Etude sur la sécurisation des réseaux LoRaWAN®

Thierry Didi,
Juillet 2025
Tidiwi

Les technologies cellulaires opérant dans les bandes licenciées, telles que LTE-M et NB-IoT, sont aujourd'hui largement déployées. Leur émergence aurait pu marginaliser les solutions concurrentes fonctionnant sur des bandes non licenciées, notamment LoRaWAN® et Sigfox®. C'est d'ailleurs ce qui s'est produit pour Sigfox. En France, de la même manière, l'opérateur Bouygues Telecom a mis fin à son réseau LoRaWAN. Pourtant, LoRaWAN a su « tirer son épingle du jeu » en s'imposant dans de nombreux domaines : villes intelligentes, bâtiments connectés, agriculture de précision, etc.

De nombreuses métropoles ont déployé des réseaux LoRaWAN privés afin de superviser un vaste parc de capteurs urbains. On retrouve également une forte présence de capteurs LoRaWAN dans la gestion technique des bâtiments tertiaires ainsi que dans les infrastructures agricoles. Le protocole cherche aussi à s'étendre vers d'autres secteurs, comme l'habitat intelligent.

Ce document est structuré de la manière suivante :

- Les technologies LoRa® et LoRaWAN®
  Il débute par une présentation du protocole LoRaWAN, indispensable pour aborder les aspects liés à la sécurité. Il ne détaille cependant pas les éléments techniques de bas niveau tels que la modulation LoRa, les Spreading Factors ou les débits atteignables, pour lesquels de nombreuses ressources spécialisées sont disponibles en ligne.
- Les menaces contre les réseaux LoRaWAN
  L'analyse se concentre ensuite sur les menaces auxquelles ces réseaux sont confrontés, et aux conséquences potentielles pour le client final.
- La sécurisation d'un réseau LoRaWAN
  Enfin, il décrit les bonnes pratiques à mettre en en œuvre pour sécuriser un réseau LoRaWAN en fonction de la topologie déployée.

Remarque préliminaire : Ce document s'appuie exclusivement sur des sources et des spécifications publiquement disponibles. Aucune information confidentielle n'a été utilisée. Les opinions et conclusions présentées ici sont celles de l'auteur, et ne reflètent en aucun cas celles de la société Semtech ni de la LoRa Alliance.

Si vous avez des remarques ou des suggestions d'amélioration, n'hésitez pas à m'en faire part par email : thierry.didi[at]tidiwi[dot]com.

©Tidiwi 2025

1 Introduction

Avant de rentrer dans les détails des protocoles, il est utile de clarifier quelques éléments de terminologie :

- **ABP : « A**ctivation **B**y **P**ersonnalization » est une des deux méthodes d'activation d'un terminal LoRaWAN.

- **AES :** est un algorithme de chiffrement symétrique robuste. Dans le protocole LoRaWAN, il est utilisé avec des clés de 128 bits. AES permet de chiffrer des messages de 128 bits (16 octets).

- **AES ECB :** AES-ECB est un mode de chiffrement basé sur l'algorithme AES, qui permet de chiffrer des messages dont la longueur est un multiple de 16 octets. Chaque bloc de 16 octets est chiffré indépendamment avec la même clé AES. Ce mode n'offre aucune protection contre la répétition de données : si l'on chiffre deux fois le même message avec la même clé, ou si le message contient des blocs identiques, les blocs chiffrés seront eux aussi identiques. Cela rend AES-ECB inadapté à la plupart des usages cryptographiques.

- **AES CTR :** AES-CTR est un mode de chiffrement basé sur l'algorithme AES, qui permet de chiffrer des messages de taille arbitraire. Pour que le chiffrement soit sécurisé, AES impose l'utilisation d'un vecteur d'initialisation (IV) — ou un compteur initial — différent à chaque exécution. Ainsi, si l'on chiffre deux fois le même message avec des IV différents, on obtiendra des résultats chiffrés différents. En revanche, si le même IV est réutilisé avec la même clé, les résultats seront identiques, ce qui compromettra la sécurité.

- **AppKey :** Clé AES connue du terminal et du réseau, qui permet de dériver les clés de chiffrement qui seront utilisées pour sécuriser les échanges entre un terminal et le réseau. La clé AppKey d'un terminal est le paramètre de sécurité le plus critique qu'il faut absolument protéger.

- **ARM®** est une société qui conçoit des microprocesseurs, mais qui ne les fabrique pas. La plupart des processeurs et des *SoC*s développés par les grands fabricants comme STMicroelectronics, Texas Instruments, NXP Semiconductors … sont basés sur des architecture ARM.

- **Classe A, B, C :** LoRaWAN définit trois classes d'équipements — A, B et C. Les équipements de classe A sont les plus répandus. Leur radio est éteinte la majeure partie du temps : ils se réveillent de manière autonome pour émettre un message, ouvrant ensuite deux courtes fenêtres de réception pour éventuellement recevoir une réponse. Les équipements de classe C, quant à eux, sont en écoute permanente (hors transmission), ce qui leur permet de recevoir des messages à tout moment. Enfin, les équipements de classe B fonctionnent comme ceux de classe A, mais ouvrent en plus des fenêtres de réception régulières synchronisées avec un signal de balise (beacon), afin de permettre des réceptions programmées.

©Tidiwi 2025

- **Downlink :** décrit le sens des communications, partant du réseau vers un terminal radio.

- **FUOTA : F**irmware **U**pdate **O**ver **T**he **A**ir.

- **Gateway :** Appareil qui reçoit les messages des équipements terminaux et les transmet au *LNS*, et qui reçoit les messages provenant du *LNS* pour les transmettre aux équipements terminaux.

- **Join procedure** : procédure par laquelle un terminal LoRaWAN activé en *OTAA* obtient son adresse et ses clés de chiffrement.

- **JS (J**oin **S**erver**)** : Serveur qui connaît les clés *AppKey* de tous les terminaux du réseau, et dérive les clés de session utilisées pour sécuriser les échanges entre le *LNS* et les terminaux. L'intérêt de ce serveur est qu'il peut être opéré par le client final quand le LNS est opéré par un opérateur : le client final est donc le seul à connaître les clés AppKey de ses terminaux. Ce serveur peut aussi être intégré dans le LNS si la connaissance des clés *AppKey* par le LNS ne pose pas de problème de sécurité.

- **LNS (L**oRaWAN **N**etwork **S**erver**) :** Ce serveur est responsable de l'enregistrement des terminaux sur le réseau LoRaWAN, puis de la gestion de la confidentialité et de l'intégrité des messages LoRaWAN. Il communique avec les terminaux avec le protocole LoRaWAN au travers des Gateways LoraWAN, et avec les serveurs applicatifs avec le protocole TCP/IP via l'internet ou via un réseau privé.

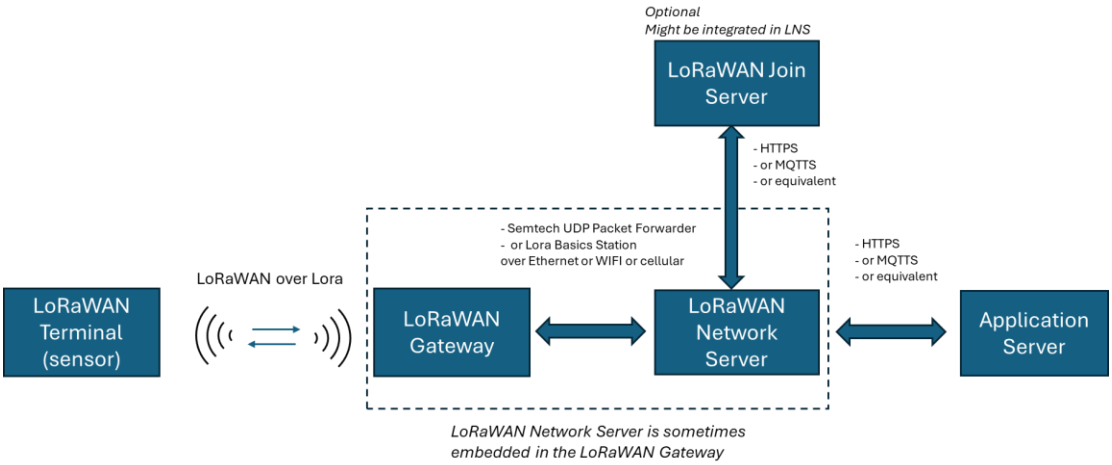


Figure 1 - LoRaWAN Network Architecture

-

MIC : Message Integrity Code. Il s'agit d'une valeur de 32 bits calculée par l'émetteur d'un message, à partir du contenu du message et d'une clé de chiffrement. L'émetteur ajoute ce code MIC à la fin du message transmis. Le récepteur, qui dispose de la même clé de chiffrement, peut vérifier que le message n'a pas été altéré par un acteur malveillant.
Note : dans les protocoles traditionnels, ce code est plutôt appelé code MAC. Mais le terme MAC a une signification particulière dans les protocoles radio, et il a donc été remplacé par MIC.

©Tidiwi 2025

- **Module Radio** : Un module Radio est composé de plusieurs composants électroniques assemblés par un constructeur sur un circuit imprimé, et le plus souvent encapsulés dans un blindage métallique. Les composants intégrés dans un module peuvent être des processeurs, des *transceivers*, des *SoC*s, des quartz, et quelques composants passifs (résistances, condensateurs, selfs). Ils sont pré-certifiés pour respecter les normes Radio en vigueur, et leur utilisation simplifie donc grandement le développement des équipements radioélectriques. Les modules LoRaWAN sont aussi certifiés par rapport au label « LoRaWAN certified[1]».

