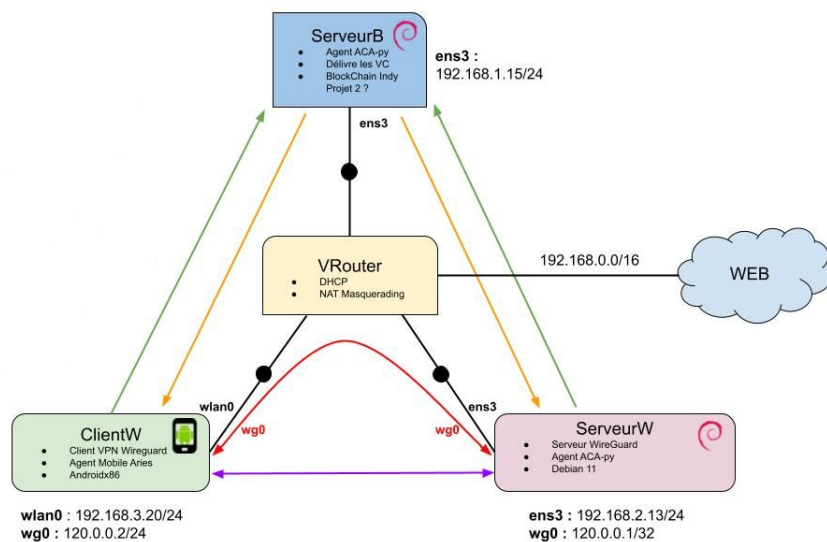


MASTER 2 INFORMATIQUE RCI

2021-2022

Rapport de PFE : Déploiement d'un VPN sur des équipements mobiles ou IoT



Sommaire

1	Introduction	2
1.1	Contexte	2
1.2	Présentation du projet	2
2	Étude de l'existant	3
2.1	Wireguard	3
2.2	Verifiable Credentials	3
2.3	Aries Hyperledger	5
2.3.1	Aries Cloud Agent	6
2.3.2	Aries Mobile Agent React-Native	7
2.4	QEMU	7
2.5	NEmu	7
3	Scénarios fonctionnels	7
4	Architecture et implémentation	8
4.1	Réseau virtuel NEmu	8
4.2	Cloud Agent du Serveur BlockChain	8
4.3	Cloud Agent du Serveur WireGuard	8
4.4	Mobile Agent du Client WireGuard	8
5	Analyse du fonctionnement & Tests	9
6	Conclusion	9
6.1	Limitations	9
6.2	Extensions	9
7	Bibliographie	9
8	Annexe	9

1 Introduction

1.1 Contexte

De nos jours l'emploi de réseaux privés virtuels (VPN, Virtual Private Network) est de plus en plus démocratisé. On s'en sert généralement pour masquer son adresse ip, ou pour créer un canal sécurisé chiffré avec un destinataire. De nombreuses applications, services et protocoles de VPN différents existent, que ce soit sur Ordinateur ou Smartphone.

En tant qu'utilisateur, se connecter à un serveur VPN nécessite d'en connaître son adresse ip ainsi qu'échanger des clés de chiffrement (symétrique ou asymétriques) avec celui-ci.

Cependant pour garantir l'authenticité de la connexion, et identifier le serveur/client avec lequel le tunnel VPN s'établit, on peut recourir aux Certificats. Lorsqu'un client veut se connecter à un serveur, il lui demande son certificat afin de prouver son identité. Ce certificat étant délivré par un tier de confiance, à savoir l'Autorité de certification, le client peut donc avoir une preuve de l'identité du serveur. Cependant, ce système étant centralisé, il dépend des autorités de certification et peut présenter différents problèmes. D'une part la possible censure ou contrôle de la part de cette autorité, mais aussi le fait que si jamais cette autorité est attaquée, alors tous les certificats délivrés par celle-ci sont compromis. Pour pouvoir contrer ces difficultés, de nouvelles méthodes basées sur la décentralisation des autorités de certifications existent, comme par exemple basées sur la Blockchain.

