Woleet SAS.
2 bis rue de la Châtaigneraie
35577 Cesson-Sévigné
woleet.io

UFR Informatique et Informatique
Campus de Beaulieu
bâtiment 12D
35042 RENNES Cedex
www.istic.univ-rennes.fr

Master Informatique 2e année

parcours Génie Logiciel en alternance

2016-2017

Mémoire Professionnel

Ancrage de données dans la blockchain Bitcoin

Auteur : Maxime Tissier

Maître d'apprentissage : Vincent Barat

Remerciements

Je tiens à remercier tous les membres de Woleet avec lesquels j'ai passé une excellente année. Je tiens à dire que cela a été un véritable plaisir de travailler avec eux.

Vincent Barat pour la patience et la détermination dont il fait preuve en encadrant mon travail pendant plus d'un an; ainsi que pour l'exigence quant à la qualité de rendu qu'il a su m'inculquer tant bien que mal, m'immunisant alors contre le moindre espace mal placé. Je tiens à dire que cela à été un véritable plaisir de travailler avec lui.

Gilles Cadignan pour l'ensemble des connaissances et la passion qu'il a su me transmettre concernant le Bitcoin et les cryptomonnaies, m'ouvrant alors les portes d'un tout petit monde ou tout est possible et reste encore à faire.

Je tiens aussi à remercier tous les membres de l'ISTIC pour leur disponibilité et toutes les notions abordées durant cette année.

Résumé

Dans ce rapport, nous aborderons dans un premier temps les activités de l'entreprise et les technologies sur lesquelles sa solution repose. Finalement, nous détaillerons l'ensemble des projets sur lesquels j'ai été amené à travailler durant cette année d'alternance.

Sommaire

Introduction.....5
Chapitre I : Contexte.....6
1. Présentation de Woleet.....6
2. À propos de la preuve d’existence sur la blockchain Bitcoin.....7
3. Environnement de travail.....8
4. Service d’horodatage de Woleet.....9
Chapitre 2 : Projets.....10
1. Outils relatifs à la vérification.....10
1.1. woleet-weblibs.....12
1.2. woleet-widget.....15
2. Support de la signature.....16
2.1. Woleet-clients : ajout du module de signature.....17
3. Le problème de la preuve d’identité.....19
3.1. woleet-backendkit.....20
Conclusion.....22

Introduction

Le Bitcoin et la technologie blockchain sous-jacente a créé un nouveau-paradigme de la confiance sur internet : en effet, il est désormais possible de se passer d'intermédiaire pour effectuer des transactions financières.

Toutefois les cas d'usages de ce gigantesque registre public ne s'arrête pas là…

Durant cette année d'alternance j'endosserai le rôle de développeur « full-stack » qui me permettra d'explorer les multiples facettes du métier : sous la supervision de Vincent Barat (directeur technique), ma mission en entreprise consistera à :

- contribuer à la mise à disposition d'un ensemble d'outils open source ;

- maintenir et étendre les fonctionnalités des services existants.

Chapitre I : Contexte

1. Présentation de Woleet

Woleet est une entreprise créée en février 2016 et incubée à Télécom Bretagne, elle a pour objectif de démocratiser l'accès à la preuve d'existence de documents[1] et à la signature numérique de données[2] en se basant sur la technologie blockchain[3].

Pour cela, elle propose une plateforme multi-clients en mode SaaS, facilement accessible, permettant d'ancrer[4] dans le registre Bitcoin[3] des preuves d'existence de documents pour faire valoir leur propriété ou leur intégrité.

Partant du postulat que la blockchain Bitcoin est incorruptible, il est alors possible de fournir un moyen sûr de prouver l'authenticité de n'importe quel type de donnée numérique. Ainsi, Woleet n'est pas un tiers de confiance, mais un facilitateur d'accès à une technologie de confiance décentralisée.