- **OTAA : « O**ver **T**he **A**ir **A**ctivation » est la méthode à privilégier pour activer enregistrer un terminal LoRaWAN auprès d'un *Serveur LoRaWAN*.

- **Payload :** Un message radio contient un certain nombre d'informations permettant au récepteur de se synchroniser sur l'émetteur ou de détecter/corriger des erreurs de transmission. Mais ce message radio a pour vocation de transmettre un message applicatif, qui est appelée la payload du message radio. Un message radio est donc composé d'une payload, et d'un certain nombre d'autres octets destinés à acheminer cette payload.

- **Replay Attack :** une attaque de type replay consiste à capturer un message envoyé par un équipement (une commande d'ouverture de porte par exemple), et de le « rejouer » plus tard pour faire croire au récepteur qu'une nouvelle commande a été envoyée.

- **Serveur Applicatif :** Le Serveur Applicatif est l'interlocuteur « final » des terminaux connectés à un réseau LoRaWAN (Cf Figure 1 - LoRaWAN Network Architecture). Les messages émis par les terminaux sur l'interface radio (par exemple les mesures prises par un capteur) sont reçus par une *gateway* LoRaWAN, qui les transmet à un *LNS* éventuellement intégré à la gateway LoRaWAN. Le *LNS* les transmet au Serveur Applicatif qui les interprète. Les messages provenant du Serveur Applicatif et destinés aux équipements terminaux suivent le chemin inverse.

- **SoC** : System On Chip. Dans ce document, un SoC est un composant électronique qui intègre un processeur, de la mémoire volatile, de la mémoire non volatile et un *transceiver* Radio.

- **Terminaux :** Dans ce document, les terminaux sont les capteurs ou actionneurs qui utilisent le protocole LoRaWAN pour communiquer.

- **Transceiver**. Un transceiver est un composant électronique qui permet de **trans**mettre et de **recev**oir des ondes radio. Il peut s'agit d'un composant discret qui sera connecté à un processeur via une interface série (SPI, UART, I2C), ou il peut être intégré à l'intérieur d'un *SoC*.

- **Uplink :** décrit le sens des communications, partant d'un terminal radio vers le réseau.

[1] https://lora-alliance.org/lorawan-certification/

©Tidiwi 2025

2 La technologie LoRa®/LoRaWAN®

On utilise souvent le terme LoRa en parlant du protocole LoRaWAN. Il est donc utile de clarifier ce que représentent effectivement ces deux termes.

2.1 LoRa®

LoRa est une modulation brevetée par Semtech®, dérivée de la modulation CSS (Chirp Spread Spectrum[2]). Semtech est le seul fondeur à fournir des transceivers Lora. Il est possible de se procurer des *SOCs* (System On Chip) qui intègrent un processeur *ARM* et un *transceiver* LoRa chez des fondeurs tels que ST Microelectronics[3], ou des *modules* qui intègrent un processeur et un *transceiver* LoRa chez la plupart des fournisseurs de modules Radio. L'intérêt de la modulation Lora est qu'elle permet de transmettre des signaux radio sur des distances de plusieurs kilomètres et qu'elle offre aussi une bonne pénétration dans les environnements intérieurs profonds (« Deep Indoor »).

Remarque
Lora n'est donc pas un standard ouvert, à l'inverse d'autres technologies comme LTE-M, NB-IOT, Thread ou Zigbee. Semtech est le seul fournisseur de cette technologie, même si certains fondeurs comme STMicroelectronics ont conclu des accords avec Semtech pour intégrer un transceiver Semtech dans certains de leurs SoCs. Par exemple, il n'existe pas de SoC Lora chez Texas Instruments, Silicon Labs, NXP Semiconductors ...

Ce point peut être considéré comme une faiblesse de cette technologie (un seul fournisseur), mais aussi comme une force : la compatibilité entre deux transceivers Lora est garantie puisqu'ils proviennent du même fondeur.

LoRa ne définit pas les paramètres de sécurité, ni la technique d'accès à l'interface radio, ni l'adressage, mais seulement la modulation à utiliser et un format de trame Radio « très simple », illustré ci-dessous :

Figure 2 – Lora frame structure

Une trame radio Lora contient un « Préambule » destiné à réveiller un récepteur, un « Header » qui contient des informations sur la longueur de la *payload* et sur le codage utilisé pour la transmettre, et essentiellement une *payload* qui est le vrai contenu du message. Une trame LoRa peut aussi contenir un « CRC » (Contrôle de Redondance Cyclique) qui permet au récepteur de vérifier si le contenu de la trame n'a pas été altéré pendant son transport sur l'interface Radio.

Le but de ce document n'est pas de décrire dans le détail la modulation LoRa ni la structure des trames LoRa. Pour plus de détail, vous pouvez vous référer à la datasheet des transceivers SX1276/77/78/79[4] qui explique ceci dans les détails.

[2] https://en.wikipedia.org/wiki/Chirp_spread_spectrum
[3] ST Microelectronics: STM32WL5x - Wireless MCUs with LoRa support - STMicroelectronics
[4] https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE

©Tidiwi 2025

Il est possible d'implémenter beaucoup de protocoles au dessus de Lora, dans des topologies de réseau en étoile (« Star Networks ») ou de réseau maillé (« Mesh Networks ») mais le protocole le plus répandu actuellement est le protocole LoRaWAN qui fait l'objet de ce document.

2.2 LoRaWAN®

LoRaWAN est un protocole ouvert dont les spécifications[5,6] sont publiées par la Lora Alliance® et qui définit :

- **La couche radio à utiliser**
  - LoRaWAN fonctionne en principe sur la couche physique Lora,
  - Une évolution récente[7] a aussi spécifié LoRaWAN sur une autre couche physique, LR-FHSS (Long Range Frequency Hopping Spread Spectrum). Cette autre couche est adaptée à des communications très longue portée (plusieurs centaines de kilomètres) notamment pour les communications (uplink uniquement) vers les satellites. LR-FHSS est aussi brevetée par Semtech. Actuellement, les SOCs de la série STM32WL de ST Microelectronics intègrent un *transceiver* Semtech qui supporte les deux modulations Lora et LR-FHSS.

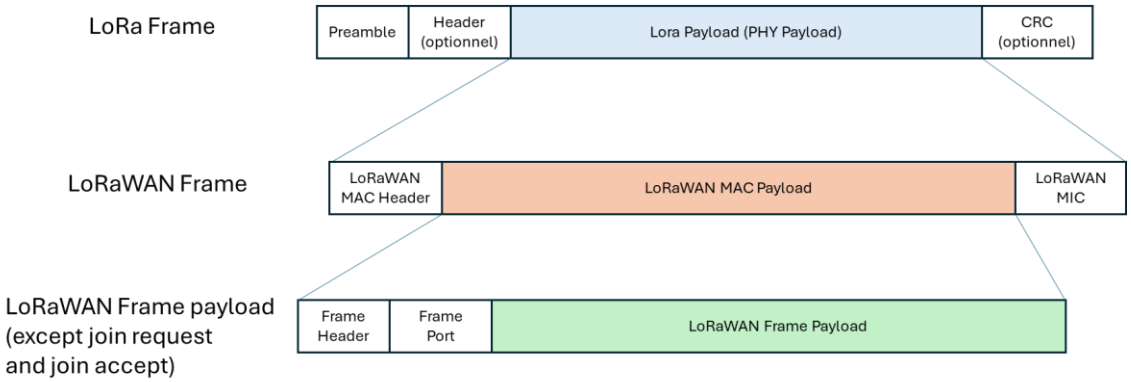
- **La plage de fréquences à utiliser**, qui dépend de la localisation des appareils.
  - LoRaWAN fonctionne dans la bande ISM 863-870MHz en Europe[10].

- **La taille maximum des payloads des messages LoRaWAN**
  Cette taille maximum dépend de la région.
  En Europe, cette taille maximum est définie dans le document « RP002-1.0.3 LoRaWAN® Regional Parameters». Elle varie entre 59 octets et 230 octets en fonction du Spreading Factor utilisé.

- **Le format des trames LoRaWAN** qui seront transportées dans la payload d'une trame LoRa (ou éventuellement d'une autre modulation autorisée dans la spécification). Ce format est illustré ci-dessous :

[5] LoraWan Technical Specification V1.0.4 : https://resources.lora-alliance.org/technical-specifications/ts001-1-0-4-lorawan-l2-1-0-4-specification
[6] RP2-1.0.3 LoRaWAN® Regional Parameters: https://resources.lora-alliance.org/technical-specifications/rp2-1-0-3-lorawan-regional-parameters
[7] Lora Specification V1.0.4 – Octobre 2020

©Tidiwi 2025



Les contenus des champs représentés ci-dessus sont décrits dans la spécification du protocole LoRaWAN[7]. Cette spécification décrit :

- Le format des adresses des appareils (4 octets),
- Le chiffrement des messages LoRaWAN (AES-CTR),
- L'intégrité des messages LoRaWAN (AES-CMAC),
- Les procédures pour envoyer/recevoir des données entre un terminal et un serveur.