1.2 Présentation du projet

Déploiement d'un VPN sur des équipement mobiles ou IoT est un projet dont le but est de réussir à installer et configurer le VPN WireGuard sur un client Android. Cet Android sera une machine virtuelle qui s'appuie sur les logiciels QEMU et KVM, et membre d'un réseau virtuel NEmu.

WireGuard est un VPN nécessitant des couples de clés publique/privée de chiffrement asymétrique. Afin de garantir l'authentification et l'identité du serveur VPN WireGuard auquel le client Android se connectera, cela nécessitera l'emploi de Verifiable Credentials (VC), un équivalent des Certificats mais dont l'autorité de certification repose sur la décentralisation, à savoir un noeud de blockchain déjà existant. Le projet Aries Hyperledger soutenu par la fondation Linux permet de développer des mécanismes basés sur les VCs. Nous utiliserons donc Aries Mobile Agent sur le client Android pour communiquer avec un Aries Cloud Agent relié à un réseau Hyperledger Indy dont le rôle est de délivrer les VC. Nous utiliserons également un Aries Cloud Agent sur le serveur WireGuard afin de communiquer avec le noeud Indy et récupérer un VC.

2 Étude de l'existant

2.1 Wireguard



WireGuard est un VPN fonctionnant sur la couche 3 du modèle OSI. Il est implémenté comme une interface réseau virtuelle du noyau pour Linux. Il est pensé pour remplacer les VPN IPsec et ceux basés sur TLS comme OpenVPN, tout en se voulant plus sûr, performant et facile d'utilisation de part son implémentation en moins de 4000 lignes de code facilement compréhensibles et vérifiables sur les systèmes Linux.

Il a été initialement déployé pour les systèmes Linux, mais il dispose maintenant de portages sous Android, Windows ou macOS principalement. Pour un client, WireGuard ne nécessite qu'un échange de clés publiques et d'informations de connexions comme l'adresse ip du serveur hôte. De courtes clés statiques pré-partagées en Curve25519 (basé sur Diffie-Hellman) sont utilisées pour l'authentification mutuelle. Le protocole proposé assure une confidentialité forte ainsi qu'un haut degré de dissimulation d'identité. Au niveau du transport, il utilise le cryptage authentifié ChaCha20Poly1305 pour l'encapsulation des paquets en UDP. La clé publique construite avec Curve25519 est utilisée pour créer une interface réseau qui lui est associée. Ces interfaces font partie de la 'cryptokey routing table', qui peut être configuré et à laquelle on peut ajouter des règles de routage supplémentaires. Quand des paquets sont envoyés vers une machine par un tunnel Wireguard (donc, par l'interface dans la cryptokey routing table), ils sont chiffrés avec la clé publique de la machine qui reçoit le paquet.

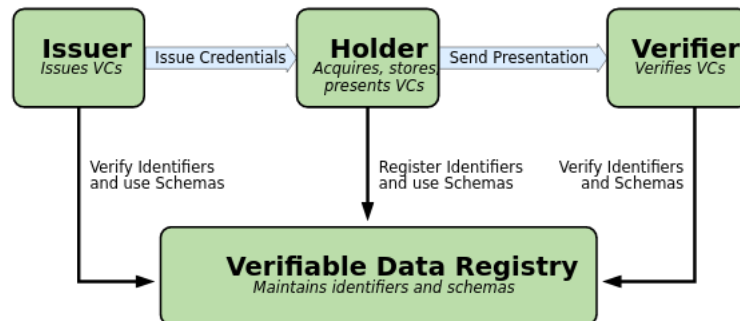
Il est cependant à noter que les couples de clés asymétriques publiques/privées générées par WireGuard à l'aide de son algorithme ne sont pas adaptables aux Certificats X509 de part leur format (32 caractères 64 bits).

