|  |  |
| --- | --- |
| Programmation par composants |  |
| Document de spécification du composant 5 : Vérificateur du bloc | |
| Version 1.2 – 05/04/2019 |  |
|  | |

|  |  |  |
| --- | --- | --- |
| Photo.png | [Commentaires ] |  |
| [Résumé] |  |
|  |  |  |

Fiche de suivi du document

**Identification**

|  |  |
| --- | --- |
| Nom du projet | Programmation par composants |
| Intitulé du document | Document de spécification du composant 5 : Vérificateur du bloc |
| Référence du document | [Référence Mustang] |
| Société |  |
| Service Métier |  |
| Service MOA |  |
| Service MOE |  |
| Domaine d’activité |  |
| Application |  |
| Responsable | [Responsable] |
| Rédacteur(s) | VO Quoc Huy, ANDRIAMBELOMAHERY Sam Michael, THAI Linh Linh |

**Documents de référence**

|  |  |  |
| --- | --- | --- |
| Référence | Titre | Fournisseur |
|  |  |  |

**Visas**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Approbateur | Fonction | Date |
| 1.0 | [Responsable] |  | 18/03/2019 |
| 1.1 |  |  | 25/03/2019 |
| 1.2 |  | Verifier(), verifierInitial(),verifierSignature() | 05/04/2019 |

**Diffusion**

|  |  |  |
| --- | --- | --- |
| Destinataire, Société | Pour action | Pour information |
|  |  |  |
|  |  |  |

**Historique des changements majeurs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Date | État | Auteur | Modifications |
|  |  | (Choisissez) |  |  |

Sommaire

[1 Objet du document 4](#_Toc4079279)

[2 description 5](#_Toc4079280)

[2.1 Contexte 5](#_Toc4079281)

[2.2 Schéma du bloc 5](#_Toc4079282)

[2.2.1 Schéma du bloc 5](#_Toc4079283)

[2.2.2 Zoom sur les transactions 6](#_Toc4079284)

[2.2.3 Blockchain complet 7](#_Toc4079285)

[2.3 Interface 8](#_Toc4079286)

[3 Test 9](#_Toc4079287)

[3.1 Plan de test 9](#_Toc4079288)

[3.2 Programme de test 9](#_Toc4079289)

# Objet du document

Ce document a pour objectif de :

* Spécifier les exigences liées au composant 5 Vérificateur du bloc,
* Présenter le schéma bloc incluant les composants connexes,
* Décrire l’interface et l’interaction avec chaque autre composant,
* Définir un plan et un programme de test.

# description

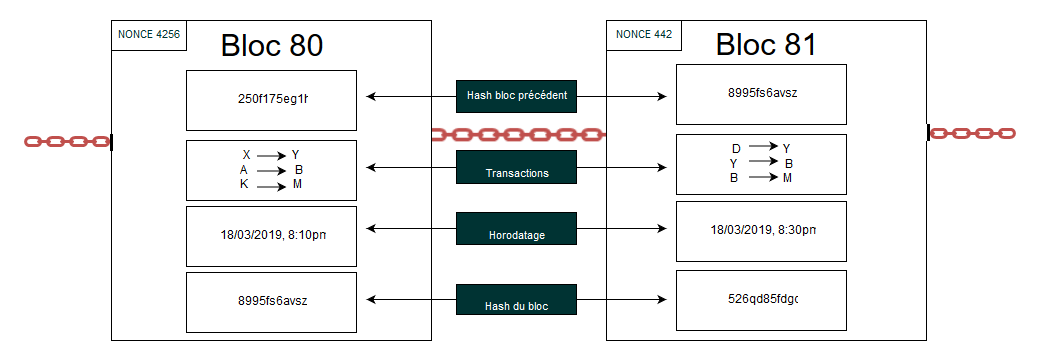
## Contexte

L’objectif de ce projet est de développer une blockchain complet à travers la programmation par composants. Parmi les six composants de la blockchain, le composant « Vérificateur du bloc » est un composant important permettant de vérifier :

* Vérifier la composition du bloc,
* Vérifier les transactions.

## Schéma du bloc

### Schéma du bloc



* Description

Un bloc est une structure contenant du numéro de bloc, des données de transactions, du numéro NONCE, de l’horodatage, du previousHash (hash du bloc précédent) et du hash cryptographique de ce bloc.

Un hash est un nombre créé par une fonction de hachage qui convertit une entrée de lettres et de chiffres en une sortie cryptée de longueur fixe.

Une blockchain est une liste croissante de blocs pour lesquels chaque bloc contient une référence au bloc précédent : le hash. Ils sont donc chaînés grâce au hash. De par sa conception, une blockchain est résistante à la modification des données. Une petite modification d’un bloc entraine la perte de son signature. Ce bloc et l’ensemble des blocs qui suivent ne sont plus signés. Il est donc très facile de savoir si la moindre information d’une blockchain a été changée et dans quel bloc.

* Validation l’intégrité des blocs

À n’importe quel moment, il faut pouvoir définir si un bloc ou une chaine de blocs est valide en terme d’intégrité. Un bloc est valide si :

* L’index du bloc est plus grand que celui du bloc précédent,
* Le previousHash du bloc correspond au hash du bloc précédent,
* Le hash du bloc lui-même est valide.

NB : Le premier bloc d’une blockchain est le seul bloc de la chaîne ne possédant pas de previousHash. Pour valider ce premier bloc, les règles sont les mêmes sauf que le previousHash doit être nul.