2.3 Principales fonctionnalités de LoRaWAN

Les sections suivantes décrivent quelques fonctionnalités importantes de LoRaWAN

- La bidirectionnalité dans LoRaWAN
- La gestion des canaux radio
- L'activation d'un équipement
- La gestion du chiffrement
- La gestion de l'intégrité
- La gestion des répétitions
- La protection contre les attaques de type « Replay Attacks »
- Le protocole entre les gateways et le serveur LoRaWAN
- Les mises à jour du Firmware Over The Air (*FUOTA*)

Remarque
Il est important de noter que SemTech propose en open source une implémentation complète de la spécification LoRaWAN et des packages optionnels, notamment FUOTA. Le code de cette implémentation nommée Lora Basics Modem[8] est disponible sous github. Ce point est important car la majorité des terminaux LoraWAN utilisent ce code, ce qui améliore grandement l'interopérabilité avec les gateways et les LNS, sachant que l'interopérabilité est un enjeu majeur pour l'adoption d'une technologie.

[8] https://github.com/Lora-net/SWL2001/tree/master

©Tidiwi 2025

2.3.1 La bidirectionnalité dans LoRaWAN

Pour économiser leur batterie, les terminaux LoRaWAN de Classe A gardent le plus souvent leur radio éteinte, et ne peuvent donc pas recevoir de messages. Le mécanisme suivant a été spécifié pour permettre à ces équipements de revoir des messages :

- Après une émission, un appareil de classe A ouvre une fenêtre de réception (RX1) sur le même canal radio que celui qu'il a utilisé pour transmettre. S'il ne reçoit rien, il ouvre une autre fenêtre (RX2) une seconde plus tard sur un canal prédéfini. Ce mécanisme est représenté ci-dessous :

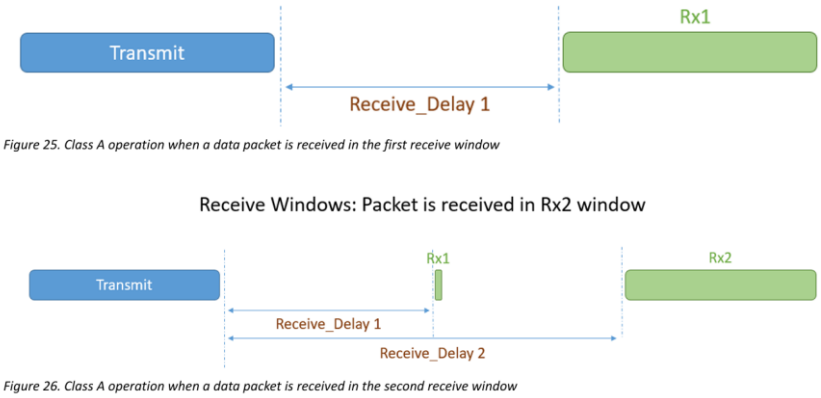


Figure 3 - Source : Semtech AN1200.86

- Les délais entre l'émission d'une trame et l'ouverture des fenêtres RX1 et RX2 sont configurés par le LNS[9].

2.3.2 Les canaux / fréquences définis par LoRaWAN

En Europe[10], un terminal LoRaWAN doit obligatoirement implémenter[11] les trois canaux suivants qui seront utilisés en *Uplink* et en *Downlink* : 861.1MHz, 861.3MHz, 861.5MHz. Ces canaux sont notamment utilisés pour émettre les messages « join requests » vers le *Serveur LoRaWAN*. Dans sa réponse au « Join Request », un *serveur LoRaWAN* peut optionnellement ajouter 5 canaux additionnels. Enfin il peut encore 8 autres canaux dans la bande 867-870MHz en cas réseau très dense via un autre message du protocole[12].

Les trois canaux obligatoirement supportés par tous les terminaux LoRaWAN sont les suivants en Europe :

Canal	Fréquence	Bande passante	DR (DataRate) / Bitrate	Duty cycle
0	868.1 MHz	125 kHz	DR0 à DR5 / 0.3-5kbps	1%
1	868.3 MHz	125 kHz	DR0 à DR5 / 0.3-5kbps	1%
2	868.5 MHz	125 kHz	DR0 à DR5 / 0.3-5kbps	1%

[9] Les délais Receive_Delay1 et Receive_Delay2 sont définis dans le message RXParamSetupReq.
[10] Pour les autres régions, Cf RP002-1.0.3- LoRaWAN Regional Parameters
[11] https://resources.lora-alliance.org/technical-specifications/rp2-1-0-3-lorawan-regional-parameters
[12] Le message NewChannelReq permet d'ajouter ou de modifier des canaux.

©Tidiwi 2025

En Europe, les cinq canaux configurés par le serveur dans la réponse au « join request » sont libres, dans la mesure où ils respectent la règlementation, et pourraient être configurés par le serveur de la manière suivante :

Canal	Fréquence	Bande passante	DR (DataRate) / Bitrate	Duty Cycle
4	867.1 MHz	125 kHz	DR0 à DR5 / 0.3-5kbps	1%
5	867.3 MHz	125 kHz	DR0 à DR5 / 0.3-5kbps	1%
6	867.5 MHz	125 kHz	DR0 à DR5 / 0.3-5kbps	1%
7	867.7 MHz	125 kHz	DR0 à DR5 / 0.3-5kbps	1%
8	867.9 MHz	125 kHz	DR0 à DR5 / 0.3-5kbps	1%

- Un canal spécial RX2[13] est dédié au *downlink*, avec un meilleur Duty Cycle et une puissance plus importante[14]. Le *LNS* peut décider, pour des raisons qui lui sont propres, d'utiliser ce canal pour envoyer une requête ou un acquittement à un terminal. Ce canal doit donc aussi être supporté par les terminaux, en réception uniquement.

[13]valeur par défaut = 869.525MHz, bande passante=125kHz

[14] Le canal RX2 supporte un duty cycle de 10% au lieu de 1% pour les autres canaux et permet d'émettre à 27dBm au lieu de 14dBm pour les autres canaux.

©Tidiwi 2025

2.3.3 L'activation d'un équipement

Configuration en usine

En premier lieu, un terminal doit avoir été configuré au moment de son installation avec le minimum de paramètres suivants :

- **devEUI** : Un identifiant de terminal unique codé sur 64 bits. Ce paramètre peut être public.
- **joinEUI** : Un identifiant du Serveur de « join » codé sur 64 bits. Ce paramètre peut être public.
- **AppKey** : Une clé secrète AES-128 qui sera utilisée lors de la procédure de « join », nécessaire seulement dans le cas d'une activation OTAA comme décrit ci-dessous. Cette clé sera aussi connue du LNS (ou du Join Server). **Ce paramètre doit rester secret**.

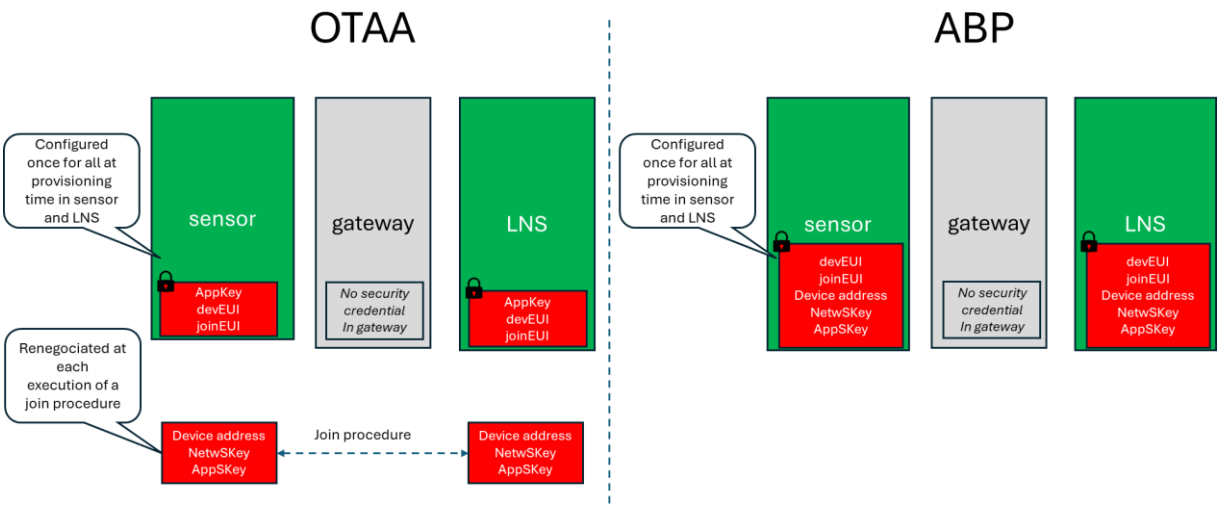
Activation

Pour pouvoir communiquer avec un LNS, ce terminal doit être activé. Le but de cette activation est de doter cet équipement :

- D'une **adresse** (32 Bits) : allouée par le serveur dans le cas *OTAA* ou configurée à l'installation dans le cas *ABP* comme ce sera décrit plus loin.
- D'une clé AES 128 bits (**NwkSKey**) pour chiffrer la payload des messages contenant des commandes réseau, et pour calculer le code MIC des trames LoRaWAN.
- D'une clé AES 128 bits **(AppSKey**, à ne pas confondre avec **AppKey)** pour chiffrer la payload des messages contenant des données de l'application finale. L'intérêt de cette deuxième clé est que le fournisseur du réseau ne sera pas capable de déchiffrer les messages applicatifs[15].

Cette activation peut se faire de deux manières :

- Activation By Personnalisation (*ABP*) : mode d'activation NON recommandé.
- Activation Over The Air (*OTAA*) : mode d'activation recommandé.



[15] Ce point est un peu « théorique » si le Join Server est intégré au LNS puisque dans ce cas le LNS connaît les clés AppSKey. Donc, si le LNS est compromis, les messages applicatifs le sont aussi.

