Cryptographie **I**ndustrielle **A**vancée // TP 3

Benchmark de Signatures

Loïc Rouquette

7 octobre 2025

Introduction

Ce projet en binôme consiste à bâtir un prototype de mise à jour logicielle (OTA) pour l'IIoT, sécurisé par une **signature hybride** combinant un algorithme classique (ECDSA) et un post-quantique (Dilithium). Le prototype servira ensuite de plateforme pour analyser le coût et les avantages de cette approche.

L'objectif d'une signature hybride est de garantir la sécurité à la fois contre les attaquants actuels (grâce à ECDSA, un algorithme éprouvé) et les futurs ordinateurs quantiques (grâce à Dilithium). La mise à jour n'est valide uniquement si **les deux signatures sont valides**.

Environnement technique.

Pour les deux parties du TP, je recommande les biblothèques suivantes :

- `cryptography` (pour RSA, ECDSA, AES-GCM) ;
- `oqs-python` (pour Dilithium, Kyber) ;
- `matplotlib` & `pandas` (pour l'analyse) ;
- `request` & `http.server` (pour la communication).

Séance 1 : Construction du Prototype de Mise à Jour Sécurisée

L'objectif est de développer un prototype fonctionnel de la chaîne de mise à jour sécurisée de bout en bout en utilisant directement le mécanisme de double signature.

1. Squelette de la Communication

- **Serveur** (`server.py`) : Mettez en place un serveur HTTP de base qui peut servir un fichier binaire `firmware.bin` (créez-en un de 1 Mo avec des données aléatoires).
- **Client** (`client.py`) : Écrivez un client capable de se connecter au serveur, de télécharger le fichier `firmware.bin` et de le sauvegarder localement.

2. Intégration de l'Authenticité Hybride

"Provisioning" des clés (`provision.py`) :

Écrivez un script qui genère et sauvegarde **deux paires de clés :**

- Une paire **ECDSA** (`secp256r1`).
- Une paire **CRYSTALS-Dilithium** (`Dilithium2`).

Côté Serveur (Double Signature) :

- Le serveur charge ses **deux clés privées** (ECDSA et Dilithium).
- Pour un `firmware.bin` donné, le serveur calcule **deux signatures distinctes** : `sig_ecdsa` et `sig_pqc`.
- Il **concatène** ces deux signatures pour former une unique signature hybrique.
- Le serveur expose une route `/hybrid_signature` qui renvoie ce bloc de données binaire.

Côté Client (Double Vérification) :

- Le client charge les **deux clés publiques du serveur**.
- Après avoir téléchargé le firmware et la signature hybride, le client doit d'abord séparer la signature hybride en ses deux composantes.

> **Astuce**
>
> La taille de la signature ECDSA pour `secp256r1` est fixe, ce qui facilite la séparation.

- Le client doit ensuite effectuer **deux vérifications :**
  - Vérifier `sig_ecdsa` avec la clé publique ECDSA.
  - Vérifier `sig_pqc` avec la clé Dilithium.
  - Le client affiche `[v] SIGNATURE HYBRIDE VALIDE` **uniquement si les deux vérifications réussissent**. Si l'un ou l'autre échoue, il doit afficher `[!] ALERTE : SIGNATURE HYBRIDE INVALIDE !`.

3. Sécurisation du Canal (Confidentialité).

Le firmware peut contenir des données sensibles. Vous allez maintenant chiffrer son transfert.

1. **Protocole d'Échange de Clés :** Implémentez un échange de clés basé sur le KEM post-quantique **CRYSTALS-Kyber** (`Kyber512`).
   - Le client initie la communication en envoyant sa clé publique Kyber au serveur.
   - Le serveur génère un secret partagé et le renvoie encapsulé ("ciphertext").
   - Le client décapsule le "ciphertext" pour retrouver le même secret partagé.
2. **Chiffrement du Transfert :**
   - Une fois le secret établi, le serveur chiffre le `firmware.bin` en utilisant **AES-256-GCM**.
   - Le client reçoit le firmware chiffré, le déchiffre avec le secret partagé, puis procède à la vérification de la signature ECDSA comme à l'étape précédente.

> **Attention**
>
> À la fin de cette séance, vous devez avoir un prototype complet et fonctionnel qui garantit l'authenticité et la confidentialité de la mise à jour.

Séance 2 : Benchmarking de Stratégies de Signature

L'objectif est d'utiliser le prototype pour comparer la performance de l'approche hybride par rapport aux approches à signature unique.

