Étude de la Sécurité

LoRa & LoRaWAN

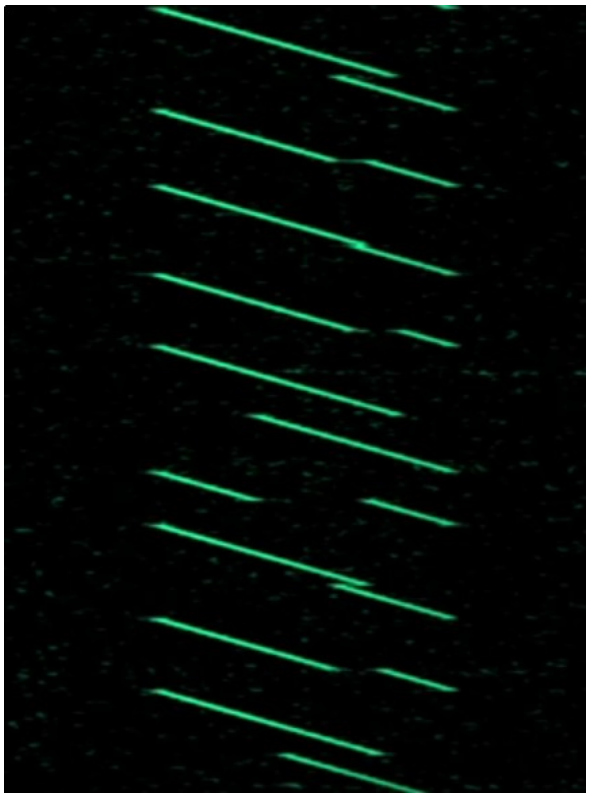


Table des matières

Introduction

La sécurité de l'internet des objets est tout aussi majeure que l'expansion du parc d'objets connectés lui-même.

Ce document n'est pas une étude de risque formelle du protocole LoRa. C'est une brève évaluation des menaces qui reste un pré-requis à toute mise en place de mesures de protection afin de s’assurer du bien fondé de celles-ci. Sont donc simplement rassemblés ici des axes de réflexions sur la sécurité de ce protocole.

Nous allons voir dans un premier temps les risques proprement liés au protocole, les risques internes. Ensuite nous allons explorer les risques qui ne sont pas directement dus à LoRa mais plutôt à l'implémentation du protocole ou a des attaques extérieures.

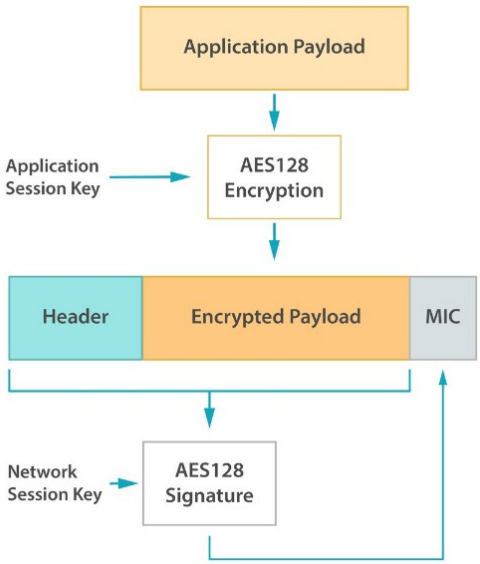
Risques internes

Chiffrement des paquets

La sécurité de LoRa se repose en grande partie sur un triplet de clés. La première est échangée symétriquement avec un *network server* et un *end-device* et sert à la dérivation de deux autres clés dites de session.

