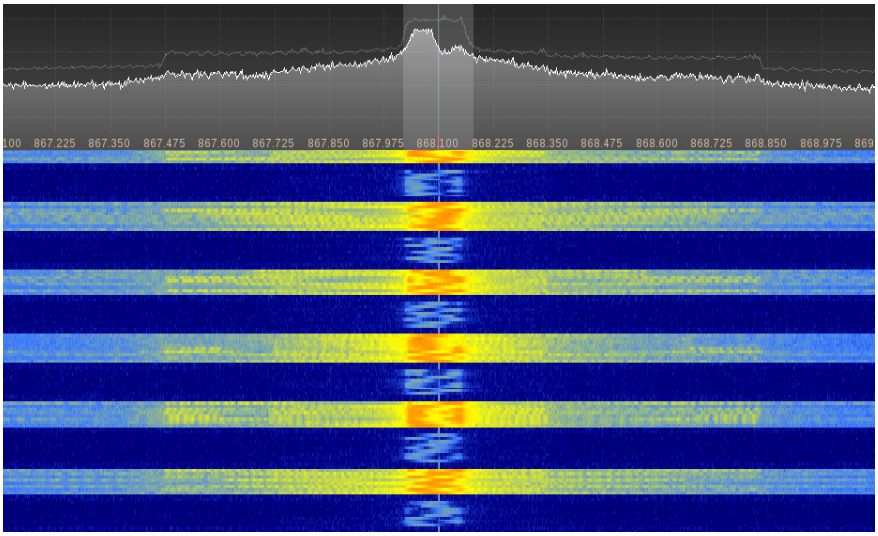


Rapport de projet de fin d'étude
Sécurité de l'Internet des Objets

2016/2017

Jérémie Denéchaud
IMA5

Encadré par

Alexandre Boé *Xavier Redon* *Thomas Vantroys*

Table des matières

Index des illustrations

Introduction

L'internet des objets (Internet of Things – IoT) désigne l'appropriation de l'internet par de nombreux objets connectés variés. Les prévisions du nombre de capteurs montrent une très forte croissance dans les années à venir pour des domaines comme la domotique, l'industrie ou encore la géolocalisation. La sécurité de ces réseaux est tout aussi majeure que l'expansion de ce parc d'objets connectés. Parmi les protocoles créés sur mesure pour s'adapter aux nouveaux besoins de cet IoT, nous retrouvons le couple LoRa et LoRaWAN.

LoRa (Long Range – réseau étendu à longue portée) est le nom donné à une technologie de modulation à étalement de spectre inventée en 2010 et brevetée en 2012 par la start-up française Cycleo. Cette entreprise grenobloise a été rachetée par le fabricant américain de semi-conducteur Semtech pour 5 millions de dollars. C'est la partie propriétaire de la technologie.

LoRaWAN décrit le protocole de transport qui utilise la modulation LoRa. Cette couche est open-source et relativement bien documentée. Cette récente technologie est au centre de l'actualité et de toutes les attentions puisque Semtech a trouvé des investissements à hauteur de 50 millions de dollars et la technologie a été bien accueillie au CES 2017. La technologie concurrente Sigfox a levé quant à elle 150 millions de fonds.

Ce rapport présente dans un premier temps un **état de l'art** de ces protocoles. Ensuite **l'aspect de la sécurité** de LoRaWAN sera approfondi.

État de l'art

Les réseaux IoT

Voici un schéma qui résume les quatre grandes étapes de la communication d'un réseau IoT.

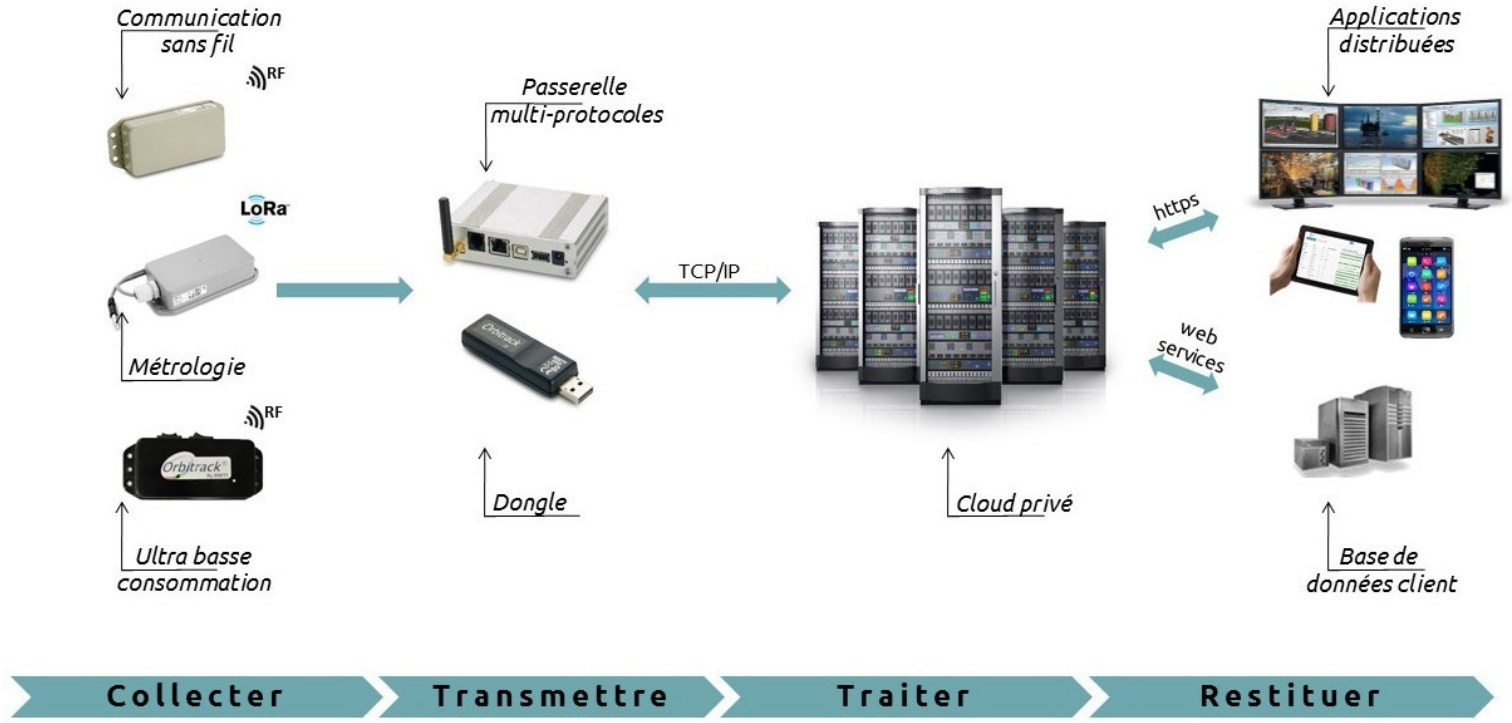


Illustration 1: Schéma d'un réseau IoT

Dans l'IoT, on considère souvent des objets (capteurs) qui ont des contraintes matérielles et logicielles qui ne leur permettent pas de se connecter directement au réseau Internet. Ils s'y connectent à travers une passerelle (gateway). En effet, d'un côté internet n'est pas dimensionné pour gérer l'adressage d'autant d'objets connectés. D'un autre côté l'ipv6 est un protocole souvent trop lourd pour être exploité directement par les capteurs. Aujourd'hui l'ipv6 est utilisé via le protocole 6LowPAN (I*Pv6 Low power Wireless Personal Area Networks*) qui est exploité au-dessus des protocoles réseaux régis par l'IEEE 802.15.4.

Les gateways traduisent aussi les protocoles propriétaires vers un protocole compréhensible sur internet et certaines peuvent jouer le rôle d'agrégateurs de réseaux. Les réseaux qui les connectent doivent aussi répondre à ces besoins de légèreté, de simplicité et de faible coût. Ainsi, les opérateurs télécoms doivent s'adapter car utiliser les réseaux cellulaires pour transmettre quelques kilo d'octets par capteur coûte très cher. De plus, le réseau 2G est en cours de démantèlement. Ceci offre une opportunité pour développer des réseaux sans fils

Machine↔Machine (M2M) à longue portée, basse énergie et très bas débit **qui coûtent beaucoup moins cher que les réseaux cellulaires**.

