****

|  |
| --- |
|  |
| Spécifications fonctionnelles détaillées |
| *Mise en place d’une blockchain locale* |

|  |
| --- |
| 28/02/2018 |

**Auteurs:**

DJEUKAM YOUDJEU Richelle

NAJI Mohammed Hamza

RASOLOHERY Tianome

HILAHY Nawfal

**SUIVI DU DOCUMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Date* | *Auteur* | Objet de mise à jour |
| 1.0 | 15/02/2018 | Groupe5 | Version initiale |
| 1.1 | 28/02/2018 | Groupe5 | * Redéfinition des relations entre composants * Mise à jour des headers. * Ajout des tests. |

Contexte

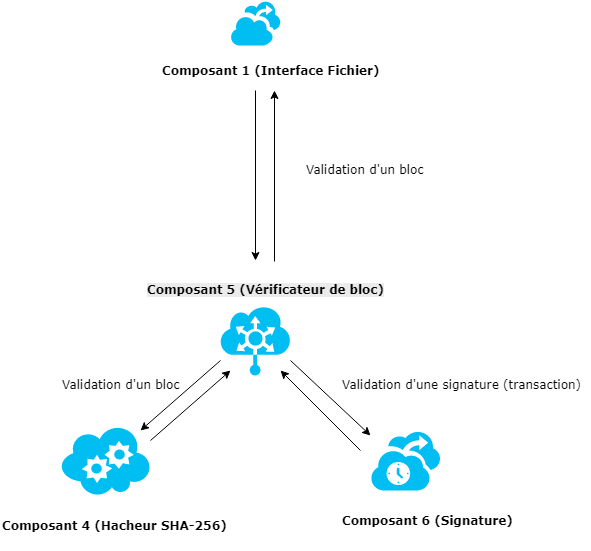
La blockchain est une technologie de stockage et de transmission d’informations, transparente, sécurisée, et fonctionnant sans aucun organe central de contrôle. Ce projet vise donc à mettre en place une blockchain locale à travers la réalisation d’un ensemble de composants qui permettront l’acheminement sécurisée et complète d’une transaction.

Lorsqu’un hôte A a effectué une transaction vers un hôte B, celle-ci va être acheminée dans un bloc; Ce bloc va être validé par les nœuds du réseau; une fois le bloc validé, il est ainsi ajouté à la blockchain et la transaction ainsi générée sera visible au destinataire et à l’ensemble du réseau.

Afin d’assurer la sécurité au sein de la blockchain, chaque bloc et chaque transaction doivent vérifier certaines conditions de validation.

Ce document illustre ainsi les différentes spécifications ou encore les différentes étapes de vérification par lesquelles chaque transaction et chaque bloc devront passer afin d'être validés et par la suite être acheminés dans la blockchain.

Schéma bloc incluant les composants connexes



Explication des interactions avec les composants connexes

A partir de ce schéma, nous avons remarqué que ce dernier interagit principalement avec les composants : 1, 4 et 6.

En effet, le composant 1 « Interface Fichier » fait appel au composant 5 « Vérification de bloc » pour la vérification de tous les blocs de la blockchain, ce dernier fera appel ensuite au « hacheur SHA-256 » (composant 4) pour savoir si un bloc donné est valide ou non afin d’éviter de propager des informations erronées..

La deuxième fonctionnalité de ce composant est la validation d’une transaction. Cette partie intervienne quant à elle à la création d’une transaction, ainsi elle communiquera avec le composant « Signature » (composant 6) pour faire l’évaluation de celle-ci.

Résumé (headers)

Dans cette partie, on va présenter les signatures des classes et des fonctions qui devront être implémentées pour la validation d’un bloc et la vérification d’une transaction.

# Vérifier l'état d’un bloc

Un bloc est constitué de:

. Numéro du bloc

. Hash du bloc précédent

. Nonce

. Transactions

. Gain du mineur

La fonction vérification du bloc prend en entrée le bloc à vérifier, le hash du bloc N-1 et le nombre de zéros définis par la difficulté. Elle appelle la fonction du hacheur qui calcule le hash en fonction du bloc et du hash du bloc N-1. Le résultat du hacheur sera comparé au hash contenu dans le bloc. La fonction vérification prend ensuite ce résultat et le nombre de zéros pour renvoyer True ou False pour dire si le résultat de la vérification est juste ou bien le bloc est erroné. La fonction de vérification du bloc intervient au niveau du composant 1 dans l’étape de vérification du bloc et à la réception du bloc dans le composant 4.

Signature de la méthode : bool verify\_bloc (Bloc\* b);

# Vérification de transaction:

La fonction de vérification de la transaction prend en entrée une UTXI (unspent transaction input) et renvoie True ou False pour dire si la transaction est valide ou non. Cette méthode fait appel à la fonction « verif\_signature » du composant 6 pour valider la signature d’une transaction.

Signature de la méthode : bool verify\_transaction (UTXI\* ui);

Test :

La sécurité dans une blockchain est d’une importance capitale ; de ce fait, il est important de s’assurer que chaque étape a été bien effectuée et que les principales spécifications des différents composants ont bien été respectés.

Pour ce composant, nous allons tester principalement les fonctionnalités qui seront développées pour vérifier la robustesse de notre blockchain.

On a dans un premier temps un test qui s’assure que la transaction a bien été vérifiée et dans un deuxième temps on s’assure que le block est bien vérifié avant de transiter dans le réseau.

Afin de pouvoir faire des tests intéressants, nous avons opté pour une stratégie itérative selon laquelle :

* Ecriture des cas de tests (les plus simples possibles : initialisation des variables de test à NULL) voués à l’échec et exécution.
* Génération du code faisant passer ces tests et respectant les caractéristiques du composant.
* Ajouts des cas de test (passant et non passant) avec utilisation des composants dont dépend le composant5 (composant4 et 6 par exemple) et exécution.
* Ajustement du code (si nécessaire).
* Exécution des tests.
* Validation du fonctionnement du composant.

Actuellement le premier point est mis en en place avec les méthodes :

”TestVerifyBloc” et “TestVerifyTransaction”

Pour l’exécution, il suffit d’aller dans l’explorateur de tests et choisir « **Exécuter tout** ». Sinon on peut également sélectionner le test à exécuter.