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

Les deux clés de sessions sont utilisées ensuite comme ceci :

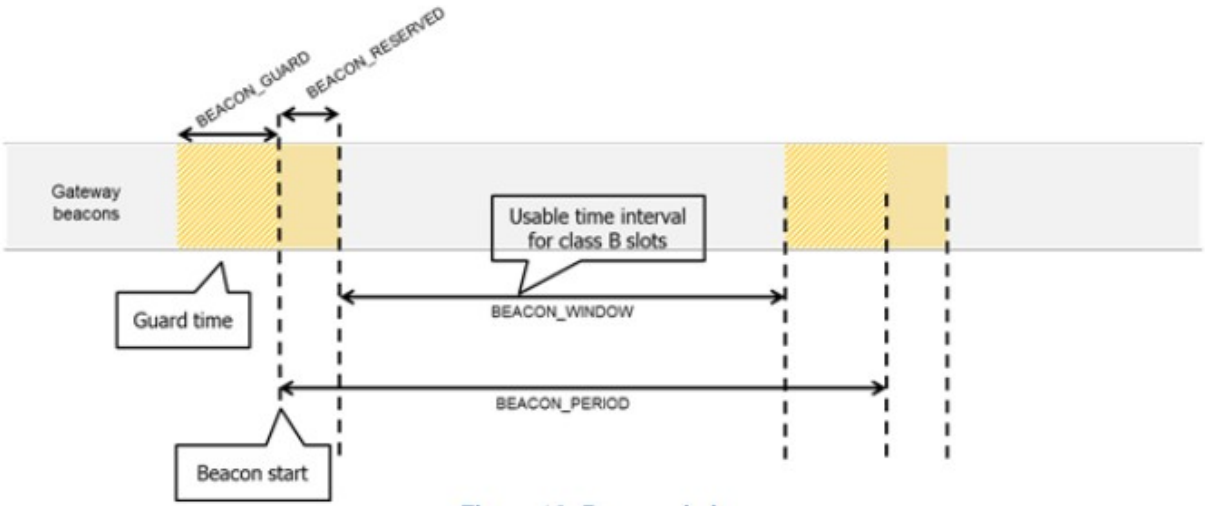


Le chiffrement avec l'AppSKey contient un désavantage majeur : il ne change pas la la taille du message envoyé. En effet, voici le processus de chiffrement :

- Le message utile (payload) est découpé en bloc de 16bits
- Une matrice de chiffrement est élaboré **pour chacun des blocs**
- XOR entre le message et l'ensemble des matrices de chaque blocs

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 et de l'indice du bloc. Ceci évite le rejeu. Renaud Lifchitz explique que : *« Il est donc possible de distinguer les messages en fonction de leur longueur, ce qui peut se révéler utile si les messages sont assez répétitifs ».*

Dans certains cas, l'*end-device* peut être configuré pour « écouter » la gateway. On parle alors de classe B et C. 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 ce n'est tout simplement pas prévu par le protocole. La spécification prétend supporter cette fonction mais le problème est évident : comment chiffrer 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 nativement être multi-cast et sécurisée.



De même pour les gateway dites « beacon » qui diffusent en clair un timestamp et leur 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 extrèmes (1 janvier 1970, ou 19 janvier 2038 etc.).

Processus d'activation On-The-Air

Voici le détail de l'échange qui permet à un *end-device* de s'activer et s'authentifier au sein du réseau. Signer une trame revient à calculer un code d'intégrité du message (MIC).

End-device

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

Gateway

Converti la requête LoRaWAN dans un autre protocole (IP, IPv6 etc.)

Network Server

Vérifie que le nonce n'ai jamais été utilisé

Génère et compare le MIC

Vérifie les identifiants du device et de l'application

Génère un nouveau nonce

Envoi un join_accept **signé** et **chiffré** par l'AppKey contenant ce nonce

Génère la paire de clés (NwkSKey, AppSKey)

