18/02/2019



Département MIDO

Année universitaire 2018 – 2019d

Sagodira Yoan– Lellouche Aaron – Uzan Raphaël – Sagodira Sébastien

[nom de la société]

Projet Programmation par composants

Spécification – Composant 4 : Hacheur SHA -256

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Modification** |
|  |  |  |
| V 1.0.4 | 24/02/2019 | Plan de test |
| V 1.0.3 | 21/02/2019 | Schéma blocs composants connexes |
| V 1.0.2 | 20/02/2019 | Ajout du contexte & interaction entre composant |
| V 1.0.0 | 18/02/2019 | Création du document & Plan détaillé des spécifications |

Table des matières

[Description 3](#_Toc4013420)

[Contexte 3](#_Toc4013421)

[Blockchain 3](#_Toc4013422)

[Hash dans la Blockchain 4](#_Toc4013423)

[Schéma bloc incluant les composants connexes 5](#_Toc4013424)

[Interface et interaction avec chaque autre composant 6](#_Toc4013425)

[Résumé: headers 6](#_Toc4013426)

[Cas d’erreurs 6](#_Toc4013427)

[Test 7](#_Toc4013428)

[Plan de test 7](#_Toc4013429)

[Programme de test: mode d’emploi 7](#_Toc4013430)

# Description

## Contexte

### Blockchain

Une Blockchain, est une liste d'enregistrements, appelés blocs, qui sont liés à l'aide d'une cryptographie. Chaque bloc contient un hachage cryptographique du bloc précédent, un horodatage et des données de transaction.

La Blockchain a été inventée par une personne utilisant le nom de Satoshi Nakamoto en 2008 pour servir de registre des transactions publiques du crypto-monnaie bitcoin. L'identité de Satoshi Nakamoto est inconnue. L'invention de la blockchain pour Bitcoin en a fait la première monnaie numérique à résoudre le problème de la double dépense sans recourir à une autorité de confiance ou à un serveur central.

De par sa conception, une blockchain est résistante à la modification des données. C'est "un grand livre ouvert” et distribué qui peut enregistrer les transactions entre deux parties de manière efficace, vérifiable et permanente .

Une fois enregistrées, les données d’un bloc donné ne peuvent plus être modifiées rétroactivement sans modification de tous les blocs suivants, ce qui nécessite un consensus de la majorité du réseau. Bien que les enregistrements ne soient pas inaltérables, les Blockchain peuvent être considérées comme sécurisées de par leur conception.

### Hash dans la Blockchain

Afin de sécuriser les données de chaque blocs, nous utilisons la cryptographie.

“La cryptographie est une science permettant de convertir des informations en clair en informations codées, c'est à dire non compréhensibles, puis, à partir de ces informations codées, de restituer les informations originales.“ *source : Securiteinfo.com*

Nous pouvons mettre en place les principes de la cryptographie grâce au Hachage.

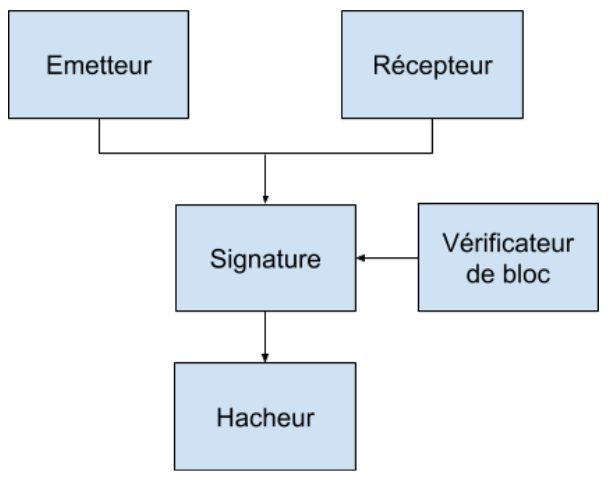
Une fonction de hachage cryptographique est une classe spéciale possédant diverses propriétés, ce qui la rend idéale pour la cryptographie.

