OPC UA, un protocole sécurisé pour l'automatisme industriel

Mise en œuvre d'un serveur OPC UA sécurisé et de sa supervision

La Revue 3E.I
Culture Sciences de l'Ingénieur

Édité le 05/06/2024

école normale supérieure paris-saclay

Louis LALAY[1] – Anthony JUTON[2]

[1] *Étudiant agrégé de sciences de l'ingénieur, doctorant à Télécom Paris*

[2] *Professeur agrégé, ENS Paris Saclay, DER Nikola Tesla*

Cette ressource fait partie du N°112 de La Revue 3EI du 2ème trimestre 2024.

L'objectif de cette ressource est de présenter une application pratique de OPC UA, utilisant un nano-ordinateur raspberry Pi 4 bon marché (90 euros), pour simuler un système de château d'eau dialoguant en OPC UA avec le superviseur. Ce dispositif facile à dupliquer peut alors être un support pour des travaux pratiques autour du protocole OPC UA (supervision, étude des mécanismes de sécurisation du protocole). L'objectif étant l'étude d'OPC UA en tant que protocole sécurisé d'automatisme industriel, nous avons limité au maximum le besoin de connaissance de Linux et de python.

En cas de difficulté ou d'éléments peu clairs dans la ressource, il est possible d'échanger sur la liste https://groupes.renater.fr/sympa/info/revue3ei *pour résoudre les problèmes et améliorer la ressource.*

Pour travailler sur une seule machine (ce qui nous semble moins pédagogique), il est possible de remplacer le nano-ordinateur au choix :

- Par une machine virtuelle (de préférence Linux pour pouvoir utiliser OpenSSL),
- En exécutant les scripts python directement sur le système d'exploitation Windows où fonctionne Panorama

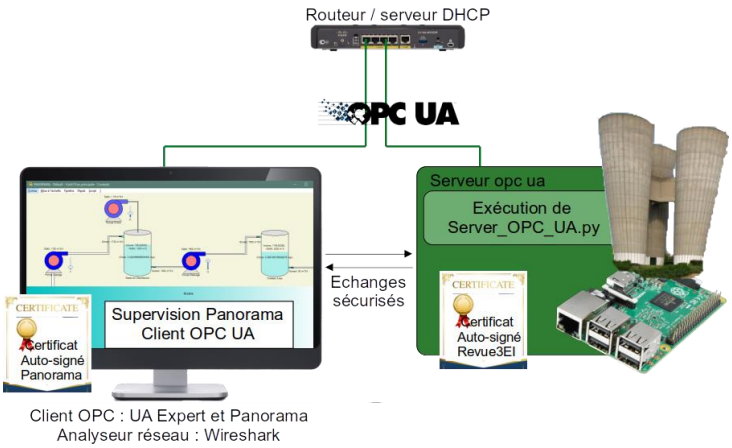

Figure 1 : Schéma synoptique de l'installation simulée présentée dans cette ressource

La ressource commence par une rapide présentation d'OPC UA et de son fonctionnement pour pouvoir aborder ensuite la mise en œuvre dans le cadre de l'activité pratique présentée. Cette présentation peut être avantageusement complétée par les vidéos d'Hervé Discours [6]. Le lecteur pourra ensuite approfondir avec les supports mis à disposition par OPC Uacademic [2].

1 - Présentation de OPC UA

1.1 - Histoire de la fondation [1]

Dans les années 90, Microsoft COM et DCOM[1] dominent la communication industrielle. En 1995, Fisher-Rosemount, Intellution, Opto 22 et Rockwell Software s'associent pour créer un standard de communication basé sur COM et DCOM nommé OPC, raccourcis de OLE[2] for Process Control. La fondation OPC (opcfoundation.org) est officiellement créée en 1996. Fin 1996, OPC-DA[3], une version simplifiée des spécifications OPC voit le jour : c'est la version qui a rendu OPC populaire. Cette version se base sur des protocoles Microsoft, et n'est donc supportée que par Windows.

En 2006, OPC UA[4] est créé afin de devenir indépendant de la plateforme hôte. Cette version est donc compatible avec plus de services. L'ancienne version est alors appelée OPC Classic.

Depuis, la fondation a surtout mis à jour ces deux protocoles et créé des liens avec différents industriels. En effet, un organisme peut intégrer la fondation pour faciliter la mise en œuvre des spécifications OPC à son propre matériel. La fondation OPC compte actuellement 850 membres, dont tous les acteurs importants de l'automatisme industriel (Siemens, Rockwell Automation, Wago, B&R), de la supervision (Arc Informatique, Codra, …), de la robotique industrielle (Fanuc, Staubli, Omron, …), de la production d'électricité (ABB, Alstom, General Electric, Schneider Electric, …), et même des fabricants de composants (Microchip, ST Microelectronics, ...)

Aujourd'hui, OPC signifie Open Plateform Communications. Dans la suite, on s'intéressera uniquement à OPC UA, dans un contexte multiplateforme et sécurisé.

1.2 - Domaines d'utilisation principaux

Le protocole OPC UA est surtout utilisé en automatisme, pour la communication entre un superviseur et des automates, ou pour la communication des automates entre eux. OPC UA permet la communication sécurisée de données, avec une multitude de services utiles à la supervision d'installations industrielles, par exemple :

- Alarmes,
- Horodatage,
- Historisation,
- etc.

Intérêt majeur de OPC UA par rapport aux protocoles de supervision traditionnels (Modbus TCP, BACnet Profinet, …), la communication se fait au travers de sessions ouvertes entre des clients et des serveurs et sécurisées par le populaire protocole SSL. Le client (un logiciel de supervision par exemple) envoie des requêtes aux serveurs (des automates par exemple) pour obtenir des informations du processus en cours et/ou pour donner des consignes afin de modifier ce processus.

[1] Distributed Component Object Model
[2] Microsoft Object Linking & Embedding
[3] OPC for Data Acces
[4] OPC Unified Architecture

1.3 - Rôle de la fondation OPC

Figure 2 : Logo de la fondation OPC

La fondation OPC est en charge du développement et du maintien des spécifications du protocole. C'est ensuite aux constructeurs de mettre en œuvre ces spécifications pour créer un serveur OPC UA ou un client OPC UA. Par exemple, Siemens implémente un serveur OPC UA sur ses automates S7-1200. De la même manière, Codra, entreprise développant le logiciel de supervision Panorama, implémente un client OPC UA sur Panorama, ce qui permet ainsi la supervision sécurisée des automates Siemens. Siemens et Codra faisant partie de la fondation OPC, leurs implémentations du protocole est validée par la fondation.

2 - Fonctionnement de OPC UA

Cette partie présente différentes possibilités offertes par OPC UA et leur fonctionnement.

2.1 - Session de communication

On s'intéresse ici à la structure de la communication OPC UA, le support utilisé, les différentes couches et leur rôle.

OPC UA utilise les couches TCP/IP (couches 4 et 3) sur des couches liaison de données / physique Ethernet ou Wifi (couches 2 et 1). Une fois la connexion TCP établie, une session OPC UA est ouverte au niveau de la couche 5 (voir Figure 3). Les interactions entre les clients et les serveurs requièrent un modèle à état. Les informations de l'état sont définies dans la session, qui est une connexion logique entre les clients et les serveurs. Les sessions sont indépendantes des protocoles de communication sous-jacents, et ne sont fermées que sur une requête de fermeture ou l'inactivité d'un client. Dans une session, on retrouve par exemple des détails sur les utilisateurs (nom d'utilisateur, mot de passe, autorisations, etc.), le mode de communication (Publisher/Subscriber ou Request/Response), etc.

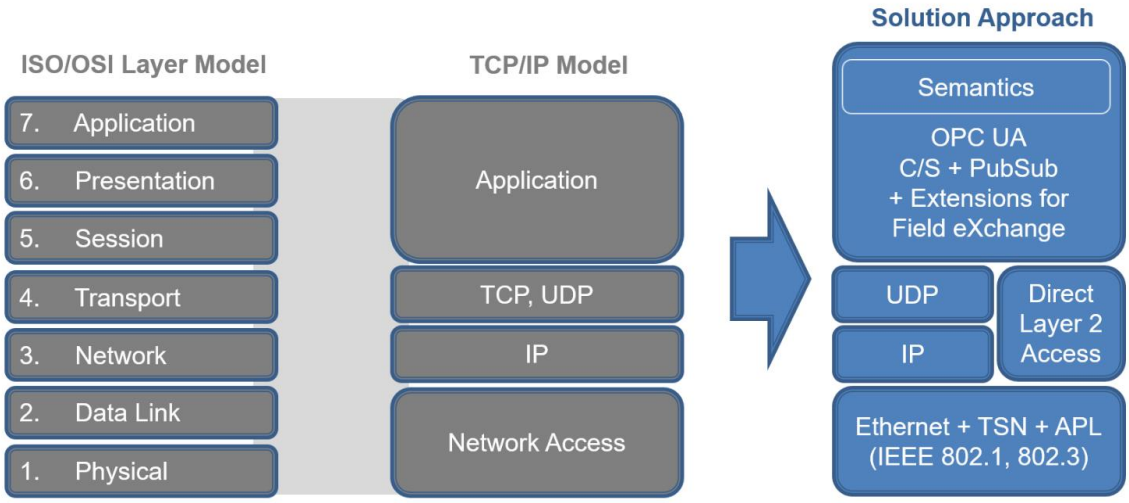

Figure 3 : Couches réseaux OPC UA (source OPC UAcademic)

2.2 - Sécurité

La sécurité est un des points forts de OPC UA, comparé aux anciens protocoles de supervision (Modbus par exemple). La sécurisation des communications en OPC UA s'appuie sur OpenSSL, comme HTTPs (cf ressource [13]).

Le service de sécurisation est indépendant du fonctionnement de l'application OPC UA, ce service de sécurisation reposant sur la *Communication Stack*. La *Communication Stack* communique via un Secure Channel. Un Secure Channel est une connexion logique longue durée entre un client et un serveur. Il est mis en place par un échange de certificats X.509 (client et serveur) et de clés publiques (clé client et clé serveur) aboutissant à un échange de clés de chiffrement symétrique permettant le chiffrement des échanges suivants, jusqu'à la fermeture du Secure Channel. Une application OPC UA ignore tout message qui ne suit pas la politique de sécurité. Une session est associée à un unique *Secure Channel*. Lors d'une connexion, le *Secure Channel* est d'abord ouvert, puis la session est ensuite ouverte en utilisant ce canal sécurisé, comme cela sera présenté dans la partie pratique un peu plus loin dans cette ressource.

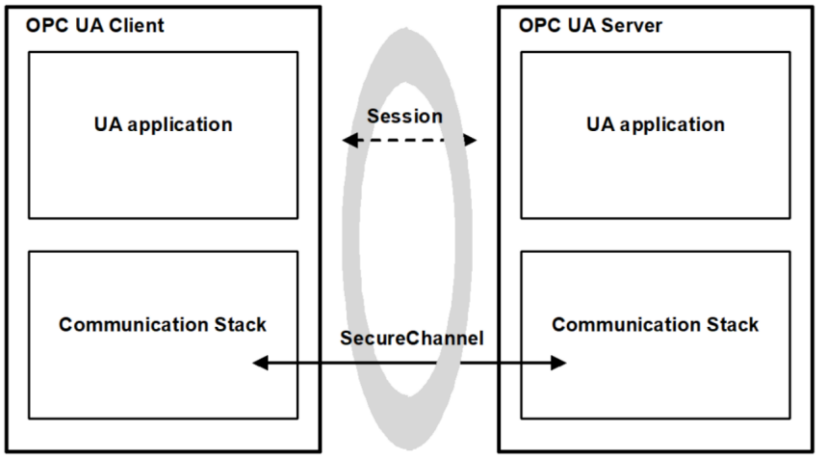

Figure 4 : Sécurité (source opcfoundation.org)

Pour les échanges de données via le *Secure Channel*, il existe 3 niveaux de sécurité :

- Aucun
- Avec chiffrement
- Avec chiffrement et authentification

2.3 - Discovery

Le service *Discovery* est décrit en détail dans la Partie 12.4 de la référence OPC UA. Ce service permet aux applications OPC UA de rechercher d'autres applications. En général, ce sont les serveurs qui proposent ce service afin que des clients s'y connectent, mais certains clients peuvent avoir une connexion inversée.

Avant même de créer une session, le service de *Discovery* permet d'identifier un serveur. Cela permet aussi d'avoir une vue d'ensemble de l'application, avec par exemple les variables qu'elle propose. Les variables étant désignées par un nom explicite, le développeur évite ainsi les erreurs d'adressage ou d'unité, typiques de modbus (on y désigne l'adresse de la variable et on récupère sa valeur brute, sans unité).

2.4 - Espace d'adresses

Une fois que la connexion est établie et sécurisée, il est possible d'accéder à l'espace d'adresses et de commencer à échanger des messages.

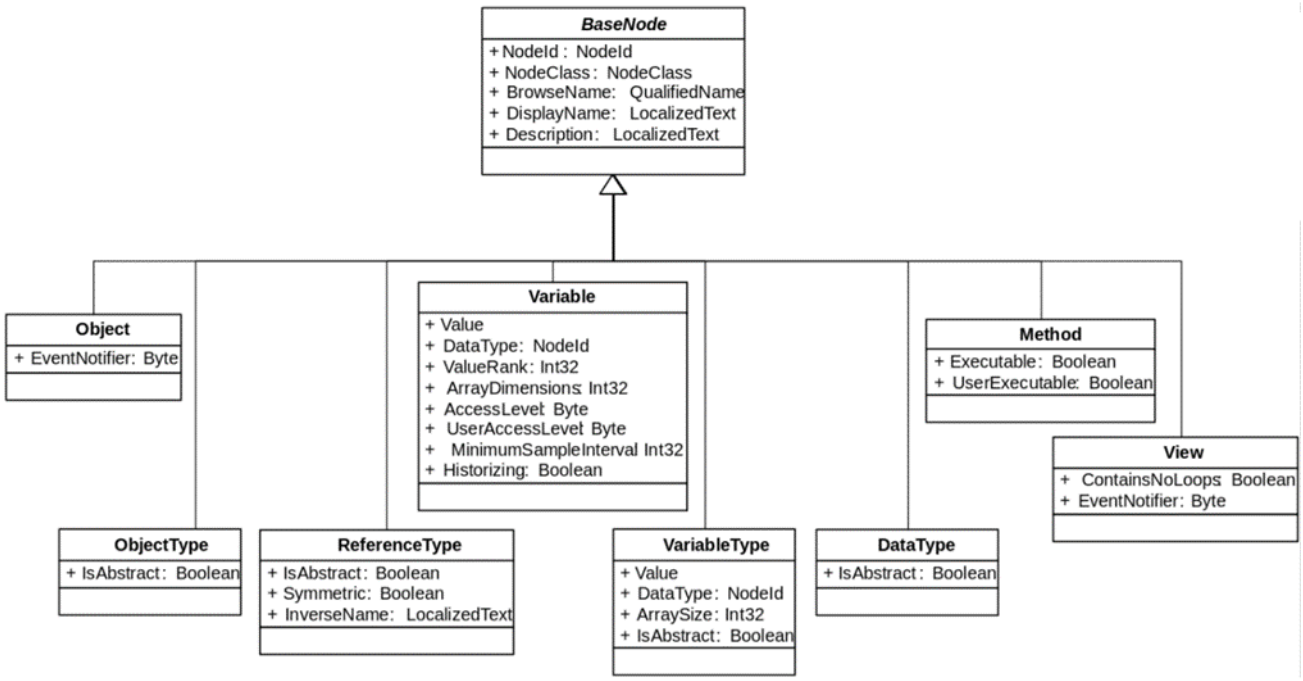

Figure 5 : Address Space (Source OPC UAcademic)

L'espace d'adresses définit comment est organisée une application OPC UA. C'est ici que se trouvent les variables par exemple. Chaque élément de l'espace d'adresse est un nœud, qui peut être une des 8 classes présentées sur la figure 5 ci-dessus et définies comme suit :

- **Object** : Utilisé pour représenter des systèmes, des composants, des objets réels, etc. Les objets sont liés entre eux par des références ;
- **ObjectType** : Définition pour les objets ;
- **ReferenceType** : Définition pour les références ;
- **Variable** : Stocke des données pour un objet ;
- **VariableType** : Définition pour les variables ;
- **DataType** : Type pour les données ;
- **Method** : Fonction ;
- **View** : Définit un sous ensemble de l'espace d'adresses. La vue par défaut est l'espace d'adresses en entier.

Un exemple d'espace d'adresses simple est proposé sur la figure 6. La racine de l'espace d'adresse est l'URL du serveur. On a ensuite (à la fin de la liste ici) un objet présentant les informations relatives au serveur (mode de communication, type de sécurité, etc.) puis les objets liés au process, avec des valeurs associées. L'espace d'adresses se construit à partir de l'installation réelle, en encapsulant les variables dans des objets. On peut ensuite encapsuler les objets dans d'autres objets plus grands. Les *View* permettent ensuite de sélectionner une partie de l'arborescence.