Le réseau LoRa présenté ensuite a principalement été conçu pour l'utilisation d'un schéma en étoile. Ceci a l'avantage d'avoir une grande simplicité de déploiement et une faible consommation énergétique de la part des équipements émetteurs. Les particularités des technologies sans fil longue distance permettent ce type d'architecture.

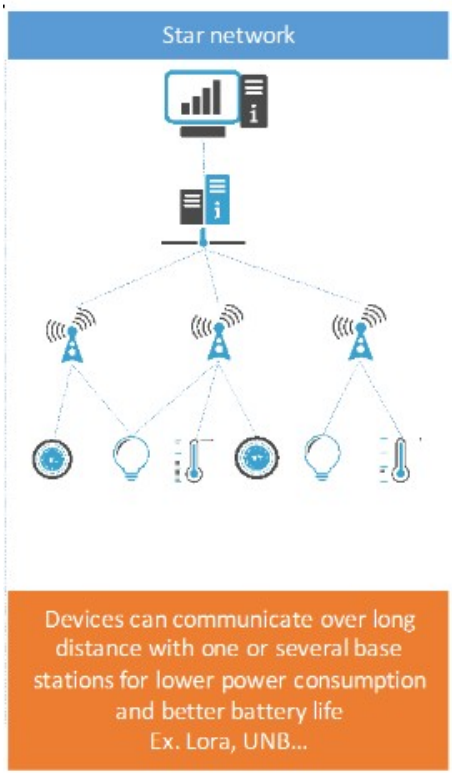


Illustration 2: Réseau en étoile

Voici quelques technologies et protocoles actuels autour du monde de l'IoT :

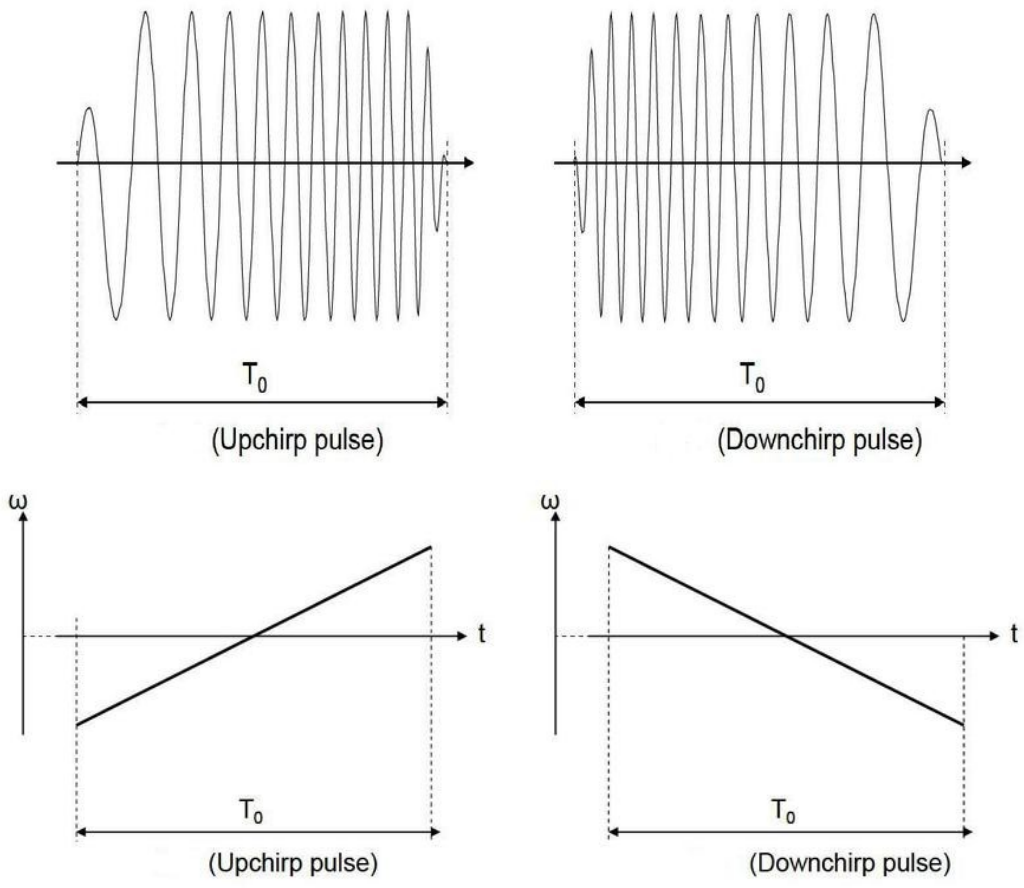
- LoRaWAN
- Sigfox
- ZigBee
- 6LoWPAN
- OCARI
- DASH-7
- Bluetooth Low-Energy (BLE)
- NFC

Couche physique – LoRa

Modulation

Il existe principalement deux documents officiels publiés par Semtech qui nous aident à étudier LoRa: la spécification du brevet déposé et l'*Application Note 1200.22*. Cependant il faut prendre ces informations avec prudence, car il n'y a aucune garantie que le protocole réel corresponde exactement à ce qui est décrit. En effet la documentation fournie par Semtech cherche aussi à **offusquer le fonctionnement réel** pour protéger la technologie.

LoRa utilise le **Chirp Spread Spectrum**. L'évolution de la fréquence d'un tel signal est toujours strictement croissante (upchirp) ou décroissante (downchirp) et linéaire au cours du temps. L'information est transmise par des déplacements de fréquence, on peut donc faire le rapprochement entre LoRa et une modulation par déplacement de fréquence multiple (*multiple frequency-shift keying* – MFSK). Comme il y a étalement de spectre, cela rend la transmission très résistante aux interférences tout en restant peu gourmande en énergie. Cette modulation de fréquence est typiquement utilisée par les Radars. La fréquence est généralement étendue sur une bande passante de 125 kHz à 500 kHz. La robustesse du signal est vitale pour les technologies émettant dans les bandes industrielles, scientifiques et médicales (ISM) qui sont utilisées par plusieurs services et donc très « bruyantes ». En Europe, la bande de fonctionnement de LoRa est 868MHz. On peut également communiquer sur les bandes 915MHz et 433MHz.



Encodage

Le *coding rate* (CR) définit le ratio de bits utile servant effectivement au transport d'information par rapport au nombre de bits de détection et de correction d'erreur. Cette valeur varie entre 1 et 4 et indique le nombre de fréquences sur lesquelles la modulation peut se « déplacer ». L'ensemble du CR et de la bande passante (*bandwidth* – BW) est appelé *datarate*. Grâce à un tel signal on peut transmettre entre 7 et 12 bits par symbole, ce nombre est appelé *spreading factor* (SF).

En modulant ces paramètres qui sont interdépendants, on peut optimiser la modulation pour une utilisation particulière de LoRa. Si la communication nécessite un fort débit, on privilégiera un SF faible. On aura donc un chirp moins étalé, plus rapide et moins résistant aux perturbations (faible distance) mais avec un débit plus grand. Ci-dessous on voit l'influence du SF sur l'allure du chirp. Il est commun de varier dynamiquement ces paramètres pour s'adapter en temps réel aux conditions d'émission.

