DEPARTMENT DE SCIENCES ET INGÉNIERIE

PROJET DE RECHERCHE

Implémentation d'un Smart-Contract lié à l'Identité et de la Balance des Interactions

Autheurs:
Messilva MAZARI
Chan Yeong HWANG
Karima SADYKOVA
Elina JANKOVSKAJA

Superviseurs:
Nour EL MADHOUN

En collaboration avec
Daniel MALDONADO-RUIZ
de l'EPN(École Polytechnique Nationale de Quito)

March 2022

Contents

1 Contexte du projet

Ce projet découle d'une recherche doctorale intitulée "Decentralised Identities based on Next Generation Autonomous Networks" (Identités décentralisées basées sur les réseaux autonomes de nouvelle génération), qui vise à créer un réseau autonome dans lequel les utilisateurs peuvent stocker des informations sans dépendre d'une tierce partie pour valider les informations stockées. À partir de cela, on va chercher à stocker et à administrer toutes les informations relatives à l'identité de l'utilisateur afin de développer un réseau autonome où les utilisateurs peuvent stocker leurs informations, en utilisant les fonctionnalités de la blockchain [1] comme support de sécurité.

2 Objectifs

- Créer une structure de smart contracts[2] qui peut prendre en charge le stockage des informations d'identité d'utilisateurs et associer ces informations à une identité spécifique sur la blockchain.
- Utiliser la cryptographie à courbe elliptique[3] pour créer une paire de clés publiques et privées pour chaque utilisateur afin de garantir la sécurité des informations de chaque utilisateur.
- Définir un tableau de signatures numériques[4] pour certifier la validation des informations stockées dans le smart contract afin que chaque utilisateur de la blockchain puisse savoir que les informations stockées sont vérifiées.
- Créer une structure de vérification à l'intérieur d'un smart contract afin de connaître la validité et le temps d'expiration de l'information stockée ainsi que le nombre de changements qui ont été effectués dans celle-ci.

[1] Une blockchain est un registre, une grande base de données qui a la particularité d'être partagée simultanément avec tous ses utilisateurs, tous également détenteurs de ce registre, et qui ont également tous la capacité d'y inscrire des données, selon des règles spécifiques fixées par un protocole informatique très bien sécurisé grâce à la cryptographie, *Assemblée nationale*.

[2] Smart contracts: protocoles informatiques qui facilitent, vérifient et exécutent la négociation ou l'exécution d'un contrat. Les smart contracts ont généralement une interface utilisateur et émulent la logique des clauses contractuelles, *Wikipédia*.

[3] Courbe élliptique: proposée indépendamment par Victor Miller et Neal Koblitz en 1985 une courbe algébrique va permettre de générer des clés sécurisés à l'aide d'une équation très simple. Cette équation génère une courbe. On choisit ensuite un point de la courbe et on effectue sa tangente. L'opposé de cette tangente nous donne un point à partir duquel on va encore faire une tangente et ainsi de suite. Finalement, on aura une paire de clés très efficaces mais impossibles à retrouver.

[4] Signature numérique: mécanisme permettant de garantir la non-répudiation d'un document électronique et d'en authentifier l'auteur, par analogie avec la signature manuscrite d'un document papier, *Wikipédia*

3 Introduction du problème

Le problème consiste donc à décentraliser l'accès aux données d'un utilisateur afin de lui permettre de sauvegarder et gérer ses informations de manière décentralisée sur le réseau. On va donc chercher à créer un espace de stockage d'identité.

Dans la Fig. 1, on peut voir les schémas proposés dans le cadre de la recherche doctorale. Bien que la plupart des caractéristiques sont au-delà de la portée de ce projet, cela permet de schématiser la structure du «core blockchain»: une structure privée interne où les paires de clés liées à la sécurité du réseau sont stockées en toute sécurité. La structure appelée «main blockchain» quant à elle est celle où seront stockés les smart contrats conçus dans ce projet.