Gateway

Converti la requête

End-device

Vérifie le MIC

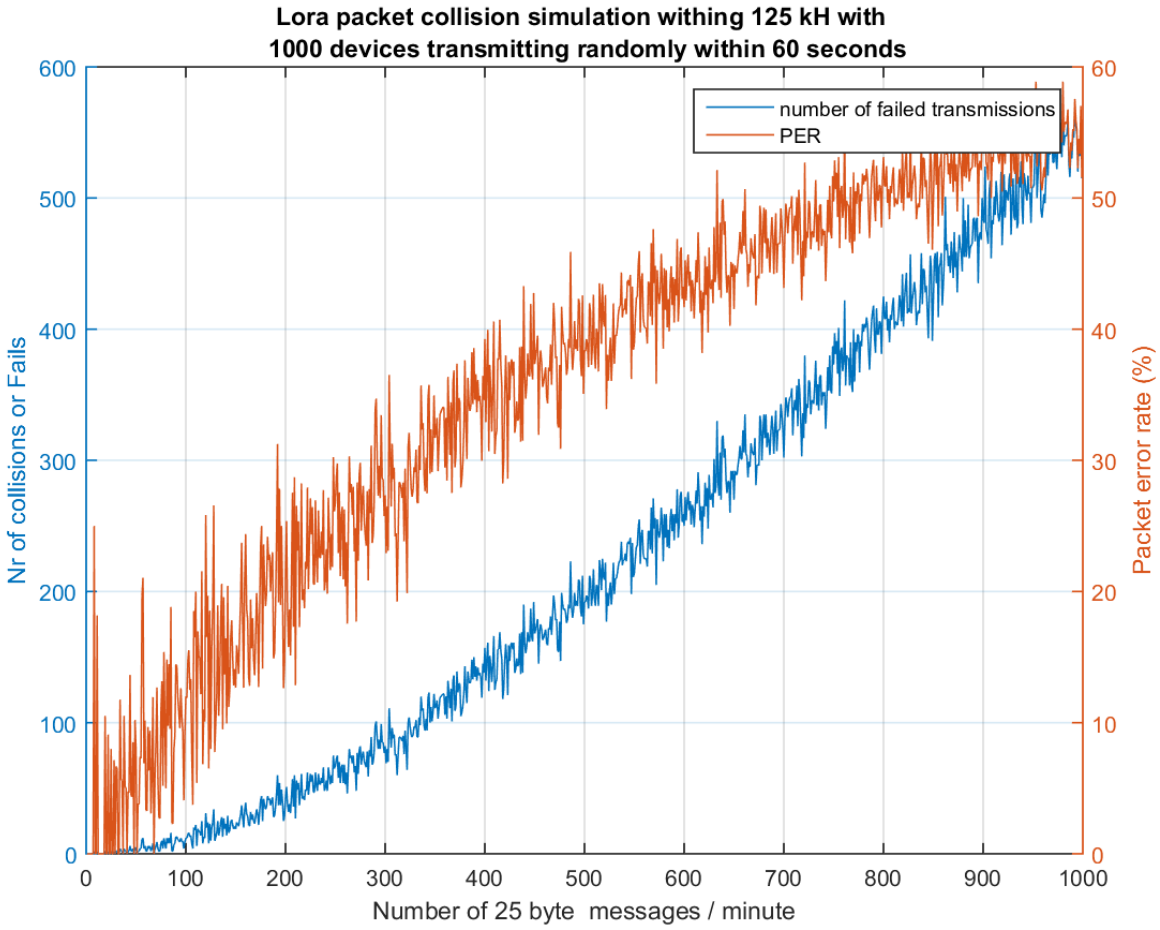
Déchiffre le join_accept avec l'AppKey

Génère la paire de clés (NwkSKey, AppSKey) avec les informations du join_accept

Si ce processus est bien implémenté, un end-device pour s'activer en toute sécurité.

Performances

Pour un *end-device* de classe A, la règle d'émission est similaire au protocole ALOHA dans sa première version (*pure ALOHA*) : aucune écoute avant émission, aucune gestion de collision de message, 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. Maarten Weyn à analysé les pertes de paquets dans des conditions optimales selon les législation européenne (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.