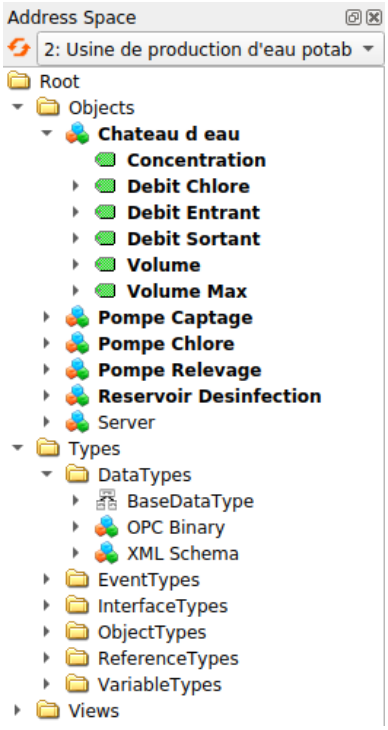

Figure 6 : Espace d'adresse de l'application château d'eau

2.5 - Modes de communication

Il existe deux modes de communication entre les applications. Les deux formes peuvent être adoptées et vont dépendre des cas d'utilisation. Dans les deux types de communication, les variables sont horodatées. La figure 7 présente les deux modes de communication.

Client/Server : Ce mode de communication est établi entre un client et un serveur. Lors de l'ouverture de la session, le serveur met en place un abonnement du client sur les variables qui l'intéressent. La session restant ouverte, le client n'est notifié que lorsque les valeurs ont changé. Le temps de rafraîchissement des valeurs côté serveur est réglable.

Publication/Subscription : Ce mode de communication met en relation un éditeur et des abonnés. Les applications n'échangent pas les messages directement, mais passent par un intermédiaire. Les *Publishers* envoient leurs données à l'intermédiaire sans savoir s'il y a des abonnés. De leur côté, les abonnés signalent à l'intermédiaire qu'ils sont intéressés par une donnée, sans savoir si la donnée est encore mise à jour. Lorsqu'une donnée est modifiée (et uniquement lorsqu'elle est modifiée), les abonnés sont notifiés du changement et peuvent récupérer la donnée à jour. Cela permet à plusieurs clients de s'abonner aux mêmes données sans ouvrir de session supplémentaire.

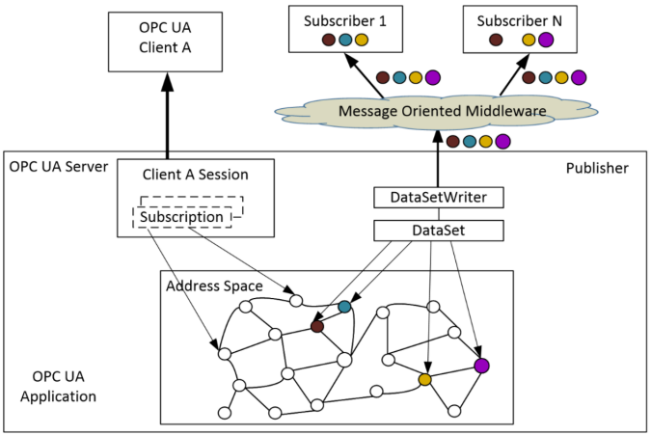

Figure 7 : Modes de communication (source Partie 1 6.6)

2.6 - Alarmes

La Partie 9 de la référence OPC UA explicite le fonctionnement des alarmes dans leur détail.

Les applications OPC UA permettent la création d'alarmes, et donc de simplifier la supervision de processus. Une alarme se déclenche par exemple lorsqu'une mesure sort d'une plage de valeur définie et alerte l'opérateur (via le logiciel de supervision).

L'application présentée ici en exemple gagnerait à utiliser les alarmes pour indiquer par exemple un débordement du château d'eau.

3 - Mise en œuvre du serveur OPC UA / château d'eau

Dans cette partie, on s'intéresse à la mise en œuvre d'un serveur OPC UA simulant un château d'eau, en 3 étapes.

- Démarrage d'un serveur OPC UA FreeOpcUa, qui implémente une grande partie des fonctionnalités OPC UA en langage Python (et en C++, inutilisé ici). Il n'est pas nécessaire de maîtriser python pour mettre en place ce qui suit. Le Client OPC UA est le logiciel gratuit UAExpert.
- Sécurisation de la connexion, toujours avec le serveur OPC UA FreeOpcUa et le client UAExpert
- Supervision avec le logiciel Panorama, client OPC UA, du serveur OPC UA simulant un château d'eau. Ce serveur utilise là encore FreeOpcUa.

3.1 - OPC UA avec Python

FreeOpcUa est un projet open source et ne fournit donc aucune garantie, mais il est très complet pour en faire un projet de démonstration sur un nano-ordinateur comme une carte Raspberry Pi. FreeOpcUa implémente les fonctionnalités essentielles de OPC UA, notamment la sécurisation des échanges.

3.2 - Installation du serveur

On configure un serveur OPC UA en langage Python. L'objectif est d'implémenter les fonctionnalités de base du serveur, afin de pouvoir observer les trames avec des outils tels que Wireshark, pour mettre en évidence les échanges, non sécurisés pour l'instant, via OPC UA.

Installation du système d'exploitation sur le nano-ordinateur

Pour installer le système d'exploitation Raspberry OS sur le nano-ordinateur raspberry Pi 4 et l'accès à distance, on peut suivre le document référencé ici [8]. On obtient ainsi un nano-ordinateur avec un système d'exploitation linux dont le bureau à distance est accessible via VNC Viewer. Il est aussi possible d'utiliser une machine virtuelle ou de faire tourner le serveur directement sur l'OS du client.

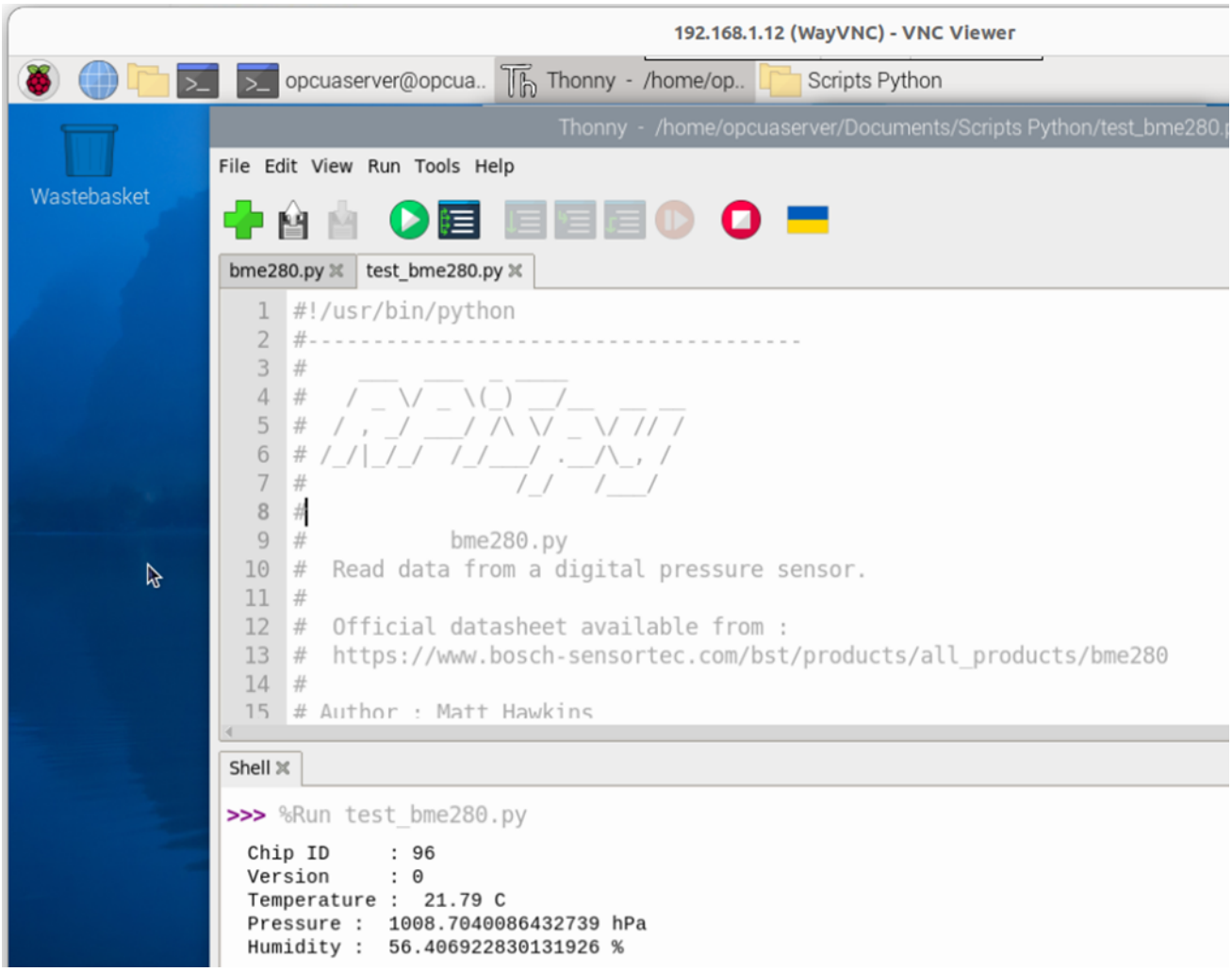

Figure 8 : Exécution d'un programme test_bme280.py via l'accès au bureau à distance du nano-ordinateur Raspberry Pi 4 par VNC Viewer

Installation du module serveur OPC UA

On installe tout d'abord le module python Free OPC-UA nommé **asyncua**. Le nano-ordinateur ne servant que de serveur OPC UA, on ne s'encombre pas du système d'environnements virtuels python, d'où l'option --break-system-packages.

```
pip install asyncua --break-system-packages
```

3.3 - Test simple du serveur OPC UA

Pour commencer, on teste sur une configuration simple :

- Pas de sécurité sur le serveur (prog *test_server_OPC_UA.py*)
- Juste une variable qui s'incrémente sur le serveur,
- Un client tout prêt : UA Expert,
- L'analyseur de réseau Wireshark pour observer les échanges.

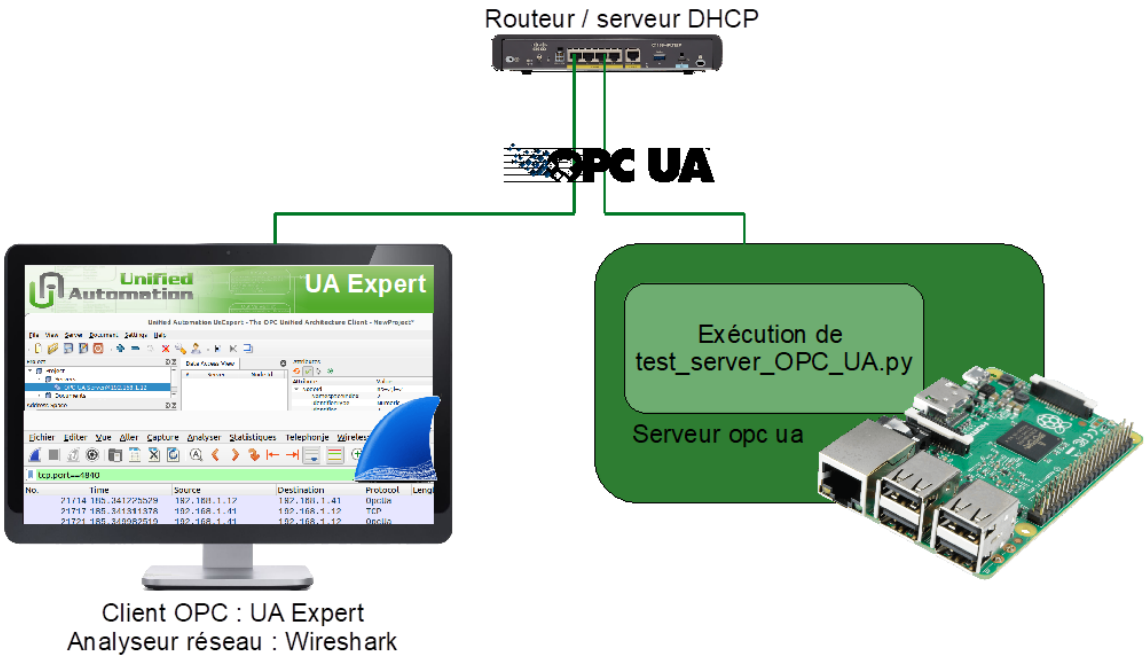

Figure 9 : Première application avec un serveur OPC UA simple, le client UA Expert et Wireshark

Côté serveur (Raspberry Pi 4 avec le module python asyncua)

Le programme du serveur *test_server_OPC_UA.py* est fourni avec cette ressource. Attention, l'adresse IP indiqué ligne 9 doit être l'adresse IP du serveur (donc de la raspberry Pi) :

```
#Programme de test du serveur OPC UA
from opcua import Server, ua
import time

def main():
    # Instanciation du serveur
    server = Server()
    # URL de la Raspberrypi qui hÃ©berge le serveur
    server.set_endpoint("opc.tcp://192.168.1.12:4840/UA/SampleServer")
    # nom du serveur
    server.set_server_name("OPC-UA-Server")
    # Sécurité (rien sur le programme de test)
    server.set_security_policy([ua.SecurityPolicyType.NoSecurity])
    # Nom de l'espace d'adresse
    name = "progDeTest_OPCUA"
    noeud_test = server.register_namespace(name)
    # Création d'un noeud racine
    node = server.get_objects_node()
    # Ajout d'un noeud pour les variables que l'on veut partager
    monObjet    = node.add_object(noeud_test, "monObjet")
    # Création de 2 variables, dont une accessible en écriture
    maVariable1 = monObjet.add_variable(noeud_test,"variableALire",0)
    maVariable2 = monObjet.add_variable(noeud_test,"variableAEcrire",0)
    maVariable2.set_writable()
    ancienneValeur2 = maVariable2.get_value()
    valeur1 = 3.14

    # Demarrage du serveur
    server.start()
    while True:
        # Lecture de la valeur de la variable accessible en écriture
        nouvelleValeur2 = maVariable2.get_value()
        if nouvelleValeur2 != ancienneValeur2 :
            ancienneValeur2 = nouvelleValeur2
            print("valeur recue : ",nouvelleValeur2)
        # Incrémentation de la valeur de maVariable1
        valeur1 = valeur1 + 0.1
        maVariable1.set_value(valeur1)
        time.sleep(1)

if __name__ == "__main__":
    main()
```

On exécute le programme sur la raspberry Pi, avec l'éditeur python Thonny par exemple :

Quelques lignes de commande permettent de vérifier le bon fonctionnement du serveur OPC, directement depuis sur un terminal du nano-ordinateur Raspberry Pi, avec bien sûr l'adresse IP correcte :

- uals --url=opc.tcp://192.168.1.12:4840
- uals --url=opc.tcp://192.168.1.12:4840 --nodeid i=85
- uaread --url=opc.tcp://192.168.1.12:4840 --nodeid "ns=2;i=2"
- uaread --url=opc.tcp://192.168.1.12:4840 --path "0:Objects,2:monObjet,2:variableALire"

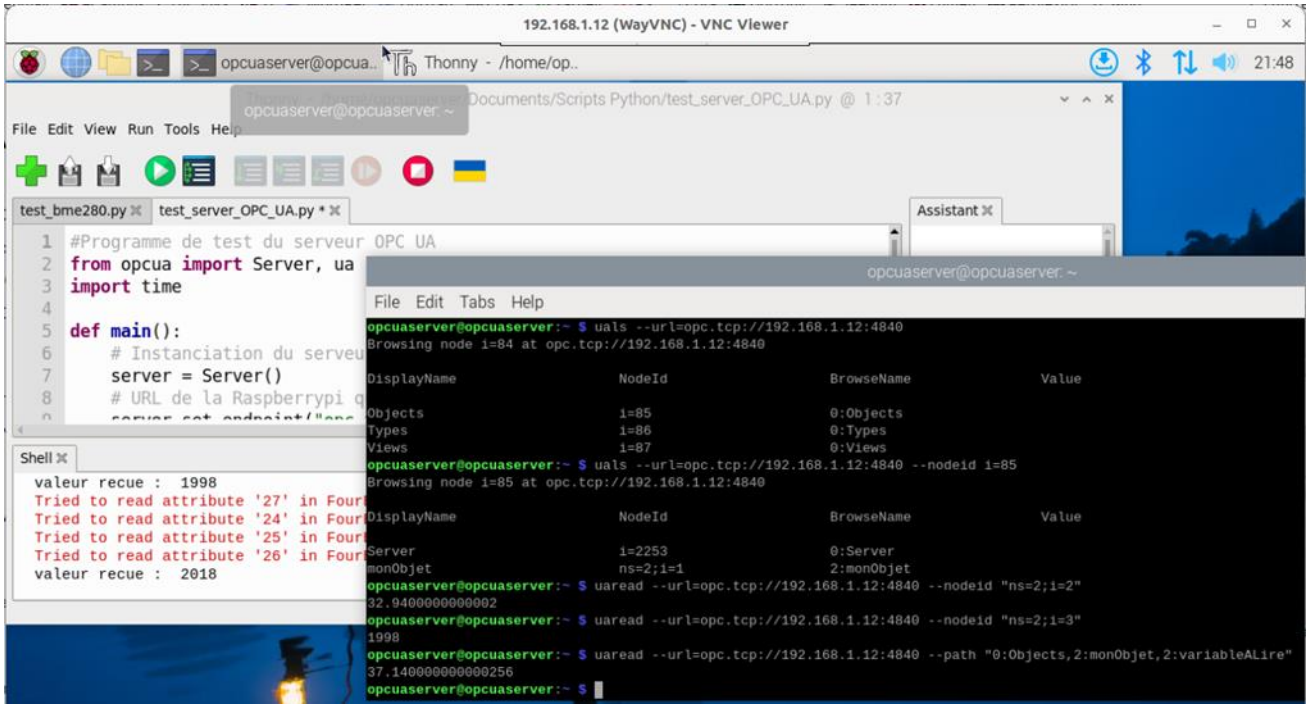

Figure 10 : Bureau du nano-ordinateur Raspberry Pi, serveur OPC UA avec l'IDE Thonny pour l'exécution du programme et la console pour les tests

Côté client (UA Expert sur le PC)

UA Expert [9] est un client OPC UA gratuit, robuste, permettant d'utiliser la plupart des fonctionnalités OPC UA.