Nous utiliserons pour notre projet la version mobile Android de WireGuard côté client, ainsi que la version Linux de WireGuard côté serveur.

2.2 Verifiable Credentials

Un "credential" correspond à une accréditation, un certificat ou une référence. Par exemple dans le monde physique cela peut être une pièce d'identité, un passeport, un permis ou encore un diplôme universitaire. Avec l'avènement de la BlockChain, il est maintenant possible de représenter numériquement et de manière sûre un credential. On parle alors de Verifiable Credential, car grâce à la BlockChain il est possible de les vérifier facilement et rapidement. En Novembre 2019 le W3C a entamé une procédure de normalisation en publiant une recommandation sur les Verifiable Credentials, dans le but de garantir un format générique reconnu mondialement. Il s'agit donc d'une technologie novatrice et en pleine construction. Il y est stipulé qu'un détenteur (ou "Holder") de VC peut générer une présentation à partager à quelqu'un voulant une preuve d'accréditation selon certaines caractéristiques, et ce sans forcément transmettre l'entièreté des caractéristiques de son VC. Par exemple pour obtenir un

service nécessitant d'être majeur, une preuve de notre âge peut nous être demandée. Il n'est alors pas obligatoire de transmettre le champ de son VC d'identité contenant sa date de naissance, mais par exemple juste prouver dans la présentation que l'on a "plus de 18 ans". Pour mieux comprendre les différents rôles et informations concernant les VCs, voici ci-dessous le schéma proposé par la W3C :



Des mécanismes de preuves et de signatures numériques sont nécessaires afin d'assurer la protection d'un Verifiable Credential. L'obtention de la validation des preuves peut dépendre de la syntaxe de la preuve, cependant dans le cadre de ce projet les VCs correspondront à des JSON Web Tokens sécurisés par l'utilisation de JSON Web Signatures. Voici ci-dessous un exemple de de VC JWT présenté par le W3C :

[illegible]

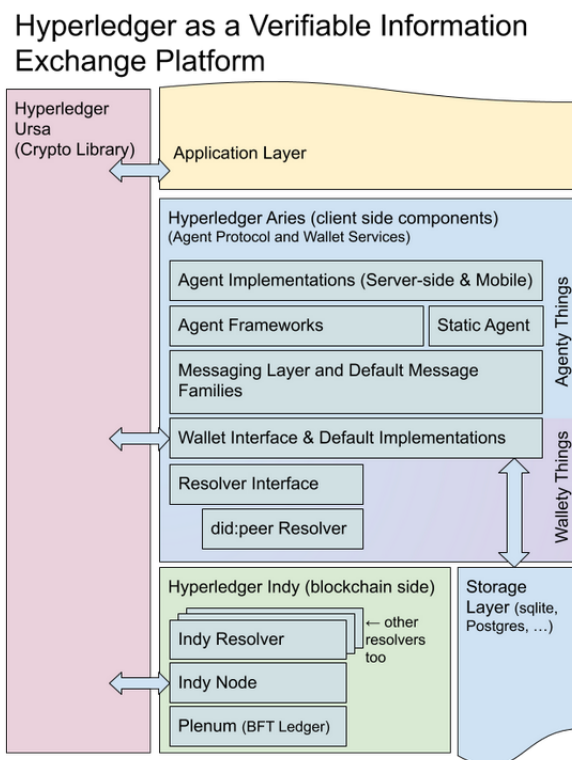
2.3 Aries Hyperledger

Aries Hyperledger est un projet développé par l'Hyperledger Foundation. Hyperledger Foundation est une communauté centrée sur le développement d'outils, bibliothèques et frameworks permettant de déployer des blockchains qui seront majoritairement mises au service d'entreprises.

Il y a différents projets lancés par la Fondation Hyperledger. Aries Hyperledger est le 13ème projet lancé par cette communauté. Aries Hyperledger est une infrastructure permettant l'échange de données en relation à la blockchain ainsi que l'échange de messages peer-to-peer.