4. Refactoring pour la Crypto-Agilité

Votre prototype actuel est codé en dur pour utiliser l'approche hybride. Modifiez-le pour supporter différentes **stratégies de signature**.

1. **Abstraction des Stratégies :** Refactorez votre code (serveur et client) pour qu'il puisse opérer selon trois modes, sélectionnables par un simple paramètre de configuration :
   - `ecdsa_only` : Seule la signature **ECDSA** est générée et vérifiée.
   - `pqc_only=<SCHEME>` : Seule la signature **Dilithium**, **Falcon** ou **SPHINCS+** est générée et vérifiée.
   - `hybrid=<SCHEME>` : Les deux signatures sont générées et vérifiées (comportement par défaut du TP1.)
2. Cette modularité vous permettra de lancer des tests comparatifs sans avoir à modifier le code pour chaque cas.

5. Implémentation du Protocole de Mesure

1. **Script de Benchmark (`benchmark.py`) :** Ce script orchestrera le lancement du serveur et du client pour chaque stratégie de signature.
2. Boucle d'Expérimentation : Le script doit boucler sur les différentes stratégies. Pour chaque stratégie :
   - Il exécute le scénario de mise à jour complet **100 fois**.
   - **Collecte des Matriques :**
     - Temps de signature : Le temps total pour générer la ou les signatures sur le serveur.
     - Temps de vérification : Le temps total pour vérifier la ou les signatures sur le client.
     - Taille de la signature : La taille totale de la signature envoyée sur le réseau.

> **Astuce**
>
> Votre script doit calculer la moyenne de ces métriques pour qu'il soit possible de les traiter. Cependant, avoir 100 résultats est plus qualitatif que d'avoir uniquement une moyenne. Cherchez comment exploiter cette information supplémentaire dans vos graphiques.

6. Analyse de l'Approche Hybride

1. **Visualisation :** Créez des graphiques comparant les trois stratégies sur les axes du temps d'exécution et de la taille des données.
2. **Analyse Critique :** Rédigez la conclusion de votre rapport. L'analyse ne doit plus seulement comparer les algorithmes entre eux, mais évaluer la stratégie hybride. Répondez aux questions suivantes en vous basant sur vos données :
   - Quel est le surcoût (en latence de vérification et en bande passante) de l'implémentation d'une signature hybride par rapport à une solution purement classique (ECDSA) ?

— Comment ce surcoût se compare-t-il à une solution purement post-quantique (Dilithium) ?
— Dans le contexte d'un appareil IIoT aux ressources limitées, ce surcoût est-il un **compromis acceptable** pour garantir la sécurité pendant la période de transition vers le PQC ? Argumentez votre réponse.

> **Important**
>
> Rédigez la conclusion finale de votre rapport en vous appuyant sur les données collectées via **votre prototype**. Discutez des compromis et fournissez une recommandation argumentée pour un déploiement IIoT, en considérant les contraintes de performance (CPU du client), de bande passante (réseau) et de sécurité à long terme (résistance post-quantique).

> **Attention**
>
> Votre conclusion ne doit pas être "X est le meilleur", mais plutôt une discussion argumentée des compromis. Par exemple : "Pour une vitesse de vérification optimale, l'algorithme Y est le choix évident. Cependant, la taille de sa signature le rend prohibitif pour notre réseau LoRaWAN. Une solution alternative serait Z, qui offre une signature compacte au détriment d'un temps de vérification 5 fois plus long, ce qui reste acceptable au vu de notre cahier des charges..."

Livrables Attendus (par binôme)

Une archive `.zip` unique contenant :

1. **Code Source Complet :**
   — Le code final du prototype modulaire (server.py, client.py, provision.py).
   — Le script d'orchestration du benchmark benchmark.py.
   — Le README.md et le requirements.txt.
2. **Rapport d'Analyse (PDF - 4 pages maximum) :**
   — **Partie 1 : Architecture du Prototype Hybride**
     — Description de l'architecture et du protocole de double vérification.
     — Discussion sur la modularité implémentée pour permettre la crypto-agilité.
   — **Partie 2 : Analyse de Performance des Stratégies de Signature**
     — Le tableau complet des mesures comparant les trois stratégies.
     — Les graphiques de visualisation.
     — **L'analyse critique sur le coût et la pertinence de la signature hybride** dans un contexte IIoT, en s'appuyant sur les données mesurées.