Activation OTAA

L'activation OTAA (Over-The-Air Activation) est le mode recommandé pour garantir un niveau de sécurité élevé. Contrairement à la méthode ABP (décrite plus loin), OTAA permet de générer dynamiquement deux clés de session qui seront utilisées pour assurer la confidentialité et l'intégrité des communications : la NwkSKey (Network Session Key) et l'AppSKey (Application Session Key). Ces clés ne sont pas préenregistrées en usine dans l'équipement et sont renouvelées à chaque nouvelle procédure d'activation, ce qui renforce considérablement la sécurité du dispositif.

Dans le détail, cette procédure fonctionne de la manière suivante :

Emission d'un message Join-Request

Pour rejoindre le réseau, le terminal envoie un message Join-Request, dont la charge utile (payload) contient notamment :

- Le DevEUI (identifiant unique du terminal)
- L'AppEUI (identifiant de l'application ou du réseau)
- Le DevNonce, un nombre unique utilisé pour éviter les attaques par rejeu.

> Le DevNonce ne doit jamais être réutilisé, même après un redémarrage ou un changement de batterie pour ne pas compromettre la sécurité du système. Il doit donc être stocké en mémoire non volatile, généralement sous forme de compteur incrémental à chaque tentative d'activation.

Le message Join-Request est envoyé en clair sur l'un des trois canaux obligatoires (0 à 2) avec n'importe quel débit LoRa autorisé. Le code MIC (Message Integrity Code) est calculé à l'aide de la clé AppKey, préconfigurée en usine.

Si aucune réponse n'est reçue, le terminal réémet le message en sautant aléatoirement sur un autre canal autorisé, jusqu'à obtenir une réponse valide.

Réception d'un message Join-Accept

En réponse, le serveur envoie un Join-Accept, dont la payload est :

Size (octets)	3	3	4	1	1	(16) optional
Join-Accept payload	JoinNonce	NetID	DevAddr	DLSettings	RXDelay	CFList

Table 54: Join-Accept payload format

Cette payload contient essentiellement :

- JoinNonce : Un nonce généré par le LNS
- DevAddr : l'adresse du terminal allouée par le LNS
- NetID, DLSettings, RXDelay : des paramètres réseau.
- CFList : S'il est présent, permet de spécifier jusqu'à 5 canaux supplémentaires (canaux 3 à 7) que le terminal pourra utiliser. Par la suite, le LNS peut aussi ajuster dynamiquement les canaux et débits via des messages LinkADRReq (ADR = Adaptive Data Rate).

À partir des informations contenues dans le Join-Accept, notamment le JoinNonce, le terminal et le LNS dérivent de manière synchronisée les deux clés de session :

©Tidiwi 2025

- NwkSKey : utilisée pour l'intégrité des messages échangés avec le réseau.
- AppSKey : utilisée pour le chiffrement/déchiffrement des messages applicatifs.

La clé AppSKey est transmise par le *LNS* (ou par le *Join Server*) au *Serveur Applicatif*, afin que ce dernier puisse interpréter les messages en provenance du terminal.

Activation par personnalisation (APB)

Ce mode n'est pas recommandé car la sécurité est minimaliste. L'adresse du terminal ainsi que les clés **AppSKey** et **nwkSKey** sont configurées dans le terminal avant qu'il ne puisse se connecter au réseau. Il n'y a donc pas de procédure de *join*. Ces clés doivent être uniques pour chaque terminal, pour que la compromission d'un terminal ne compromette pas tout le réseau. Ces clés ne pourront jamais être changées, ce qui va à l'encontre de la nouvelle directive RED en Europe qui impose que les actifs de sécurité puissent être mis à jour si le terminal gère des données à caractère personnel.

2.3.4 La gestion de la confidentialité

LoRaWAN utilise des algorithmes de chiffrement symétrique pour assurer la confidentialité et l'intégrité des messages échangés. Les mêmes clés de chiffrement doivent donc être connues par les deux entités qui communiquent, à savoir un terminal et un LNS.

Si une trame LoRaWAN contient une payload, celle-ci est encryptée avec l'algorithme AES-CTR, et une clé AES de 128 bits.

- Si la payload contient uniquement des commandes LoRaWAN, elle est encryptée avec la clé **NwkSKey**.
- Sinon, elle contient des données applicatives et elle est encryptée avec la clé **AppSKey**. Cela permet d'assurer que le gestionnaire du réseau ne pourra pas déchiffrer les messages applicatifs.

2.3.5 La gestion de l'intégrité

Toutes les trames LoRaWAN contiennent un code *MIC* (Message Integrity Code). Ce code *MIC* est calculé par l'algorithme AES-CMAC, en utilisant la clé **NwkSKey**, sauf les messages utilisés lors de la procédure de join dans le cas d'activation *OTAA* dont le code *MIC* est calculé avec la clé **AppKey** puisque la clé **NwkSKey** n'a pas encore été négociée entre le terminal et le réseau.

2.3.6 La gestion des répétions

Pour améliorer la qualité du réseau, le *LNS* peut indiquer aux terminaux qu'ils doivent répéter chaque message émis un certain nombre de fois[16]. Dans ce cas, les terminaux répèteront les messages plusieurs fois (sauf s'ils reçoivent un message du *Serveur LoRaWAN*), en changeant de canal à chaque retransmission. Ce mécanisme permettra de pallier les interférences radio

[16] Paramètre NbTrans du message LinkADRReq

©Tidiwi 2025

générées par un équipement qui se trouverait à proximité de la *gateway* Lora et qui génèrerait un signal Radio dans un des canaux Lora utilisés par le terminal, ou les éventuelles collisions avec un autre message provenant d'un autre capteur.

2.3.7 La protection contre les attaques de type « replay »

Pour se protéger contre les attaques de Replay, la spécification LoRaWAN définit pour chaque terminal deux compteurs de 32 bits de trames pour *l'uplink*[17] et le *downlink*[17] :

- A chaque fois qu'il émet un nouveau message (donc pas en cas de répétition d'un même message), un terminal incrément son compteur *uplink*, et intègre ce compteur[18] dans son message. Le LNS mémorise ce compteur, et ignorera les trames qui auraient le même compteur, et qui ne seraient pas des répétitions.
    - Donc, il sera impossible pour un utilisateur malveillant de rejouer un message qu'il aurait capturé car ce message serait considéré comme une répétition d'un message déjà reçu, et il serait donc ignoré par le réseau.

- A chaque fois qu'il émet un nouveau message vers un terminal, le serveur incrémente son compteur *downlink*. Un terminal doit donc ignorer une trame qui contiendrait un compteur *downlink* déjà reçu.

- Si le terminal supporte « l'activation over the air » (OTAA) les compteurs *uplink* et *downlink* sont remis à zéro à la fin de chaque séquence d'activation. Sinon, le terminal supporte « l'activation par personnalisation » (ABP) et les compteurs ne doivent jamais être remis à zéro, même en cas de redémarrage, même en cas de remplacement de la batterie.

2.3.8 Le protocole entre les gateways et les serveurs LoRaWAN

Il existe plusieurs protocoles pour interfacer une gateway avec un LNS.

- Le protocole « Semtech UDP packet forwarder »
- Lora Basics™ Station

Semtech UDP packet forwarder

Historiquement, Semtech avait spécifié le protocole « Semtech UDP packet forwarder »[19] entre les gateways LoRaWAN et les LNS. Mais ce protocole est basé sur UDP, et ne fournit aucun mécanisme de confidentialité ni d'authentification. Il est donc possible d'espionner les communications entre un LNS et une Gateway, ou d'émettre des messages vers un LNS en se faisant passer pour une gateway officielle, et vice versa.

Ce protocole n'est évidemment pas recommandé pour un déploiement, à moins que les gateways et le LNS ne fassent partie d'un même réseau privé, qui serait lui-même protégé contre les attaques extérieures.

[17] Compteur Uplink = FCntUp et Compteur Downlink = FCntDown
[18] En fait, seulement les 16 bits de poids faible de ce compteur
[19] semtech-udp/PROTOCOL.TXT at master · helium/semtech-udp · GitHub

©Tidiwi 2025

Lora Basics™ Station

Par la suite, Semtech a développé une implémentation open source d'un « packet forwarder » qui propose une sécurisation des communications : Lora Basics Station[20]. Ce protocole permet de définir plusieurs niveaux de sécurité entre une gateway et un LNS. En fonction de la configuration du système, la communication peut être en clair et sans authentification mutuelle, ou bien imposer une authentification du LNS par la gateway, ou encore imposer une authentification mutuelle entre la gateway et le LNS. Dans ces derniers cas, les protocoles HTTPS[21] ou WSS[22] sont utilisés.

Par ailleurs, certaines gateways LoRaWAN intègrent un LNS. Elles permettent donc d'implémenter un réseau LoRaWAN sans aucune interaction avec le monde extérieur. Dans ce cas, les messages applicatifs sont transmis au *Serveur Applicatif* en utilisant des protocoles standards comme MQTTS ou HTTPS qui sortent complètement du cadre des spécifications ou des recommandations publiées par Semtech ou par la LoRaWAN alliance.

2.3.9 Les mises à jour du Firmware des terminaux Over The Air (*FUOTA*)

La Lora Alliance a publié un ensemble de spécifications techniques[23] pour décrire la mise à jour des firmwares des terminaux LoRaWAN. L'implémentation de FUOTA est optionnelle.

Le fonctionnement repose sur les étapes suivantes :

- À la demande du serveur applicatif, un serveur dédié, appelé MDS (Multicast Distribution Server), constitue un groupe multicast rassemblant tous les terminaux dont le firmware doit être mis à jour.