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 de 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 ouvert 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-même 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 de debug 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 a l'heure actuelle est MQTT (Message Queue Telemetry Transport). La mise en place de ce protocole souffre souvent d'une sécurité négligée.

Rôle de l'opérateur

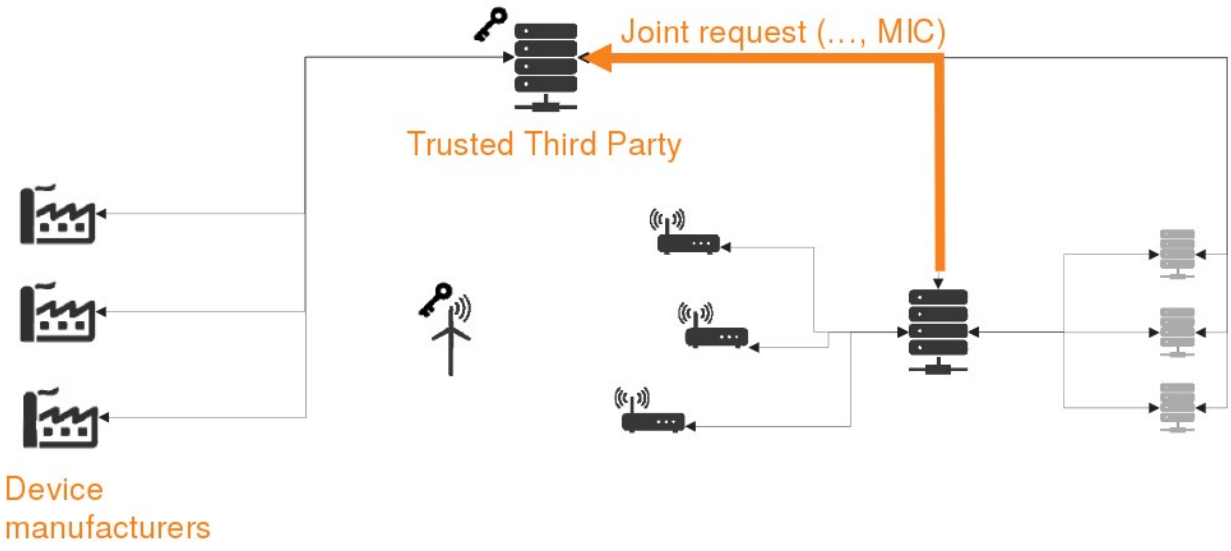
Le périmètre de ces clés doit être bien défini. La plupart des opérateurs (à l'heure actuelle, tous) possède les network servers qui stockent les trois clés. La confiance dans la sécurité du protocole LoRa est donc limitée par la confiance dans son opérateur. Orange confirme par exemple dans son guide du développeur LoRaWAN que leur serveur génère et stock la paire de clé NwkSKey et AppSKey. L'opérateur a donc toutes les informations nécessaire 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.*