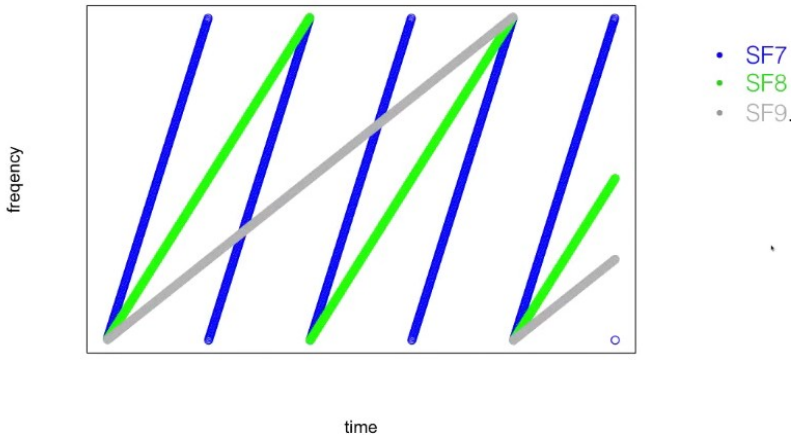


Illustration 3: Influence du spreading factor sur l'allure du chirp

$$BitRate = SF \cdot \frac{1}{\left[\frac{2^{SF}}{Bandwidth}\right]} bits/sec$$

$$BitPerSymbol = SF \cdot \frac{4}{(4+CR)} bit/symbol$$

Exemple avec un cas commun, pour une bande passante de 125 kHz, SF = 8 et CR = 4 :

BitRate = 3904 **bits/sec** = 488 octets/sec	BitPerSymbol = 4 **bit/symbole** 2^4 → 16 fréquences de shift

Avant de transmettre les symboles, LoRa décrit 2 étapes principales :

- Le signal est conjugué avec une séquence de bruit blanc définie pour ajouter de l'entropie (*whitening*)
- Entrelacement entre les bits utiles et les bits de correction/ détection d'erreur (*interleaving & error coding*)

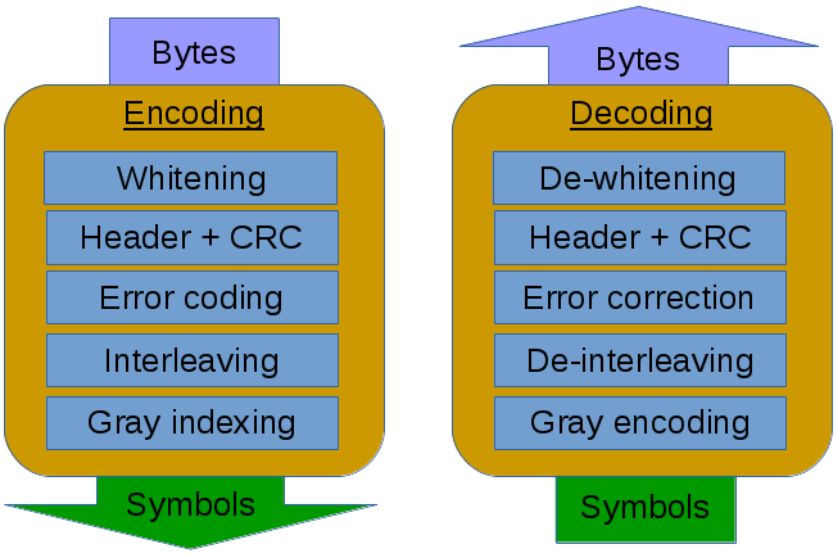


Illustration 4: Étapes de modulation et démodulation

La modulation LoRa permet de capter des signaux émis même quand le rapport signal à bruit est inférieur à 1. On peut par exemple établir une communication même avec un rapport signal à bruit de -20dB avec un SF de 12.

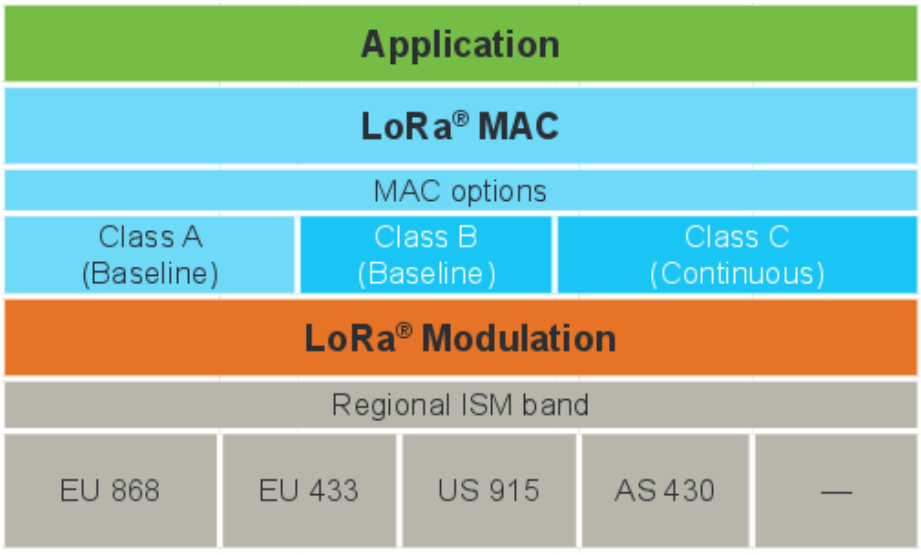


Illustration 5: Couche MAC et couche de modulation

Rétro-ingénierie

L’ingénieur Matt Knight a réalisé une analyse du protocole physique LoRa par rétro-ingénierie. En partant du signal brut transmis, il explique chaque étape de la modulation en la confrontant à la documentation fournie par Semtech. La première étape consiste à capter un message transmis et à analyser grossièrement l'allure du signal (modulation, *chirp*, préambule du message). La figure ci-dessous montre l'évolution de la fréquence (abscisse) et de l'amplitude (cote) en fonction du temps (ordonnée) d'un signal LoRa. On y voit les discontinuités provoquées par les changements de fréquence.

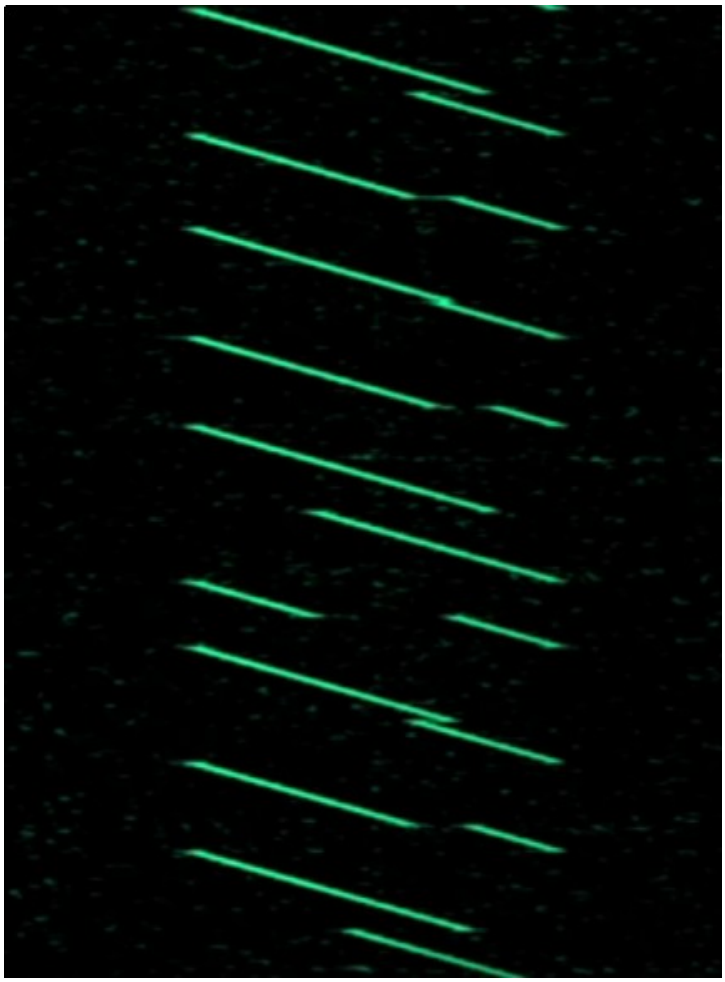


Illustration 6: Spectre fréquentiel d’une communication LoRa

Ensuite nous pouvons ramener l'étude du signal à l’étude d'une MFSK en combinant le message transmis avec des *chirps* opposés.