Aries Hyperledger inclut plusieurs services dans son infrastructure :

- Une couche interface appelée resolver qui permet de créer et signer des transactions blockchain
- Un wallet sécurisé permettant de garder des secrets et autres informations
- Un système de messagerie encrypté pour l'échange entre clients hors blockchain
- Une implementation des W3C Verifiable Credentials
- Une implementation du Decentralized Key Management System (DKMS)
- Un mécanisme qui permet de construire protocoles et des API



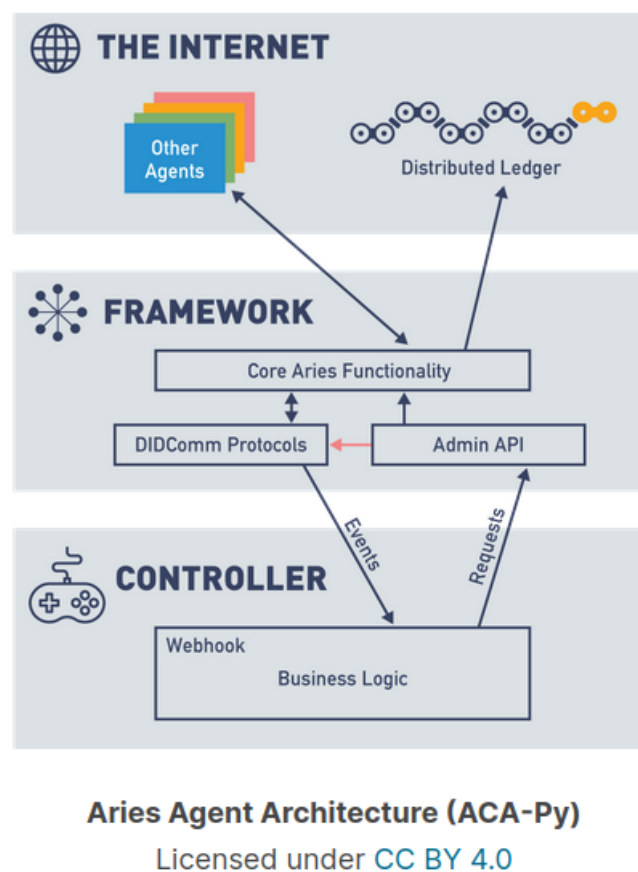
Le plus grand objectif du projet Aries Hyperledger est de pouvoir offrir une infrastructure qui peut travailler et s'adapter à d'autres technologies développées par Indy ou qui se servent d'autres technologies blockchain.

2.3.1 Aries Cloud Agent

Hyperledger Aries Cloud Agent Python (ACA-Py) est un projet visant à servir de base pour construire des agents pouvant utiliser des Verifiable Credentials. Ses protocoles et fonctionnalités de base permettent de délivrer, vérifier et stocker des Verifiable Credentials. Les agents ACA-Py peuvent contrôler des Verifiable Credentials de format Hyperledger Indy AnonCreds et de format W3C, dans ce projet nous nous intéressons au dernier format.

Aries Cloud Agent fonctionne avec des requêtes HTTP et des notifications webhook. Ceci ouvre la possibilité d'écrire un contrôleur qui 'discute' avec notre agent en n'importe quel langage pouvant gérer des requêtes HTTP.