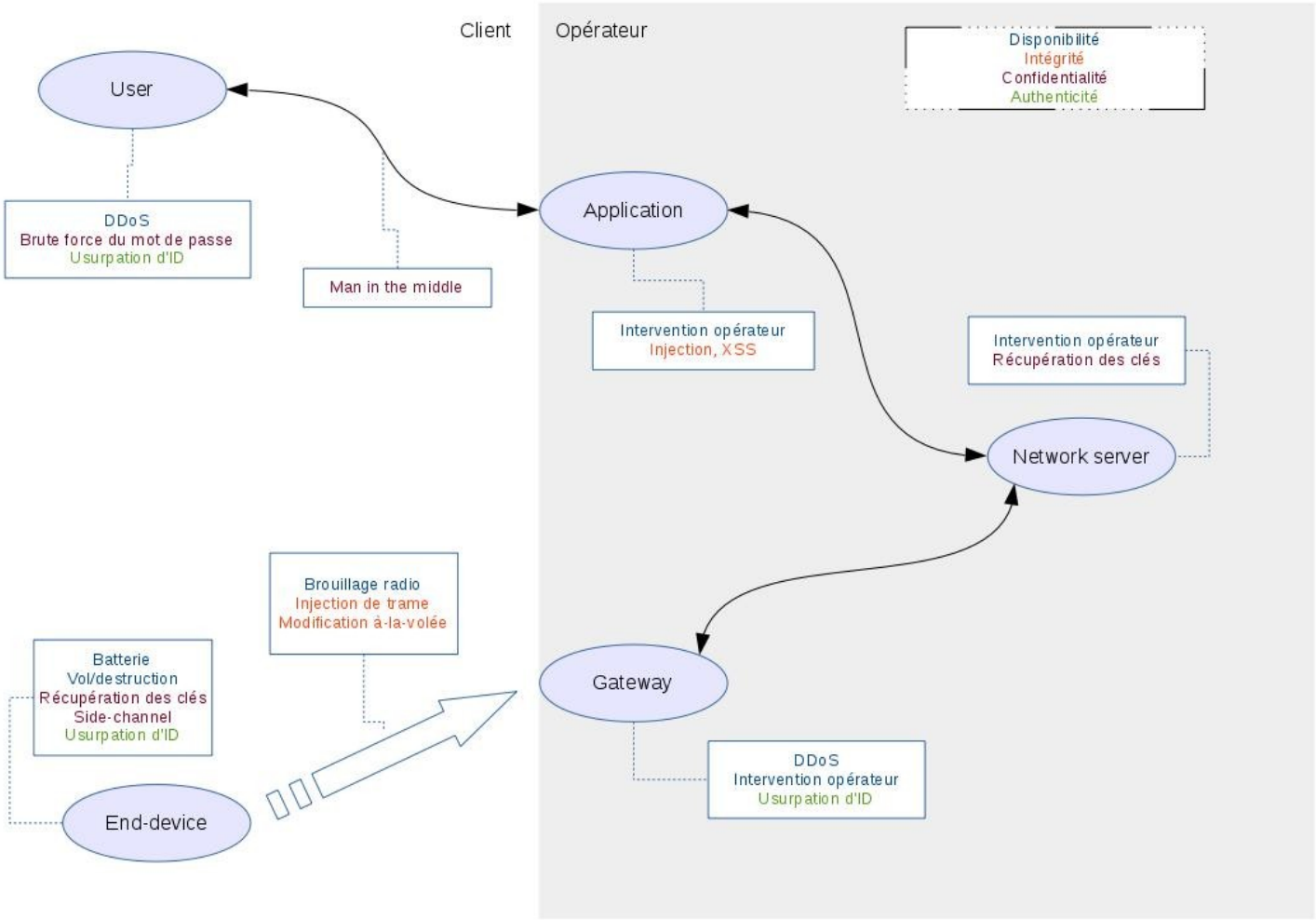
»

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 :



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.

Récapitulatif des risques



Sources

Maarten Weyn analyse LoRa en cas de communication dense
https://github.com/maartenweyn/lpwansimulation

Orange et Actility - Guide du développeur LoRaWAN
https://partner.orange.com/wp-content/uploads/2016/04/LoRa-Device-Developer-Guide-Orange.pdf

Robert Miller (MWR InfoSecurity) - Analyse de la sécurité
https://labs.mwrinfosecurity.com/assets/BlogFiles/mwri-LoRa-security-guide-1.2-2016-03-22.pdf

Libelium LoRaWAN Networking Guide
http://www.libelium.com/downloads/documentation/waspmote-lorawan-networking-guide.pdf

Pierre Girard (Gemalto) – Sécurité des réseaux LPWAN
https://docbox.etsi.org/Workshop/2015/201512_M2MWORKSHOP/S04_WirelessTechnoforIoTandSecurityChallenges/GEMALTO_GIRARD.pdf