Les cas d'usage sont nombreux : horodatage et certification de documents administratifs ou de diplômes, réconciliation de partenaires commerciaux sur l'authenticité d'une donnée, protection de la propriété intellectuelle…

---

1 – Preuve d'existence de document : certifier qu'une donnée existait bien à une certaine date.
2 – Signature numérique de données : mécanisme permettant de vérifier l'authenticité d'une donnée (voir Signature numérique)
3 – Blockchain : voir point suivant.
4 – Ancrage de données : écriture de données extra-transactionnelles dans une transaction Bitcoin.

2. À propos de la preuve d'existence sur la blockchain Bitcoin

Avant d'entrer dans le vif du sujet, il semble nécessaire de définir ce que sont la blockchain et le Bitcoin :

La blockchain est une technologie de stockage décentralisée. Dans le cas spécifique de bitcoin il s'agit d'un registre public contenant l'ensemble des **blocs** (de données) ordonnés et datés dans lesquels les transactions sont inscrites, chaque bloc contenant l'empreinte numérique (hash) du bloc le précédant : ils sont donc **chaînés** (on ne peut pas en changer un sans invalider tous ceux qui suivent).

Le Bitcoin est un système décentralisé d'échange de jetons (token), il est constitué d'un ensemble de blocks, chacun contenant un ensemble de transactions. C'est le fait que ces blocks soient chaînés qui garantit que les jetons échangés ne soient pas utilisés deux fois.

Bitcoin permet d'associer 80 octets à une transaction, il est donc possible de prouver qu'une donnée existait bien à une certaine date.

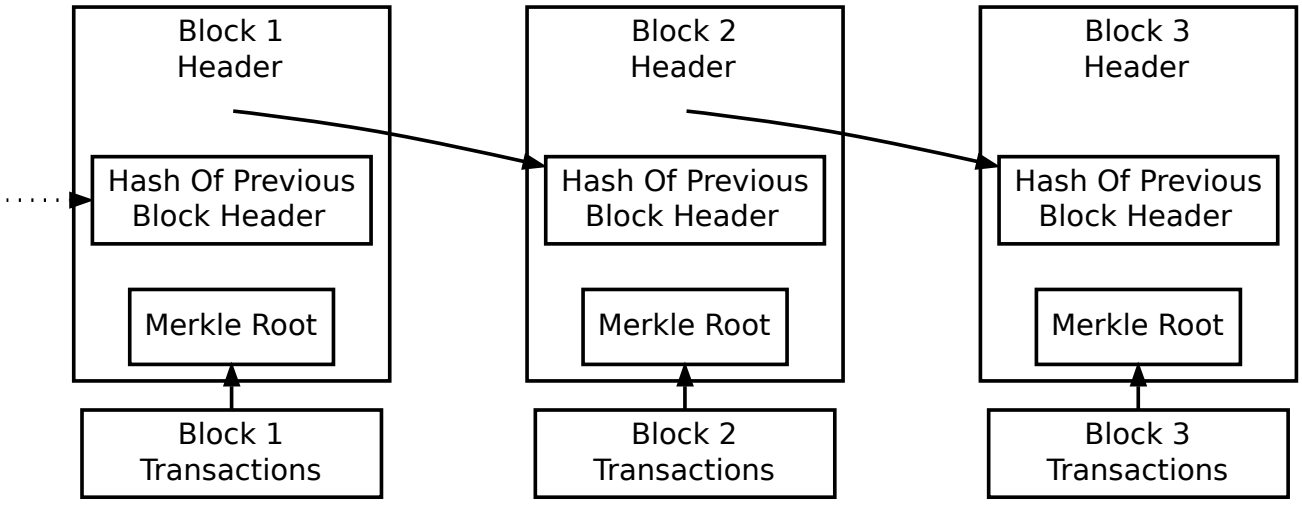


Illustration 1: blockchain Bitcoin simplifiée source: bitcoin.org

3. Environnement de travail

Pour la gestion de projets nous utilisons Git ainsi que Gitlab pour le déploiement automatique de certains projets et la gestion des images docker.

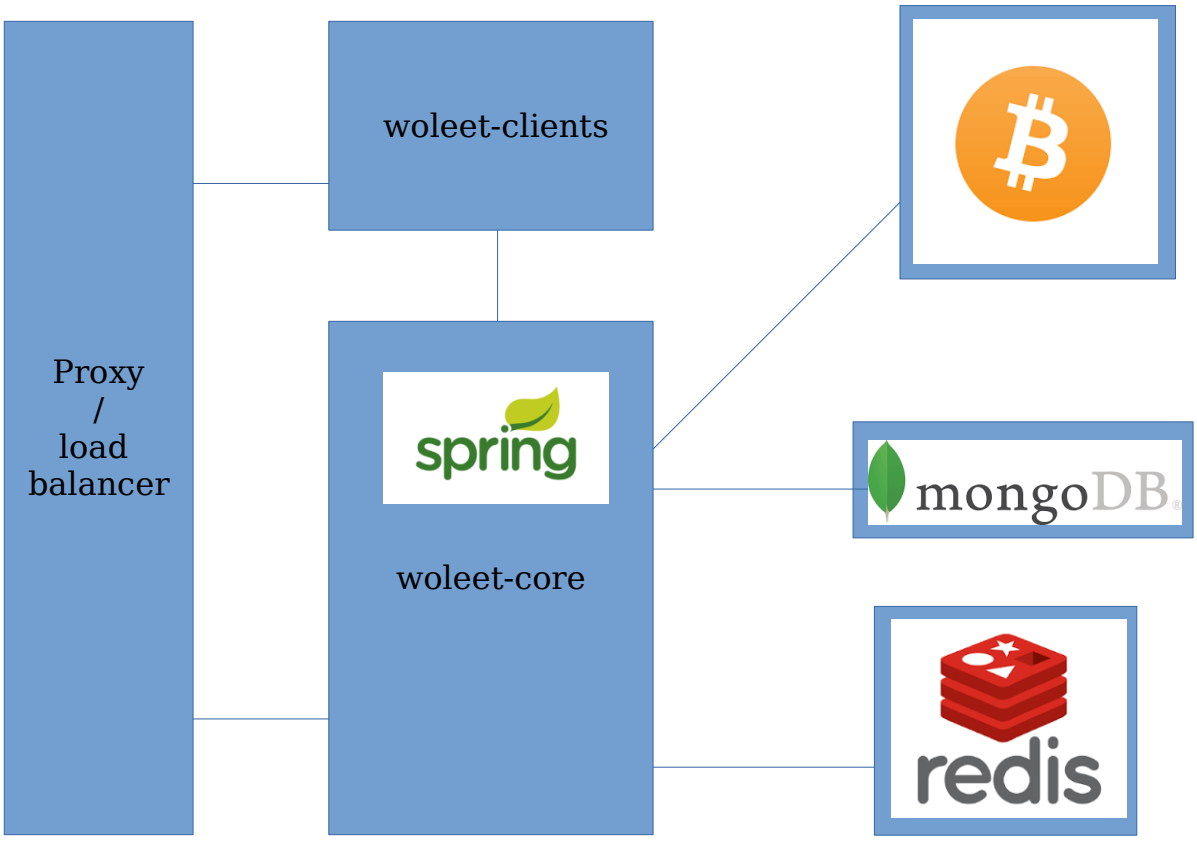
Les taches sont assignées et suivies avec JIRA, tandis qu'une documentation interne est maintenue avec confluence.

Pour les développements, nous utilisons principalement IntelliJ.

4. Service d’horodatage de Woleet

Le service d’horodatage (timestamping) de documents que propose Woleet est une API permettant en autres d’envoyer des empreintes numériques de fichiers et de récupérer, après traitement, un reçu contenant les informatisations nécessaires pour constituer une preuve d’existence.

L’API est servie par le projet « woleet-core » et est utilisable via une interface web « woleet-clients »



Note : ce schéma est une version simplifiée de l’architecture de Woleet, les différents services ayant connu des modifications a des fins de passage à l’échelle.

Chapitre 2 : Projets

L'objectif de Woleet durant cette année d'alternance était de parvenir au passage à l'échelle de sa plateforme, d'y ajouter des fonctionnalité afin de proposer à ses clients la signature de documents en plus de leur horodatage ainsi que de leur fournir un ensemble d'outils facilitant l'utilisation de sa solution.

Mon travail portait sur l'écriture de ces outils, et l'implémentation de la signature.

1. Outils relatifs à la vérification

Woleet souhaite développer des solutions open-source afin de faciliter la vérification des ancres que la société émet (ainsi que toute preuves d'existence de document émise au format Chainpoint).

Ces vérifications doivent pouvoir se faire sans indépendamment des services proposés par Woleet.

Le « reçu Chainpoint » est une partie de la preuve, il contient les données nécessaires qui, combinées avec celles ancrées dans la blockchain, constitue une preuve de document. Il est constitué, notamment, d'un arbre de Merkle (structure de donnée utilisée a des fin de passage à l'échelle au niveau des transactions) de l'identifiant de la transaction Bitcoin, et d'un « merkle_root » qui correspond à la donnée inscrite avec la transaction.

Les « feuilles » de l'arbre de Merkle correspondent aux empreintes des documents, et la racine (merkle_root) le résultat de hashs récursivement effectués sur chaque paire de feuilles.

```
       ABCDEEEE .........racine de l'arbre de Merkle
      /        \
     ABCD        EEEE
    /    \      /
   AB    CD    EE .......dans le cas d'un nombre impair de feuilles sur un niveau
  /  \  /  \  /                    la feuille est appairée à elle-même
  A  B  C  D  E .........Empreintes de fichiers
```

Exemple de reçu Chainpoint :


```
{
    "header": {
        "chainpoint_version": "1.0",
        "merkle_root":    "6b5c97f0bb9bc96136f6b82adf71243f812c420af62b2539235c4a3155402121",
        "tx_id":          "29f4e54688390e95f36706d7c512422b45cfc2dcd609607b1cdf090162ff2571",
        "hash_type": "SHA-256",
        "timestamp": 1500628951
    },
    "target": {
        "target_hash":    "b7c0ce7a1d7e18755a4924007e3b97691fc0534fde422c48ecd9baf0daf566c5",
        "target_proof": [
            {
                "parent": "b9422307f1907650d8b97c915a1be1edfed0dd85927706bbfa5d0013337365f9",
                "left":   "b7c0ce7a1d7e18755a4924007e3b97691fc0534fde422c48ecd9baf0daf566c5",
                "right":  "7a44fb95a771b0d2786b2f355b7bce9d371bec087bb50beec420ba33d64228ed"
            },
            {
                "parent": "55340aa45094ab253151cbbc5fa3d73035646172f982a259de20ac24ada8498c",
                "left":   "b9422307f1907650d8b97c915a1be1edfed0dd85927706bbfa5d0013337365f9",
                "right":  "79f81401dea203e615b90e4f306ae9d44017110f8a3ffc94299da1dc975b4aae"
            },
            {
                "parent": "2b3e4e06399991e7dc8ff7d6d17ec95f1c95b97ed1d14b8d97cb9bc85b4c6c1c",
                "left":   "55340aa45094ab253151cbbc5fa3d73035646172f982a259de20ac24ada8498c",
                "right":  "7594af5bb8caaddaf5a92ba3a8798d3b83d029a20fb3608fab3ebab413a0891e"
            },
            {
                "parent": "ef7ad0a74a54d083cf79512da388d72db1625678d7fcd1d25694507ef9778a65",
                "left":   "2b3e4e06399991e7dc8ff7d6d17ec95f1c95b97ed1d14b8d97cb9bc85b4c6c1c",
                "right":  "2b3e4e06399991e7dc8ff7d6d17ec95f1c95b97ed1d14b8d97cb9bc85b4c6c1c"
            },
            {
                "parent": "6b5c97f0bb9bc96136f6b82adf71243f812c420af62b2539235c4a3155402121",
                "left":   "d65f474f2dbbc82ff07003c0a9ec7b64f3e287464fd7264b4c4c2fc7aa0b45bf",
                "right":  "ef7ad0a74a54d083cf79512da388d72db1625678d7fcd1d25694507ef9778a65"
            }
        ]
    }
}
```

1.1. woleet-weblibs

Dépôt git: https://github.com/woleet/woleet-weblibs

Bibliothèque JavaScript open-source et modulaire permettant la vérification des preuves émises par Woleet indépendamment de ses services.

Le but de cette bibliothèque est de fournir à nos clients une solution clé en main pour :

- vérifier de preuves d'existence de donnée émises au format Chainpoint, le hachage de fichier
- hacher un fichier dans un navigateur ;
- vérifier une signature bitcoin (ECDSA).
- vérifier une identité associée à une signature bitcoin *

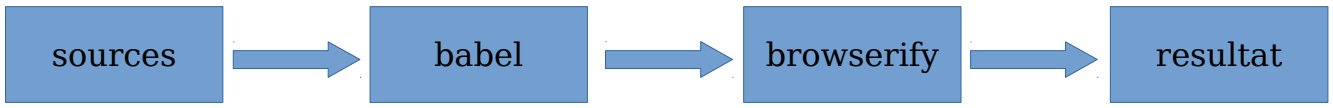
Cette bibliothèque sera par la suite intégrée au client web de Woleet (afin de factoriser les fonctions de hachage de documents et de vérification de reçus).

développement

Comme dans la plupart des projets à dominance JavaScript, j'avais pour ce projet une grande liberté quant aux choix de l'implémentation et des bibliothèques à utiliser.

Cette bibliothèque en utilise plusieurs autres :

elle embarque *bitcore-message* pour la vérification de signatures et *crypto-js* pour le hash de fichiers ; *babel* et *browserify* sont utilisés pour le portage du JavaScript moderne (ES6) vers l'ancien et pour permettre l'utilisation de modules nodejs dans un navigateur, respectivement.



La bibliothèque *jasmine* est utilisée pour les tests, les mêmes tests sont utilisés pour nodejs et le navigateur.

Architecture

Consiste en 7 fichiers sources :

- api : fonction concernant les appels réseaux (récupération des transactions notamment)

- chainpoint: fonctions de vérifications d'arbres de Merkle et de reçus Chainpoint

- crypto : interface la fonction sha256 de la bibliothèque crypto-js selon la bibliothèque native ''crypto'' de nodejs

- hashfile : définis une classe de hachage de fichier servant à la fois à empiler les taches de hash en cours (évite les dépassements mémoire/crash de navigateurs) et sélectionne pour chaque fichier de la pile la méthode la plus adaptée pour le hash du fichier :

  - api crypto du navigateur (si api existante et fichier < 500Mo)
  - via un Web Worker si fichier > 500Mo
  - dans le contexte d'exécution courant sinon

- hashfile-worker : fonction de hash pour Web Workers

- signature : vérification de signature et d'identité

- verify : vérification des preuves d'existences de documents (avec ou sans signature)

Travail réalisé

difficultés rencontrées :

hash de fichier dans un navigateur

Le hash de fichiers dans un navigateur est une opération périlleuse quand le fichier dépasse une certaine taille. Dans presque tous les cas le hash d'un fichier de plus de 100Mo conduit systématiquement à un crash de la page (dû à la forte consommation de mémoire de l'interpréteur JavaScript), la solution trouvée pour pallier à ce problème est d'utiliser des Web Workers (seule façon de lire un fichier séquentiellement mais impose un contexte d'exécution séparé). Toutefois le support des Web Workers varie selon les navigateurs.

1.2. woleet-widget

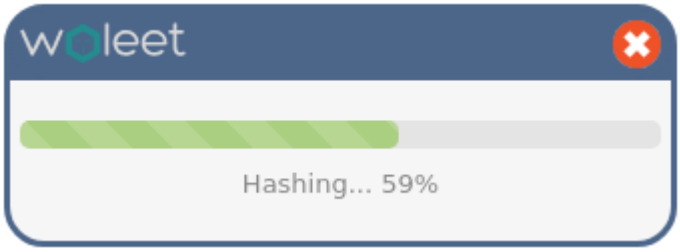
Dépôt git: https://github.com/woleet/woleet-widget

Page de démonstration : http://identity.woleet.io/

Outil graphique ayant pour but d'être facilement intégré à un site web et facile d'utilisation.

développement

Cet outil repose essentiellement sur la bibliothèque woleet-weblibs et se compose de deux classes : l'une gérant les différents états, l'autre permettant de générer du code HTML (et de maintenir une référence vers les différents éléments). Le code généré est injecté sur la page, dans les éléments HTML répondant à certains critères.



2. Support de la signature

Comme évoqué en introduction, il est souhaitable de pourvoir – en plus de créer une preuve d'existence de document – ancrer une signature de document dans la blockchain et ainsi pouvoir en identifier la source.

Les cas d'usages relatifs a cette nouvelle fonctionnalité sont multiples :

- Signature de documents :

Un document pourra être envoyé à de multiples acteurs auxquels il sera demandé une signature.

- Identifier (et certifier) la source d'une donnée :
  - pour une donnée quelconque venant de l'IoT ;
  - pour des documents pouvant servir de preuve de résidence, de factures, etc.
  - pour les émetteurs diplômes ;

2.1. Woleet-clients : ajout du module de signature

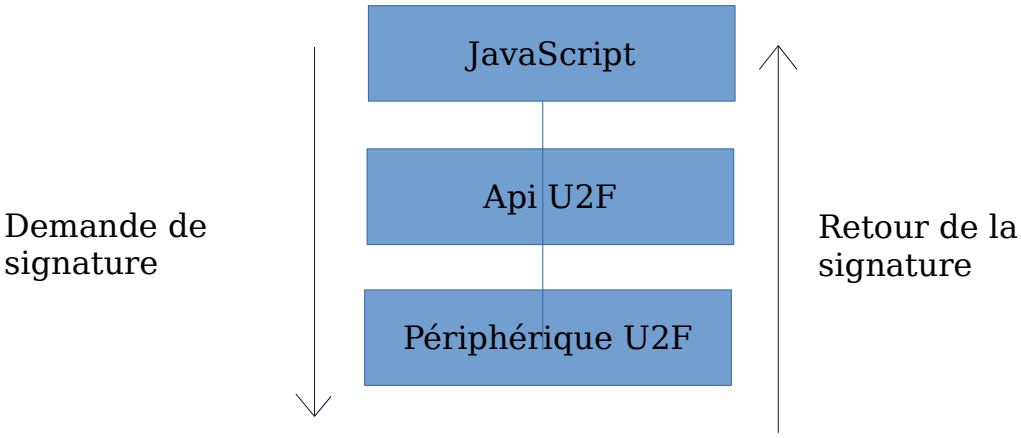
Illustration 2: Ledger Nano S

Ma mission sur ce projet (en plus du maintien de l'existant) consistait à l'ajout d'un module permettant à l'utilisateur de signer le hash d'un document en utilisant un appareil cryptographique (Ledger).

Ledger est une entreprise fabriquant des portefeuilles matériels pour cryptomonnaies, leur intérêt étant de pouvoir signer une transaction Bitcoin sans jamais exposer la clé privée de l'utilisateur. Nous l'utilisons afin de signer le hash d'un document.

La communication entre le navigateur et l'appareil se fait via l'API U2F (Universal 2nd Factor) du navigateur (Chrome/Opera), cette dernière est une norme conçue pour permettre a un utilisateur de se connecter sans avoir à utiliser de mot de passe, à la place en utilisant un mécanisme de signature à la place.

La difficulté de cette tache est de prendre en compte l'état de connexion de l'appareil : il n'est en effet pas possible de savoir s'il est branché ou non, de même, sa déconnexion ne déclenche aucun événement. Une autre difficulté a été de comprendre comment fonctionnait la communication entre le navigateur et l'appareil via U2F : Ledger détourne l'U2F pour communiquer avec ses produits en utilisant son propre protocole.



futur développement

Demande de signature :

Actuellement, Woleet développe une solution pour que ses clients puissent demander la signature d’un document à une tierce partie.

3. Le problème de la preuve d’identité

Une fois en mesure de signer un document, il est nécessaire de pouvoir s’assurer de l’identité de son émetteur. En effet rien n’empêche par exemple un étudiant de générer une paire de clés et de prétendre qu’elles appartiennent à une université.

Il faut alors trouver un moyen de lier une identité Bitcoin à *quelque chose* capable d’attester de l’identité d’un acteur.

La solution retenue par Woleet, est de s’appuyer sur les autorités de certifications existantes (faute de gestionnaire d’identité décentralisée au point et largement utilisé). Ces fournisseurs de certificats sont ceux utilisés pour sécuriser les communications sur le web (en liant une clé cryptographique aux informations d’une organisation).

Lier une adresse bitcoin et une identité peut alors se faire en associant les informations contenues dans un certificat SSL et une adresse Bitcoin (clé publique).

Pour prouver que l’on est bien détenteur d’une adresse Bitcoin (avec laquelle des documents ont étés signés, par exemple), le client doit exposer une URL capable de signer des données aléatoires.

3.1. woleet-backendkit

Dépôt git: https://github.com/woleet/woleet-backendkit

Page de démonstration : http://identity.woleet.io/identity

Le but de ce projet est de fournir aux clients de l'entreprise un moyen facilement configurable et déployable leur permettant de :

- gérer leur identité Bitcoin (paire de clés) de façon simple
- signer des hashs en utilisant leur identité Bitcoin en appelant un endpoint interne.
- Exposer un endpoint d'identité permettant a quiconque de vérifier leur identité.

Développement

Comme pour weblibs (1.1) cette application embarque *bitcore-message* pour la signature. Elle consiste en un serveur nodejs (ou deux selon les paramètres), prenant divers paramètres en entrée (chemin du certificat SSL, restauration de clé privé…) et peut-être utilisé avec ou sans docker.

Le endpoint de signature étant privé il est possible de définir un port d'écoute différent du port de preuve d'identité : pour cela nous utilisons une factory prenant en entrée une liste de endpoints à définir et créant une instance de serveur prêt à l'écoute.

travail réalisé

Une fois l'application finalisée, je me suis intéressé aux performances de celle-ci et ait réalisé un benchmark de l'application, une opération de signature est assez lente, même en langage compilé, en JavaScript, cela donnait environ 40 ops/s avec 100 connexions concurrentes, l'utilisation de la bibliothèque cluster de nodejs a permis de passer la barre des 200 ops/s avec des temps de réponse moyen passant respectivement de 900 ms à 400 ms.

futur développement

Il est possible de stocker les clés privées (générées ou restaurées) dans une section spécialisée des processeurs Intel récents (Intel Software Guard Extensions), l'utilisation de cette fonctionnalité est une des pistes à explorer pour sécuriser davantage ce programme.

Conclusion

Cette année d’alternance m’a permis d’acquérir de nombreuses compétences. J’ai pu prendre part aux réflexions de l’équipe quant aux différentes façons de résoudre un problème donné ainsi qu’ à la prise de décision quant aux choix des technologies à utiliser en ayant à garder à l’esprit durant le développement d’un projet, les conditions permettant de rendre le projet facile à maintenir et à déployer et à s’approprier.

En définitive l’ensemble des projets réalisés m’a permis d’acquérir de l’expérience sur le développement logiciel et une technologie au futur très prometteur.