Implémentation de la sécurité – Projet TAILS

Ce document présente un résumé des étapes franchies pour la mise en place de la sécurité entre le drone (LoRa) et la station de base, à travers des tests locaux et scripts Python.

# 1. Étapes réalisées & Résultats

Chaque étape est accompagnée d’une capture d’écran de la sortie terminale et d’une courte explication technique.

## 1.1 Génération de trame chiffrée (sender.py)

- Test en mode « dry-run » du script côté drone  
- Vérifie que la trame est bien générée au format attendu (header, nonce, ciphertext\_tag)  
- Le chiffrement AES-GCM est opérationnel

**Instruction:**

cd .\TAILS-Embedded

python .\OnBoardNode\apps\dependencies\lora\sender.py

## 

## 1.2 Déchiffrement local & Vérification anti-rejeu

Cette étape illustre la différence entre une **trame acceptée** et une **trame rejetée** lorsqu’on tente de la réinjecter.  
Le mécanisme repose sur la vérification du **compteur CTR**, utilisé pour empêcher les attaques par rejeu.

-Réinjection de la même trame  
- Le système rejette la trame (CTR déjà vu)  
- Vérifie la protection contre les attaques par rejeu

Cas 1: Trame ACCEPTÉE

A screen shot of a computer

AI-generated content may be incorrect.

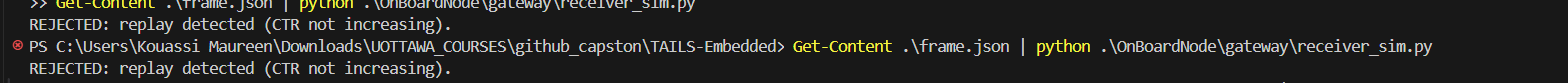
Commande utilisée:

echo '{"header": "010000000700000001", "nonce": "00000007000000001150f3e63", "ciphertext\_tag":"17108d0fbedfa5e69a3245cce9f031a6d2889836f27a54866f018fc2c52f86a82194c42b23ee4d33f39df4f1dbd077593a7fe880d91c9992cee33bc9f597fcb0d8"}' | python .\OnBoardNode\gateway\receiver\_sim.py

**Ce que cette étape vérifie :**

* Le programme reçoit une trame au format JSON en entrée standard. La trame est transmise directement au script receiver\_sim.py via la commande echo.
* La trame est **décryptée correctement** grâce à la clé partagée (dans keys.json).
* Le compteur CTR est **valide et unique**, donc le message est accepté.
* Les données GPS déchiffrées sont affichées sous forme JSON dans le terminal.

Cas 2 : Trame REJETÉE (rejeu)



Commandes utilisées :

Set-Content -Path .\frame.json -Value '{"header":"010000000700000001","nonce":"0000000700000001150f3e63","ciphertext\_tag":"17108d0fbedfa5e69a3245cce9f031a6d2889836f27a54866f018fc2c52f86a82194c424b23ee4d33f39df4fdb10d7077539a7fe886d91c999c2ee33bc9f597fcb0dd8"}'

Get-Content .\frame.json | python .\OnBoardNode\gateway\receiver\_sim.py

**Ce que cette étape vérifie :**

* La même trame est réinjectée une deuxième fois en utilisant un fichier temporaire (frame.json).
* Le système détecte que le compteur CTR n’a pas augmenté et rejette la trame.
* Cela vérifie la protection contre les attaques par rejeu.

## 1.4 Pipeline complet local

- Exécution en une seule commande (sender → receiver)  
- Simule l’envoi LoRa réel sans matériel  
- Confirme que le flux chiffré/déchiffré est stable

Elle permet de vérifier le **fonctionnement de bout en bout** du flux local (génération → chiffrement → transmission simulée → déchiffrement → affichage des données GPS), ainsi que la **protection anti-rejeu**.

Remove-Item .\OnBoardNode\gateway\last\_ctr.json -ErrorAction Ignore

$env:JSON\_ONLY="1"

$env:CTR="1"

python .\OnBoardNode\apps\dependencies\lora\sender.py | python .\OnBoardNode\gateway\receiver\_sim.py

**Première exécution — ACCEPTED**

*(La trame est acceptée et les données GPS sont correctement affichées.)*

A black screen with colorful text

AI-generated content may be incorrect.

**Seconde exécution avec le même CTR — REJECTED**

*(La trame est rejetée car le compteur CTR n’a pas augmenté, prouvant le mécanisme anti-rejeu.)*

A screen shot of a computer program

AI-generated content may be incorrect.

# 2. Fichiers créés / modifiés

- OnBoardNode/apps/dependencies/lora/sender.py  
- OnBoardNode/gateway/crypto.py  
- OnBoardNode/gateway/replay\_guard.py  
- OnBoardNode/gateway/receiver\_sim.py  
- OnBoardNode/gateway/keys.json  
- OnBoardNode/tools/gen\_frame.py (utilisé)

# 3. Conclusion

Les étapes ci-dessus montrent que la logique de chiffrement AES-GCM, le format des trames, et la protection anti-rejeu sont fonctionnels localement. Ces scripts pourront ensuite être exécutés sur le Raspberry Pi (drone) et la passerelle Heltec pour établir la communication sécurisée réelle.