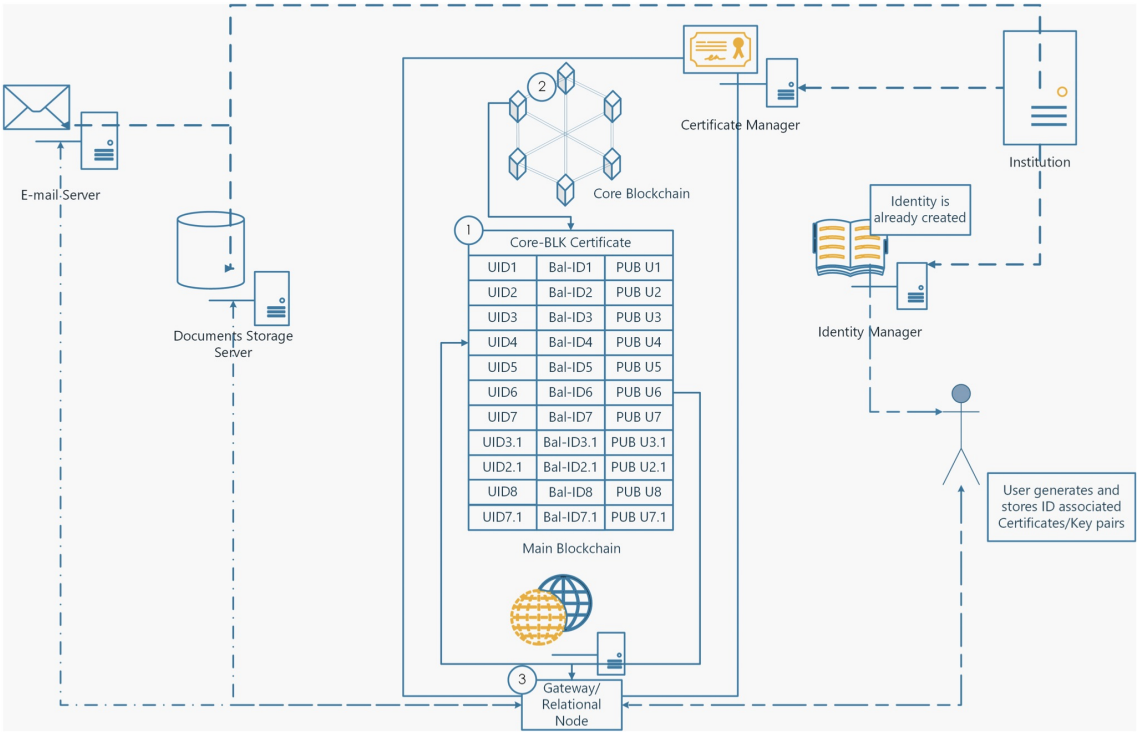

Figure 1: Schémas de la proposition de réseau autonome.

La Fig. 2 nous montre la structure de notre smart contract. Pour réaliser ce smart contract, le processus est divisé en trois étapes.

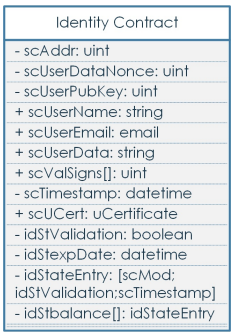

Figure 2: Création du nouvel organigramme des blocs.

4 Réalisation du projet

La réalisation du projet se divise en plusieurs étapes:

La première étape est décrite dans la Fig. 3: l'utilisateur envoie ses informations à travers deux processus différents, l'un où l'utilisateur valide son identité en utilisant ses informations secrètes avec lesquelles sa paire de clés a été créée, et le second où il envoie ses informations (nom, e-mail, données supplémentaires d'identité) pour être stockées dans le smart contract. La partie validation n'est pas destinée à être implémentée dans ce projet, donc comme le montre la Fig. 4, l'acquisition des informations d'identification et le calcul de la clé publique doivent être effectués dans un module externe non lié au smart contract. Lorsque les informations sont stockées dans le smart contract, ce dernier hachera la clé publique et les informations d'identité pour les transformer en une sorte de certificat qu'il va ensuite stocker en enregistrant sa date de création.

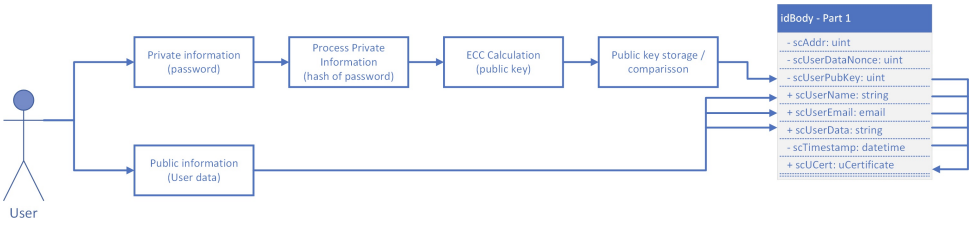

Figure 3: Acquisition des données utilisateur à stocker.

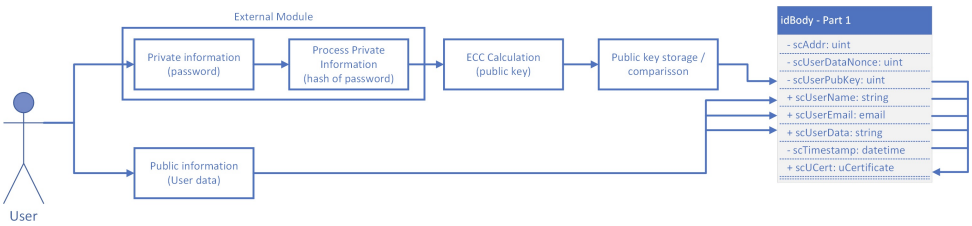

Figure 4: Implementation of the acquisition in this project.

La deuxième étape consiste à (décrite à la figure 5) hacher les informations et signer consécutivement par deux signatures différentes. Les deux signatures sont ensuite stockées dans un tableau pour une validation externe.

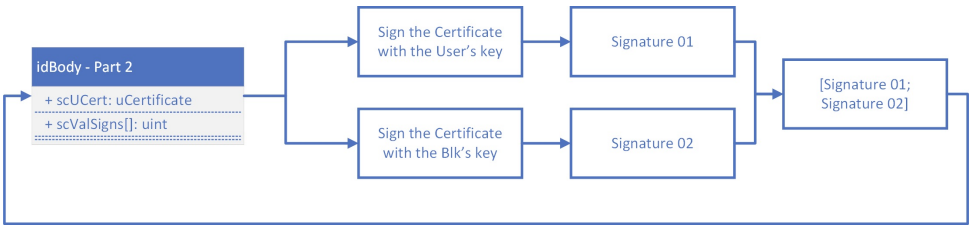

Figure 5: Signing process of the acquired information.

La première signature est effectuée par la clé privée de l'utilisateur (acquise avec le module externe) et la seconde est fournie par la structure de sécurité interne du

réseau, représentée par la Core Blockchain (2) sur la Fig. 1. Dans ce projet, le module externe créera une paire de clés supplémentaire appelée "Network", avec laquelle seront effectuées toutes les deuxièmes signatures des informations dans le smart contract.

Finalement, la Fig. 6 représente les variables dans lesquelles on enregistre les validations et les modifications du smart contract. Ces informations (heure de validation, heure d'expiration, certificat d'utilisateur) sont stockées dans un tableau pour savoir combien de fois le smart contract et les informations qu'il contient ont été modifiés et mis à jour. Cette étape termine la création du smart contract et met à jour la blockchain.

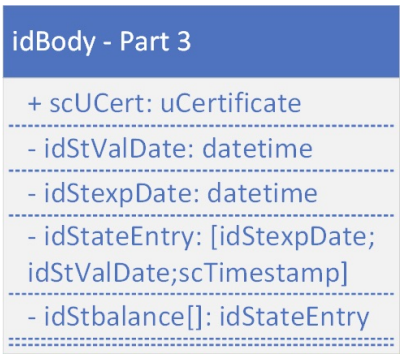

Figure 6: Update of the validation structure.

5 Tâches procédurales

- Créer un prototype du smart contract proposé sur un réseau Ethereum et définir tout le champ nécessaire à sa mise en œuvre.
- Créer un module pour générer des paires de clés publiques-privées en utilisant la cryptographie à courbe elliptique pour fournir au smart contract les informations de l'utilisateur et le processus de signature.
- Implémenter la première étape, en validant la création d'une structure de type certificat qui sera utilisée pour valider l'identité de l'utilisateur.
- Mise en œuvre de l'étape deux, où les informations de l'utilisateur sont signées et validées par le réseau.
- Mettre en œuvre l'étape trois et créer le solde de validation pour chaque utilisateur avec un smart contract.
- Tester la structure complète sur le réseau Ethereum pour mesurer le temps nécessaire à la création d'un smart contract pour chaque utilisateur dans les conditions du projet.
- Tester le nouveau prototype de blockchain, mesurer la puissance de calcul et la comparer aux mesures des blockchains traditionnelles.

6 Avancement sur le projet

Les trois premières semaines de notre projet ont été consacrées à la compréhension de la structure de la blockchain ainsi que de la place des smarts contracts au sein de celle-ci. Nous avons ensuite survolé des notions basiques de cryptographie notamment le chiffrement asymétrique à deux clés. Durant ce temps, nous avons également appris à manipuler Solidity, un langage orienté objet dédié à l'écriture de smart contracts qui est utilisé sur diverses blockchains, notamment Ethereum.

Une fois les contrats créés, il fallait pouvoir les tester et pour cela on a eu besoin de différents outils: Ganache (Fig. 7) pour nous permettre d'effectuer des tests sans devoir acheter des Ether (la monnaie utilisée sur la blockchain Ethereum) et Remix IDE pour compiler et identifier les failles de sécurité de notre contrat plus rapidement et plus efficacement.

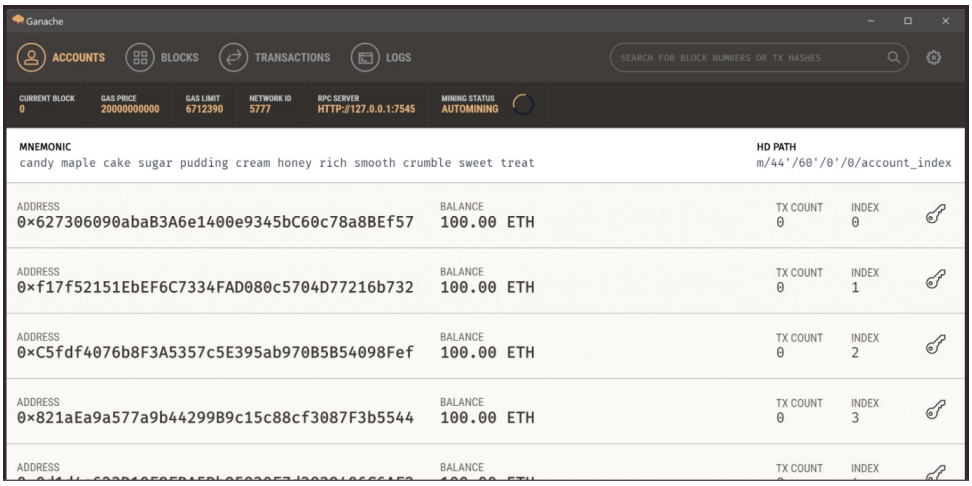

Figure 7: Image de Ganache, l'outil de développement high-end

A la suite de cette étape préliminaire, nous avons pu définir notre projet ce qui nous a permis de commencer à le réaliser. Une fois le contrat solidity fait selon ces spécifications, nous avons cependant rencontré certains problèmes: nous n'avions pas généré la paire de clés et notre interface utilisateur n'était pas prête.

6.1 Clé privée et Clé publique

Nous avons choisi de générer des paires de clés à l'aide de Javascript. Ensuite on a rajouté des fonctions nous permettant de récupérer l'identité de l'utilisateur. Ces nouveaux éléments ont donc complété notre contrat. Cependant, une identité sur la blockchain n'est pas uniquement composée d'une paire de clés et de données personnelles. Comme nous l'avons vu plus haut, celle-ci nécessite aussi une signature électronique et un hachage pour sécuriser notre contrat suivant la norme x.509 (une norme spécifiant les formats pour les certificats à clé publique qui nous a été imposée dans le cadre de ce projet).

Ainsi, nous avons généré deux hachages, un du côté de l'utilisateur et l'autre du côté de Solidity et nous les avons comparés pour parer une éventuelle attaque

venant s'interposer entre nos deux programmes. Par la suite, ces hachages et la clé privée de l'utilisateur permettrons la génération de la signature.

6.2 La signature électronique

La signature nécessite l'ensemble des informations de l'utilisateur ce qui nous as poussé à la produire dans l'interface de l'utilisateur sauf que cela rend difficile l'envoi de cette signature au contrat Solidity. Nous sommes en ce moment entrain de chercher des solutions à ce problème: on envisage par exemple de générer la signature dans le contrat Solidity à la place. Par la suite, nous allons devoir générer une deuxième signature en provenance de la blockchain qui nous permettrait de doublement signer le contrat.

7 Conclusion

Jusqu'à présent nous avons réussi à implémenter un smart contract, générer une paire de clés privée et publique pour l'utilisateur, un certificat répondant aux normes x509. Les difficultés rencontrées nous ont permis d'acquérir des bases solides en cryptographie asymétrique et en langage javascript, ce qui nous permettra de trouver des solutions aux problèmes que nous rencontrons actuellement avec la signature électronique plus facilement. Ce projet nous a permis de nous familiariser avec le domaine de la recherche et de la blockchain qui nous étaient inconnus jusqu'à maintenant, de travailler en équipe, et enfin de relever un challenge sur des problématiques concrètes.

References

[1] A Pure-solidity implementation of the SHA1 hash function by Nick Johnson. https://github.com/ensdomains/solsha1. Visité le 15/02/2022.

[2] Simon Pontie.Simon Pontie. Architecture d'un crypto processeur ECC sécurisé contre les attaques physiques. Journées Nationales du Réseau Doctoral en Microélectronique (JNRDM'14), May 2014, Lille, France. pp.4. ffhal-01091030f https://hal.archives-ouvertes.fr/hal-01091030/document. Visité le 25/02/2022.

[3] A native js function for hashing messages with the SHA-1 algorithm. https://github.com/pvorb/node-sha1. Visité le 16/02/2022.

[4] « Cryptographie - Certificats » issu de l'encyclopédie informatique « Comment Ça Marche ». www.commentcamarche.net. Visité le 28/02/2022.

[5] Secure Hash Algorithms. Brilliant.org. Retrieved 17:22, March 14, 2022, from https://brilliant.org/wiki/secure-hashing-algorithms/. Visité le 23/01/2022.