Le hachage signifie prendre une chaîne d'entrée de n'importe quelle longueur et donner une sortie d'une longueur fixe. Dans le contexte de crypto-monnaies telles que le Bitcoin, les transactions sont prises en entrée et exécutées via un algorithme de hachage qui produit une sortie de longueur fixe.

Au cours de ce projet, nous allons mettre en place un SHA-256 (Secure Hash Algorithm 256).

Le SHA est un algorithme de hachage utilisé par les autorités de certification pour signer certificats et CRL. Il a été Introduit en 1993 par la NSA avec le SHA0, il est utilisé pour générer des condensats uniques (donc pour "hacher") de fichiers.

## Schéma bloc incluant les composants connexes



## Interface et interaction avec chaque autre composant

**Interface et interaction avec le composant mineur :**

Le composant mineur va appeler le hacheur SHA-256 pour hacher un bloc.

**Interface et interaction avec le composant signature :**

La signature numérique est un schéma mathématique permettant de présenter l'authenticité de messages ou de documents numériques. Une signature numérique valide donne au destinataire une raison de croire que le message a été créé par un expéditeur revendiqué (authentification), que l'expéditeur ne peut pas nier avoir envoyé le message (non-répudiation) et que le message n'a pas été modifié en transit (intégrité).

Notre composant va permettre de valider la signature numérique en étant réquisitionné par le composant signature afin de valider une signature

## Résumé: headers

Le composant hacheur reçoit un bloc et complète avec des zéros si nécessaire.

Par la suite, il renvoie la chaîne SHA.

Nous devons vérifier que le bloc est bien reçu ainsi que la chaîne SHA et retourner “vrai” ou “faux”.

**Méthodes :**

std::string hacher(std::string chaine\_a\_hacher)

bool verifier\_hachage(std::string chaine\_a\_verifier, std::string chaine\_hache)

## Cas d’erreurs

Nous avons identifié quelques cas d’erreurs possibles liés à notre composant.

Cas d’erreur 1 : L’information reçu n’est pas un bloc

Cas d’erreur 2 : La chaine renvoyée n’est pas haché 256

# Test

Nous avons décidé de mettre en place différents tests afin de nous assurer que notre composant fonctionne correctement et effectue bien ce qui est attendu.

## Plan de test

Voici notre plan de test qui se décompose en 4 tests permettant de vérifier que l’ensemble des fonctionnalités attendues pour notre composant s’effectuent correctement.

Test 1 : Vérifie que l’on reçoit un bloc et la chaîne SHA-256

Grâce à ce test nous allons vérifier que nous recevons correctement le bloc à hacher ainsi que la chaîne SHA-256.

Test 2 : Vérifie que l’on complète avec des zéros si nécessaire

Une des “fonctionnalités” de notre composant est de compléter le bloc avec des zéros si cela est nécessaire. Ce test va alors nous permettre de vérifier que dans le cas où l’ajout de zéros est requis, le composant réalise l’ajout correctement.

Test 3 : Vérifie que l’on renvoie la chaîne SHA

Ce test va nous permettre de vérifier qu’après avoir reçu le bloc et complété avec des zéros si nécessaire, notre composant renvoie la chaîne SHA.

Test 4 : Vérifie que l’on retourne “vrai” ou “faux”

Ce test va nous permettre de vérifier que l’on renvoie “vrai” lorsque le hash correspond au résultat attendu et “faux” lorsque ce n’est pas le cas.

## Programme de test: mode d’emploi

Nous avons mis en place un mode d’emploi afin d'effectuer nos tests.

Test 1 : Vérifie que l’on reçoit un bloc et la chaîne SHA-256

Test 2 : Vérifie que l’on complète avec des zéros si nécessaire

Test 3 : Vérifie que l’on renvoie la chaîne SHA

Test 4 : Vérifie que l’on retourne “vrai” ou “faux”