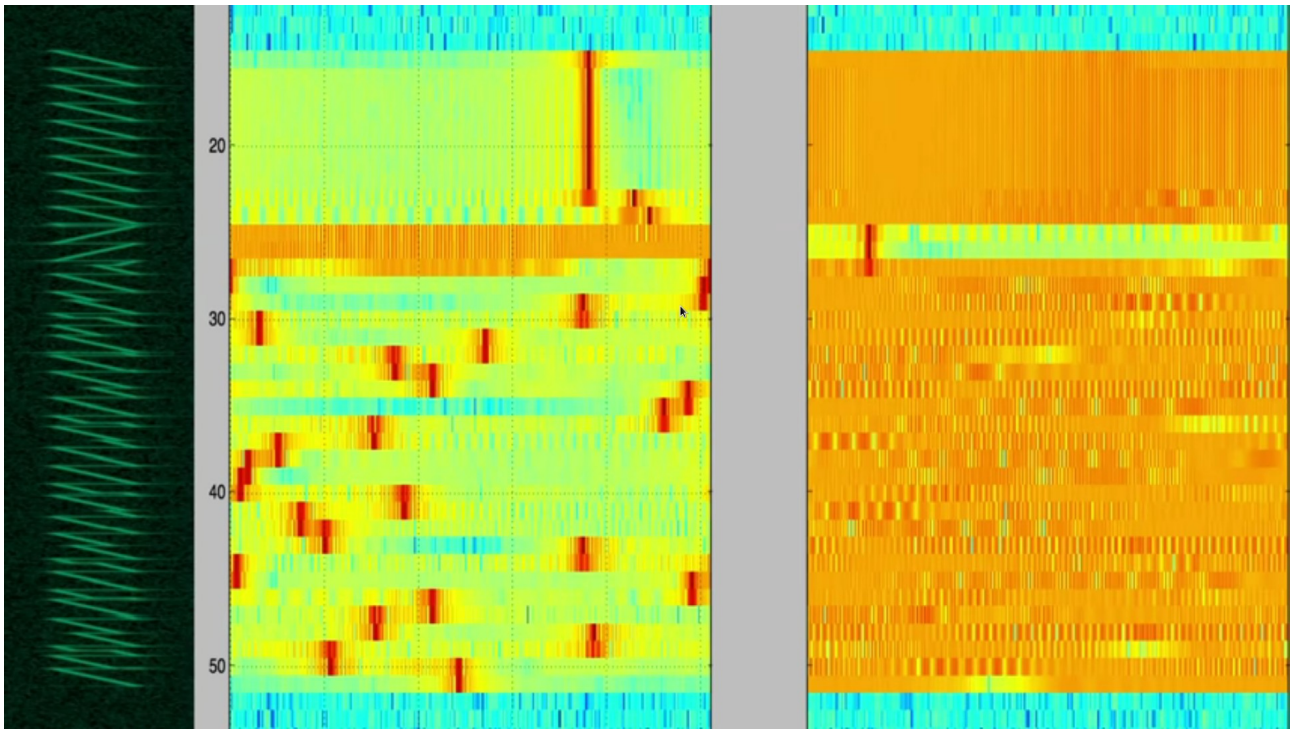


Illustration 7: Saut de fréquence LoRa

On peut voir sur le graphe ci-dessus le signal original à gauche, le signal conjugué avec des *upchirps* ensuite, et à droite le signal conjugué avec des *downchirps*. Les pentes des *chirps* se compensent et laissent apparaître des constantes. Ces constantes indiquent les sauts de fréquences et contiennent l’information (symboles). Le préambule est une série de *downchirps* suivit de 2 *upchirps* qui servent à la synchronisation. Nous sommes ici en présence d'un SF valant 8 et d'un CR valant 4 d'où les symboles codés sur 16 fréquences.

Ensuite se pose la question de ce que représentent ces symboles. En effet ici le message transmis à subit les étapes de *whitening* et d'entrelacement. Lors de l'étape de *whitening*, le message subit un XOR avec une matrice aléatoire connue de toutes les radios LoRa. En passant un message rempli de 0 on retrouve donc exactement cette matrice aléatoire.

Couche transport – LoRaWAN

Généralités

LoRaWAN (Long Range Wide Area Network) est la couche de contrôle d'accès au support (*media access control* – MAC) pour le protocole LoRa. La spécification de la première version a été publiée en janvier 2015 et une nouvelle version est en cours d'élaboration.

On distingue dans cette partie les *end-device* (capteur) de la *gateway* (agrégateur). Les *end-devices* sont classés en fonction de leur comportement. Nous allons principalement étudier le protocole avec des émetteurs de classe A.

Class name	Intended usage
A (« all »)	**Battery powered sensors**, or actuators with no latency constraint Most energy efficient communication class. Must be supported by all devices
B (« beacon »)	**Battery powered actuators** Energy efficient communication class for latency controlled downlink. Based on slotted communication synchronized with a network beacon.
C (« continuous »)	**Mains powered actuators** Devices which can afford to listen continuously. No latency for downlink communication.

Illustration 8: Classe d'émetteur LoRa

Pour gérer les différentes parties d'une communication au niveau de la couche MAC, L'INSEE 802.15.4 spécifie trois parties :

- MCPS – *MAC Common Part Sublayer*

Fonctions d'activation, accusé de réceptions

- MLME – *MAC layer Management Entity*

Fonctions d'intégrité

- MIB – *MAC Information Base*

Clé de chiffrement, taux d'encodage, adresses…

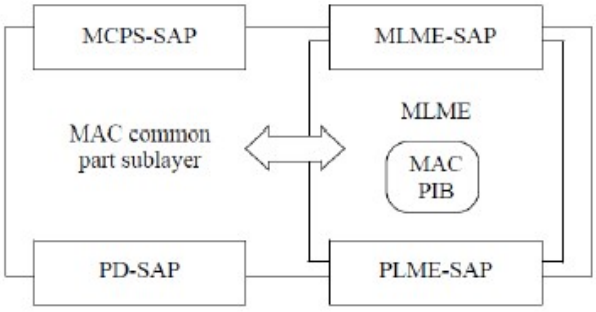


Illustration 9: Sous-couches MAC

LoRaWAN est compatible avec plusieurs options mais peu sont obligatoires. Le chiffrement de bout en bout par exemple est supporté et recommandé mais aucunement obligatoire. On peut également citer d'autres fonctionnalités comme l'adaptation automatique du taux d'échange de données, adaptation dynamique du SF. De manière native, les *end-device* changent automatiquement de canal dans la bande ISM données pour être plus robuste aux interférences. Dans la bande des 868MHz par exemple, LoRaWAN décrit **3 canaux minimum**. Toujours dans cette bande, il faut faire attention aux contraintes sur le temps d'émission qui est réglementé et qui ne doit pas dépasser 1 % pour les *end-devices*.

Structure des paquets

Voici la structure générale d'un paquet LoRaWAN décrite dans la spécification officielle :