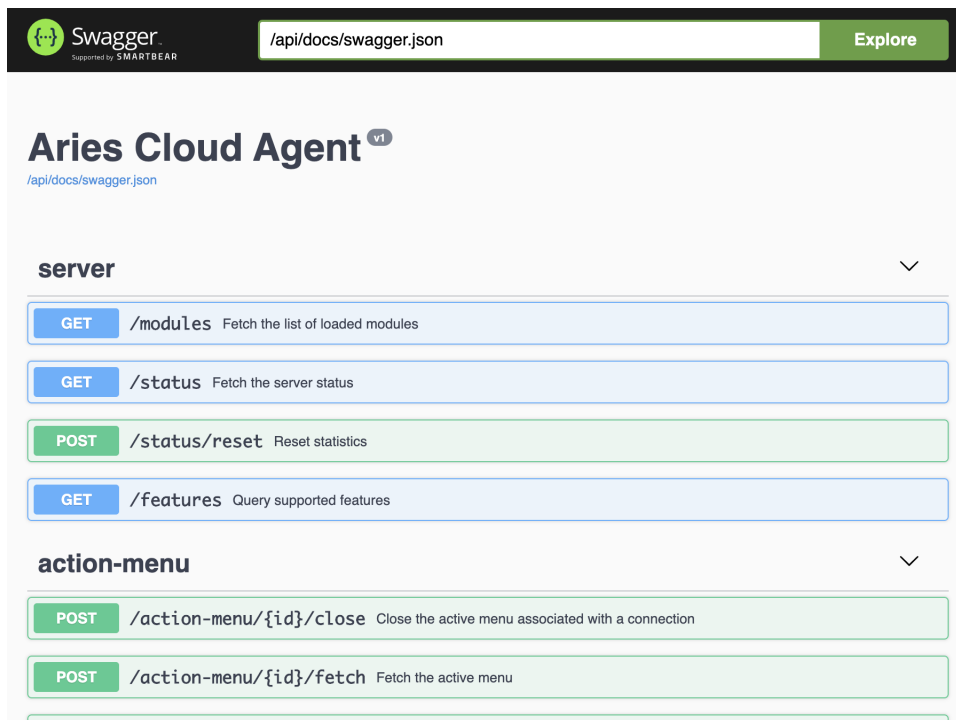
L'Agent Aries Cloud met aussi en œuvre une interface OpenAPI REST pouvant servir à comprendre comment les protocoles dans notre agent fonctionnent. Le contrôleur peut donc utiliser ceci pour gérer le comportement de l'agent. Voici un résumé de l'architecture de l'agent Aries Cloud :



Dans notre architecture, nous pouvons voir que notre Agent communique avec un Distributed Ledger, donc un Blockchain. Pour précision, dans notre projet nous utilisons Von Network comme Blockchain.

Pour contrôler et faire des actions dans l'API il suffit juste de retrouver le endpoint qui fera l'action qu'on veut, le sélectionner et rentrer les champs nécessaires s'il en faut. C'est un outil très utile pour comprendre et développer son agent.

Voici l'API d'un agent Aries Cloud :



2.3.2 Aries Mobile Agent React-Native

TODO

2.4 QEMU

QEMU est un logiciel libre pouvant émuler un processeur ou une architecture différente. Il peut émuler un système ou juste le virtualiser, dépendant du système de l'hôte. QEMU peut exécuter différents systèmes d'exploitation et leurs applications de manière isolée sur une même machine physique ainsi que simuler les périphériques.

Dans notre projet nous nous sommes servis de QEMU pour émuler deux machines Debian et une machine Android

2.5 NEmu

NEmu est un environnement permettant de mettre en place des réseaux virtuels. NEmu construit une topologie d'un réseau virtuel avec des Machines Virtuelles QEMU. NEmu met en place une API python rendant le travail plus facile. Il y a aussi la possibilité de simuler des dispositifs comme des routers ou des switchs pour mieux administrer le réseau.

Nous avons utilisé NEmu en combinaison avec QEMU pour construire un réseau de machines virtuelles.