Télécharger, installer et lancer UA Expert.

Ajouter le serveur :

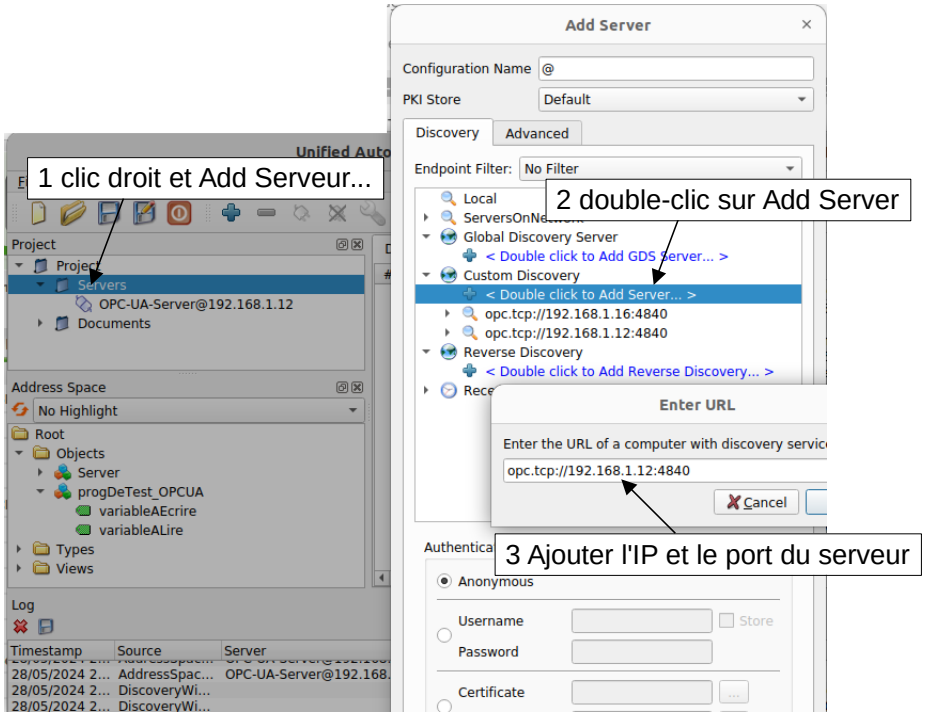

Figure 11 : Ajout d'un serveur sur le client UA Expert

Une fois le serveur dans la liste des Servers de la zone Project, un clic droit dessus permet de s'y connecter.

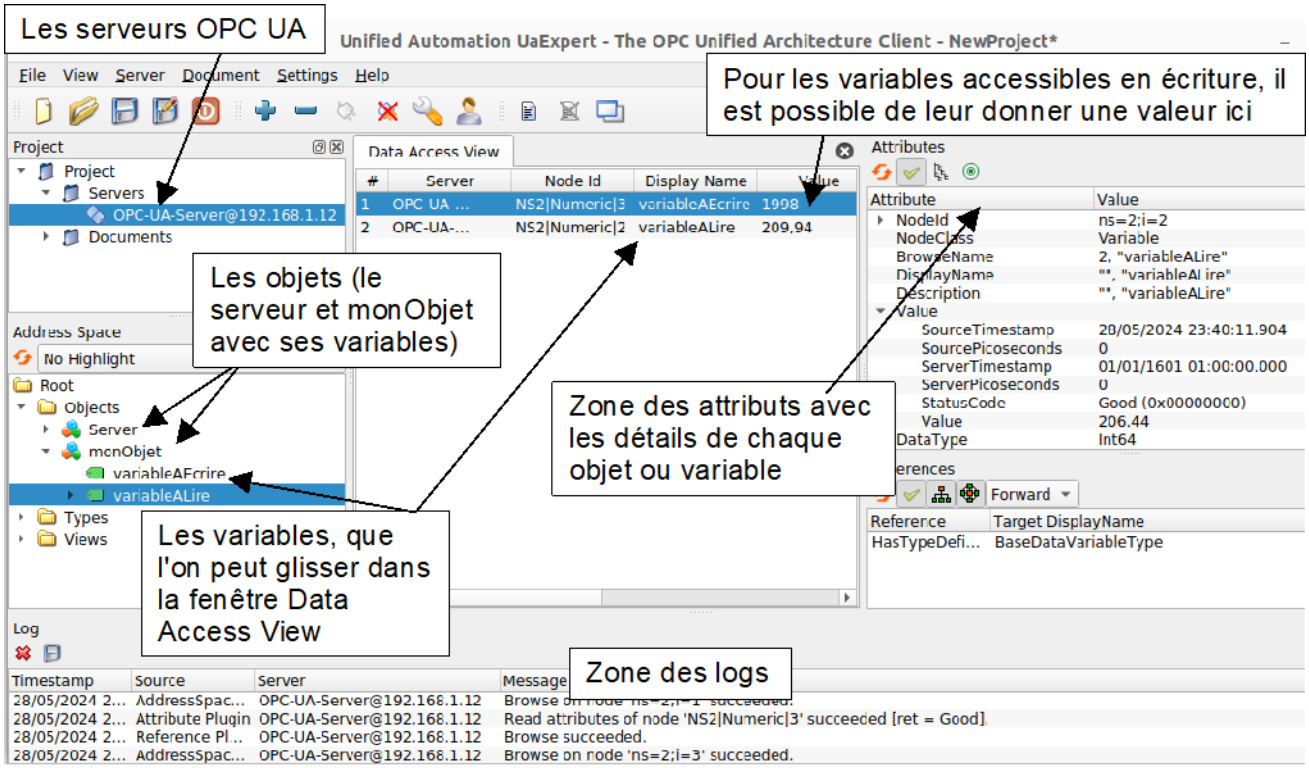

Figure 12 : Utilisation de UA Expert pour afficher et modifier les variables supervisées

Espionnage des échanges (Wireshark sur le PC)

Le logiciel analyseur de réseau Wireshark [10] permet d'observer les échanges entre le client et le serveur. Il est possible de l'installer sur le PC et/ou sur le nano-ordinateur raspberry Pi. L'espionnage de la mise en place de la connexion est intéressant :

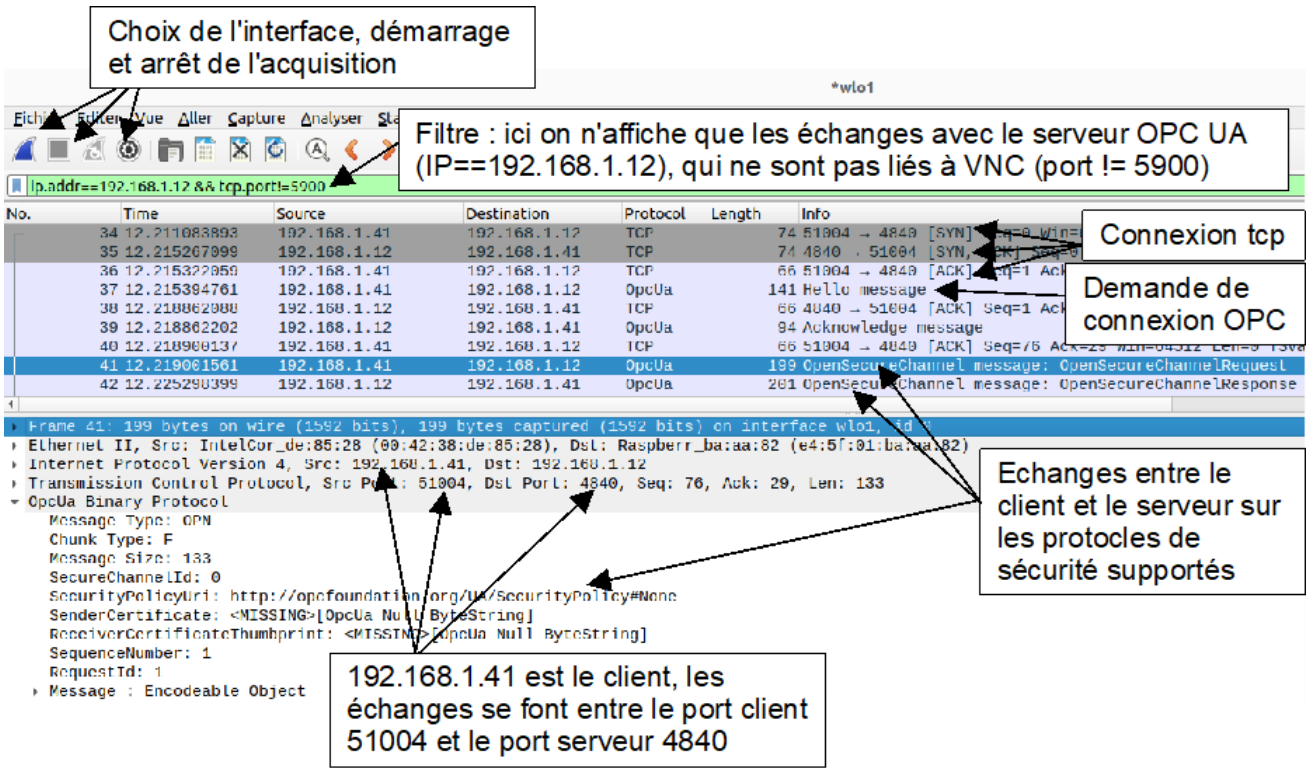

Figure 13 : Espionnage des échanges entre le client et le serveur OPC UA avec Wireshark, lors de la connexion

L'espionnage des échanges une fois la connexion établie permet de mettre en évidence le mode publisher/subscriber.

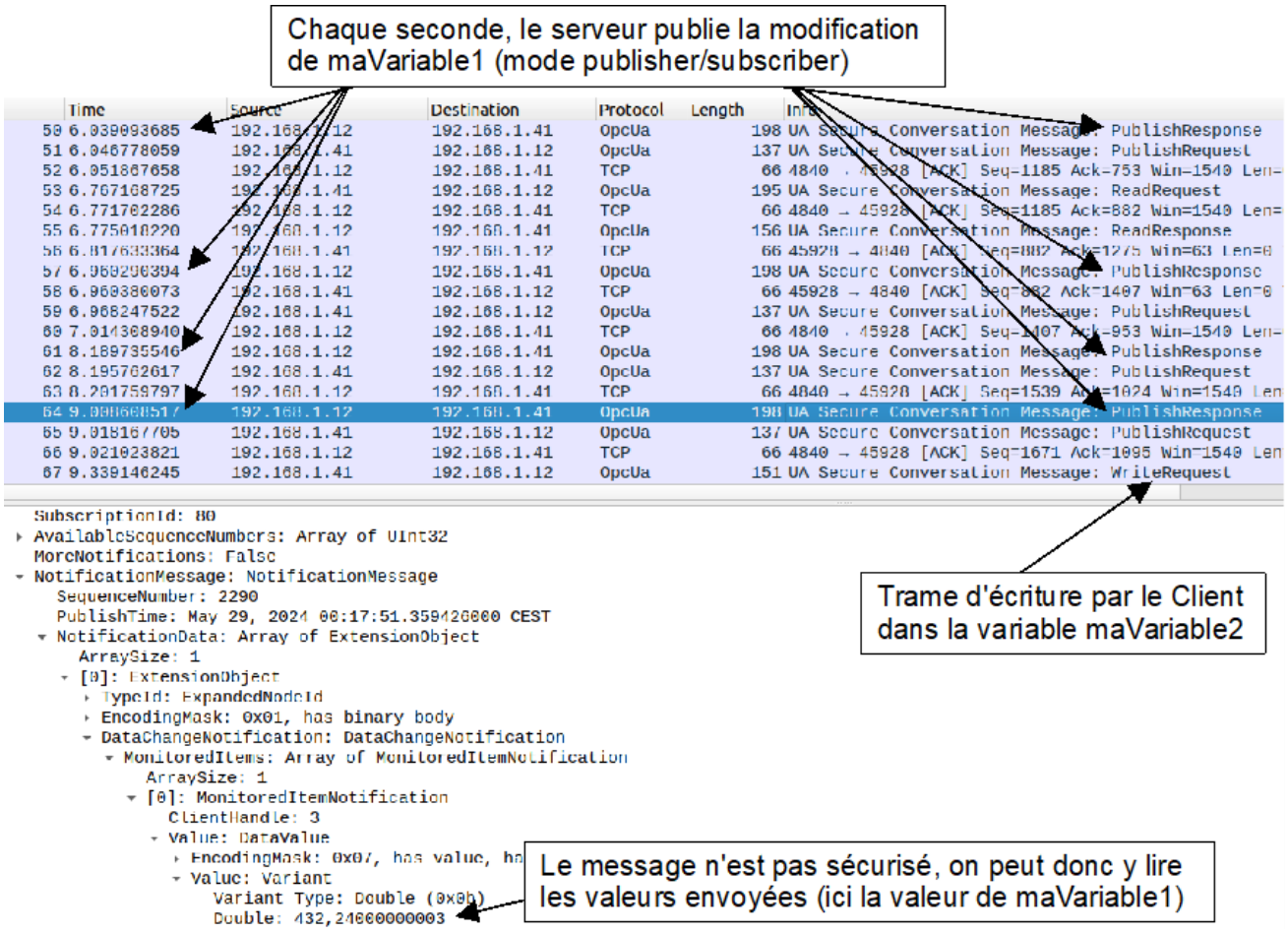

Figure 14 : Espionnage des échanges des valeurs maVariable1 (Serveur -> Client) et maVariable2 (Client->Serveur)

3.4 - Connexion sécurisée OPC UA

Dans ce 2ème exemple, on reprend l'exemple simple, en ajoutant la sécurisation de la connexion. On choisit le chiffrement des échanges et l'authentification (SignAndEncrypt).

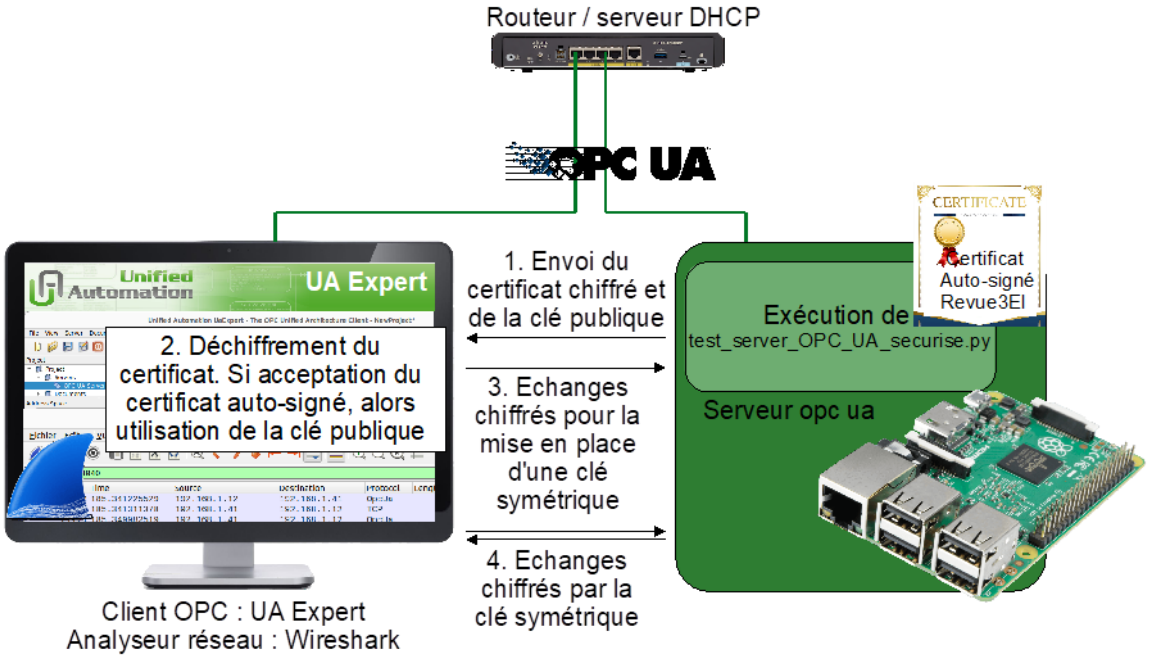

Figure 15 : Mise en place de la connexion OPC-UA sécurisée

La meilleure pratique est de privilégier l'utilisation d'une autorité de certification au lieu d'un certificat autosigné. Cependant, dans un environnement pédagogique, cela est peu accessible. L'authentification est donc basée ici sur des certificats X.509 auto-signés, associés à des clés asymétriques : le serveur envoie sa clé publique et un certificat chiffré par sa clé privée. Le client vérifie que la clé publique déchiffre bien le certificat. Il reste à approuver le certificat (une autre machine pourrait se faire passer pour le serveur), ce qui pour des certificats auto-signés se fait manuellement.