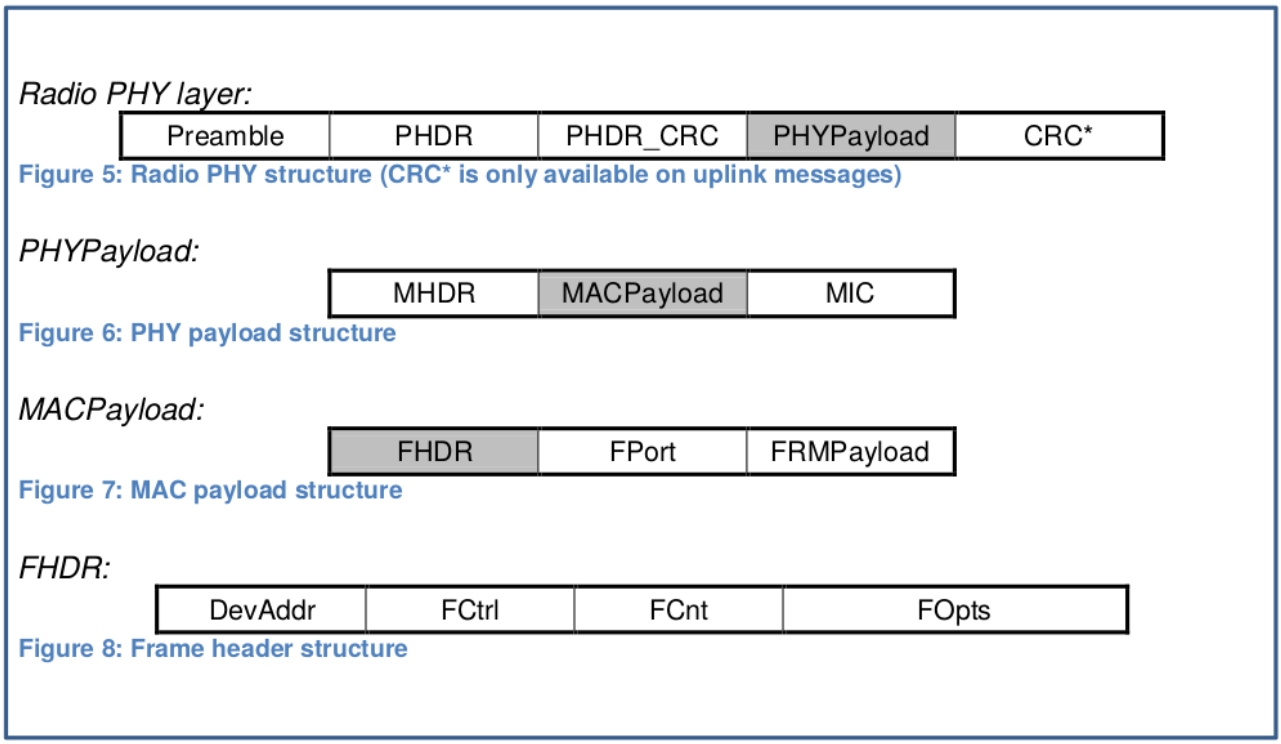


Illustration 10: Structure de paquets LoRaWAN

Processus d’authentification et d’activation

Avant de communiquer, tous les *end-devices* doivent être authentifiés et activés au sein d'un réseau. Il existe deux méthodes pour cela. La première consiste à authentifier et activer manuellement tous les *end-devices* ainsi que configurer le serveur qui va traiter la demande (*network server*). La deuxième et la plus pratique est celle dite « à-la-volée » (On-The-Air Activation) qui va authentifier et activer automatiquement l'*end-device*. Sur un réseau de taille réduite, le serveur peut être associé à la gateway (Raspberry Pi avec un module LoRa par exemple). Cette procédure a pour but de générer sur l'end-device et sur le serveur une paire de clé pour chiffrer tout futur message. Évidemment, les clés ou les éléments servant à les générer ne doivent pas être compromis pendant la procédure.

Au préalable il est nécessaire de préparer le *device* en lui renseignant :

- Un identifiant global unique (DevEUI)
- Un identifiant global d’application (AppEUI)
- Une clé AES-128 unique pour chaque device (AppKey)

Le serveur auquel le *device* fait sa demande possède au préalable les informations suivantes :

- L’AppKey du *device*
- Un identifiant global unique de réseau (NwkID)
- Une adresse réseau libre à fournir au device (NwkAddr)

Seulement alors peut commencer le processus d'authentification et d'activation à la volée. Un premier paquet, la demande, est émis en clair sur des canaux par défaut par le device. Le message est envoyé en clair mais un code d'intégrité du message (*Message Integrity Code* – MIC) est calculé pour signer le paquet. Il permettra d'authentifier cette demande et de garantir la légitimité de celle-ci auprès du serveur. Un nonce est un code utilisable qu'une seule fois. Uniquement si toutes les informations sont correctes, le serveur lui répond par un message *join_accept*.

À la fin de la procédure, grâce aux champs NetID et DevAddr, *l'end-device* possède les informations nécessaires pour communiquer au sein du réseau. En effet les informations reçues permettent de dériver une paire de clés à partir de l'AppKey. L'AppSKey chiffre le message alors que la NwkSKey sert à calculer le MIC.

NwkSKey *= aes128_encrypt(AppKey, 0x01 | AppNonce | NetID | DevNonce | pad16)*
AppSKey *= aes128_encrypt(AppKey, 0x02 | AppNonce | NetID | DevNonce | pad16)*

Voici en détail les échanges lors d'une telle procédure.

End-device

Envoi un *join_request* contenant un nonce aléatoire, **signé** par l'AppKey

Size (bytes)	8	8	2
Join Request	AppEUI	DevEUI	DevNonce

Illustration 11: Paquet join_request

Network Server

Vérifie que le nonce n'a jamais été utilisé
Génère et compare le MIC
Vérifie les identifiants du *device* et de l'application
Génère un nouveau nonce aléatoire
Envoi un *join_accept* **signé** et **chiffré** par l'AppKey contenant ce nonce
Génère la paire de clés (NwkSKey, AppSKey)

Size (bytes)	3	3	4	1	1	(16) Optional
Join Accept	AppNonce	NetID	DevAddr	DLSettings	RxDelay	CFList

Illustration 12: Paquet join_accept

End-device

Vérifie le MIC
Déchiffre le join_accept avec l'AppKey
Génère la paire de clés (NwkSKey, AppSKey) avec les informations du join_accept

La sécurité de LoRaWAN

Risques internes

Chiffrement des paquets

La sécurité de LoRa se repose en grande partie sur le triplet de clés évoqué précédemment. Si on désigne par opérateur un fournisseur qui possède la *gateway* et le *network server, v*oici la portée théorique de ces clés :

- AppKey est connue du client uniquement
- AppSKey est connue uniquement du client
- NwkSKey est connue du client et de l'opérateur

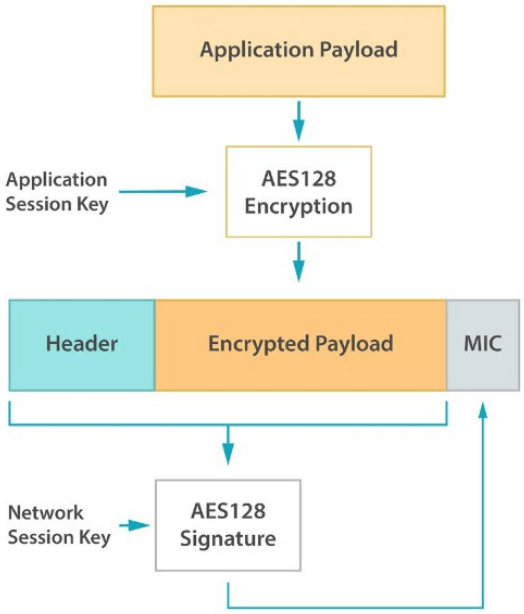


Illustration 13: Utilisation des clés

Les deux clés de sessions sont utilisées ensuite comme décrit dans la figure ci-dessus. Le chiffrement avec l'AppSKey contient un désavantage majeur : il ne change pas là la taille du message envoyé. En effet, voici le message utile (payload) est d'abord découpé en bloc de 16bits. Ensuite une matrice de chiffrement est élaboré **pour chacun des blocs** puis une opération XOR entre le message et l'ensemble de ces matrices est effectué.

L'unicité de la matrice de chiffrement est entre autre garantie par l'utilisation lors de la génération de la matrice du compteur de message (nonce) et de l'indice du bloc. Ceci évite le rejeu. Il est alors possible de distinguer les messages en fonction de leur longueur, ce qui peut se révéler utile si les messages sont assez répétitifs.

End-device de classe B et C

Dans certains cas, l'*end-device* peut être configuré pour « écouter » la *gateway*. Dans ce cas-là d'autres enjeux apparaissent. En effet la tentation d'effectuer un message *multi-cast* est grande, pour divulguer rapidement une commande MAC de la *gateway* vers tous les objets connectés. Mais diffuser de tels messages de façon sécurisée n'est tout simplement pas prévu par le protocole. La spécification prétend supporter cette fonction mais le problème est évident : comment authentifier le message si tous les *end-device* ont une clé NwkSKey différente ? En effet les messages MAC sont forcément chiffrés par le *network server* avec NwkSKey, la seule clé qu'il est censé posséder. La communication ne peut donc pas être nativement multi-cast et sécurisée.