- Le serveur notifie individuellement chaque terminal ciblé de son inclusion dans le groupe. Il lui communique également l'adresse multicast associée, ainsi que les clés de chiffrement spécifiques qui seront utilisées pour sécuriser la transmission des blocs de données contenant le nouveau firmware.

- Le MDS demande ensuite à ces terminaux de mettre en place des fenêtres de réception périodiques afin de recevoir les blocs de firmware. Concrètement, cela revient à activer temporairement un comportement de type classe B, le temps de la session de téléchargement.

- Les blocs du firmware peuvent être réémis plusieurs fois pour compenser d'éventuelles pertes ou erreurs de transmission.

- Une fois le téléchargement terminé, le MDS vérifie terminal par terminal si l'intégralité du nouveau firmware a bien été reçue. En cas de succès, il peut alors demander au terminal de valider et activer le nouveau firmware, soit immédiatement, soit à une date ultérieure.

[20] https://doc.sm.tc/station/index.html
[21] https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure
[22] https://fr.wikipedia.org/wiki/WS-Security
[23] https://resources.lora-alliance.org/technical-specifications

©Tidiwi 2025

> *Cette fonctionnalité est particulièrement stratégique pour assurer la mise en conformité des équipements avec la directive RED, notamment en facilitant les mises à jour à grande échelle.*

©Tidiwi 2025

3 Les menaces contre les réseaux LoRaWAN

Avant de s'intéresser en détail aux mesures à mettre en place pour sécuriser un réseau LoRaWAN, il est important d'identifier les menaces et les risques associés à ce type de déploiement. Il appartiendra ensuite à l'utilisateur final de juger si ces menaces sont pertinentes par rapport à l'utilisation qu'il veut faire du réseau, et de mettre en place les mesures pour se protéger contre ces menaces. La plupart de ces menaces sont communes avec tous les protocoles de communication sans fil. En résumé, les risques sont les suivants :

- Rendre le réseau complètement inopérant, et donc réduire le retour sur investissement à néant.
- Le vol de données personnelles, si le réseau est utilisé pour gérer des données privées (ouvertures de portes, fenêtres … dans un déploiement de type smart home par exemple).
- Le vol de données monétaires, si le réseau est utilisé pour gérer des données ayant une valeur monétaire (compteur d'eau, parcmètres …. Dans un déploiement Smart City par exemple).

Les menaces principales sont décrites dans les sections suivantes :

- Clonage de terminaux
- Usurpation d'identité
- Attaques de replay
- Vol de données
- Altération des données transmises
- Denial Of Service

3.1 Le Clonage de terminaux

Cette menace concerne plutôt le fabricant du terminal que l'utilisateur final. Il appartiendra tout de même à l'utilisateur final de s'assurer qu'il a acquis des terminaux originaux et pas des copies.

Les risques associés sont les suivants pour l'utilisateur :

- Un terminal non-original peut inclure un logiciel malveillant qui transmettra de fausses données, ou perturbera le réseau LoRaWAN. Il peut donc rendre le système déployé complètement inutilisable.
- Perte de la garantie du constructeur, et potentiel problème sur la qualité des terminaux.

3.2 L'Usurpation d'identité

Un acteur malveillant émet des données erronées vers une gateway en se faisant passer pour un terminal faisant partie du réseau, ou émet des données erronées vers le LNS en se faisant passer pour une gateway officielle, ou vice-versa.

Les risques associés sont les suivants pour l'utilisateur :

- Si un seul terminal est compromis, ce terminal deviendra complètement inutilisable dans le système puisque le LNS recevra de sa part des données incohérentes, composées d'un mélange de données réelles et de fausses données. Si le terminal réel est lui-même attaqué

©Tidiwi 2025

(détruit, ou simplement empêché de transmettre des données) le système déployé sera inutilisable.

3.3 Les Attaques de Replay

Un acteur malveillant écoute une communication entre un capteur et une gateway, ou entre une gateway et un LNS, puis rejoue ce message.

- Par exemple, un acteur pourrait écouter une commande réseau demandant d'ouvrir une vanne, ou une serrure ... puis la rejouer beaucoup plus tard pour forcer une nouvelle exécution de cette action.
- Si le rôle du capteur est de générer des alarmes, un acteur malveillant pourrait aussi déclencher des alarmes intempestives en rejouant un message d'alarme véritable et perturber le fonctionnement d'un bâtiment tertiaire par exemple.

3.4 Vol de données (Ecoute passive)

Un acteur malveillant espionne les échanges entre un capteur et une gateway, ou entre une gateway et un LNS, et en extrait des informations.

- Si les terminaux sont utilisés dans un système de contrôle d'accès par exemple, l'écoute passive permettra à un acteur malveillant d'identifier les horaires d'entrée et de sortie (dans une application de type smart home par exemple), et éventuellement les identités des personnes qui entrent et sortent en fonction du système déployé (dans une application de type Smart Building par exemple).

3.5 Alteration des données transmises

Un acteur malveillant modifie les données émises par un terminal ou une gateway avant qu'elles n'arrivent au LNS, ou vice-versa.

- Par exemple, il pourrait modifier la valeur de l'index d'un compteur d'eau pour réduire sa facture.
- Dans un système de gestion d'arrosage en ville ou en agriculture, la transmission de fausses données d'humidité du sol pourrait compromettre les cultures et rendre le système inutilisable.
- Dans un système de gestion de la circulation ou de places de parking, le système deviendrait encore une fois complètement inutilisable.

3.6 Attaques de Denial Of Service (DOS)

Un acteur malveillant empêche les données provenant des capteurs ou des gateways d'arriver au LNS, ou vice-versa.

- Ce type d'attaque rend le réseau complètement inopérant.

©Tidiwi 2025

4 La sécurisation d'un réseau LoRaWAN

Remarque préliminaire sur la sécurité de l'algorithme AES

L'algorithme AES permet de chiffrer un message à partir d'une clé symétrique. Si on chiffre deux fois le même message avec la même clé, on obtient le même résultat. Cet aspect est une faiblesse de ce type d'algorithme, qui peut être exploitée par un acteur malveillant pour casser la sécurité du protocole.

Pour cette raison, les constructions basées sur l'AES telles que l'AES-CTR ajoutent un « nonce » au début du message à chiffrer : ce « nonce » est un nombre qui ne doit jamais être réutilisé avec la même clé de chiffrement. Dès lors, si on chiffre deux fois le même message avec la même clé, on obtiendra deux résultats différents, pourvu qu'on n'ait pas réutilisé un « nonce » qui aurait déjà été utilisé.

La plupart des messages LoRaWAN (en dehors de la procédure de join) sont chiffrés avec la clé NetwSKey et l'algorithme AES-CTR. Dans ce cas, le nonce est un simple compteur, incrémenté à chaque nouvelle opération de chiffrement. Il est donc important que ce nonce ne soit pas remis à zéro au redémarrage d'un terminal, car un acteur malveillant qui un accès physique au terminal pourrait provoquer des redémarrages, et le terminal serait amené à chiffrer plusieurs fois un message avec le même nonce, ce qui serait une faille majeure de sécurité.

Le message Join Request n'est pas chiffré, mais il inclut un code MIC calculé avec la clé AppKey. Pour se protéger contre les attaques de replay, la payload du message Join Request contient elle-même un nonce qui ne doit jamais se répéter, même en cas de redémarrage du terminal ou de remplacement de sa batterie.

4.1 La version du protocole LoRaWAN

Le protocole LoRaWAN a évolué au cours du temps, notamment pour en améliorer la sécurité. Les versions déployées actuellement sont les suivantes :

- 1.0.2
- 1.0.3
- 1.0.4

Il existe aussi une version 1.1 mais elle n'est pas largement déployée à ma connaissance. Cependant, les évolutions majeures en termes de sécurité introduites dans la version 1.1 ont été reportées dans la version 1.0.4.

Les versions 1.0.2 et 1.0.3 ont un certain nombre de failles en termes de sécurité[24] qui sont corrigées dans la version 1.1 (et donc dans la version 1.0.4). Certaines failles sont assez « théoriques » car elles supposent que l'attaquant connaît exactement les instants où les messages sont transmis, mais d'autres failles sont plus simples à exploiter. Notamment, les failles les plus importantes corrigées dans les versions 1.1 et 1.04 sont les suivantes[25] :

[24] Les compteurs des trames uplink sont remis à zéro à chaque démarrage en cas d'activation ABP, ce qui entraine une faille de sécurité. Les nonces utilisés dans les messages join request sont des valeurs aléatoires, et il y a donc un risque qu'un même nonce soit réutilisé après un certain nombre de messages. Dans la version 1.0.4, c'est un compteur incrémenté à chaque nouvelle procédure de join qui tient lieu de nonce. Ce compteur ne doit pas être remis à zéro en cas de reset.

[25] LoRaWAN : Partie 2, attaques sur le protocole - Kereval

©Tidiwi 2025

- Les compteurs de trames (nonce) des terminaux activés en ABP sont remis à zéro en cas de redémarrage d'un appareil.
    - **Compromission de la sécurité des appareils activés en ABP**
      Un acteur malveillant qui aurait accès à un terminal pourrait provoquer des redémarrages pour que l'appareil génère plusieurs trames avec le même nonce. Cela compromettrait entièrement la sécurité offerte par AES-CTR.

    - **Denial Of Service sur les appareils activés en ABP**
      Un acteur malveillant qui écouterait les communications d'un terminal activé en ABP pourrait forcer un terminal à redémarrer, puis rejouer des messages enregistrés. Cela aurait pour effet que les messages émis par le vrai terminal après son démarrage seraient ignorés par le LNS car les compteurs de ces messages seraient inférieurs aux compteurs des trames rejouées : cela revient à exécuter une attaque Denial Of Service sur ce terminal.