Génération des certificats

La génération de certificats auto-signés est gratuite, mais il faut approuver le certificat serveur manuellement, soit à la demande du client lors de la connexion, soit en ajoutant le certificat dans le dossier des certificats approuvés. Le serveur approuve le certificat du client (il répond à tous les clients).

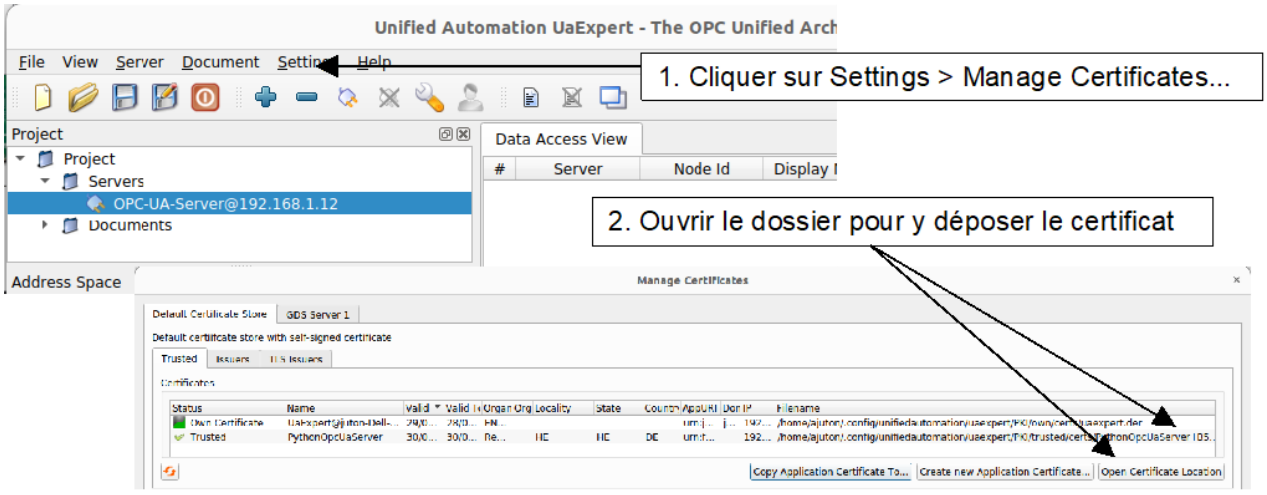

Figure 16 : Management des certificats par UAExpert, avec le dossier de dépôt des certificats

UA Expert crée lui-même le certificat client (*Own Certificate* sur la figure sous-dessous).

Pour le serveur, FreeOPCUA propose un générateur de certificats X.509 auto-signés, avec la configuration pré-remplie, à adapter quelque peu pour que le certificat soit conforme et accepté aussi bien par UAExpert que Panorama (pour la suite) :

On crée le fichier ssl.conf (fourni en pièce jointe à cette ressource). OpenSSL est installé de base sur linux.

```
[ req ]
default_bits = 2048
default_md = sha256
distinguished_name = subject
req_extensions = req_ext
x509_extensions = req_ext
string_mask = utf8only
prompt = no

[ req_ext ]
basicConstraints = CA:FALSE
nsCertType = client, server
keyUsage = nonRepudiation, digitalSignature, keyEncipherment, dataEncipherment, keyCertSign
extendedKeyUsage= serverAuth, clientAuth
nsComment = "OpenSSL Generated Certificat"
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
subjectAltName = URI:urn:freeopcua:python:server, IP: 192.168.1.12

[ subject ]
countryName = FR
stateOrProvinceName = IDF
localityName = Saclay
organizationName = Revue3EI
commonName = PythonOpcUaServer
```


Figure 17 : Fichier ssl.conf pour la génération du certificat

On génère alors la clé avec OpenSSL :

```
openssl genrsa -out key2.pem 2048
```

On génère ensuite le certificat X.509, aux formats pem (ascii) et der (binaire) :

```
openssl req -x509 -days 365 -new -out certificate2.pem -key key2.pem -config ssl.conf
openssl x509 -outform der -in certificate2.pem -out certificate2.der
```

Linux permet d'afficher le certificat X.509 généré :

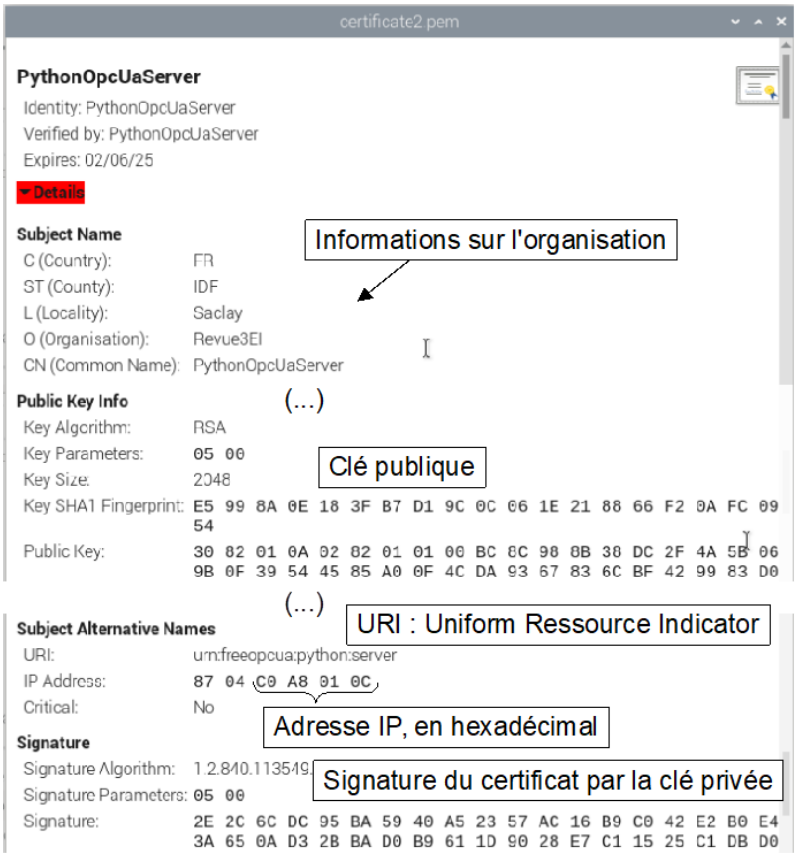

Figure 18 : Affichage du certificat généré

Démarrage du serveur

Une fois certificate2 et key2 copiés dans le dossier du fichier *test_server_OPC_UA_securise.py*, on lance le serveur, avec la sécurité activée :

test_server_OPC_UA_securise.py

```
4
5  def main():
6      # Instanciation du serveur
7      server = Server()
8      # URL de la Raspberrypi qui héberge le serveur
9      server.set_endpoint("opc.tcp://192.168.1.12:4840/UA/SampleServer")
10     # nom du serveur
11     server.set_server_name("OPC-UA-Server")
12     # Sécurité
13     server.set_security_policy([ua.SecurityPolicyType.Basic256Sha256_SignAndEncrypt])
14     server.load_certificate("certificate2.der")
15     server.load_private_key("key2.pem")
16     # Nom de l'espace d'adresse pour éviter les ambiguïtés de noms de noeuds
17     name = "progDeTest_OPCUA"
```

Figure 19 : Début du fichier test_server_OPC_UA_securise.py avec l'intégration du certificat et de la clé

Démarrage du client

Il est possible de se connecter au serveur. Celui-ci n'acceptant que les connexions sécurisées désormais, il propose son certificat au client, que celui-ci doit accepter. 2 possibilités avec UAExpert :

Soit on accepte manuellement le certificat au moment de la première connexion :

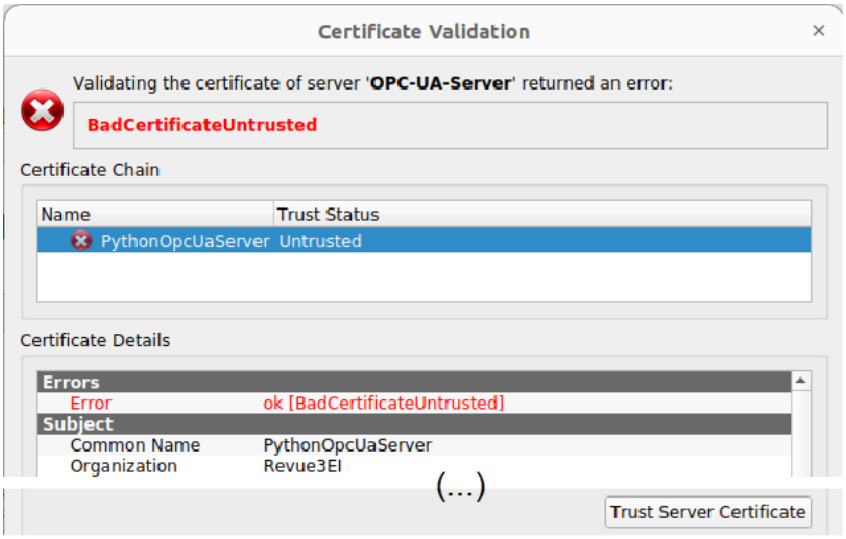

Figure 20 : Fenêtre UAExpert de demande l'acception du certificat

Il est aussi possible de copier à l'avance le certificat dans le dossier des certificats de confiance (voir **Erreur ! Source du renvoi introuvable.**).

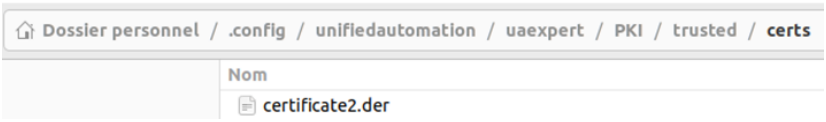

Figure 21 : Fichier certificat copié dans le dossier des certificats de confiance de UAExpert

La communication sécurisée est alors opérationnelle :

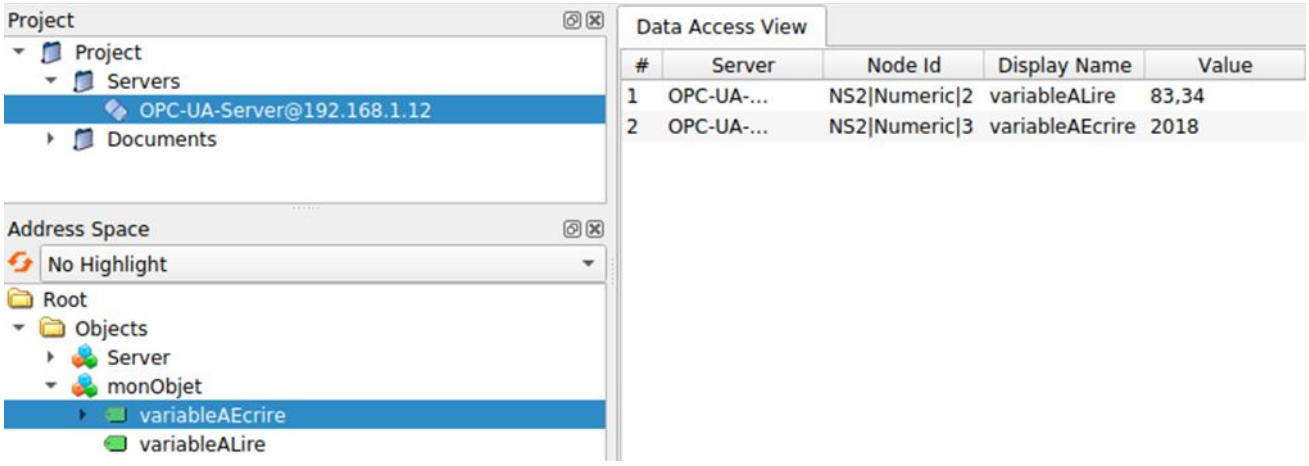

Figure 22 : Communication du client UAExpert avec serveur OPC UA

Observation des échanges OPC UA sécurisés par wireshark

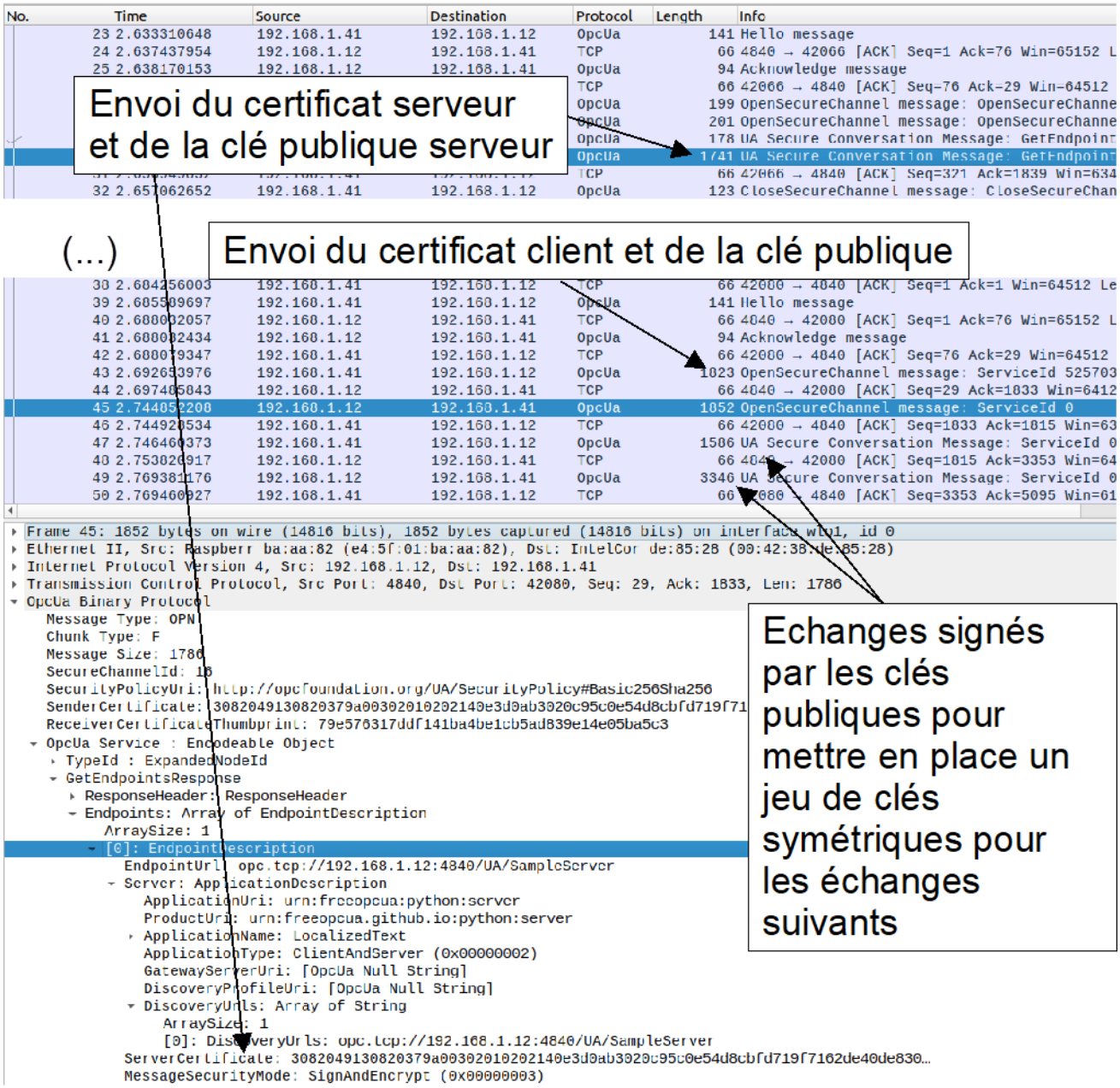

Figure 23 : Mise en place des échanges sécurisés lors de la connexion du client UA Expert au serveur OPC UA

3.5 - Supervision d'un château d'eau simulé via OPC UA

Les échanges supervisés étant en place entre UAExpert et le serveur OPC UA, il est possible de lancer le serveur OPC UA du château d'eau et le client OPC UA sur le superviseur Panorama.

Démarrage du serveur

Sur le serveur, l'ensemble des fichiers nécessaires est fourni en pièce jointe à cette ressource.

Figure 24 : Fichiers du dossier ScriptsPython, pour l'exécution du simulateur de château d'eau

Il faut lancer d'abord le fichier *simulation_usine_eau.py* qui gère le fonctionnement simulé du château d'eau. Ensuite, on exécute *server_OPC_UA.py* qui, comme son nom l'indique gère la mise à disposition des variables par OPC. Les échanges entre les 2 process (simulation_usine_eau et server_OPC_UA) se font par l'intermédiaire de 3 fichiers texte : *fichier_commande_OPC.txt*, *fichier_retour_pompes.txt* et *fichier_retour_reservoirs.txt*.