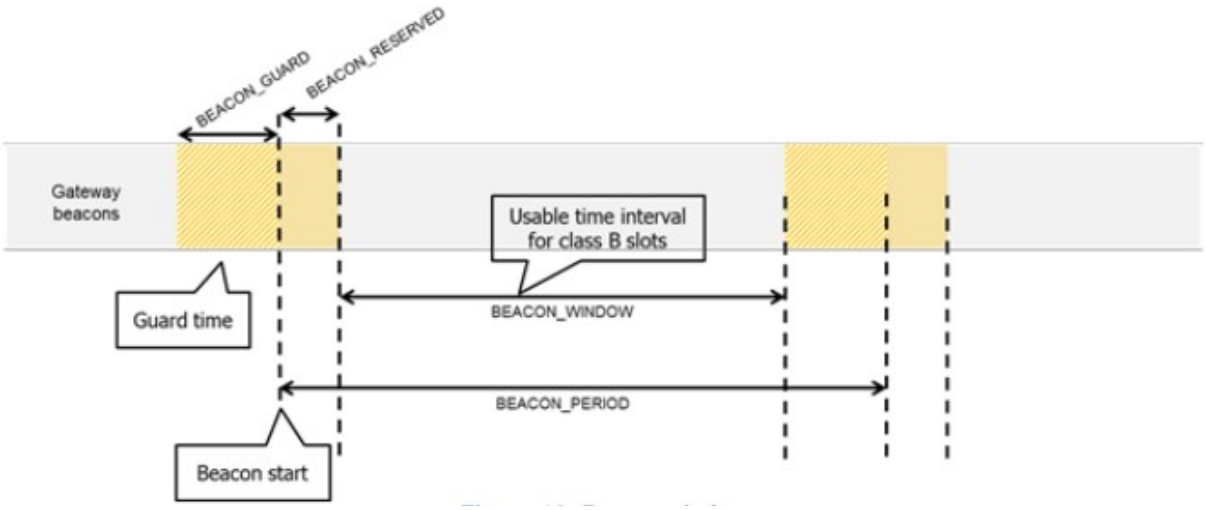


Illustration 14: Balise émise par une gateway Beacon

De même pour les *gateway* qui diffusent en clair un timestamp et leurs coordonnées GPS, cela introduit de nouvelles problématiques. Les coordonnées sont évidemment des informations précieuses pour mener vers les gateways elle-mêmes. Des comportements de bords peuvent aussi être étudiés en diffusant malicieusement des timestamp extrêmes (1 janvier 1970, ou 19 janvier 2038, etc.).

Performances

Pour un *end-device* de classe A, la règle d'émission est similaire au protocole ALOHA dans sa première version (*pure ALOHA*). Il n'existe aucune écoute avant émission, aucune gestion de collision de message et on réémet périodiquement le message jusqu'à bonne réception. La transmission n'est donc pas optimisée. La conséquence directe est qu'un parc d'objets connectés conséquent crée un environnement dense dans lequel il est difficile de communiquer. Le chercheur Maarten Weyn à analysé les pertes de paquets dans des conditions optimales selon les législations européennes (contrainte de puissance, de temps d'émission continue, etc.). Le graphe ci-dessous représente le nombre de collision de paquet et le taux d'erreur d'un paquet. Pour une centaine d'*end-device* qui émettent la même minute, on atteint déjà les 10 % d'erreur dans le paquet transmis.

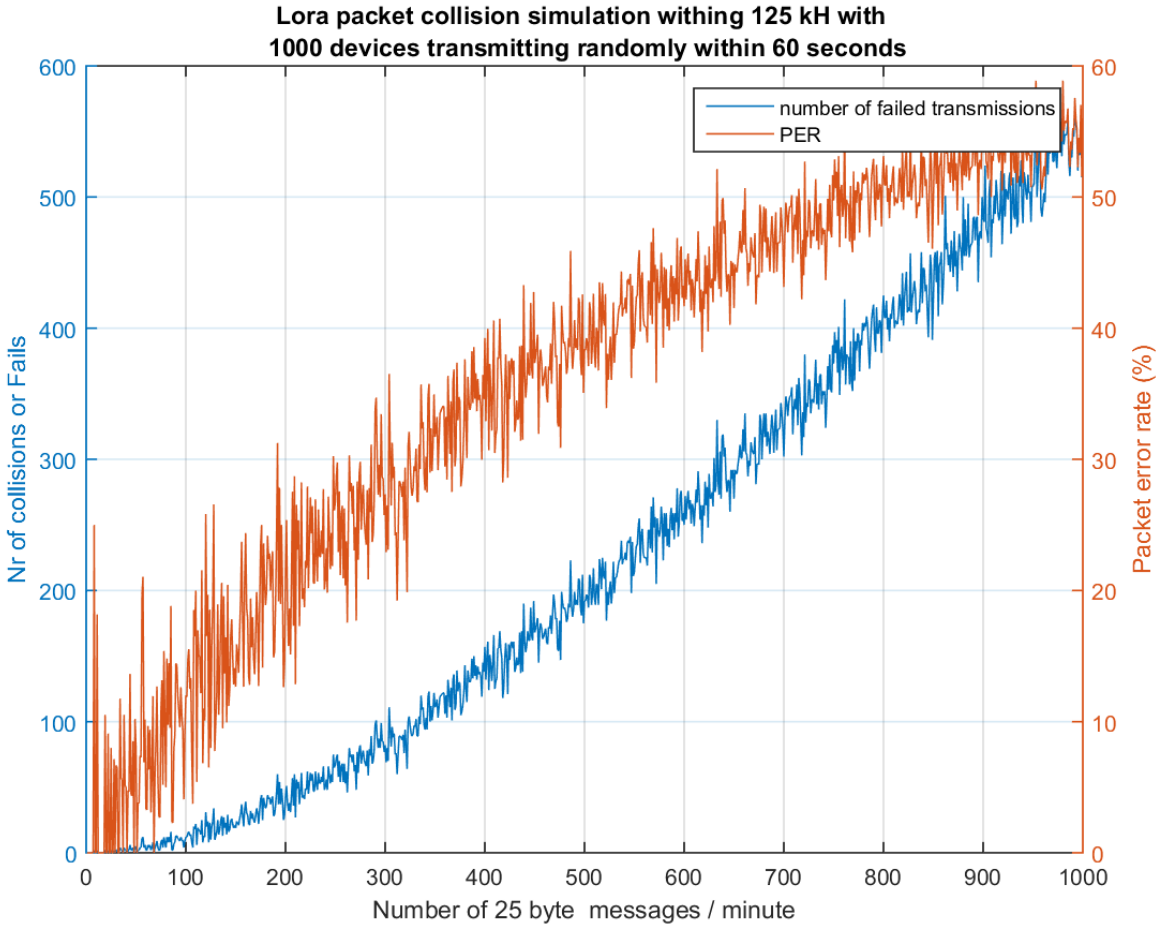


Illustration 15: Collision de paquet pour 1000 end-devices

L'effet pernicieux est que plus le facteur d'étalement (SF) est élevé, plus le signal sera transmis loin (résistance aux interférences) mais plus le risque de collision et d'échec de transmission est élevé. Le compromis est difficile à trouver. Le protocole LoRaWAN prend aussi en compte la limite de cycle d'émission pour respecter la législation européenne. Ceci implique que pour respecter la spécification du protocole le maximum de transmission continue pour un *end-device* est de 36 secondes toutes les heures.

Risques externes

Implémentation

Le protocole prévoit des sécurités mais libre à celui qui l'implémente d'adapter l'architecture à son utilisation et son matériel. Au vu du faible des coûts des capteurs, une forte population non sensibilisé à la sécurité peut déployer son propre matériel. Les risques qui en découlent sont très nombreux sur les réseaux ouverts comme *The Things Network*. La meilleure solution actuelle pour ce réseau est de ne pas chiffrer le trafic et d'avertir qu'aucune garantie n'est fournie. La clé NwkSKey de ce réseau par exemple était pendant longtemps publiée sur leur site. Ceci représentait un risque énorme puisque les messages de contrôles d'accès au média sont uniquement chiffrés en utilisant cette clé. On pouvait donc très facilement se faire passer pour une *gateway* du réseau et causer d'énormes dégâts (changement des fréquences utilisées, changement des fenêtres d'écoute, etc.). Plus globalement, aucune implémentation *open source* d'un *network server* respectant la spécification LoRaWAN n'est disponible.

La recommandation principale de sécurité est d'utiliser une clé d'application unique pour chaque *end-device*. Ainsi la compromission d'une seule clé ne remet pas en cause la sécurité de tout le parc d'objets connectés. De nouveaux risques apparaissent de la génération de ces nombreuses clés et de leur transfert vers l'*end-device*. Cela dépasse le cadre de notre étude mais la solution encore une fois adoptée par de nombreux opérateurs est de générer eux-mêmes l'AppKey.

On peut également citer les attaques par *side-channel*. Le faible coût des *end-device* là aussi ne pousse pas à investir dans la prévention des risques liés à ces attaques. Très souvent les solutions hardware sont des systèmes simples sans protection de mémoire et des ports série encore actifs. Il faut évidemment relativiser la compromission d'un seul capteur sur un parc de potentiellement plusieurs milliers. Les gateways ont cependant un impact bien plus grand. Le coût d'une telle attaque semble assez dissuasifs pour ne pas préoccuper les opérateurs.