    - **Denial Of Service sur les appareils activés en OTAA**
      La spécification 1.0.3 indique que les nonces utilisés dans les messages Join Request et Join Accept doivent être choisis au hasard par les terminaux et par les LNS. Pour se protéger contre les attaques de replay, les serveurs et les terminaux doivent mémoriser « un certain nombre » de nonce déjà reçus. Les serveurs ont assez de mémoire pour mémoriser 65535 nonces pour terminal, mais les terminaux n'ont sûrement pas assez de mémoire pour faire la même chose. Un attaquant peut donc répondre à un message join request émis par un terminal avec un join accept qu'il aurait mémorisé, et qui date d'un certain temps. A partir de ce moment, le terminal serait connecté à un « LNS fantôme », et serait incapable de communiquer avec le LNS réel. Comme indiqué, ce problème est corrigé dans la version 1.0.4 où le nonce est un simple compteur, ce qui simplifie la mémorisation du dernier nonce utilisé.

Ces failles supposent que l'attaquant peut écouter les échanges Radio, ce qui est à la portée de n'importe qui en possession d'un récepteur Lora. Elles sont donc facilement exploitables. Elles supposent parfois que l'attaquant a un accès physique aux terminaux, et qu'il est capable de déclencher un redémarrage du terminal. Ce point est aussi très plausible dans un déploiement de type Smart City ou Smart Building puisque les capteurs ne sont pas protégés physiquement. Par ailleurs, il suffit de retirer les batteries du capteur pour provoquer un redémarrage.

Il est donc recommandé d'utiliser des terminaux qui implémentent une version du protocole supérieure ou égale à 1.0.4

4.2 La certification des terminaux

La LoRa Alliance a défini un programme de certification[26] permettant de s'assurer qu'un terminal respecte la spécification LoRaWAN. L'utilisation de terminaux disposant de cette certification permet notamment de garantir qu'ils gèrent correctement les aspects relatifs à la sécurité définis dans la specification LoRaWAN. Notamment :

- La négociation des clés AppSKey et NtwSKey dans la procédure OTAA.

[26] https://lora-alliance.org/lorawan-certification/

©Tidiwi 2025

- La gestion du chiffrement et de l'intégrité des trames LoRaWAN.
- La gestion des nonces utilisés dans la procédure de join, et la persistance de ces nonces en cas de redémarrage du terminal.
- La gestion des compteurs uplink et downlink utilisés comme IV pour le chiffrement des trames LoRaWAN.

4.3 La sécurisation des clés AppKey

L'actif de sécurité qui est absolument critique dans un réseau LoRaWAN est la clé *AppKey* stockée dans les terminaux et dans le *Join Server*, lui-même souvent intégré dans le *LNS*. En effet cette clé est utilisée pour exécuter la *procédure de join*, qui sera elle-même utilisée pour définir les clés de session. Par ailleurs, cette clé ne peut pas être modifiée une fois que le terminal a été déployé.

> **Un attaquant qui dispose de cette clé peut exécuter toutes les attaques mentionnées plus haut. Il peut donc rendre un terminal, ou même le réseau complètement inutilisable sans qu'il soit possible de corriger le problème sans accéder physiquement au terminal**.

Dans la mesure où cette clé est connue du terminal et du Join Server sa protection n'est pas un problème simple :

- Un acteur malveillant pourra facilement subtiliser un terminal déployé, l'emmener dans son laboratoire pour l'analyser et essayer d'en extraire la clé AppKey.
- Un utilisateur (éventuellement un stagiaire …) qui a accès au Join Server (ou au LNS si le Join Server est intégré dans le LNS) peut accéder à l'ensemble des clés AppKey de tous les terminaux déployés.

Les mesures recommandées pour sécuriser ces clés sont les suivantes :

- **Utiliser une clé AppKey unique pour chaque terminal**
  Cette clé doit absolument être unique pour chaque terminal pour éviter que la compromission d'un terminal ne compromette tout le réseau. Cette clé ne doit pas pouvoir être déduite facilement du numéro de série du produit par exemple, mais plutôt obtenue grâce à une « fonction de dérivation » qui met en œuvre des informations connues et une clé secrète[27].

- **Contrôler les accès aux AppKeys dans les serveurs**
  Les clés AppKey sont aussi stockées dans le Join Server (ou dans le LNS). Elles sont donc potentiellement accessibles à un opérateur qui a accès au LNS ou au Join Server. Il faut donc aussi s'assurer que les accès à ces clés ne sont disponibles qu'à un nombre très restreint d'utilisateur, et pas à tous les opérateurs de ces serveurs.

  Dans le cas où le client final sous-traite l'infrastructure à un opérateur télécom ou à un intégrateur il est recommandé d'utiliser un Join Server distinct du LNS. Ce *Join Server* doit être géré par le client final et pas par l'opérateur. Dans ce cas, les opérateurs du LNS n'auront pas accès aux clés des terminaux, et il « suffira » de sécuriser les accès au Join Server. Cette option n'est possible qu'à partir de la spécification LoRaWAN 1.0.4.

[27] https://cyber.gouv.fr/sites/default/files/2021/03/anssi-guide-selection_crypto-1.0.pdf

©Tidiwi 2025

- **Vérifier la sécurisation des clés dans les terminaux**
  Cette vérification n'est pas forcément réalisable par l'utilisateur final qui n'a le plus souvent pas accès à l'intérieur des terminaux. Il est cependant possible d'auditer les fournisseurs de terminaux pour connaître leur stratégie de protection des actifs de sécurité. Notamment, les stratégies suivantes sont recommandées :

  - Les interfaces de DEBUG (notamment l'interface JTAG) doivent être désactivées sur les terminaux. En effet, un acteur malveillant qui pourrait se connecter à ces interfaces pourrait accéder aux clés de sécurité stockées dans le terminal, notamment à la clé *AppKey* qui permet au terminal de s'enregistrer sur le réseau. Il pourrait aussi remplacer le firmware du terminal par un firmware destiné à perturber le réseau, en envoyant de fausses données, ou en envoyant un trop grand nombre de données, ou en exécutant de nombreuses procédures de join ...

  - Les clés de sécurité, et surtout la clé AppKey, ne doivent pas pouvoir être lues dans le terminal. Idéalement, cette clé devrait être stockée dans un Secure Element, mais il existe peu de capteurs à bas coût qui sont équipés de ce type de fonctionnalité. Ces clés peuvent aussi être stockées dans la flash interne du terminal puisque l'interface JTAG est désactivée, ou bien stockées dans une mémoire externe. Dans ce dernier cas, elles doivent être chiffrées avec une clé qui ne peut pas être lue depuis l'extérieur ni déduite d'information connues publiquement (comme le numéro de série de l'appareil par exemple).

  - Si les données du capteur sont transmises sur internet - ce qui est presque toujours le cas, sauf dans le cas d'un déploiement entièrement privé où les capteurs, les gateways et le LNS sont colocalisés - et s'il est possible d'identifier un utilisateur, personne physique, à partir de ces données – ce qui est moins souvent le cas sauf dans des application de type habitat intelligent - le terminal doit implémenter la fonctionnalité *FUOTA* (Firmware Update Over The Air), ou tout au moins il doit fournir un moyen de mettre à jour son firmware. Cette contrainte est introduite par la directive RED.

4.4 La détection des fraudes

- **Détecter les comportements anormaux au niveau du LNS**
  Aucun mécanisme de protection n'étant infaillible, il est recommandé de pouvoir détecter un comportement anormal d'un ou plusieurs terminaux de manière à pouvoir les isoler ou les remplacer en cas de présomption de fraude. Ce mécanisme devrait être implémenté dans le LNS qui a une vision globale du réseau.

- **Détection d'intrusion dans les terminaux**
  Idéalement, il faudrait que le capteur implémente une fonctionnalité de détection d'intrusion, et qu'il soit capable de générer une alerte vers le serveur applicatif en cas de suspicion d'intrusion.

©Tidiwi 2025

4.5 L'activation des terminaux

S'il n'a pas d'accès physique aux terminaux ni au LNS, un utilisateur malveillant peut encore espionner l'interface radio, puisque les trames émises par les capteurs ou les gateways peuvent être écoutées par n'importe qui, même à grande distance, pourvu que cette personne dispose d'un récepteur Lora.

Les algorithmes cryptographiques comme l'AES-CTR présentent une vulnérabilité si les « nonce » utilisés pour chiffrer les messages sont utilisés plusieurs fois. La spécification LoRaWAN recommande que ces nonces ne soient jamais réutilisés, même en cas de redémarrage de l'appareil.

Préférer l'activation OTAA à ABP

La spécification LoRaWAN prévoit aussi de réduire la période pendant laquelle le système est « attaquable » en utilisant des clés éphémères, redéfinies à chaque nouvelle procédure de join, plutôt que des clés stockées une fois pour toute dans les terminaux, sans pouvoir être modifiées. C'est l'un des intérêts de la procédure d'activation OTAA : la clé AppKey n'est utilisée qu'une fois pendant la procédure de join, pour chiffrer le message join accept. Une fois que la procédure de join est terminée, les prochains messages seront chiffrés et authentifiés avec les clés **AppSKey** et **NetwSKey** qui seront négociées entre le terminal et le LNS pendant la procédure de join. Une nouvelle procédure de join pourra être déclenchée en cas de suspicion de fraude, même s'il n'est pas recommandé d'exécuter cette procédure de manière trop fréquente[28].