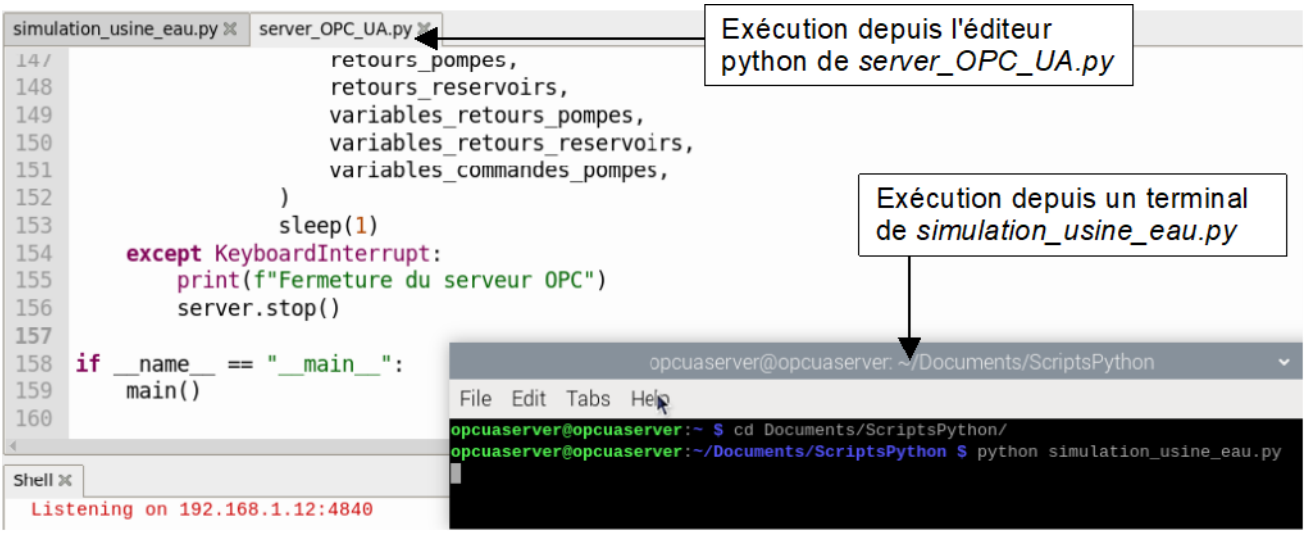

Figure 25 : Copie d'écran du serveur Raspberry Pi lors de la simulation du château d'eau

Vérification du bon fonctionnement du serveur depuis un client UAExpert

Depuis UAExpert, il est possible de se connecter au simulateur de château d'eau pour vérifier son bon fonctionnement.

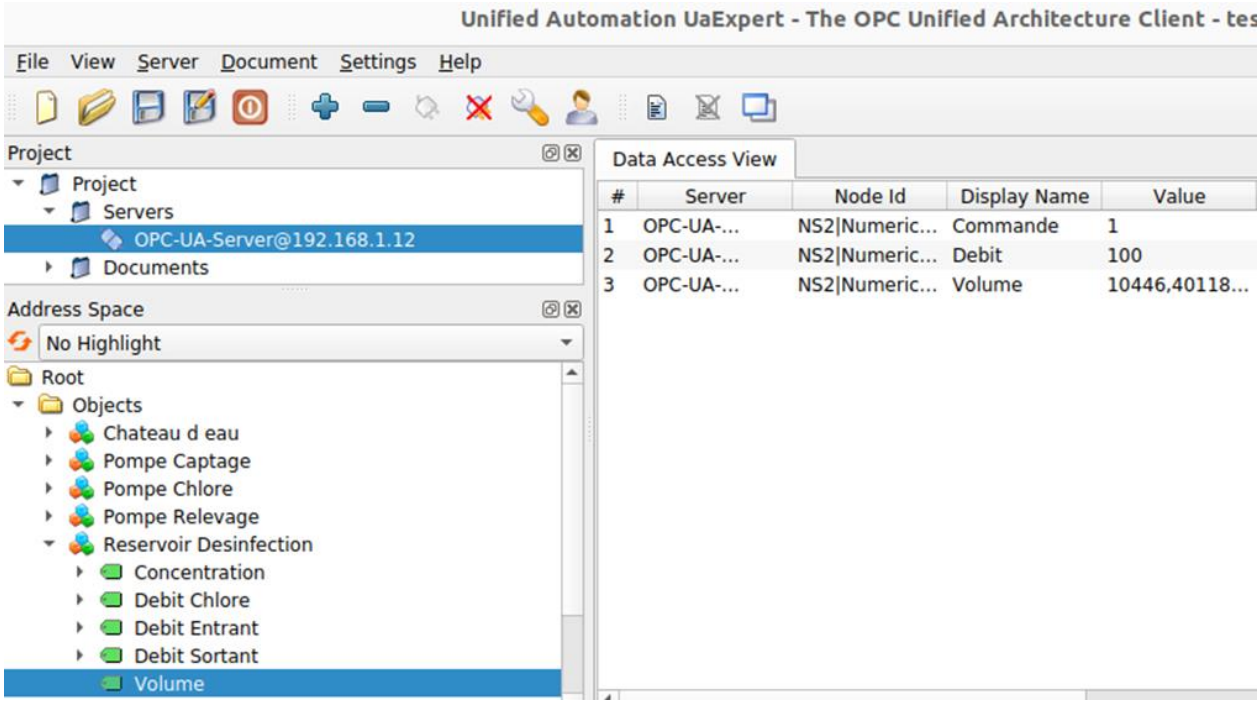

Figure 26 : Supervision des variables du simulateur de château d'eau depuis UAExpert

Cela permet de vérifier le bon fonctionnement mais aussi de mettre en évidence l'intérêt du service Discovery de OPC-UA, fournissant l'ensemble des variables accessibles.

Supervision par Panorama

Le populaire logiciel de supervision Panorama de Codra [3] inclut la possibilité de créer des clients et des serveurs OPC UA. Comme Codra fait partie de la fondation OPC, leur produit est régulièrement validé et mis aux normes. Le logiciel est disponible en version d'essai pour faire des projets de taille raisonnable pour l'enseignement (moins de 50 variables / 4h de connexion) [3]. Panorama ne fonctionne que sous Windows.

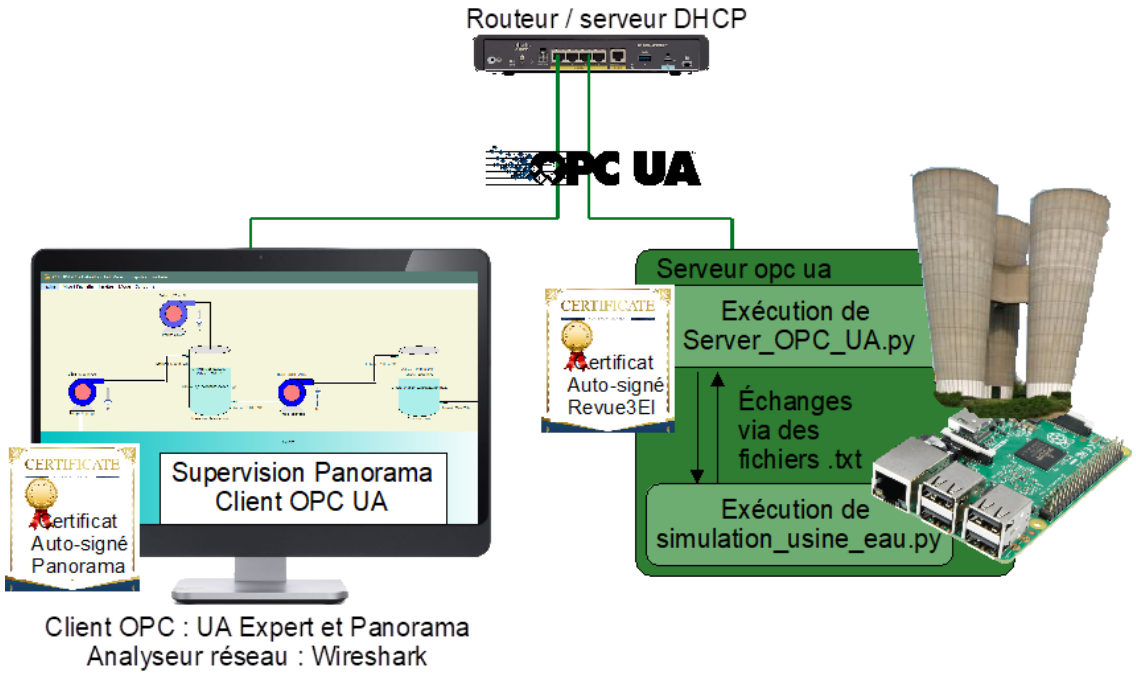

Figure 27 : Schéma de la configuration réseau pour la supervision du château d'eau simulé

Depuis Panorama Studio, ouvrir le fichier Panorama.ini du dossier SimuUsineMars2024 fourni avec cette ressource. SimuUsineMars2024 a été créé avec Panorama version 2023, il pourra donc être

ouvert avec une version plus récente. Aucune garantie par contre pour son ouverture avec une version plus ancienne.

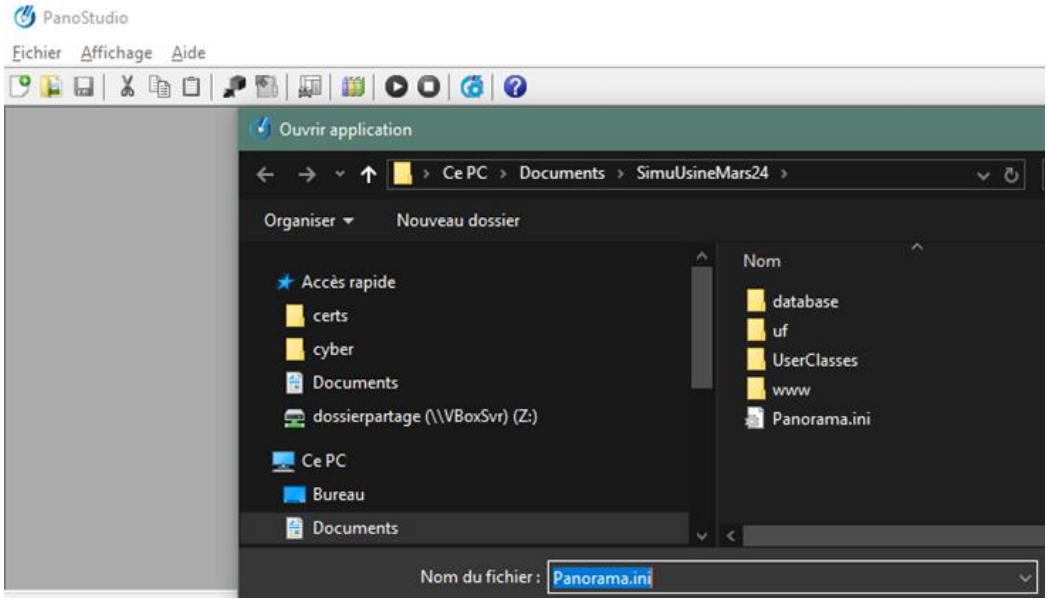

Figure 28 : Ouverture du projet

Ajuster la configuration OPC UA du projet.

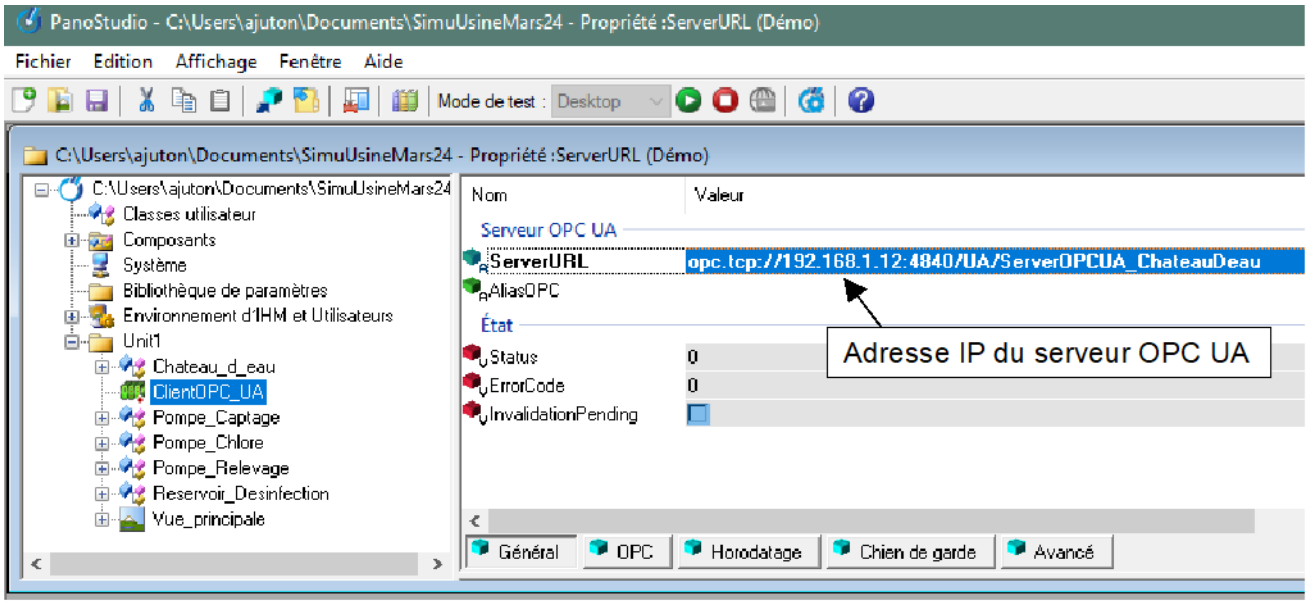

Figure 29 : Configuration de l'adresse IP du serveur OPC-UA / simulateur de château d'eau

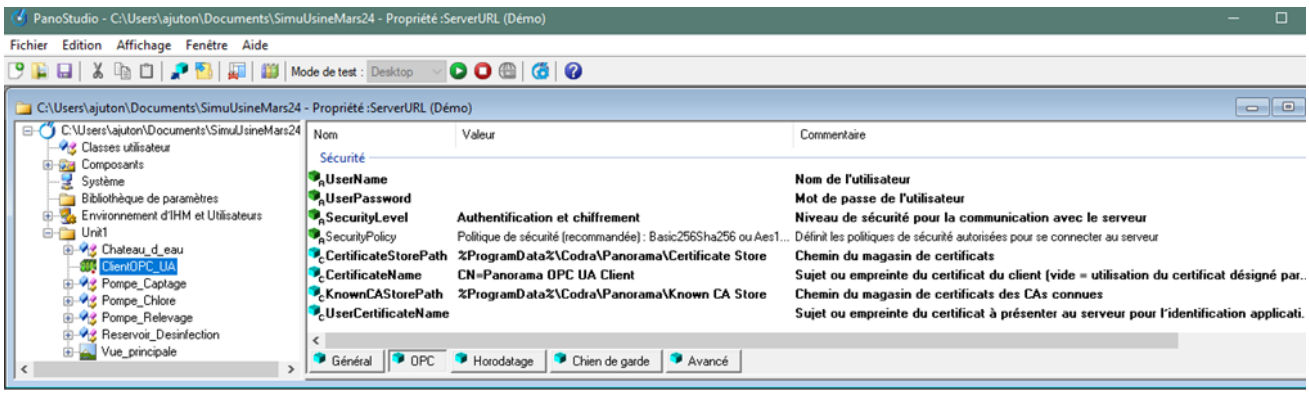

Figure 30 : Configuration des dépôts de certificat du client OPC UA

Pour se connecter de manière sécurisée, il ne manque plus à OPC-UA qu'un certificat SSL (X.509). Pour le créer depuis Windows (OpenSSL ne fonctionne que sous linux), Codra propose une solution (aide de Panorama rubriques *Création et installation du certificat Client OPC UA* et *Configuration de la sécurité*) ainsi qu'une fiche technique *FAQ080-V2.1 Création de certificats pour les fonctions Panorama* [12].