Une autre faille réside dans la couche applicative. Si un opérateur fournit une interface pour remonter les valeurs des *end-devices*, toute l'infrastructure est sensible aux attaques déjà bien connues (XSS, injection, *Man in the Middle, etc.*). La base de données, le serveur web, l'interface utilisateur ou tout autres éléments en façade d'internets sont autant d'éléments qui doivent être robustes. Une chaîne est aussi solide que son maillon le plus faible. Le protocole souvent utilisé pour remonter l'information à l'heure actuelle est MQTT (*Message Queue Telemetry Transport*). La mise en place de ce protocole souffre souvent d'une sécurité négligée.

L'algorithme de chiffrement AES-128 est très robuste. Par exemple, les documents classifiés et secrets américains peuvent être chiffrés par cette méthode. La meilleure attaque purement algorithmique (force brute) actuelle utilise 9 pétaoctets de stockage et quelques milliards d'années. L'échange de clé cependant est par contre beaucoup moins sécurisé. L'utilisation de l'AES-128-ECB semble non justifiée par rapport à un chiffrement par cipher-block chain (CBC).

Rôle de l’opérateur

Le périmètre des trois clés LoRaWAN doit être bien défini. La plupart des opérateurs, si ce n'est tous, possède les *network servers* qui stockent les trois clés. La confiance dans la sécurité du protocole LoRa est donc limitée par la confiance dans son opérateur. Orange confirme par exemple dans son guide du développeur LoRaWAN que leur serveur génère et stock la paire de clé NwkSKey et AppSKey. L'opérateur a donc toutes les informations nécessaires pour chiffrer et déchiffrer la communication d’un client.

«

> *Compared to some other systems which rely on a single key for authentication and encryption, the LoRaWAN framework separates authentication and encryption, so that Orange is able to authenticate packets and provide integrity protection. [...] The AppSKey and NwkSKey are stored by Orange for regulatory and security reasons.*

»

Comme les clés sont des données sensibles, on peut imaginer que stocker ces informations sur des serveurs sécurisés chez l'opérateur plutôt que chez le client est effectivement une mesure de sécurité. Une autre conception cependant serait de déléguer la génération des clés entièrement au client et de renseigner l'opérateur avec uniquement la clé le concernant : NwkSKey. Le compromis qu'a imaginé Gemalto est de relayer la génération des clés à un tiers de confiance :

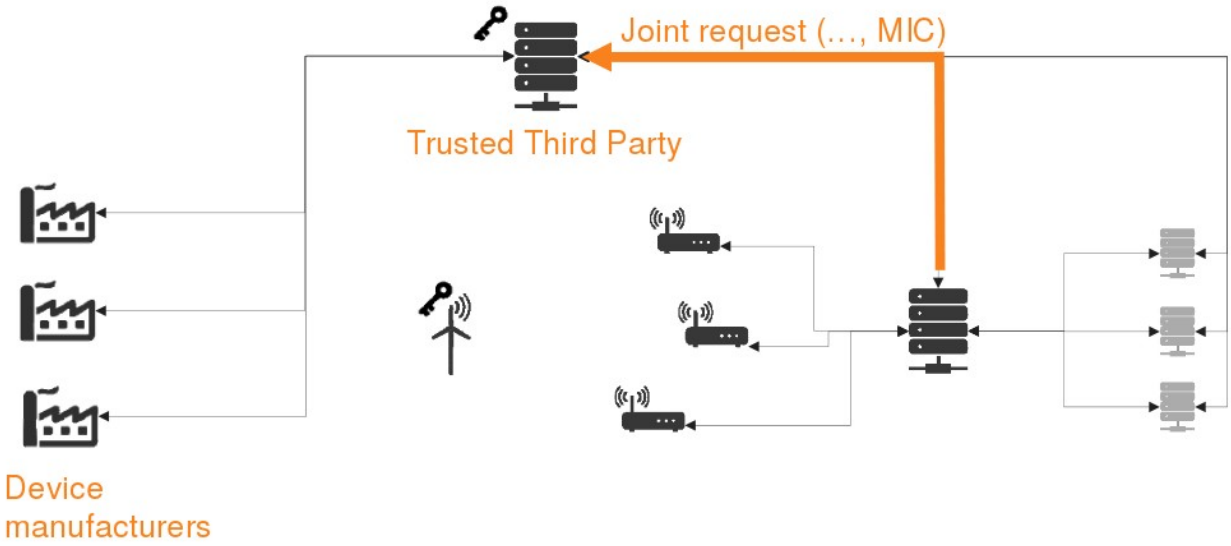


Illustration 16: Utilisation d’un tiers de confiance

La requête *join_request* est transférée au tiers de confiance. Les clés sont alors générées sans que l'opérateur ne connaisse ni la clé AppKey ni la clé AppSKey.

Analyse de consommation

La problématique de durée d'autonomie de l'*end-device* est souvent montrée comme très avantageuse avec le protocole LoRa. Il faut cependant garder en tête certains aspects qui peuvent drastiquement diminuer la durée de vie de la batterie.

Dans certains cas, les communications exigent des accusés de réception. Deux fenêtres d'écoute peuvent être alors ouvertes à des délais précis pour recevoir cet accusé. Or comme vu précédemment lorsqu'on atteint une densité conséquente d'*end-devices*, la seule solution est d'émettre jusqu'à bonne réception ou jusqu'à un nombre limite autorisé. En regardant de plus près la consommation énergétique de la radio lors de l'émission on peut dresser le bilan d'une tentative de communication. Le schéma ci-dessous illustre l'émission d'un premier message (*data0*). Après non-réception de l'accusé, re-transmission. Enfin un deuxième message (data1) est transmis correctement

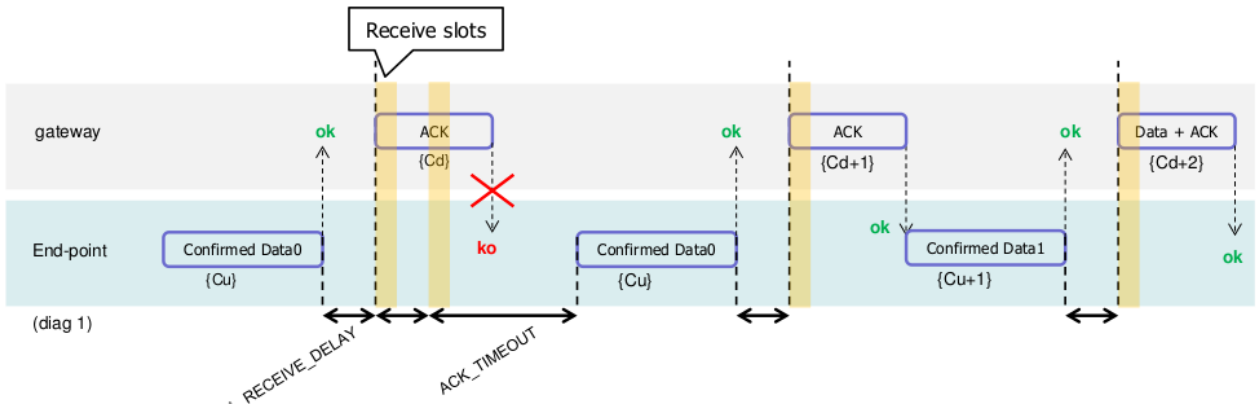


Illustration 17: Transmission de 2 paquets, avec échec du premier

Pour la puissance à l'émission nous avons sans surprise la plus grosse consommation énergétique avec 364 mW de puissance moyenne pendant 61 ms. La durée d'émission est évidemment fonction des paramètres choisis tel que le SF, le CR ou encore la bande passante.