> **Remarque**
>
> la procédure OTAA permet aussi de changer de fournisseur de réseau LoraWAN sans avoir à intervenir physiquement sur les capteurs pour y programmer les clés appSKey netSKey du nouveau fournisseur.

4.6 Choisir la topologie du réseau

Les risques associés au déploiement d'une solution basée sur LoRaWAN dépendent en premier lieu de l'architecture déployée. On peut envisager de déployer l'une des topologies suivantes :

- Réseau LoRaWAN opéré
- Réseau LoRaWAN privé avec LNS hébergé
- Réseau LoRaWAN privé avec LNS privé

4.6.1 Réseau LoRaWAN opéré

L'utilisateur final peut déléguer la gestion de son réseau à un opérateur mobile ou à un intégrateur. On rencontre ce type de déploiement dans les villes intelligentes par exemple. Dans ce cas, une grande partie de la sécurité du système reposera sur les épaules du sous-traitant (opérateur ou intégrateur), dont il conviendra d'auditer les capacités en termes de cybersécurité :

[28] Pas plus d'une fois par mois dans le pire des cas selon https://lora-alliance.org/wp-content/uploads/2021/05/TR007_Developing_LoRaWAN_Devices-v1.0.0.pdf?utm_source=chatgpt.com

©Tidiwi 2025

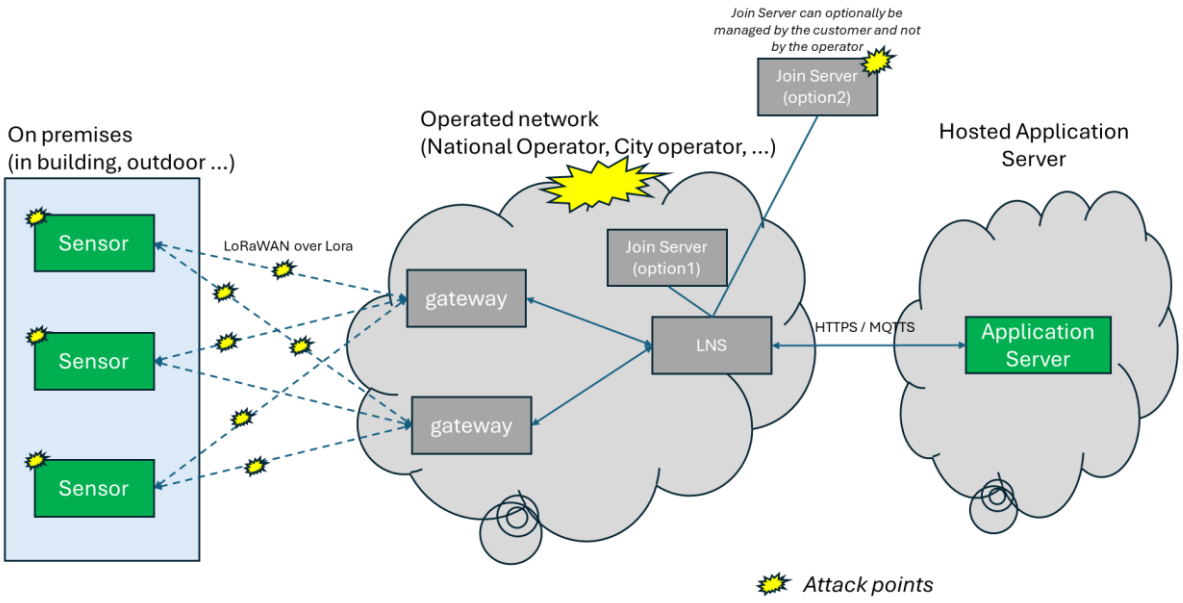


Figure 4 - LoRaWAN Operated network

Dans ce cas, en dehors des capteurs eux-mêmes et de l'interface radio, la gestion de la sécurité repose essentiellement sur l'opérateur ou le sous-traitant qui gère le réseau.

- **La protection contre le brouillage (jamming)**

  Dans ce type de déploiement, le brouillage Radio (au moins en uplink) semble très difficile car il faudrait émettre un signal radio assez fort pour perturber les gateways de l'opérateur. Par ailleurs, les signaux émis par les capteurs sont généralement reçus par plusieurs gateways, ce qui rend le problème encore plus compliqué pour un acteur malveillant. Enfin, dans la mesure où les gateways d'un opérateur gèrent un grand nombre de terminaux, un dysfonctionnement des gateways serait très rapidement détecté par l'opérateur. Ce problème n'est donc pas critiques dans ce type de déploiement, au moins pour les communications uplink.

  En downlink, le LNS peut demander que les messages soient acquittés par le terminal. Ce mécanisme permettra donc de détecter un dysfonctionnement du terminal, lié au brouillage ou à une autre cause.

- **La protection des clés AppKey**

  - **Gérer les accès dans le LNS**
    Si le LNS intègre le Join Server, il connaît les clés appKey de tous les terminaux, ce qui représente un risque majeur pour la sécurité du réseau. En effet, en fonction de la manière dont les droits d'accès au LNS seront configurés par le sous-traitant, des personnes non-autorisées (éventuellement des stagiaires ...) pourraient avoir accès à ces clés.

  - **Utiliser un Join Server géré par le client final**
    Comme il semble impossible pour un client d'auditer la configuration du LNS de l'opérateur, la solution consiste à utiliser un Join Server séparé, géré par le client lui-

©Tidiwi 2025

même et pas par l'opérateur. Dans cas, ce Join Server sera le seul à connaître les clés AppKey des terminaux, et le LNS de l'opérateur interrogera ce Join Server à chaque demande de join d'un terminal.
La majorité des opérateurs de LNS proposent d'ailleurs cette option. Il en va de même pour le LNS open source Chirpstack. Evidemment, il faudra correctement configurer les droits d'accès au Join Server pour s'assurer que seul un nombre très restreint d'administrateurs auront accès aux clés AppKey.

- **Coupler le Join Server avec un Secure Element**
  Pour éviter les risques de mauvaise configuration des droits d'accès aux clés AppKey du Join Server, on peut coupler le Join Server à un Secure Element[29], qui gèrera les clés AppKey des terminaux. Dans ce cas, même les opérateurs du Join Server n'auront pas accès aux clés AppKeys.

Il faut noter que, même dans le cas d'un Join Server séparé, la compromission du LNS géré par le sous-traitant entraînera des conséquences catastrophiques sur la sécurité du réseau. Le LNS n'a pas accès aux clés AppKey, mais il connaîtra les clés de session utilisées pour sécuriser les échanges. La sécurisation du LNS est du ressort de l'opérateur du réseau.

Note : On ne s'intéresse pas à l'interface entre le LNS et le serveur d'application car elle est complètement indépendante de LoRaWAN. La sécurisation de cette interface passe par l'utilisation de protocoles sécurisés communément utilisés sur l'internet (SSL/TLS, HTTPS, MQTTS …).

4.6.2 Réseau LoRaWAN privé avec LNS hébergé

Dans des déploiements plus petits (en termes de surface à couvrir), typiquement dans les applications de bâtiments intelligents, le gestionnaire du bâtiment pourra déployer et opérer ses propres gateways. En général, il utilisera un LNS hébergé (par exemple « *TTN* », **T**he **T**hings **N**etwork[30] ou ThinkPark[31], …). Ici encore, même dans le cas d'utilisation de protocoles sécurisés entre les gateways et le LNS, une partie de la sécurité de la solution reposera sur le fournisseur de la solution LNS hébergé.

[29] Composant Physique inviolable qui stocke les clés et qui exécute les opérations de chiffrement
[30] https://www.thethingsnetwork.org/
[31] https://www.thingpark.com/thingpark-enterprise/

©Tidiwi 2025

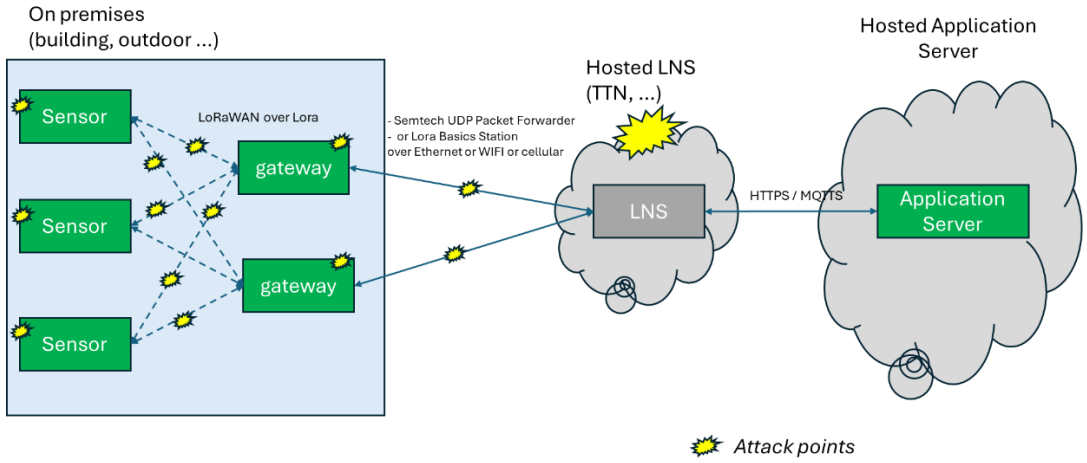


Figure 5 - Private Network with hosted LNS

- **Protection contre le brouillage**
  En règle générale, il n'est pas possible de se protéger contre le brouillage. Les seules mesures envisageables consistent à détecter le brouillage, pour déclencher une action d'investigation.

