

revising the manuscript critically for important intellectual content: Giorgio Pillon, Patrizio Pisani ____, ____,

____, ____.

Category 3

Approval of the version of the manuscript to be published (the names of all authors must be listed):

Pietro Spadaccino, Domenico Garlisi, Francesca Cuomo, Giorgio Pillon, Patrizio Pisani,

____, ____, ____, ____, ____.

Acknowledgements
All persons who have made substantial contributions to the work reported in the manuscript (e.g., technical help, writing and editing assistance, general support), but who do not meet the criteria for authorship, are named in the Acknowledgements and have given us their written permission to be named. If we have not included an Acknowledgements, then that indicates that we have not received substantial contributions from non-authors.

This statement is signed by all the authors (*a photocopy of this form may be used if there are more than 10 authors***):**

Author's name (typed)	Author's signature	Date
Pietro Spadaccino	[illegible]	11/02/2022
Domenico Garlisi	[illegible]	11/02/2022
Francesca Cuomo	[illegible]	11/02/2022
Giorgio Pillon	[illegible]	11/02/2022
Patrizio Pisani	[illegible]	11/02/2022

Declaration of interests

[✓ ☒] The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: