Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 134 (2018) 51–58

Procedia Computer Science

www.elsevier.com/locate/procedia

The 15th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2018)

Security of LoRaWAN v1.1 in Backward Compatibility Scenarios

Tahsin C. M. Dönmez[a,*], Ethiopia Nigussie[a]

[a]*Department of Future Technologies, University of Turku, Turku, Finland*
[b]*Second affiliation, Address, City and Postcode, Country*

Abstract

We present security vulnerabilities of LoRaWAN v1.1 in backward compatibility scenarios and propose countermeasures. Novel low-power wireless communication technologies are continuously introduced into the IoT ecosystem, while the existing ones continue their evolution by adding new features and fixing known problems. Since most of IoT-based applications involve communication of sensitive data with remote users, it is crucial to scrutinize the security of the emerging low-power wireless communication technologies. LoRaWAN is a Low Power Wide Area Network (LPWAN) protocol that is optimized for wireless battery-powered IoT devices. The current LoRaWAN v1.1 states only one possible backward compatibility scenario without defining all the associated security behaviors and their implications for the system's overall security. This work examines the applicability of v1.0.2 attacks in fall-back cases by consolidating the vulnerabilities of v1.0.2 and the resulting attacks. After determining the possible attacks, countermeasures to prevent DoS attacks that target the *join procedure*, replay of data, and eavesdropping in backward compatibility scenarios are proposed.

Peer-review under responsibility of the scientific committee of the 13th International Conference on Future Networks and Communications, FNC-2018 and the 15th International Conference on Mobile Systems and Pervasive Computing, MobiSPC 2018.

Keywords: LoRaWAN ; IoT ; security ; backward compatibility

1. Introduction

Internet of Things (IoT) is becoming ubiquitous through the deployment of IoT devices in virtually every corner of the economy, such as wearable, individual households, industries and vehicles. According to Gartner forecast, about 20 billions of devices will be interconnected to sense and act on their environments in 2020 [1]. A 2017 report by Ericsson [2] predicts that, by 2022, the number of wide-area connectivity (such as LTE-M, NB-IoT, Sigfox, LoRaWAN, and RPMA) based devices will reach 2.1 billions, and the number of short-range (such as Wi-Fi, Bluetooth, and ZigBee) based IoT devices will reach 15.5 billions. The large number of interconnected devices, their limited resources, and the fact that they operate autonomously without a human involvement, emphasizes the importance of security in ways which clearly reveal that security can neither be ignored nor be an afterthought for IoT systems. IoT security is a multifaceted problem due to the involvement of heterogeneous devices, various technologies and stakeholders. The multitude of vulnerable IoT devices [3] is the embodiment of the failure of free-market in creating incentives for security. Furthermore, each newly introduced connectivity technology inevitably brings along its own set of vulnerabilities, and intensify the overall security challenges of the IoT ecosystem. This work focuses on the security of one of the wide area wireless communication technologies, LoRaWAN [4].

* Corresponding author. Tel.: +358-4652-2-9553.
E-mail address: tcmdon@utu.fi

Peer-review under responsibility of the scientific committee of the 13th International Conference on Future Networks and Communications, FNC-2018 and the 15th International Conference on Mobile Systems and Pervasive Computing, MobiSPC 2018.
10.1016/j.procs.2018.07.143

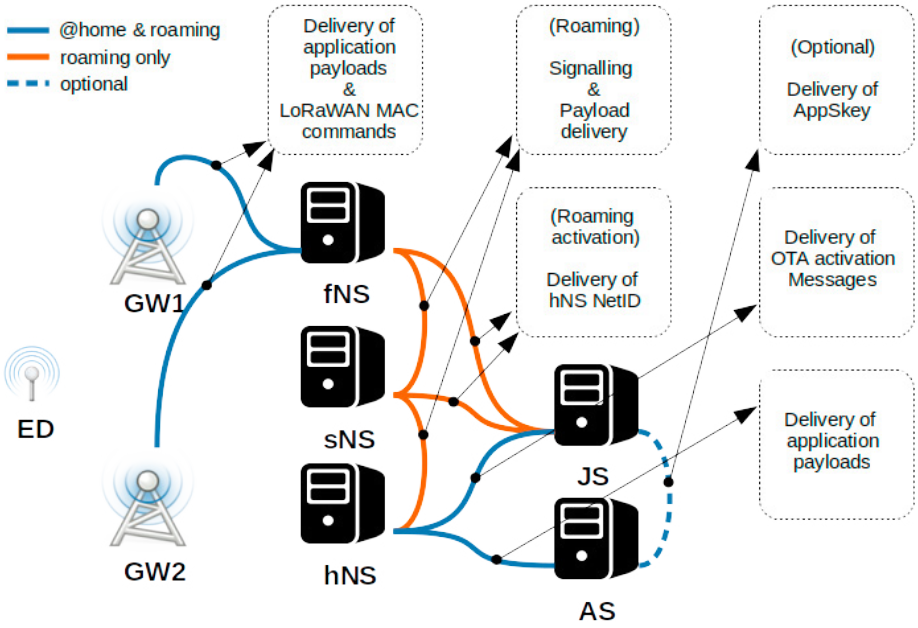

Fig. 1. LoRaWAN network elements and their interaction.

Several Low Power Wide Area Network (LPWAN) technologies are emerging to provide connectivity for IoT applications that require low-power long-distance wireless communication. These technologies often differ in their characteristics, such as radio implementation, range, power consumption, and data rate. In terms of these characteristics and the trade-offs made between them, LPWAN complements the traditional short-range wireless technologies [5]. LoRa™ is a wireless physical layer solution for long-range low-power low-data-rate applications developed by Semtech. LoRaWAN™ is a LPWAN specification optimized for wireless battery-powered end-devices based on LoRa. LoRaWAN specification version 1.0.2 [6] and 1.1 [7, 8] were released in July 2016 and October 2017, respectively. Both versions are of interest, as this work is concerned with security issues of backward compatibility scenarios. Previous works have studied the security of LoRa-enabled systems and vulnerabilities were discovered in LoRa physical layer [9, 10], specific implementations [11, 12], and in LoRaWAN protocol version 1.0 [13, 14, 15, 16]. In this paper, only LoRaWAN protocol vulnerabilities are of interest. At the time of writing this paper, the authors were unable to find any work addressing the security of LoRaWAN version 1.1.

The presented work examines the security issues in backward compatibility scenarios and proposes potential countermeasures. Though LoRaWAN version 1.1 has already addressed most of the security vulnerabilities of version 1.0.2, security issues due to these vulnerabilities still exist in case of interoperability between 1.0.2 and 1.1 devices. Providing countermeasures in this scenario is a necessity since not all devices are capable of upgrading to version 1.1. In this work, first the vulnerabilities of v1.0.2 and the resulting attacks are studied. Secondly, how v1.1 has addressed these vulnerabilities are examined. Thirdly, the LoRaWAN backward compatibility operation and the associated security issues are closely evaluated. These three investigations demonstrate that all of the studied v1.0.2 vulnerabilities and attacks reappear in case of fall-back of either end-device or back-end to version 1.0.2. Countermeasures are proposed to prevent DoS attacks that target the *join procedure*, replay of data, and eavesdropping in backward compatibility scenarios.

The rest of the paper is organized as follows. Section 2 presents the key security features of LoRaWAN v1.1. The v1.0.2 protocol vulnerabilities and how they were addressed in v1.1 are discussed in Section 3. In Section 4, the possible attacks against version 1.0.2 are described. Section 5 discusses the operation of LoRaWAN's backward compatibility. Section 6 discusses the applicability of version 1.0.2 attacks in backward compatibility scenarios, and proposes countermeasures. Finally, the conclusion is presented in Section 7.

2. Security features of LoRaWAN v1.1

Often various network elements are involved in handling the security configuration of a network. In case of LoRaWAN the involved elements are end-devices (ED), gateways (GW), network servers (NS), join servers (JS), and application servers (AS). These network elements and the interaction among them are depicted in Figure 1. In case of roaming forwarding, serving, and home NSs (fNS,sNS,hNS) may be distinct. LoRaWAN employs two activation

methods, *Activation By Personalization* (ABP) and *Over-The-Air Activation* (OTAA). An ABP end-device is tied to a specific network during its manufacturing, and unlike an OTA activated ED, it does not go through a *join procedure*. OTAA is the recommended activation method for higher security applications [7]. In several cases, security has be considered separately for ABP and OTAA EDs.

2.1. Security session context

LoRaWAN session context consists of the network session and the application session. Network session is maintained by the ED and the serving network server. Information managed in a network session are the network session keys: *NetSKeys*, frame counters: *FCntUp* and *NFCntDwn*, and the device address: *DevAddr*. Application session is maintained by the ED and the AS. Information managed in an application session are the application session key: *AppSKey*, and the frame counters: *FCntUp* and *AFCntDwn*. During a session, session keys do not change, and frame counters are only incremented and never reused.

If a NS drops the old context immediately after processing the ED request which triggered the context switch (e.g. a join request), without waiting for a confirmation from the ED, situations may arise where the ED and the NS end up using different security contexts, and therefore lose the ability to communicate with each other. For example, it may be the case that the ED never intended for a context switch (e.g. replay attack). Neither version 1.0 nor version 1.1 specify the number of unconfirmed sessions a NS maintains simultaneously per ED.

2.2. Use of cryptography

LoRaWAN uses AES with 128-bit keys. The modes of operation used in LoRaWAN are CCM*, *Cipher-based Message Authentication Code* (CMAC), and *electronic codebook* (ECB). Counter with CBC-MAC (CCM) mode follows the authenticate-then-encrypt paradigm, and provides both confidentiality and authentication. CCM* is a minor variation of CCM, which allows encryption-only and integrity-only usage modes. LoRaWAN uses CCM* in encryption-only mode for encrypting MAC commands and application data. As the name suggests, CCM mode is derived from the counter (CTR) mode. Consequently, reuse of LoRaWAN's frame counters with the same key leads to two-time padding. When LoRaWAN uses CCM* mode for encryption, it uses CMAC mode for authentication, following the encrypt-then-authenticate paradigm. This allows forwarding network elements, which are not in possession of the encryption keys, to perform integrity checks.

LoRaWAN employs ECB mode for derivation of keys from the root keys, and also for the encryption of Join-accept messages. CCM* mode can not be used for Join-accept messages, because it is not possible to use *JoinNonce* for encryption (as part of the secret block), while also having *JoinNonce* itself encrypted. For *Join-accept* messages, message integrity code (MIC) is encrypted with the message (authenticate-then-encrypt paradigm). In this case, integrity check is performed only by the receiving ED.

2.3. Security keys and key derivation

Security of LoRaWAN is built on two AES-128 root keys, AppKey and NwkKey. Root keys are specific to an ED, and are stored by both the ED and its JS. Root keys are required for deriving the two lifetime keys JSIntKey and JSEncKey, the three network session keys NetSKeys (SNwkSIntKey, FNwkSIntKey, NwkSEncKey) and the application session key AppSKey. Root keys are also needed for implementing the integrity check of *Join-request* messages, and confidentiality of *Join-accept* messages that are triggered by *Join-request* messages. JSIntKey is employed for realizing the integrity check of *Rejoin-request* messages of type 1 and *Join-accept* messages. JSEncKey is used for implementing the confidentiality of *Join-accept* messages that are triggered by *Rejoin-request* messages. SNwkSIntKey is employed for ensuring the integrity of data messages and *Rejoin-request* messages of type 0 and type 2. FNwkSIntKey is used in partial integrity checks of uplink data messages by the forwarding NSs. NwkSEncKey is employed for protecting the confidentiality of MAC commands. AppSKey is used for ensuring the confidentiality of application data.

In case of OTAA, derivation of session keys happen in both the ED and the JS. Diversity of the session keys hinges on the nonce values involved in the derivation process. The two nonce values involved in the derivation of the four session keys are JoinNonce, and one of DevNonce, RJcount0 or RJcount1, depending on the type of the message which triggered the session key derivation. Whereas in case of ABP, session keys are assigned in the manufacturing stage, and the same session keys are used throughout the ED's lifetime.

2.4. Counters and nonces

In LoRaWAN, counters and nonces are employed for replay protection, and for ensuring correct usage of block cipher modes of operations. The general strategy adopted by LoRaWAN to deal with the saturation of counters is summarized as follows. In case of session counters, a security context switch is forced, resulting in the derivation of new session keys, and the resetting of session counters. For lifetime counters, recommendations are given on how to limit the periodicity of increments, so that counters do not overflow within the planned ED lifetime.

Frame counters LoRaWAN uses three frame counters to keep track of uplink/downlink data messages: FCntUp, NFCntDown and AFCntDown. FCntUp counts uplink data messages, and is incremented with each uplink, except

retransmissions. NFCntDown counts downlink data messages carrying only MAC commands. AFCntDown counts downlink data messages carrying application data (possibly along with MAC commands). FCntUp value is sent in FCnt field of the uplink messages. Receiving network element synchronizes the FCntUp counter for the ED with the received value. Similarly, either NFCntDown or AFCntDown is sent in FCnt field of downlink messages, and the receiving ED synchronizes the corresponding local counter with the received value. In both of these cases, the synchronization occurs only if the message is authenticated, and the received counter value is incremented compared to the local counter value, which is the previously observed value. Unlike OTAA where the frame counters are reset in every session, the counters for an ABP ED are never reset.

DevNonce and JoinNonce DevNonce is a 16-bit counter managed by the ED. It is initially set to 0, and then incremented with every transmission of a *Join-request* message. JoinNonce is a 24-bit counter. JS manages one JoinNonce counter per ED. JoinNonce is initially set to 0, and then incremented with every *Join-accept* message sent to the corresponding ED. DevNonce and JoinNonce are lifetime counters, and they must never repeat.

RJcount0 and RJcount1 are 16-bit counters managed by the ED. RJcount0 is incremented with every type 0 or type 2 *Rejoin-request* message sent, and reset to 0 each time a *Join-accept* is successfully processed. RJcount0 must never repeat within a session. RJcount1 is a lifetime counter, and is never reset. It is initially set to 0, and then incremented with every type 1 *Rejoin-request* message sent.

3. LoRaWAN v1.0.2 protocol vulnerabilities and their solutions in v1.1

Vulnerabilities in v1.0.2 can affect the security of LoRaWAN v1.1 in backward compatibility scenarios. In this regard, vulnerabilities in session management, join procedure, acknowledgment mechanism, and end-to-end integrity protection of LoRaWAN v1.0.2, along with how these vulnerabilities are addressed in v1.1 are presented in this section. The implications of the presented vulnerabilities are discussed in Section 6.

Vulnerability 1 - reuse of frame counter values: in v1.0.2 reuse of frame counter values with the same session keys and device address is not prevented. This vulnerability applies to both ABP and OTAA. Reuse of frame counters occurs when the device is reset or when the counter overflows in case of ABP, while in OTAA due to counter overflow within a session. LoRaWAN v1.1 addresses this vulnerability by introducing an explicit mechanism for rekeying EDs. The mechanism is enabled by the introduction of a new message type *Rejoin-request* and a new MAC command *ForceRejoinReq*. According to the specification [7], a new session context must be established before the saturation of a frame counter. Furthermore, frame counters of a v1.1 ABP ED are stored in nonvolatile memory, and are never reset during its lifetime despite power losses and device resets.

Vulnerability 2 - reuse of nonce values: by definition, a nonce is a number that can only be used once. Nonces in v1.0.2 (DevNonce and AppNonce[1]) are pseudo-randomly generated. It is possible that the same value may be regenerated and reused unless the previously used values are tracked. In version 1.0.2, EDs do not track used AppNonce values, and NSs track not all but an unspecified number of used DevNonce values[2]. Due to this, reuse is not fully prevented. Version 1.1 addresses this vulnerability by turning all nonces into counter nonces. Reuse of nonce values is prevented by storing and tracking only the last used values.

Vulnerability 3 - lack of replay protection mechanism for Join-accept messages: this protection mechanism is missing in v1.0.2, probably because a resource constrained ED is incapable of tracking sufficiently many values. In v1.1, an ED keeps track of the last observed JoinNonce value, referred as *JoinNonce_last* in the specification and this work. If a *Join-accept* message is replayed, it will be discarded by the ED based on the *JoinNonce_last* value. ED only accepts JoinNonce values which are incremented compared to the last observed value.

Vulnerability 4 - weak replay protection mechanism for Join-request messages: because a v1.0.2 NS tracks not all but the most recently used N DevNonce values, an attacker needs to wait for at most N activations before being able to replay a *Join-request* message. A version 1.1 NS keeps track of the last DevNonce used by each ED, *DevNonce_last*. If a *Join-request* message from a certain device is replayed, it will be discarded based on the *DevNonce_last* value. NS only accepts DevNonce values which are incremented compared to the last observed value.

Vulnerability 5 - acknowledgments unassociated with data: v1.0.2 does not provide a mechanism to associate the received acknowledgments with the confirmed data messages. LoRaWAN v1.1 addresses this vulnerability by including *ConfFCnt* in MIC calculation of data messages. If ACK flag is set in the data message, then ConfFCnt is set to the FCnt field of the acknowledged message to associate it with the acknowledgment. If a captured acknowledgment (whose reception was prevented) is replayed by the attacker for acknowledging another data message, the MIC check will fail.

Vulnerability 6 - Join-accept messages unassociated with requests: v1.0.2 lacks a mechanism to associate *Join-accept* messages with the requests that triggered them. LoRaWAN v1.1 addresses the vulnerability by including DevNonce in MIC calculation of *Join-accept* messages. The ED expects a *Join-accept* in response to a *Join-request*

[1] AppNonce is renamed to JoinNonce in version 1.1.

[2] Storing all 2^{16} possibilities of the 2-byte DevNonce would cost a NS approximately $2^{16}x2 = 130MB$ of memory per ED. So, it is plausible to assume that a NS implementation puts an upper limit on the number of tracked DevNonce values.

message it previously sent using a particular DevNonce value. The MIC check will fail if the MIC of the received *Join-accept* message was calculated using a different DevNonce value.

Vulnerability 7 - failure to confirm security session context switches: in v1.0.2 any valid uplink frame which uses the new security session context is interpreted by a NS as a confirmation of the new session. EDs, on the other hand, do not verify a context switch. In this case, situations may arise where the ED and the NS end up using different security contexts, and are disassociated. LoRaWAN v1.1 addresses the vulnerability by introducing the RekeyInd/RekeyConf MAC commands. A RekeyInd command sent by the ED using the new security session context, is interpreted by the NS as a confirmation of the new session. Upon confirmation, the NS drops any old security contexts it maintains. RekeyConf command sent by the NS in response to the RekeyInd command allows the ED to verify the security context switch in the NS.

Vulnerability 8 - missing end-to-end integrity protection: MIC checks for uplink application data, and MIC calculations for downlink application data are carried out at the serving NS. This leaves the integrity of application data unprotected as it is transported from the NS to the AS. Moreover, precise modifications to the application data are possible because the encryption of application data involves an XOR operation with a secret block and bit positions are preserved. The LoRaWAN specification (both v1.0.2 and v1.1) acknowledges the situation, and leaves the decision to implement end-to-end integrity for the applications. When end-to-end integrity is not separately implemented, security of application data depends on the assumption of honest NS, and on the security of the channel between NS-AS.

4. Attacks against LoRaWAN v1.0.2

This section discusses briefly the attacks identified against LoRaWAN version 1.0 that exploit the vulnerabilities presented above. Some of the presented attacks exploit more than one vulnerabilities. More details about the attacks presented below can be referred from [13] and [14].

4.1. Replay and eavesdropping

Replay and eavesdropping attacks are possible due to vulnerabilities in replay protection mechanisms, and in the management of frame counters and nonces. These attacks are caused by Vulnerability 1 to 4.

Attack 1 - replay and eavesdrop: this attack is applicable to both ABP and OTAA devices, but is easier to launch against ABP activated devices. Reuse of frame counter values (*Vulnerability 1*) allows an attacker to replay and/or eavesdrop on messages as the attacker is able to observe and capture multiple messages that are constructed using the same session keys and have the same frame counter values. In case of a successful replay attack, the legitimate data from the ED is discarded by the NS and the AS is made to process valid-but-old data. Reuse of frame counters within the same session leads to two-time padding, and in case of a successful eavesdropping attack plaintexts for the encrypted messages are recovered by the attacker.

Attack 2 - replay and eavesdrop via fake session on ED: this attack capitalizes on vulnerabilities in nonce usage and replay protection for *Join-accept* (Vulnerability 2 and 3). If the attacker intervenes in the *join procedure* and prevents the ED from receiving *Join-accept* messages, the ED keeps generating DevNonce values. The attacker records a session, and then waits for the reuse of the DevNonce from the recorded session. When the desired DevNonce value is observed, the attacker replays the *Join-accept* message from the recorded session to the ED, eliciting the creation of a fake session which does not involve the NS. The attacker may replay downlink frames of the recorded session to the ED, or eavesdrop on data in the same way as *Attack 1*. In the OTAA version of *Attack 1* frame counters are repeated within the same session due to an overflow, whereas in this attack, session keys are reused between one captured real session and one fake session. In both cases, the attacker observes the reuse of frame-counter and session-key pairs.

Attack 3 - replay and eavesdrop via fake session on NS: this attack is somewhat similar to *Attack 2* but the fake session is created on the NS. It exploits vulnerabilities in nonce usage and replay protection for *Join-request* messages (Vulnerability 2 and 4). When the desired AppNonce value is observed, the attacker replays the *Join-request* message from the recorded session, eliciting the creation of a fake session. Then, the attacker may replay uplink frames of the recorded session to the NS, or eavesdrop on data in the same way as *Attack 1*. The session confirmation carried out by the NS does not prevent this attack because the attacker is in possession of valid uplink frames which appear to use the new context, and replaying them causes the NS to confirm the fake session. Though the session confirmation mechanism introduced in version 1.1 is different, the attacker would still be able to convince the NS to use the new session, as the recorded session possibly includes a RekeyInd command in one or more of the first few uplink frames. This attack would fail if DevAddr in addition to the nonce were not reused. However, NS assigns DevAddr arbitrarily in both v1.0.2 and v1.1, and it is likely that DevAddr remains the same between sessions.

4.2. Ack spoofing and bit flipping

Attack 4 - ack spoofing: this attack exploits the lack of association between acknowledgments and confirmed data (Vulnerability 5). Attacker captures a downlink ACK message (a message with acknowledgment flag set), and later uses it to acknowledge another confirmed uplink message from the same ED. The attacker is assumed to have the capability to prevent reception of downlink frames based on DevAddr and ACK flag, for example through selective

jamming. Replay protection mechanism cannot prevent the attack because the reception is prevented. The attack can be performed similarly in the uplink direction but in this case the attacker has to have the capability of preventing reception of uplink frames by the GWs in the listening range.

Attack 5 - bit flipping: this attack capitalizes on missing end-to-end integrity protection of application data (Vulnerability 8). The attack assumes that the transport layer security between NS-AS is non-existing or compromised, and also that the attacker has the ability to act on the channel between the NS and the AS. Then, the attacker is able to make precise modifications to application data. If alteration of application data yields results which are observable to the attacker, then confidentiality of application data may also be compromised.

4.3. Denial of service on ED

Attack 6 - DoS on ED via Join-accept replay: this attack exploits the lack of replay protection mechanism for *Join-accept* messages, association between *Join-accept* messages and requests, and security session context confirmation (Vulnerability 3, 6 and 7). An attacker replies to a *Join-request* by the ED before the NS does, by replaying a *Join-accept* message which was previously sent to the same ED. ED derives its session keys using the nonce value in the replayed *Join-accept* message, which is different from the nonce used by the NS. ED and NS end up deriving different session keys, and lose their ability to communicate with each other, resulting in a DoS on the ED.

Attack 7 - DoS on ED via Join-request replay: this attack capitalizes on the incapability of the ED to confirm security session context switches, and the vulnerable replay protection mechanism for *Join-request* messages (Vulnerability 4 and 7). When a *Join-request* message is replayed by an attacker at an arbitrary time, a new unconfirmed session is created on the NS. The attacker is unable to make the NS confirm the new session and drop the old one, as the attacker is unable to craft valid uplink frames. However, by waiting for the ED to send a *Join-request* message, and replaying a previously captured *Join-request* message before the arrival of a session confirmation from the ED, the attacker is able to desynchronize the ED and the NS. If the NS keeps at most N unconfirmed sessions per ED, by replaying more than N requests, the NS can be forced to drop the session context ED switched to without waiting for any confirmation.

5. Backward Compatibility in LoRaWAN v1.1

The serving NS is responsible for making decision on which protocol version to be used and it chooses the highest common version between itself and the ED. This decision is transported to the JS within the *MACVersion* field of a *JoinReq* or *RejoinReq* message. In case of handover roaming, the decision on the protocol version is made by the network server which is in the process of becoming the new serving NS during the *handover roaming start* procedure, and is first transported to the current serving NS within the *MACVersion* field of a *HRStartReq* message. The JS processes the *Join-request* or *Rejoin-request* sent by the ED according to the value of the MACVersion field. The JS sets the OptNeg flag in the *Join-accept* message according to the received MACVersion value, so that the ED also learns about the decision of the NS and behaves accordingly. For example, the ED chooses the session key derivation strategy accordingly, and performs further version negotiation with the NS if required, via the RekeyInd/RekeyConf MAC commands.

For the discussion that follows, it is useful to note that version 1.0 features a single root key named AppKey, whereas version v1.1 features two root keys. The specification of v1.1 discusses only one fall-back case, where ED is v1.1 and back-end is v1.0.2. In this work, one more possible fall-back scenario is determined and presented.

5.1. Fall-back Case 1: ED v1.1 and Back-end v1.0.2

Due to the older version NS, a v1.1 ED falls back to v1.0.2 behavior for session-key derivation, and derives all the session keys, including AppSKey, using the root key NwkKey. MIC checks, MIC calculations, and encryptions are performed by the end-device as specified in the v1.0.2 specification [6]. Current specification [7] does not mention any other fall-back behaviours. However, a version 1.1 ED has to do a fall-back to v1.0.2 also for other tasks. For example, the v1.1 ED cannot use the replay-protection mechanism involving *JoinNonce_last* since a v1.0 back-end generates AppNonce values pseudorandomly. The session context confirmation mechanism introduced in version 1.1 cannot be used either, as the RekeyInd/RekeyConf MAC commands are not known at both ends.

5.2. Fall-back Case 2: ED v1.0.2 and Back-end v1.1

Though, this fall-back case is not mentioned in the v1.1 specification [7], it is clear that the back-end network elements have to exhibit similar fall-back behaviour, if they want to successfully communicate with a version 1.0.2 ED. The JS has to fall back to v1.0 behaviour for session-key derivation, and derive all session keys using the single root key (AppKey) of the ED. MIC checks, MIC calculations, and encryptions have to be performed by the back-end as specified in the 1.0.2 specification [6]. Other mechanisms affected by the fall-back are 1) the v1.1 back-end cannot use the replay protection mechanism involving *DevNonce_last* because a v1.0 ED generates pseudo-random DevNonce values; 2) the session context confirmation mechanism introduced in version 1.1 cannot be used; and 3)back-end is unable to rekey an ED, as the *ForceRejoinReq* MAC command is not known to the ED.

6. Applicability of v1.0.2 attacks in backward compatibility scenarios and possible countermeasures

In case of the above two fall-back scenarios, the previously mentioned v1.0.2 attacks are reinstated. In this section, the applicability of these attacks are discussed and possible countermeasures for replay, eavesdrop and denial of service attacks are proposed. Table 1 maps the attacks with the fall-back cases that reinstated them.

6.1. Replay and eavesdrop

Attack 1 - replay and eavesdrop: is applicable to Fall-back Case 1. Even though ABP device resets do not help an attacker in this case, waiting for a frame counter overflow still works in both ABP and OTAA ED. In case of frame counter saturation, the NS will not send a ForceRejoinReq MAC command to rekey the OTAA ED. In order to prevent the replay and eavesdrop attacks, we suggest that a v1.1 ED does not rely on rekeyings initiated by the NS. ED must itself recognize frame counter saturation and initiate the establishment of a new session. In the absence of a ForceRejoinReq, the ED can force rekeying by sending a *Join-request* message [8]. In Fall-back Case 2, both resetting an ABP device and waiting for a frame counter overflow can be used as ways to launch the replay and eavesdrop attack. As the NS has no means to force the establishment of a new session in this fall-back scenario, we suggest that a v1.1 NS in session with a v1.0 OTAA ED recognizes frame counter saturation and refuses communication with the ED until the ED initiates an activation. In the case of an ABP ED, we suggest that communication is indefinitely discontinued, as it is not possible to rekey an ABP device.

Attack 2 - replay and eavesdrop via fake session on ED: is not applicable to Fall-back Case 1, because a v1.1 ED does not reuse a DevNonce value. Even though the replay protection mechanism involving *JoinNonce_last* cannot be used in this configuration, exploiting the replay protection vulnerability is insufficient for carrying out the attack. In Fall-back Case 2, the attack is applicable as-is, because it targets solely the ED.

Attack 3 - Replay and Eavesdrop via fake session on NS is applicable to Fall-back Case 1 as is, because it targets solely the NS. The attack is not applicable to Fall-back Case 2, because a v1.1 JS does not reuse a JoinNonce value. On the other hand, the improved replay protection mechanism involving *DevNonce_last* cannot be used in this configuration.

6.2. Ack spoofing and bit flipping

Attack 4 - ack spoofing: the mechanism introduced in v1.1 to solve ack spoofing is the new MIC calculation, which involves (part of) the frame counter of the acknowledged frame. The MIC calculation and MIC check have to be carried out in a consistent way. Using the highest common version is the only option, and in case of these two fall-back scenarios, that version can only be v1.0.2. So, the same attack in v1.0.2 is reinstated in both Fall-back Case 1 and Case 2.

Attack 5 - bit flipping: applicability of this attack in both fall-back cases are the same as their applicability in normal operation using compatible ED and back-end (v1.1 or v1.0.2). In other words, a fall-back does not affect the applicability of this attack. The success of this attack in case of normal operation depends on 1) compromised integrity of communication channel between NS-AS, and 2) attacker capability to perform man-in-the middle attack on this channel.

6.3. Denial of service

Attack 6 - DoS on ED via Join-accept replay: is applicable to Fall-back Case 1, because neither the replay protection mechanism involving *JoinNonce_last*, nor the v1.1 *Join-accept* MIC calculation allowing association with the requests can be used. Furthermore, the ED is not able to verify security context switch in the NS. Thus, the attack is applicable as is (i.e. replay of a single message is sufficient). If the ED were able to verify security context switch, it would eventually revert to join state due to the absence of a RekeyConf MAC command. Consequently, replay of a single message would not be sufficient, and the attacker would have to continue replaying *Join-accept* messages in order to achieve a DoS. We suggest that, when communicating with a v1.0 back-end, a v1.1 ED verifies a session context change in a way similar to how a v1.0 NS verifies a context change, that is, any valid downlink frame from the NS which uses the new security session context is to be interpreted as a confirmation. This way, an attacker would not be able to achieve a long-term DoS via the replay of a single *Join-accept* message. In Fall-back Case 2, the attack is applicable as is, because it targets solely the ED.

Attack 7 - DoS on ED via Join-request replay: is applicable to Fall-back Case 1, because the back-end is vulnerable to *Join-request* replays, and the ED is unable to verify security context switches. We suggest that, when communicating with a v1.0 back-end, a v1.1 ED verifies a session context change in a way similar to how a v1.0 NS verifies a context change: any valid downlink frame from the NS which uses the new security session context is to be interpreted as a confirmation. Even though *Join-request* replays are still possible, the ED would keep the initial state as the confirmed state, and would not discard it in the absence of a confirmation from the NS. The attack is also applicable to Fall-back Case 2, because a v1.1 JS has to revert to the vulnerable replay protection mechanism, and a v1.0 ED does not validate security context switches. We suggest that, when communicating with a v1.0 ED, a v1.1 back-end keeps track all of used DevNonce values and do not reply to *Join-request* messages with reused DevNonce values. As pointed out in [14], this approach requires memory efficient techniques. Even though the vulnerability in

session context confirmation is exploitable, the attack cannot be carried out as the attacker would not be able to replay *Join-request* messages.

Table 1. Applicability of the attacks in backward compatibility scenarios.

Attack	Fall-back Case 1 (ED v1.1 & Back-end v1.0.2)	Fall-back Case 2 (ED v1.0.2 & Back-end v1.1)
Attack 1 - Replay & Eavesdrop	**Yes**[*]	**Yes**[*]
Attack 2 - R & E via fake session on ED	No	**Yes**
Attack 3 - R & E via fake session on NS	**Yes**	No
Attack 4 - ACK spoofing	**Yes**	**Yes**
Attack 5 - Bit flipping	Maybe[‡]	Maybe[‡]
Attack 6 - DoS via Join-accept replay	**Yes**[†]	**Yes**
Attack 7 - DoS via Join-request replay	**Yes**[*]	**Yes**[*]

[*] A countermeasure is proposed.
[†] A partial countermeasure is proposed.
[‡] Applicability in versions 1.0.2 and 1.1 depends on a strong assumption. Applicability is not affected by fall-back.

7. Conclusion

In this work, the security issues of LoRaWAN version 1.1 in backward compatibility scenarios were examined and countermeasures were proposed. LoRaWAN v1.1 specification addresses only one backward compatibility case, ED v1.1 and back-end v1.0.2. It describes session key derivation but fails to specify how the other security mechanisms should be implemented for this fall-back case. Despite the high probability of occurrence, the other backward compatibility case, ED v1.0.2 and back-end v1.1, is not addressed at all. This work consolidated the vulnerabilities of v1.0.2 and the resulting attacks, and then defined the missing fall-back case. Based on these, the applicability of the attacks in the two fall-back cases are examined and countermeasures are proposed. Though v1.1 has addressed replay, eavesdropping, ACK spoofing, and DoS attacks, the investigations in this work revealed that all of these attacks are reinstated in fall-back cases. The proposed countermeasures enable prevention of replay and eavesdrop attacks that exploit reuse of frame counter values, as well as denial of service attack. These countermeasures are easily implementable without causing major changes to the protocol.

Acknowledgements

This work is supported in part by Tekes under the project Wireless for Verticals (WIVE). WIVE is a part of 5G Test Network Finland (5GTNF).

References

[1] "Gartner forecast", February 2017. URL:https://www.gartner.com/newsroom/id/3598917.
[2] "Ericsson Mobility Report", June 2017 https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf
[3] Nicola Dragoni, Alberto Giaretta, and Manuel Mazzara. (2018) "The Internet of Hackable Things", in Paolo Ciancarini et al. (eds) *PROCEEDINGS OF 5TH INTERNATIONAL CONFERENCE IN SOFTWARE ENGINEERING FOR DEFENCE APPLICATIONS, Advances in Intelligent Systems and Computing*, Springer, Vol. 717, pp. 129–140. doi: 10.13140/RG.2.2.19643.72482
[4] "LoRa Alliance Technology", https://www.lora-alliance.org/technology
[5] U. Raza, P. Kulkarni and M. Sooriyabandara, "Low Power Wide Area Networks: An Overview," in IEEE Communications Surveys & Tutorials, vol. 19, no. 2, pp. 855–873, Secondquarter 2017. doi: 10.1109/COMST.2017.2652320
[6] LoRa Alliance Technical Committee. LoRaWAN™ Specification, July 2016. LoRa Alliance, version 1.0.2.
[7] LoRa Alliance Technical Committee. LoRaWAN™ Specification, Oct 2017. LoRa Alliance, version 1.1.
[8] LoRa Alliance Technical Committee. LoRaWAN™ Backend Interfaces 1.0 Specification, Oct 2017. LoRa Alliance, version 1.0.
[9] E. Aras, G. S. Ramachandran, P. Lawrence and D. Hughes. (2017). "Exploring the Security Vulnerabilities of LoRa," in 3rd IEEE International Conference on Cybernetics (CYBCONF), Exeter, 2017, pp. 1-6. doi: 10.1109/CYBConf.2017.7985777
[10] E. Aras, N. Small, G. S. Ramachandran, S. Delbruel, W. Joosen, D. Hughes. (2017). "Selective Jamming of LoRaWAN using Commodity Hardware". doi: 10.1145/3144457.3144478.
[11] S. Tomasin, S. Zulian and L. Vangelista, "Security Analysis of LoRaWAN Join Procedure for Internet of Things Networks," 2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), San Francisco, CA, 2017, pp. 1-6. doi: 10.1109/WCNCW.2017.7919091
[12] Notes on LoRaWAN security. Feb 1, 2017. URL: https://medium.com/@brocaar/notes-on-lorawan-security-7e741a8ee4fa (visited on 2017).
[13] Xueying Yang. "Lorawan: Vulnerability Analysis and Practical Exploitation". Delft University of Technology, 2017. URL: https://repository.tudelft.nl/islandora/object/uuid:87730790-6166-4424-9d82-8fe815733f1e?collection=education
[14] Gildas Avoine, Loïc Ferreira, "Rescuing LoRaWAN 1.0," unpublished. URL: https://fc18.ifca.ai/preproceedings/13.pdf
[15] Jaehyu Kim and JooSeok Song, "A Dual Key-Based Activation Scheme for Secure LoRaWAN," Wireless Communications and Mobile Computing, vol. 2017, Article ID 6590713, 12 pages, 2017. doi:10.1155/2017/6590713
[16] SeungJae Na, DongYeop Hwang, WoonSeob Shin and Ki-Hyung Kim, "Scenario and countermeasure for replay attack using join request messages in LoRaWAN," 2017 International Conference on Information Networking (ICOIN), Da Nang, 2017, pp. 718-720. doi: 10.1109/ICOIN.2017.7899580