1. Installer [11] et démarrer PowerShell en administrateur

2. Exécuter les commandes suivantes (un peu différente de celles proposées par Panorama pour gérer l'utilisation de certificat auto-signé), en adaptant le chemin de destination du couple certificat/clé. Attention, la clé et le certificat doivent avoir le même nom, seule l'extension les différenciant.

```
PS C:\Users\ajuton\Downloads> $Cert = New-SelfSignedCertificate `
>> -Type Custom `
>> -Subject "CN=Panorama OPC UA Client" `
>> -HashAlgorithm sha256 `
>> -KeyAlgorithm RSA `
>> -KeyLength 4096 `
>> -KeyExportPolicy Exportable `
>> -Provider "Microsoft Enhanced RSA and AES Cryptographic Provider" `
>> -KeySpec KeyExchange `
>> -NotAfter (Get-Date).AddYears(5) `
>> -CertStoreLocation "Cert:\LocalMachine\My" `
>> -KeyUsage DigitalSignature,NonRepudiation,KeyEncipherment,DataEncipherment `
>> -TextExtension("2.5.29.37={text}1.3.6.1.5.5.7.3.2",
"2.5.29.17={text}URL=urn:localhost:CODRA:Panorama OPC UA Client")

PS C:\Users\ajuton\Downloads> Export-Certificate -Cert $Cert -FilePath
C:\Users\ajuton\Downloads\MyOpcUaClientCert5.der

PS C:\Users\ajuton\Downloads> $thumb = $Cert.Thumbprint

PS C:\Users\ajuton\Downloads> $CertPasswordEmpty = new-object System.Security.SecureString

PS C:\Users\ajuton\Downloads> Export-PfxCertificate -Cert "Cert:\LocalMachine\My\$thumb" -FilePath
C:\Users\ajuton\Downloads\MyOpcUaClientCert5.pfx -Password $CertPasswordEmpty
```

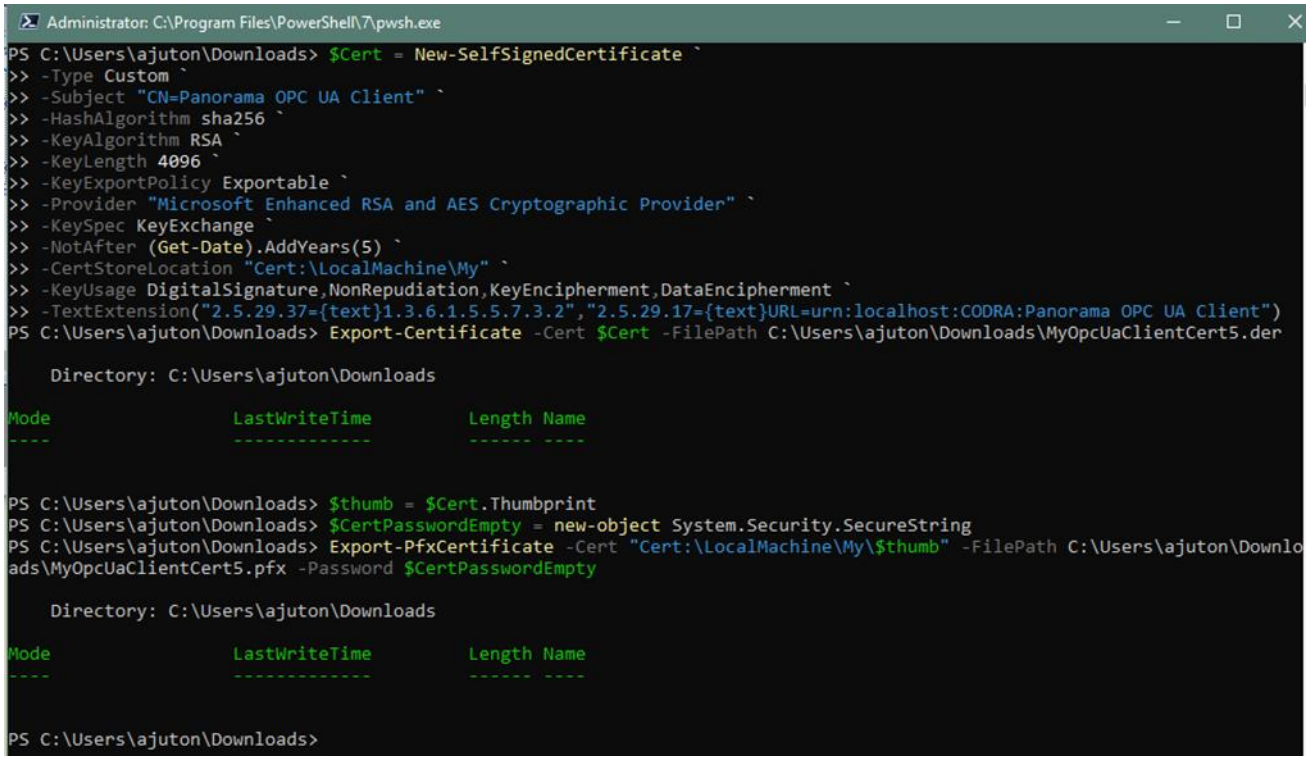

Figure 31 : Copie d'écran des commandes PowerShell de création du certificat X.509 et d'exportation de la clé privée associée

Copier le certificat généré, ainsi que le certificat du serveur OPC UA dans le dossier des certificats de Panorama :

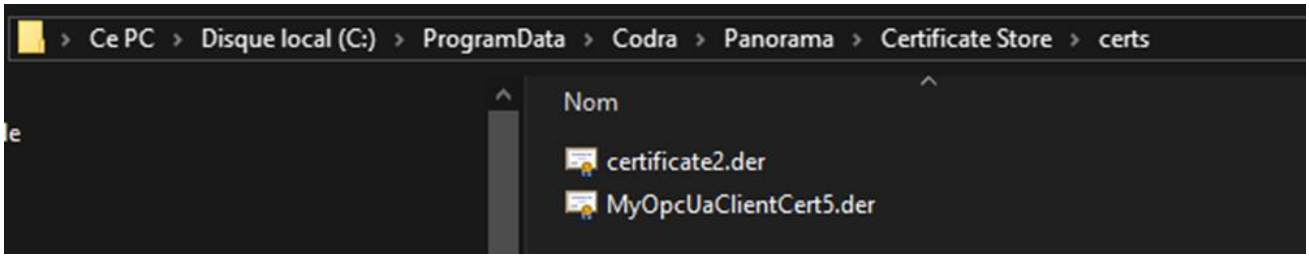

Figure 32 : Dossier certificats de Panorama

Faire de même avec la clé privée associée au certificat Client.

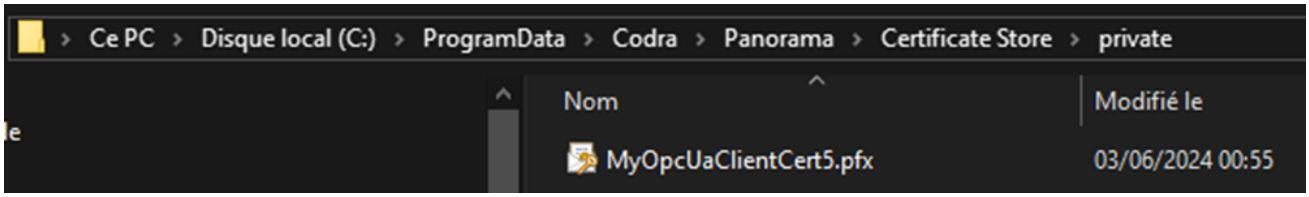

Figure 33 : Dossier clés privées de Panorama

Lancer le logiciel Codra Traceur (installé en même temps que Panorama) et éxécuter la supervision, en choisissant la vue principale. Le château d'eau est alors visualisé et contrôlable (on peut contrôler le débit de chaque pompe).

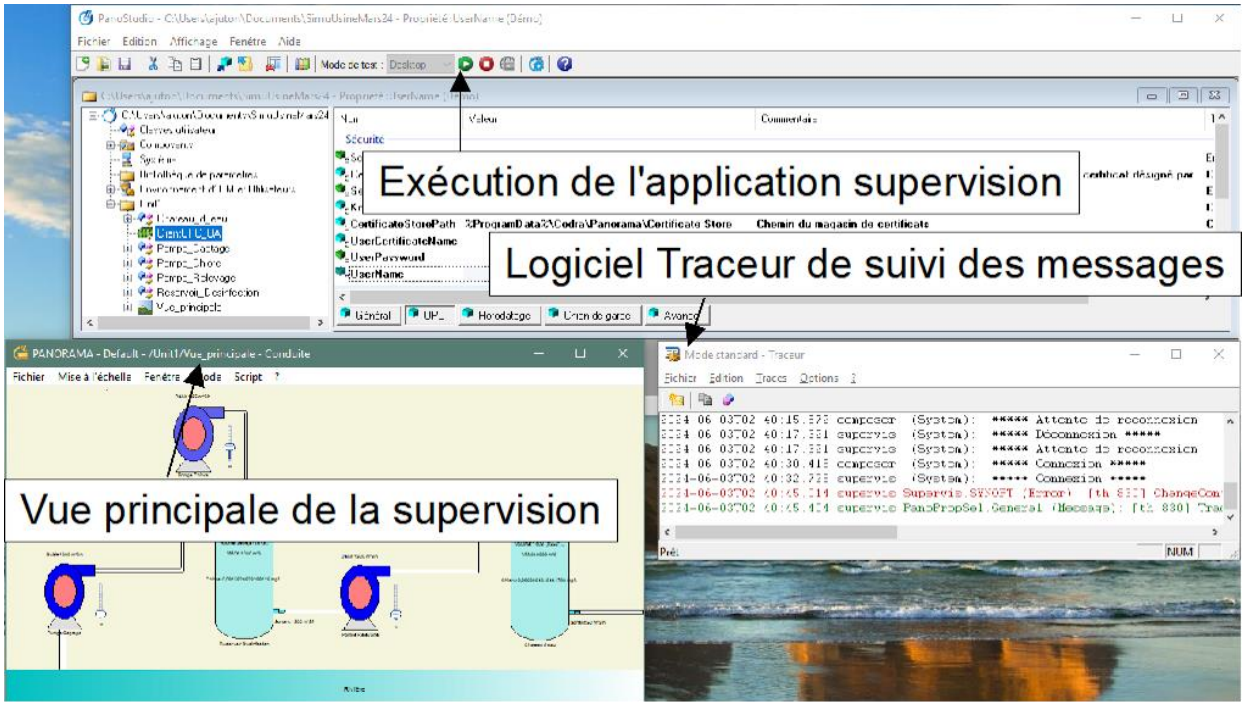

Figure 34 : Vue de la fenêtre de supervision du client OPC UA Panorama

Il est possible de connecter les clients OPC UA UAExpert et Panorama en même temps, montrant par là-même que le serveur accepte plusieurs clients simultanés. Wireshark peut être utilisé pour mettre en évidence la mise en place de la connexion sécurisée et les échanges (peu parlants lorsque la communication est chiffrée)

4 - Conclusion

Cette ressource s'est intéressée essentiellement à l'aspect sécurité d'OPC UA. Une des raisons du succès d'OPC UA est lié à l'utilisation et à la réputation des mécanismes SSL. Le monde de

l'automatisme industriel (OT – Operationnal Technology) tire parti des technologies développées et fiabilisées par l'informatique (IT – Information Technology). Le mode publisher/subscriber et les possibilités temps réels (basées sur la couche réseau TSN Time Sensitive Network) sont d'autres atouts d'OPC UA qui mériteraient également d'être présentés.

Les constructeurs intègrent de plus en plus souvent des serveurs OPC UA à leurs automates programmables. Par exemple, le S7-1200 de Siemens embarque un serveur OPC UA permettant d'accéder aux variables internes de l'automate. Hervé Discours, professeur à l'IUT de Cachan, a fait quelques vidéos très intéressantes sur le sujet [6].

Références :

[1] https://opcfoundation.org

[2] OPCUAcademics propose des ressources pédagogiques sur OPC UA. L'accès gratuit à ces ressources se demande sur la page https://opcfoundation.org/resources/opcuacademic/

[3] Codra, page de présentation de Panorama : https://codra.net/fr/offre-logiciel/plateforme-supervision/logiciel-panorama-suite/ et serveur web de téléchargement de Panorama - https://my.codra.net/

[4] Installation de Raspberry OS sur raspberry Pi4 et mise en place du bureau à distance.

https://github.com/ajuton-ens/CourseVoituresAutonomesSaclay/blob/main/Bibliotheques_logicielles/Installation_RaspberryOS_CoVAPSy_v1re2.pdf

[5] Documentation de FreeOPCUA

https://github.com/FreeOpcUa/opcua-asyncio

https://opcua-asyncio.readthedocs.io/en/latest

[6] Chaîne Youtube de Hervé Discours :

OPC UA - Initiation : https://www.youtube.com/watch?v=iN4qKm5W35g

OPC UA – Cybersécurité : https://www.youtube.com/watch?v=58FUQzWxs3Y

[7] Using the BME280 I2C Temperature and Pressure Sensor in Python, Matt Hawkins

https://www.raspberrypi-spy.co.uk/2016/07/using-bme280-i2c-temperature-pressure-sensor-in-python/

[8] Guide Installation de Raspberry OS sur raspberry Pi 4 :

https://github.com/ajuton-ens/CourseVoituresAutonomesSaclay/blob/main/Bibliotheques_logicielles/Installation_RaspberryOS_CoVAPSy_v1re3.pdf

[9] Site officiel de Unified Automation pour le téléchargement de UA Expert

https://www.unified-automation.com/products/development-tools/uaexpert.html

[10] Site officiel de Wireshark : https://www.wireshark.org/

[11] Installation de PowerShell : https://learn.microsoft.com/fr-fr/powershell/scripting/install/installing-powershell

[12] Fiche technique FAQ080-V2.1 Création de certificats pour les fonctions Panorama :

https://my.codra.net/fr/productreleases?productrelease=PS-2023&selection=technicalfiles

[13] Fondamentaux de la sécurité réseau, M. Sechehaye, A. Juton, M. Sauvergeat, février 2024, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/fondamentaux-dela-securite-reseau

Ce document est accompagné **d'une annexe zip** dont le lien est https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/opcua-protocole-securise-pour-automatisme-industriel

Ressource publiée sur Culture Sciences de l'Ingénieur : https://eduscol.education.fr/sti/si-ens-paris-saclay

Sécurité du protocole LoRaWAN

La Revue 3E.I
Culture Sciences de l'Ingénieur

Édité le 11/06/2024

école normale supérieure paris-saclay

Maxime SECHEHAYE[1]

[1] *ENS Paris-Saclay – DER Nikola Tesla*

Cette ressource fait partie du N°112 de La Revue 3EI du 2ème trimestre 2024.

Cette ressource présente les mécanismes de sécurité existants dans le protocole de communication sans fil LoRaWAN. Il ne s'agit pas d'un rapport complet sur le protocole LoRaWAN et de la technologie de modulation LoRa (déjà présentés dans le numéro 96 [5]) qu'il utilise mais plutôt d'un exposé sur les outils mis à disposition par ce protocole pour sécuriser les échanges de données. On insistera ainsi sur les choix importants qu'un développeur d'application IoT souhaitant utiliser LoRaWAN devra effectuer afin d'assurer la sécurité de son application.

Après avoir introduit le protocole LoRaWAN et rappelé son architecture, nous listerons les éléments de sécurité proposés par ce protocole puis nous étudierons dans le détail le mécanisme de connexion d'un nouveau terminal dans un réseau LoRaWAN déjà existant.

1 - Introduction

LoRaWAN est un protocole de communication radio qui permet à différents terminaux d'établir une communication sans fil et de constituer un LPWAN (*Low Power Wide Area Network*, réseau étendu à basse consommation). Ce protocole utilise la technologie de modulation LoRa pour la communication entre les terminaux et les passerelles, d'où son nom.

Sans rentrer dans les détails de fonctionnement de LoRaWAN et de la technologie LoRa, car ce n'est pas l'objectif de cette ressource, nous allons ici brièvement rappeler les éléments d'un tel réseau LoRaWAN.

L'architecture d'un réseau LoRaWAN suit une topologie de réseau en étoile. Les différents éléments de ce réseau sont :

- Les terminaux (*End Devices*) : objets connectés (capteurs, actionneurs…) qui communiquent avec les passerelles en utilisant la technologie de modulation LoRa ;
- Les passerelles (*Gateways*) : appareils faisant le lien entre les terminaux et les serveurs. Les passerelles ne réalisent aucun traitement sur l'information reçue, elles ne servent que de relais ;
- Le ou les serveurs réseau (*Network Server NS*) : pièce centrale du réseau LoRaWAN qui en assure la gestion et qui vérifie l'intégrité des échanges s'y déroulant ;
- Les serveurs d'application (*Application Servers*) : serveurs qui se chargent de traiter l'information envoyée par les terminaux et si nécessaire d'envoyer une réponse.

Sur un **réseau propriétaire**, tous les équipements appartiennent à la même entreprise, dans une zone limitée (une gare par exemple).