  - **Brouillage radio (jamming radio)**
    Dans ce type de topologie, il est assez facile pour un attaquant de localiser une gateway à l'intérieur du bâtiment et de brouiller le signal à proximité de cette gateway. Même s'il n'existe pas de protection contre le brouillage, il est plus prudent d'utiliser plusieurs gateways localisés à des endroits différents pour compliquer la tâche de l'attaquant : les messages des capteurs seront reçus par plusieurs gateways dans ce cas.
    > **Cependant, la modulation Lora est particulièrement robuste contre ce type de brouillage[32].** Un attaquant devra donc installer des brouilleurs à proximité de chaque gateway, ou bien utiliser un brouilleur puissant qui sera plus facilement détectable.

  - **Brouillage protocolaire (jamming au niveau MAC)**
    Puisque la modulation Lora résiste bien au brouillage radio, les attaquants préfèreront un « brouillage protocolaire » qui consiste à générer un grand nombre de packets LoraWAN à partir d'un terminal LoraWAN, pour que ces paquets entrent en collision avec des vrais paquets au niveau des gateways.
    Les gateways ne sont pas capables de détecter ce type de brouillage. Il faudra donc utiliser des algorithmes « intelligents » au niveau du LNS pour détecter un comportement anormal au niveau d'un site particulier, et déclencher des investigations en cas de suspicion de jamming. Les comportements anormaux sont, par exemple, la réception d'un grand nombre de paquets avec un MIC incorrect, l'absence de données provenant de capteurs qui sont censés émettre périodiquement ...

- **Protection des clés AppKey**

[32] https://www4.comp.polyu.edu.hk/~csyqzheng/papers/LoRaJamming-INFOCOM21.pdf

©Tidiwi 2025

Dans ce scénario, les considérations relatives au LNS et Join Server mentionnées dans la section précédente s'appliquent aussi.

- **Sécurisation de l'interface entre les gateways et le LNS**
  Ce scénario présente une faiblesse supplémentaire au niveau de l'interface entre les gateways et le LNS. Certaines gateways du commerce utilisent le protocole UDP Packet Forwarder pour communiquer avec le LNS. Ce protocole basé sur UDP n'offre aucun mécanisme de sécurité, de garantie de transmission, ou d'authentification mutuelle entre les gateways et les serveurs. Il ne faut donc pas utiliser ce protocole dans ce type de déploiement, mais plutôt utiliser un protocole basé sur SSL/TLS comme Lora Basics™ Station[33].

4.6.3 Réseau LoRaWAN privé avec LNS privé

Enfin, dans des déploiements encore plus petits, ou demandant un niveau de sécurité plus élevé, l'utilisateur final pourra héberger un LNS dans ses locaux. Les données ne transiteront donc pas sur internet, et l'utilisateur final ne dépendra pas d'un tiers pour assurer la sécurisation de ses données. En revanche, Il devra dans ce cas disposer des compétences nécessaires pour gérer entièrement ses gateways, et déployer et gérer un LNS sur un serveur privé. Les fournisseurs de LNS hébergés fournissent souvent la possibilité d'héberger leur LNS sur un serveur qui se trouve dans les locaux du client. L'utilisateur final pourra aussi déployer un LNS open source sur un serveur local, mais cela lui demandera de disposer des compétences spécifiques pour assurer la sécurisation de son LNS. Par exemple ChirpStark[34] est un LNS Open Source qui peut s'exécuter sur un petit serveur linux (ou même une plateforme Raspberry Pi).

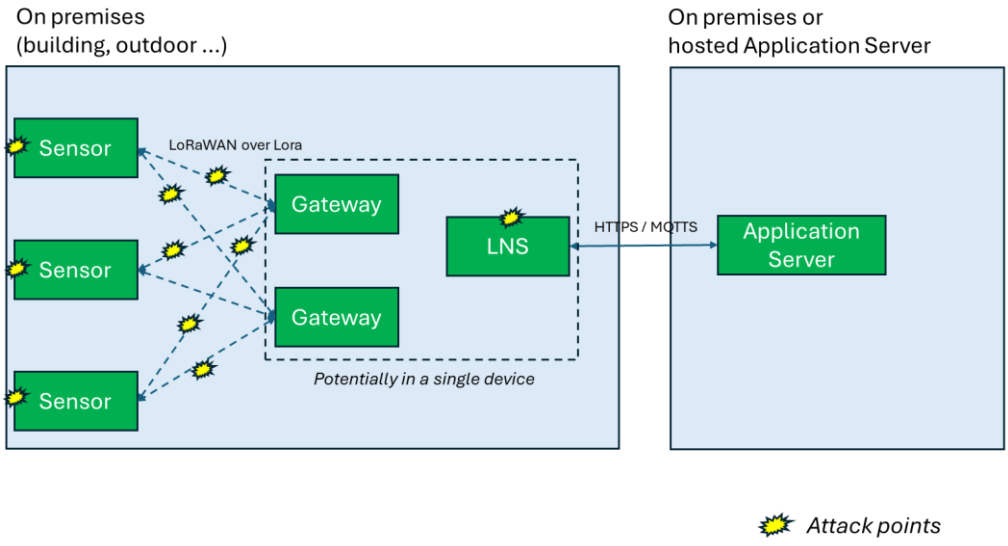


Figure 6 - Private Nework with private LNS

[33] https://doc.sm.tc/station/
[34] https://www.chirpstack.io/

Une solution encore plus simple consiste à utiliser une Gateway qui intègre elle-même un LNS, comme certaines gateways de Multitech[35], ou des gateways basées sur des Raspberry Pi[36].

Dans ce type de déploiement :

- Protection contre le brouillage
  Les considérations relatives au brouillage s'appliquent de la même manière que dans le cas précédent. Il faudra donc implémenter un algorithme « intelligent » au niveau du LNS pour détecter les situations anormales.

- Protection des clés AppKey
  Il n'est pas nécessaire de séparer le Join Server du LNS. Cependant, il faudra correctement gérer les droits d'accès au LNS, et éventuellement associer un Secure Element au LNS pour éviter que les personnes ayant accès au LNS ne puissent accéder aux clés AppKeys des terminaux.

- Le protocole utilisé entre les gateway et le LNS n'a pas d'importance, et même le protocole Semtec UDP Packet forwarder est acceptable.

4.7 Implémenter la mise à jour des Firmware Over The Air

Comme discuté plus haut, il est impossible de concevoir un système avec la certitude qu'un attaquant ne trouvera pas un jour une faille de sécurité. C'est pour cette raison qu'il est très important de prévoir la mise à jour des logiciels Over The Air.

Si les terminaux contiennent des « données personnelles », cette mise à jour est par ailleurs rendue obligatoire par les évolutions cybersécurité de la directive RED en Europe.

4.8 La protection contre l'altération des données applicatives

La confidentialité des données applicatives

Les données applicatives sont chiffrées avec l'algorithme AES-CTR et la clé *AppSKey*, connue uniquement du terminal et du serveur d'application. Pour mémoire, la clé *AppSKey* a été négociée entre le terminal et le Join Server au moment de l'activation OTAA, ou configurée dans le terminal durant la procédure d'installation dans le cas de l'activation ABP.

La confidentialité des données applicatives est donc assurée.

L'intégrité des données applicatives

Le code MIC défini dans la spécification du protocole LoRaWAN permet de vérifier l'intégrité des trames échangées entre un terminal et un LNS. Mais il ne permet pas de vérifier l'intégrité des messages applicatifs. Il serait donc possible qu'un LNS corrompu modifie le contenu d'un message applicatif sans que le serveur applicatif ne s'en rende compte, même si le message applicatif est chiffré avec la clé **AppSKey**. En effet, **le chiffrement d'une trame permet d'assurer sa confidentialité, mais pas son intégrité**.

[35] https://multitech.com/all-products/cellular/cellular-gateways/conduit-300/
[36] https://www.chirpstack.io/docs/chirpstack-gateway-os/index.html

©Tidiwi 2025

Si l'intégrité des données applicatives est requise, l'application devra implémenter son propre mécanisme de vérification de l'intégrité des messages. Ce sera notamment le cas si ces données sont utilisées pour facturer un client (dans le cas d'un compteur d'eau ou de gaz par exemple).

©Tidiwi 2025

5 A propos de TIDIWI

TIDIWI est une société de conseil spécialisée dans la conception et le développement de projets IOT. Notamment, TIDIWI a une forte expérience dans les protocoles de communication sans fil et dans les systèmes à ultra faible consommation d'énergie.

TIDIWI intervient sur des missions de conseil, de développement de logiciel embarqué, de développement de cartes électroniques, le management d'équipes de développement, la gestion de projets.

Pour plus d'information, visitez le site de Tidiwi : www.tidiwi.com

Ou contacter thierry.didi[at]tidiwi[dot]com

©Tidiwi 2025

6 Reference Documents

LoraWan Technical Specification V1.0.3 : https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf

LoraWan Technical Specification V1.0.4 : https://resources.lora-alliance.org/technical-specifications/ts001-1-0-4-lorawan-l2-1-0-4-specification

RP2-1.0.3 LoRaWAN® Regional Parameters: https://resources.lora-alliance.org/technical-specifications/rp2-1-0-3-lorawan-regional-parameters

LoRaWAN Security FAQ : https://resources.lora-alliance.org/faq/lorawan-security-faq

Semtech SX1276/77/78/7 Datasheet : https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE

Jamming of LoRa PHY and Countermeasure : https://www4.comp.polyu.edu.hk/~csyqzheng/papers/LoRaJamming-INFOCOM21.pdf

©Tidiwi 2025