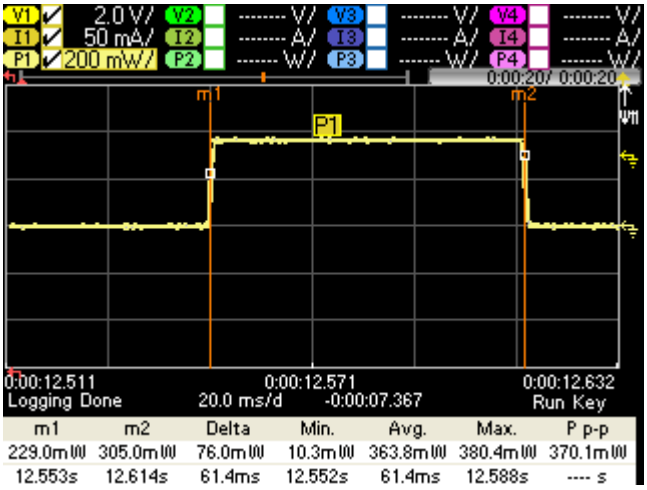


Illustration 18: Puissance consommée pour une fenêtre d'émission

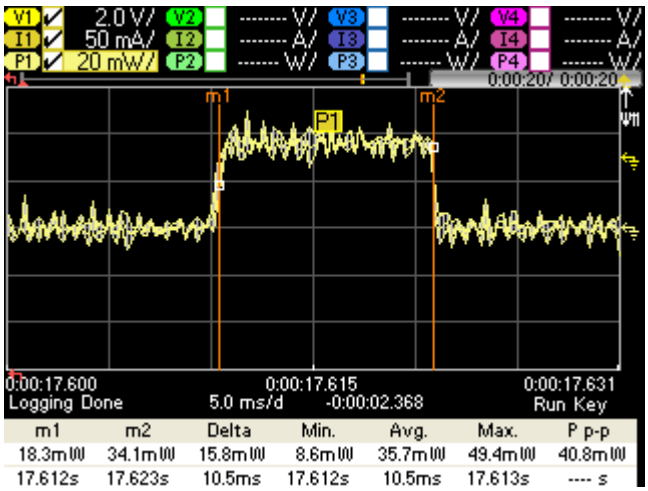


Illustration 19: Puissance consommée pour une fenêtre de réception n°1

Pour la première fenêtre de réception, nous avons 36 mW de puissance moyenne pendant 10 ms. Cette fenêtre est volontairement courte car dans des conditions idéale de réception l'accusé de réception ne doit pas avoir une empreinte énergétique trop grande.

À l'inverse, la deuxième fenêtre d'écoute est plus grande. Ceci est logique puisque si la première réception n'a pas été fructueuse, il faut prendre en compte un facteur d'erreur. Un exemple commun est une désynchronisation des horloges entre l'*end-device* et la *gateway.* Il est donc nécessaire d'avoir un temps d'écoute plus long. Nous avons une consommation de 36 mW de puissance moyenne pendant 164 ms.

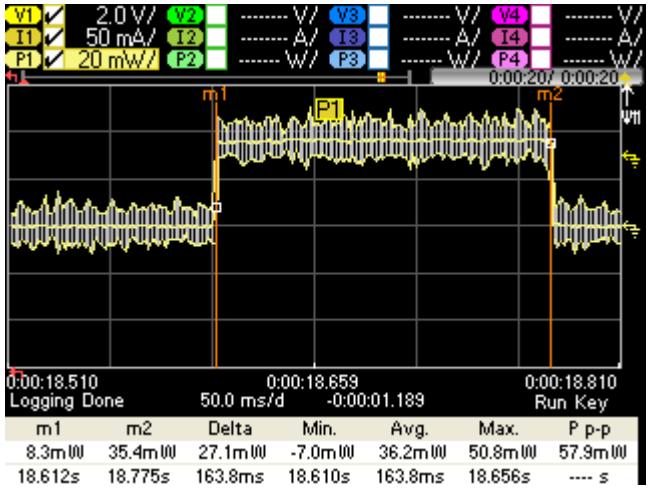


Illustration 20: Puissance consommée pour une fenêtre de réception n°2

Récapitulatif des risques

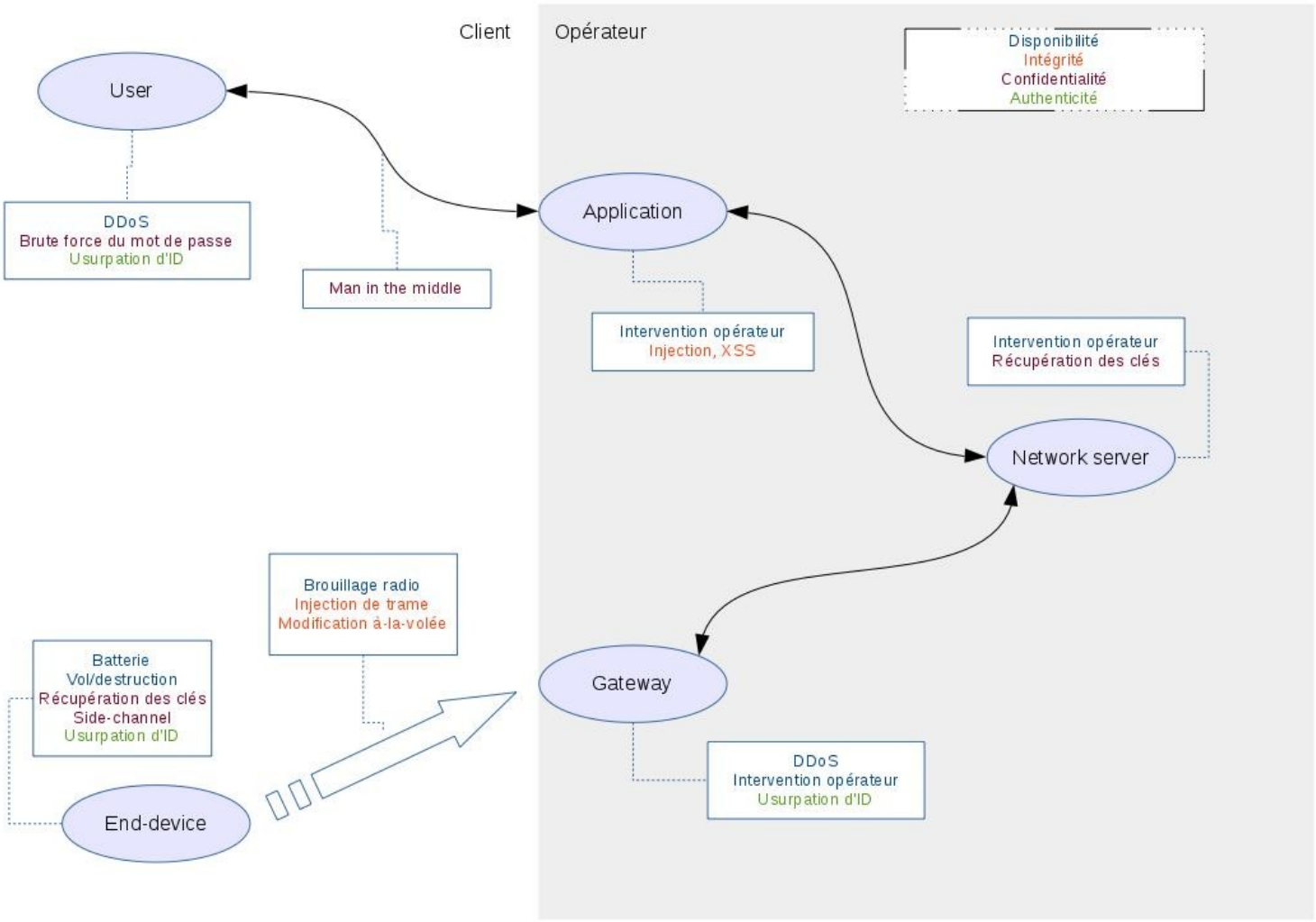


Illustration 21: Récapitulatif des risques

Conclusion

Ce projet a été une excellente occasion de découvrir le monde de l'internet des objets. Ce domaine plein d'avenir a l'avantage de concerner directement la formation IMA de Polytech Lille. La richesse se trouve notamment dans la combinaison de l'aspect purement physique (modulation, transmission radio) et de l'aspect protocolaire de la couche MAC. Après une première appréhension du côté théorique du sujet, cette richesse m'a finalement séduit.

Les objectifs ont été remplis puisqu'un état de l'art sur le protocole LoRa et son protocole associé LoRaWAN a été réalisé. Cependant, comme LoRa subit régulièrement des améliorations par sa communauté très active, il faut garder à l'esprit que les informations rassemblées ici peuvent devenir désuètes. Une réflexion sur les différents aspects de sécurité a également été menée a bien. Bien que non exhaustive, la liste des menaces donne de solides bases à la compréhension de la sécurité du protocole.