Sur un **réseau opéré**, les terminaux appartiennent à une entreprise A et les passerelles et serveurs réseau à un opérateur B. Le serveur d'application peut appartenir à l'entreprise A ou être hébergé dans un datacenter « cloud » C (OVH, AWS, Azure…). Notons que l'entreprise A peut-être un *fournisseur de service, les données appartenant alors à une entreprise D.*

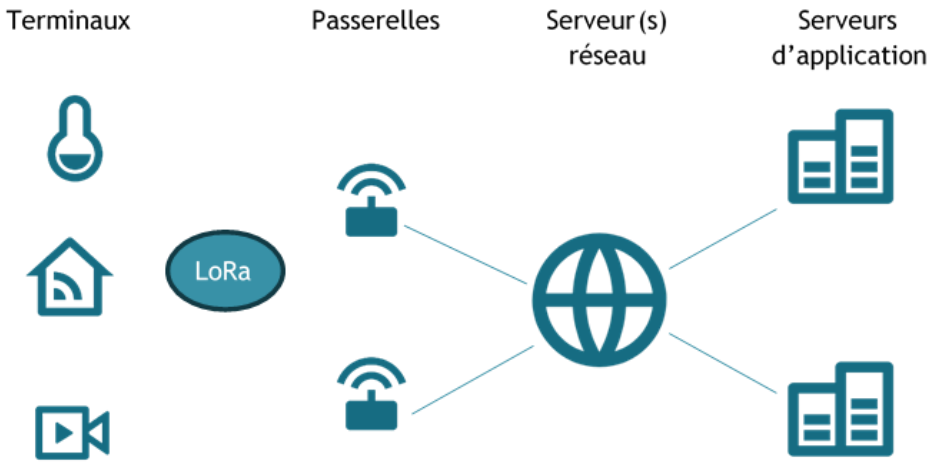

Figure 1 : Schéma de l'architecture d'un réseau LoRaWAN

Il est à noter qu'un terminal n'est pas associé à une unique passerelle. Lorsqu'un terminal souhaite envoyer une information à son serveur d'application, il transmet son message via LoRa et toutes les passerelles à proximité suffisante pour capter ce message le relaieront au serveur réseau. Si le serveur réseau s'aperçoit alors que plusieurs messages identiques arrivent, il n'en gardera qu'un. On a donc une multiplication du message qui permet de réduire la probabilité de devoir l'émettre à nouveau en cas de perte.

LoRaWan ayant une grande portée, « toutes les passerelles à proximité » reçoivent le message signifie que même les messages émis par les terminaux d'un réseau propriétaire sont reçus par les passerelles des opérateurs environnant.

2 - Éléments de sécurité d'un réseau LoRaWAN

Lorsqu'un terminal souhaite envoyer une donnée, son message transitera par au moins trois appareils différents : au moins une passerelle, un serveur réseau et un serveur d'application. Sur un réseau opéré, les 2 premiers appartiennent à l'opérateur, dont on ne maitrise pas la politique de sécurité. Il faut donc que la connexion entre chacun de ces appareils soit sécurisée pour assurer la sécurité de l'ensemble de la chaîne de transmission d'information, tout en restant compatible avec la faible consommation souhaitée et la faible puissance des processeurs.

Voici les éléments de sécurité présents dans le réseau LoRaWAN qui permettent d'augmenter la sécurité des communications établies :

- Des clés de sécurité sont générées et utilisées pour chiffrer les communications depuis les terminaux jusqu'aux serveurs d'application grâce à l'algorithme *AES-128* mais aussi pour vérifier qu'une trame n'a pas été altérée entre son émission et sa réception. Ces clés de sécurité servent donc à assurer l'intégrité et la confidentialité des échanges. Nous reviendrons en détail sur l'établissement et l'utilisation de ces clés de sécurité.
- Deux compteurs de trames (*Frame Counter*) sont utilisés lors de chaque nouvelle connexion entre le terminal et le réseau LoRaWAN et s'incrémentent chaque fois qu'un message ascendant est envoyé (depuis le terminal) ou bien chaque fois qu'un message descendant est envoyé (vers le terminal). On ne peut donc pas retransmettre une trame déjà envoyée.

3 - Établissement d'une connexion sécurisée sur LoRaWAN

Nous allons ici décrire le processus d'établissement d'une connexion sécurisée sur LoRaWAN. Comme expliqué précédemment, ce sont les clés de sécurité qui permettent d'assurer la confidentialité et l'intégrité des messages échangés. Il faut donc s'assurer que l'établissement et le partage de ces clés soient sécurisés.

Il existe deux types d'ajout d'un nouveau terminal sur un réseau LoRaWAN :

- L'activation sans fil (*Over The Air activation*, *OTAA*) où toutes les données liées à la sécurité des communications futures sont établies de manière sans fil entre le serveur du réseau (NS) et le terminal lors de l'établissement de la connexion.
- L'activation par personnalisation (*Activation By Personalization*, *ABP*) où toutes les données liées à la sécurité sont déjà stockées sur le terminal et le réseau LoRaWAN avant les premiers échanges.

Avec la méthode ABP, les mêmes clés de sécurité sont donc utilisées à chaque session et il faut les changer manuellement si besoin. C'est donc la méthode la moins sécurisée des deux.

On va donc s'intéresser ici à l'activation sans fil qui est la méthode d'activation la plus sécurisée.

4 - Activation sans fil d'un nouveau terminal sur LoRaWAN

Avant l'activation, le terminal doit posséder les informations suivantes :

- DevEUI : identifiant unique sur 64 bits attribué dès la fabrication, similaire à une adresse MAC
- AppEUI : identifiant unique d'une application sur le réseau LoRaWAN considéré, sur 64 bits et modifiable
- AppKey : clé sur 128 bits partagée par le terminal et le serveur réseau

La clé AppKey est spécifique à chaque nouveau terminal sur le point de rejoindre un réseau LoRaWAN. Elle n'est *jamais transmise sur le* réseau LoRaWAN pour des raisons évidentes de sécurité et doit donc être transmise entre le terminal et le serveur réseau par un moyen extérieur (provisionnement physique, connexion sécurisée HTTPS…)

4.1 - Demande de connexion (*Join Request*)

Comme son nom l'indique, la première étape consiste à une demande de connexion du terminal au serveur. Cette demande contient :

Le *DevNonce* est un nombre aléatoire et unique. En effet, le serveur réseau conserve les *DevNonce* précédemment utilisés par chaque terminal et rejette toutes les demandes de connexion avec un *DevNonce* déjà utilisé. Cela empêche les attaques par rejeu.

Il y a aussi un *Message Integrity Code (MIC)* qui est calculé à partir des trois champs et de l'AppKey. Il permet de s'assurer, comme son nom l'indique, de l'intégrité du message reçu et de s'assurer de l'expéditeur du message.

Le MIC est systématiquement le résultat d'un chiffrement CMAC-AES sur 128 bits sur les données du message et avec AppKey

La demande de connexion n'est pas chiffrée car elle ne contient pas d'information sensible

4.2 - Acceptation de connexion (Join Accept)

Ce message est envoyé par le serveur réseau au terminal. Il contient les champs suivants :

On va rapidement expliquer ces différents champs :

- AppNonce : Nombre aléatoire fournie par le serveur réseau
- NetID : Ses 7 bits de poids les plus forts représentent l'identifiant du réseau (NwkID) qui est unique dans une zone géographique donnée. Les autres bits donnent l'adresse du terminal dans le réseau.
- DevAddr : Adresse attribuée par le serveur réseau au terminal (similaire à une adresse IP sur un réseau local qui serait fournie par un serveur DHCP)
- DLSettings : Paramètres à utiliser par le terminal pour le *downlink* (lorsque le terminal va recevoir des messages du serveur réseau)
- RXDelay : Délai entre l'émission d'un message par le serveur réseau et le début de sa réception par le terminal
- CFList (optionnel) : Fréquences de canaux autorisées pour les communications entre le terminal et les passerelles

On calcule de nouveau un MIC et cette fois les données sont chiffrées à l'aide d'AppKey pour les protéger

4.3 - Calcul des clés de session

Le serveur réseau et le terminal peuvent alors tous les deux calculer les clés spécifiques à cette nouvelle session : NwkSKey (*Network Session Key*) et AppSKey (*Application Session Key*).

NwkSKey est utilisée pour calculer le MIC des messages mais aussi pour chiffrer les commandes MAC (*Medium Access Control*). Ces commandes sont uniquement entre le serveur réseau et le terminal et permettent de configurer et de contrôler le comportement de ce terminal LoRaWAN. La demande de connexion est un exemple de commande MAC.

AppSKey permet quant à elle de déchiffrer les données (*payload*) des messages entre le terminal et le serveur d'application.

Le serveur réseau conserve donc la clé NwkSKeyet transfère la clé AppSkey au serveur d'application. Comme le lien entre le serveur réseau et le serveur d'application peut être réalisé via n'importe quel protocole de communication, il faut s'assurer que la connexion entre ces deux serveurs soit sécurisée.

Les formules pour obtenir ces deux clés sont les suivantes :

NwkSkey = aes128_encrypt(AppKey, 0x01 | AppNonce | NetID | DevNonce | pad16)

AppSkey = aes128_encrypt(AppKey, 0x02 | AppNonce | NetID | DevNonce | pad16)

Où *pad16* signifie qu'on ajoute le nombre de zéros suffisant pour que le message chiffré avec la clé AppKey ait une taille de 16 octets.

On retrouve dans ces formules, entre autres, des données envoyées par le serveur réseau dans le message d'acceptation de connexion, ce qui justifie le chiffrement de ce dernier.

4.4 - Le terminal est intégré au réseau LoRaWAN

Maintenant que le terminal est intégré au réseau LoRaWAN, il devra garder pour cette session :

- DevAddr qui lui permet de s'identifier sur ce réseau
- NwkSKey utilisée pour calculer le MIC des messages ainsi que pour chiffrer les messages MAC
- AppSKey utilisée pour chiffrer les données utiles envoyées au serveur d'application

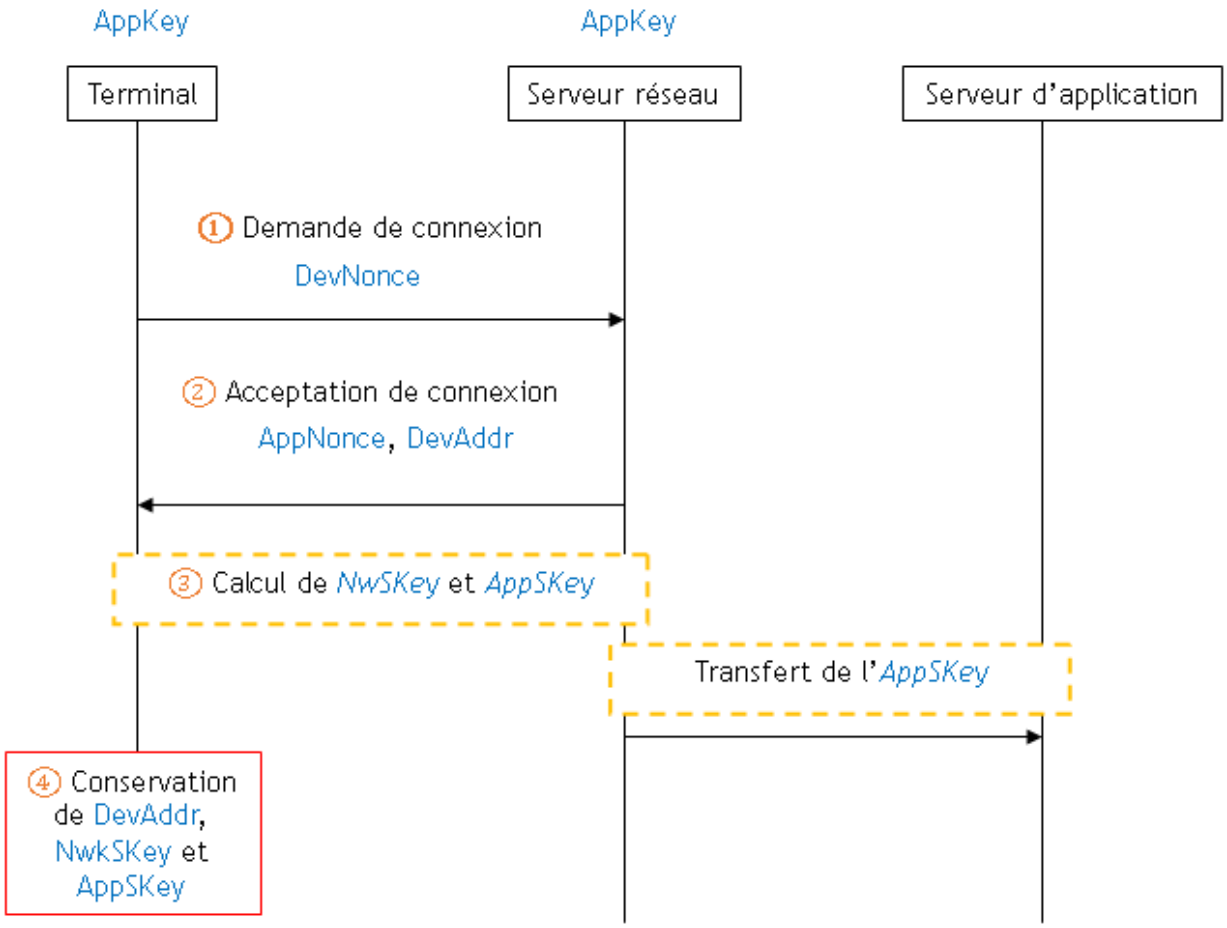

Figure 2 : Diagramme résumant les différentes étapes de l'activation sans fil d'un terminal sur un réseau LoRaWAN

5 - Différences entre LoRaWAN 1.0 et LoRaWAN 1.1

Dans LoRaWAN 1.1, tout ce qui a été exposé précédemment reste valable mais il y a quelques ajouts.

Il y a des améliorations dans la sécurité (changement du Devnonce en compteur, …).

Enfin, il y a davantage de clés de session qui permettent de déléguer au réseau visité certaines fonctions et de piloter le terminal quand le terminal est en roaming.

6 - Conclusion

Nous avons observé que le protocole LoRaWAN propose une architecture séparant les informations liées au réseau de celles liées à l'application spécifique. Pour cela, deux serveurs distincts sont utilisés dans une session : le serveur réseau et le serveur d'application. Il y aura donc une sécurité liée au serveur réseau et une autre liée au serveur d'application.

Ce protocole propose deux méthodes d'activation d'un nouveau terminal mais seule l'activation sans fil permet d'établir une session sécurisée car elle permet la création de nouvelles clés spécifiques à chaque session.

Enfin, le protocole LoRaWAN propose un mécanisme de génération et d'approvisionnement des clés de session à partir d'une clé spécifique au terminal considéré, la clé *AppKey*. Ce mécanisme est assez simple et semble sécurisé dès lors que les messages sont chiffrés, comportent un code d'intégrité ainsi qu'un compteur de trames pour éviter des attaques par rejeu.

Cependant, cette sécurité repose sur une unique clé : la clé AppKey. Ce sera toujours cette même clé qui sera utilisée lors de toute nouvelle session démarrée par le terminal considéré. Il faut donc s'assurer que cette clé est stockée de manière sécurisée et que son partage avec le serveur réseau est aussi sécurisé. Le protocole LoRaWAN n'indique rien à ce sujet, c'est à la responsabilité de l'utilisateur de ce protocole de s'assurer de la sécurité de l'approvisionnement de l'AppKey. Il en est de même pour le transfert de la clé AppSKey du serveur réseau au serveur d'application.

Voici un extrait de la spécification de LoRaWAN 1.1 [2] à ce propos :

> *Secure provisioning, storage, and usage of root keys NwkKey and AppKey on the end device and the backend are intrinsic to the overall security of the solution. These are left to implementation and out of scope of this document.*

Le protocole LoRaWAN propose donc des mécanismes de sécurité classiques et efficaces à condition d'utiliser les bonnes méthodes et d'assurer la sécurité des éléments non pris en charge par le protocole.

Références :

[1]: *LoRaWAN Specification*, LoRa Alliance Technical Committee, 2015

[2]: *LoRaWAN 1.1 Specification*, LoRa Alliance Technical Committee, 2017

[3]: *LoRaWAN Security, Full end-to-end encryption for IoT application providers*, Gemalto, Actility, Semtech, 2017

https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_security_whitepaper.pdf

[4]: *End Device Activation*, The Things Network

https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/

[5]: Réseaux très basse consommation, longue portée, bas débit, l'exemple de LoRaWAN, A. Juton, septembre 2019, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/reseau-tres-basse-consommation-longue-portee-bas-debit-exemple-lorawan

La Revue 3E.I
Culture Sciences de l'Ingénieur

Sécurité de ZigBee

Maxime SECHEHAYE[1]

Édité le 10/06/2024

[1] *ENS Paris-Saclay – DER Nikola tesla*

Cette ressource fait partie du N°112 de La Revue 3EI du 2ème trimestre 2024.

Cette ressource présente les mécanismes de sécurité présents dans le protocole de communication sans fil ZigBee. On y rappelle brièvement la structure d'un réseau ZigBee avec les différents éléments le composant avant de détailler les différents éléments qui permettent d'établir des communications sécurisées. Ces éléments mis à disposition par le protocole ne sont toutefois pas obligatoirement utilisés par des fournisseurs de solution ZigBee. On s'attachera donc à proposer des exemples de choix judicieux pour améliorer la sécurité d'un réseau ZigBee.

1 - Introduction

ZigBee est un protocole de communication qui permet à différents équipements géographiquement proches de communiquer sans fil. Il s'agit d'un protocole d'utilisation simple et peu chère, particulièrement adapté pour des réseaux sans fil personnels (*Wireless Personal Area Network*, *WPAN*) avec des applications de domotiques par exemple.

1.1 - Rappels sur l'architecture d'un réseau ZigBee

Pour pouvoir aborder sereinement le reste de cette ressource, voici un bref rappel sur l'architecture d'un réseau ZigBee avec ses différents composants.