3 Scénarios fonctionnels

TODO

4 Architecture et implémentation

4.1 Réseau virtuel NEmu

TODO

4.2 Cloud Agent du Serveur Blockchain

Le serveur Blockchain et le serveur qui est directement relié au réseau de noeuds Blockchain. Dans le cas où on veut avoir notre propre réseau de noeuds Blockchain, le serveur Blockchain sera celui qui déploie ce réseau en utilisant Von Network pour déployer les noeuds Blockchain. Nous avons aussi la possibilité de prendre comme référence un réseau de noeuds déjà existant. Le Serveur Blockchain est donc celui qui est directement relié au réseau de noeuds Blockchain. À cause de ça, c'est ce serveur qui enregistre les utilisateurs auprès de la Blockchain. Il enregistre aussi les schémas et les définitions de credentials. Les Schémas et Définitions de Credentials permettent de mettre en place un format et des champs pour les Verifiable Credentials. Les Verifiable Credentials que nous avons choisi d'utiliser ont deux champs : un champ pour la **clé publique** et un autre pour le nom de la machine qui détient ce Verifiable Credential. Il y a aussi un champ contenant la signature du Verifiable Credential. TODO ADD PICTURE OF CREDENTIAL

4.3 Cloud Agent du Serveur WireGuard

Le Serveur Wireguard est une machine Debian qui hoste un service VPN Wireguard. Dans cette machine il y a aussi un Agent Cloud. Cet Agent Cloud est capable de communiquer avec d'autres agents, notamment dans ce cas l'Agent du Serveur Blockchain et l'Agent du Client Wireguard. Il aura de différentes interactions avec chacun des deux agents.

Avec l'Agent Server Blockchain il aura des interactions qui auront comme objectif la réception de Verifiable Credentials.

Les échanges avec le Client Wireguard seront différents. En premier, le Serveur Wireguard doit répondre aux requêtes de preuves de la part du Client Wireguard. Le Serveur Wireguard doit en premier lieu produire une Verifiable Presentation puis l'envoyer. Cette Verifiable Presentation dépend de la requête du Client Wireguard, c'est à dire, la construction et les champs inclus dans cette Presentation dépendent de ce que le Client demande. Normalement, il va envoyer une Verifiable Presentation contenant sa clé publique et son 'nom'.

Dans la requête de preuve, le Client inclut sa clé publique Wireguard. Une fois que notre présentation est bien validée, le client aura récupéré notre clé publique. Finalement, le VPN Wireguard peut être mis en place des deux côtés.

Nous avons mis en place une interface graphique en python pour pouvoir gérer les différentes interactions et événements.

TODO : AJOUTER IMAGE INTERFACE +

4.4 Mobile Agent du Client WireGuard

Le Client Wireguard est donc, une machine Debian qui a comme finalité se connecter sur le VPN du serveur Wireguard.

Comme pour le Serveur Wireguard, le Serveur Blockchain doit délivrer un Verifiable Credential avec la clé publique Wireguard du client dedans.

Quand le Client voudra se connecter sur le Serveur Wireguard, il devra en premier faire une requête de preuve auprès du Serveur Wireguard, contenant la clé publique du client et demandant entre autres, la clé publique du serveur. Celui-ci envoie une Verifiable Presentation avec sa clé publique Wireguard. La requête du client contenait aussi sa clé publique, donc quand la Presentation est validée, le serveur et client Wireguard ont tous les deux les clés de l'autre. Le VPN peut être mis en place.

TODO - explique limitations

5 Analyse du fonctionnement & Tests

TODO

6 Conclusion

TODO

6.1 Limitations

TODO

6.2 Extensions

TODO

7 Bibliographie

- WireGuard : <https://www.wireguard.com/> (consulté le 06/03/2022)
- Verifiable Credentials : <https://www.w3.org/TR/vc-data-model/> (consulté le 06/03/2022)
- Aries Cloud Agent : <https://github.com/hyperledger/aries-cloudagent-python> (consulté le 10/03/2022)
- QEMU : <https://www.qemu.org/> , https://wiki.qemu.org/Main_Page
- NEmu : <https://gitlab.com/v-a/nemu>

8 Annexe

TODO