Un réseau ZigBee est composé de trois types d'appareils :

- Coordinateur : c'est le premier membre d'un réseau ZigBee. C'est donc lui qui va le configurer. Il attribue les adresses des nouveaux membres du réseau, surveille l'état du réseau et participe au routage des messages.
- Routeur : appareil permettant de transmettre les messages en choisissant le chemin optimal pour atteindre le terminal de destination.
- Terminal : dispositif émetteur et/ou récepteur qui échange des informations avec d'autres nœuds du réseau ZigBee.

Un réseau ZigBee peut être réalisé suivant une des trois topologies suivantes : en étoile, maillée ou en arbre.

Dans la topologie en étoile (voir la figure 1), il y a un coordinateur qui se charge de l'ensemble du routage. Tous les autres nœuds du réseau sont des terminaux. C'est une topologie simple et facile à mettre en place mais dont le bon fonctionnement repose uniquement sur un nœud, le coordinateur.

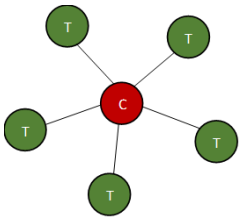

Figure 1 : Topologie en étoile d'un réseau ZigBee (C : Coordinateur, T : Terminal)

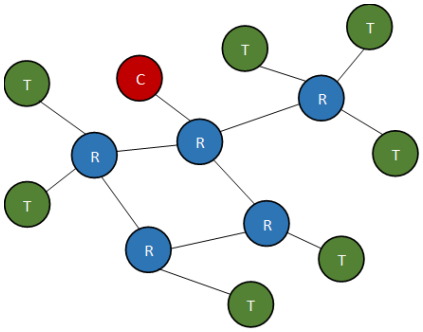

Figure 2 : Topologie maillée d'un réseau ZigBee (C : Coordinateur, R : Routeur, T : Terminal)

Dans les topologies maillée (*mesh*) et en arbre (voir les figures 2 et 3), le réseau composte des routeurs. La topologie maillée offre plus de redondance en cas de panne d'un routeur.

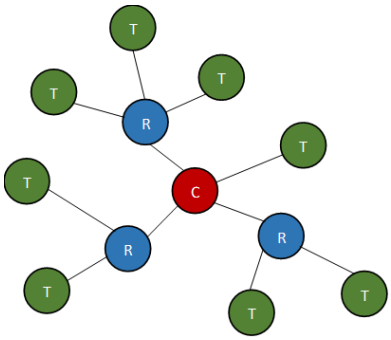

Figure 3 : Topologie en arbre d'un réseau ZigBee (C : Coordinateur, R : Routeur, T : Terminal)

1.2 - Structure d'une trame ZigBee

Voici la structure d'une trame ZigBee qui nous sera utile pour comprendre certains éléments de sécurité par la suite :

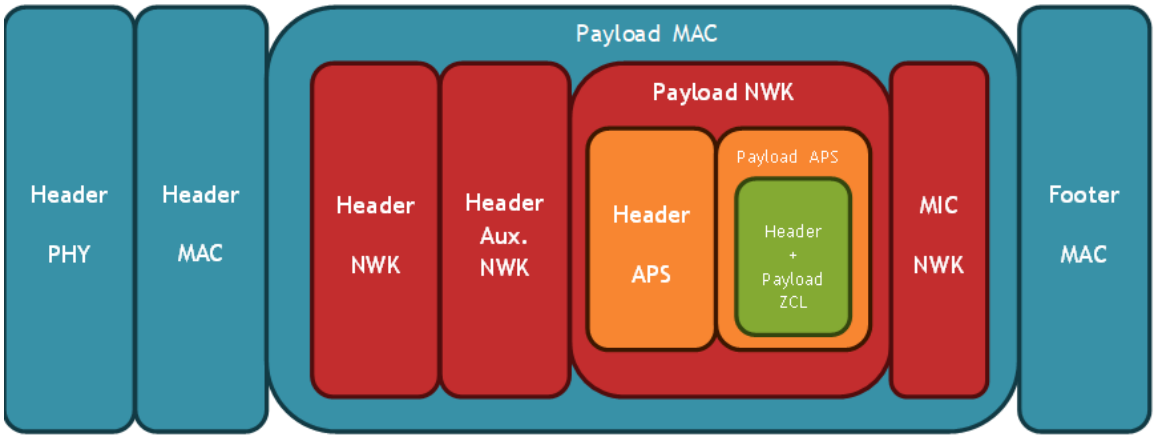

Figure 4 : Structure d'une trame ZigBee

Où PHY correspond à la couche physique, MAC signifie *Medium Access Control*, NWK représente la couche réseau, APS la couche d'application et ZCL la *Zigbee Cluster Library* (fonctionnalités courantes standardisées pour un développement plus rapide).

2 - Éléments de sécurité du protocole ZigBee

Le protocole de communication ZigBee prévoit un certain nombre d'éléments qui, si mis en place et utilisés correctement comme nous le verrons dans la suite de cet article, permettent d'améliorer la sécurité du réseau. On va ici détailler chacun de ces éléments.

2.1 - Compteur de trames

Lorsqu'un nœud du réseau ZigBee envoie un message, la trame comporte un champ correspond à son compteur de trames. Ce compteur permet d'éviter les attaques par rejeu.

Chaque nœud maintient une liste des compteurs de trames de ses voisins et de ses enfants (dans le cas d'un routeur). Ainsi, lorsqu'il reçoit un message d'un de ses voisins, il peut facilement vérifier si le compteur de trames présent dans ce message est supérieur au dernier dont il a eu connaissance. Si ce n'est pas le cas, le message est ignoré.

2.2 - Centre de confiance centralité (*Centralized Trust Center*)

Le centre de confiance centralisé est le nœud du réseau ZigBee où est centralisée la gestion de la sécurité de ce réseau. Ce nœud :

- Authentifie les nouveaux arrivants sur le réseau.
- Autorise ou non l'accès au réseau.
- Distribue les clés de sécurité.

Le choix du centre de confiance ainsi que sa politique de fonctionnement sont des choix cruciaux lors de la création d'un réseau ZigBee

Le plus souvent, le centre de confiance centralisé est confondu avec le coordinateur.

2.3 - Liste d'accès contrôlé (*Access Control List*)

Cette liste permet au centre de confiance de définir des règles d'accès au réseau ZigBee pour les nouveaux nœuds. On peut y renseigner une liste d'appareils autorisés (*white list*) ou bien une liste d'appareils interdits (*black list*). On peut également y définir des permissions accordées ou non à certains nœuds. L'implémentation de cette liste varie d'une solution ZigBee à l'autre suivant la politique de sécurité désirée.

2.4 - Clés de sécurité

Clé de sécurité du réseau (*Network Key*)

Il s'agit d'une clé de chiffrement sur 128 bits commune à tous les nœuds du réseau ZigBee et transmise lors de la procédure d'intégration au réseau. Cette clé sert à chiffrer les messages de maintenance du protocole ZigBee au sein du réseau mais aussi à chiffrer les informations réseau contenues dans les messages envoyés. C'est généralement cette clé qui est utilisée lors du calcul du code d'intégrité (*Message Integrity Code*, *MIC*) de chaque message (voir *MIC NWK* sur la figure 4).

Quand elle est transmise à un nouveau nœud, cette clé est elle-même chiffrée avec une clé préconfigurée déjà connue du centre de confiance et du nouveau nœud.

Clé de sécurité de la liaison (*Link Key*)

Cette clé de chiffrement est créée pour chiffrer les données échangées entre deux nœuds spécifiques. Les messages entre ces deux nœuds sont donc chiffrés à la fois avec la clé de liaison et la clé de réseau.

C'est le centre de confiance qui se charge de générer aléatoirement cette clé et de la transmettre aux deux nœuds concernés. Pour ce faire, une clé préconfigurée peut être partagée entre le nœud et le centre de confiance. C'est avec cette clé que le centre de confiance partagera de manière sécurisée la clé de réseau et pourra aussi générer une clé de liaison spécifique à cette session entre lui et le nouveau nœud. Ainsi, lorsque le nœud demande une clé de liaison pour chiffrer ses communications avec un autre nœud, le centre de confiance lui transmettra une clé aléatoire chiffrée avec la clé de liaison qu'il partage avec ce nœud.

Provisionnement de clé par vérification de certificat (*Certificate-Based Key Establishment*)

Il est possible d'utiliser des certificats numériques fournis par des organisations reconnues pour identifier de manière sécurisée un nouveau nœud sur le réseau ZigBee. Ce certificat permet aussi de procéder à un échange sécurisé d'une clé de liaison entre le centre de confiance et le nouveau nœud.

Politique de sécurité décentralisée (*Distributed Security*)

Depuis ZigBee 3.0, il est possible de décentraliser la sécurité du réseau. Lorsque cette politique de sécurité est choisie, ce sont les routeurs par lesquels les nouveaux nœuds entrent sur le réseau ZigBee qui sont en charge d'authentifier ces nouveaux arrivants et de distribuer les clés de sécurité. Il n'y a alors plus de nœud central qui a connaissance de tous les nœuds authentifiés du réseau.

La documentation officielle de ZigBee n'impose pas une politique de sécurité en particulier. Le choix est laissé aux fournisseurs de solutions ZigBee.

La seule obligation donnée par le protocole ZigBee est l'utilisation d'un chiffrement par bloc AES avec une clé sur 128 bits.

Sécurité « saut par saut »

Dans le protocole ZigBee, la sécurité est effectuée en mode « saut par saut » (*hop-by-hop*). Cela signifie que chaque fois qu'un paquet ZigBee passe par un routeur, ce dernier vérifie l'intégrité du paquet avec le MIC et empêche toute attaque par rejeu en vérifiant le compteur de trames.

Une fois ces vérifications effectuées, s'il n'y a aucun problème, le routeur va rechiffrer la trame avec la clé de réseau et modifier les champs du header *Aux. NWK* de la trame (voir figure 4), notamment l'adresse source ainsi que le compteur de trames. Ainsi, lorsqu'un autre routeur ou le terminal de destination recevra la trame, il pourra vérifier à nouveau qu'elle n'a pas été altérée ou rejouée.

Cette politique de sécurité permet de ne pas encombrer le réseau avec des messages corrompus ou rejoués car ils seront rapidement identifiés et ignorés. Cependant, les routeurs sont davantage

sollicités par rapport à du simple routage. C'est une des raisons pour lesquelles le protocole ZigBee n'est pas adapté pour des réseaux de grande échelle.

3 - Sécurisation de l'intégration d'un nouveau nœud au réseau

On va décrire ici les principaux éléments qui ont une influence sur la sécurité du réseau lorsqu'un nouveau nœud le rejoint.

3.1 - Protection de la clé de réseau lors de son transfert

Il est crucial de sécuriser le transfert de la clé de réseau au nouveau nœud. Le choix de la clé préconfigurée en est grandement responsable. Il existe différents choix que nous allons détailler.

Clé par défaut - « *Well-known* »

Il s'agit d'une clé préconfigurée par défaut et qui est donc connue de tous, sur tous les réseaux ZigBee. Cette clé a pour valeur hexadécimale 5A 69 67 42 65 65 41 6C 6C 69 61 6E 63 65 30 39, ce qui donne après décodage ASCII « ZigBeeAlliance09 ».

Il est évident qu'il faut absolument éviter ce choix de clé préconfigurée car il n'apporte aucune sécurité au transfert de la clé réseau.

Clé préconfigurée spécifique au réseau

On peut configurer une clé qui sera choisie comme clé de lien préconfigurée pour tous les nouveaux nœuds sur le réseau.

Le déploiement de cette solution est facile car la clé est commune à tous les nœuds sur le réseau. La sécurité du transfert de la clé de réseau est alors améliorée par rapport à la clé par défaut qui est connue même à l'extérieur du réseau ZigBee concerné.

Clé de liaison dérivée à partir d'un code d'installation

Depuis ZigBee 3.0, un code d'installation (*install code*) peut être utilisé par le nœud entrant et le centre de confiance pour générer une même clé permettant de chiffrer la clé de réseau. Ce code est généré aléatoirement lors de la conception du nœud et doit être communiqué au centre de confiance du réseau avant l'arrivée du nœud dans ce réseau.

Ce même code d'installation peut ensuite être utilisé pour générer une clé de liaison entre le centre de confiance et le nouveau nœud.

Il est vivement recommandé de transmettre hors-réseau le code d'installation au centre de confiance (par QR code par exemple).

Le code d'installation permet l'authentification du nouveau nœud et garantit l'unicité de la clé au sein du réseau. C'est donc le choix apportant le plus de sécurité au transfert de la clé de réseau.

3.2 - Mise en service hors-réseau

Il est possible de transférer hors-réseau toutes les informations nécessaires à l'intégration au réseau. On évite donc de passer par le réseau où ces informations sensibles pourraient être interceptées. Différentes solutions sont envisageables :

Pré configuration lors de la conception de l'appareil

Lorsqu'on fait ce choix, la clé de réseau est présente en permanence sur l'appareil. Il n'y a alors pas le droit à l'erreur sinon l'appareil sera inutilisable. De plus, un piratage d'un tel appareil rendrait la clé de réseau accessible depuis l'extérieur, ce qui compromettrait le réseau concerné.

NFC / QR code

On peut utiliser une communication NFC ou encore un QR code pour transmettre la clé de réseau. Ces mécanismes de communication sont très faciles d'utilisation mais il ne suffit sur place que d'un lecteur pour obtenir les informations sensibles.

Site internet

On peut enfin stocker toutes les informations de sécurité dans une base de données qui est reliée à un site internet. Si on utilise les codes d'installation pour le transport de la clé de réseau, cela peut nécessiter une base de données conséquente. De plus, on ne fait que transmettre la responsabilité de la sécurité au site internet.

4 - Sécurisation après l'intégration d'un nouveau nœud au réseau

4.1 - Empêcher les attaques par reconnexion

Si la clé préconfigurée peut être réutilisée à chaque reconnexion, il est possible de simuler une tentative de reconnexion en usurpant l'identité de l'appareil et en utilisant cette clé préconfigurée. On peut alors obtenir la clé de réseau et donc compromettre l'ensemble du réseau ZigBee.

Pour empêcher ce type d'attaque, ZigBee 3.0 propose un mécanisme de négociation de clé de liaison avec le centre de confiance. Lors de la première connexion de l'appareil au réseau ZigBee avec la clé préconfigurée, le centre de confiance génère une nouvelle clé de liaison qui sera utilisée lors des reconnexions futures.

Il est recommandé de désactiver la reconnexion avec la clé préconfigurée dans la politique du centre de confiance. Si un appareil tente de se reconnecter avec la clé préconfigurée, il faudrait mettre en place une intervention manuelle au centre de confiance pour autoriser ou non cette reconnexion.

4.2 - Changer régulièrement la clé de réseau

Puisque la clé de réseau est l'élément de sécurité majeur du réseau ZigBee, il est souhaitable de la changer régulièrement afin de s'assurer qu'aucun piratage d'un ancien appareil du réseau ne vienne compromettre ce réseau. On procède ainsi :

- Le centre de confiance diffuse la nouvelle clé de réseau à tous les nœuds en la chiffrant avec la clé actuelle qui est sur le point de de devenir obsolète

- Le centre de confiance diffuse un message *Switch Key* comprenant le numéro permettant d'identifier la nouvelle clé de réseau. C'est dorénavant cette clé qui sera utilisée sur le réseau.

Pour ne pas surcharger le réseau ZigBee de messages de diffusion, il faut trouver un juste intervalle de temps au bout duquel on change la clé de réseau. C'est un compromis entre sécurité et performance du réseau ZigBee. On pourrait le faire à chaque fois qu'un appareil quitte le réseau pour s'assurer qu'un piratage de cet appareil ne compromette pas la sécurité du réseau.

5 - Conclusion

Nous avons ici observé les éléments de sécurité présents dans le protocole de communication ZigBee. Ce protocole propose différents mécanismes de sécurité et laisse beaucoup de liberté sur son implémentation. Il revient donc à l'utilisateur de ce protocole de faire les bons choix pour assurer la sécurité de son réseau. Cette sécurité repose principalement sur celle de la clé de réseau.

Nous avons pu constater que les choix sont nombreux et qu'il faut donc être conscient de leur importance, de leurs enjeux et des conséquences que tel ou tel choix peut avoir.

Cette ressource ne se veut pas exhaustive sur les politiques de sécurité possible et sur les vulnérabilités de ce protocole. Le lecteur souhaitant approfondir ces connaissances à ce propos pourra se référer au papier de NXP sur la sécurité de ZigBee [2] ainsi qu'à la présentation lors de la conférence BlackHat des failles de sécurité possibles de ZigBee et leur exploitation [5].

Références :

[1]: *ZigBee Specification*, ZigBee Alliance, 2015
https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf

[2]: *Maximizing security in ZigBee networks*, NXP Laboratories UK, 2017
https://www.nxp.com/docs/en/supporting-information/MAXSECZBNETART.pdf

[3]: *ZigBee 3.0 Security*, Digi, 2018
https://www.digi.com/support/knowledge-base/zigbee-3-0-security

[4]: *AN1233 : ZigBee Security*, Silicon Laboratories, 2022

[5]: *ZigBee exploited – The good, the bad and the ugly*, Tobias Zillner, Sebastian Strobl, black hat USA 2015
https://www.youtube.com/watch?v=9xzXp-zPkjU&ab_channel=BlackHat
https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly.pdf