Donc, il est possible de valider tout une blockchain. Dans un premier temps, il faut valider le premier bloc et puis valider tous les blocs suivants.

### Zoom sur les transactions

TX 2

TX 1

Input

Output

Output

Input

UTXO

Montant

Montant

UTXO

**script**

Signature

Public key

Public key

Signature

* Description

Clé privée, clé publique, signature numérique

Le mécanisme cryptographique de clé privée et clé publique est essentiel pour le fonctionnement de la blockchain.

La paire de clés publique et privée est composée de deux clés cryptographiques uniques (essentiellement de longs nombres aléatoires). Il est possible de calculer la clé publique à partir de la clé privée, cependant l’inverse n’est pas vrai. Il est pratiquement impossible de retrouver la clé privée à partir de la clé publique. La clé publique est publique. Il est mis à la disposition de tous via un référentiel ou un répertoire accessible au public. D'autre part, la clé privée doit rester confidentielle à son propriétaire respectif. Par exemple, A et B représentent un expéditeur et un destinataire de message, respectivement. Chacune a sa propre paire de clés publiques et privées. A, l'expéditeur du message, envoie un message à B. Le message de A est chiffré avec la clé publique de B, tandis que B utilise sa clé privée pour déchiffrer le message reçu par A. Comme seule B a accès à sa clé privée, il est évident que seul B puisse déchiffrer les données cryptées. Même si une autre personne accède aux données cryptées, celles-ci resteront confidentielles car ils ne devraient pas avoir accès à la clé privée de B.

Un autre aspect important de la cryptographie à clé publique est sa capacité à créer une signature numérique. Une signature numérique est utilisée pour garantir qu’une personne est l'expéditeur du message d'origine. Pour vérifier cela, B utilise les étapes suivantes: B utilise la clé publique de A pour déchiffrer la signature numérique, car A doit auparavant utiliser sa clé privée pour chiffrer la signature numérique. Si elle est lisible, la signature numérique est authentifiée par une autorité de certification, ce qui permet de vérifier que A est bien l’expéditeur du message d’origine.

Transactions

Une transaction est composée d’inputs et d’outputs. Une entrée (*TXI transaction input*) contient de numéro de bloc, de numéro de UTXO et de la signature de UTXO. Une sortie (*UTXO unspent TX output*) contient de montant et la clé publique (destinataire).

* Vérification des transactions

Une transaction devrait être validée. Les règles sont les suivantes :

* Les entrées (TXI) doivent être valides,
* Retrouver la UTXO correspondante,
* La UTXO ne doit pas être référencée par une TXI dans un bloc de numéro inférieur,
* Pouvoir vérifier la signature accompagnant TXI à l’aide de la clé publique contenue dans la UTXO,
  + Si on retrouve le hash de la UTXO -> Ok
  + Sinon -> Invalide
* Note : Hash du UTXO : montant, numéro de bloc, numéro de transaction dans le bloc.

### Blockchain complet



## Interface

Les interfaces du composant 5 : Vérificateur de bloc sont :

* Composant 2 : Portefeuille/ Wallet

Le composant 5 Vérificateur de blocs est appelé par le composant 2 Portefeuille/ Wallet. Composant Wallet nous donne le numéro du bloc et le hash du bloc précédent.

* Composant 4 : hacheur SHA-256

Le composant 5 Vérificateur du bloc passe au composant hacheur le bloc ou la chaîne correspondante au hash (nous leur envoyons le hash que nous avons reçu) pour que le hacheur puisse vérifier la validité du hash et valider l’intégrité des blocs. Le composant 4 Hacheur nous dira si le hash est validé ou pas.

* Composant 6 : signature

Vérifier la signature.

On a donc les fonctions :

* verifier(Bloc bloc\_a\_verifier, Bloc bloc\_precedent, unsigned int difficulty) pour verifier un bloc intermediaire.
* verifierInitial(Bloc bloc\_a\_verifier, unsigned int difficulty) pour verifier un bloc initial.
* verifierSignature(string data, string public\_key, string signature) pour verifier la signature
* Vérification des transactions

Une transaction devrait être validée. Les règles sont les suivantes :

* Les entrées (TXI) doivent être valides,
* Retrouver la UTXO correspondante,
* La UTXO ne doit pas être référencée par une TXI dans un bloc de numéro inférieur,
* Pouvoir vérifier la signature accompagnant TXI à l’aide de la clé publique contenue dans la UTXO,
  + Si on retrouve le hash de la UTXO -> Ok
  + Sinon -> Invalide
* Note : Hash du UTXO : montant, numéro de bloc, numéro de transaction dans le bloc.
* Validation l’intégrité des blocs

Un bloc est valide si :

* L’index du bloc est plus grand que celui du bloc précédent,
* Le previousHash du bloc correspond au hash du bloc précédent,
* Le hash du bloc lui-même est valide.

NB : Le premier bloc d’une blockchain est le seul bloc de la chaîne ne possédant pas de previousHash. Pour valider ce premier bloc, les règles sont les mêmes sauf que le previousHash doit être nul.

Donc, il est possible de valider tout une blockchain. Dans un premier temps, il faut valider le premier bloc et puis valider tous les blocs suivants.

# Test

## Plan de test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N°** | **Thème** | **Sujet** | **Description/Cas de test** | **Résultat attendu** |
| **Test unitaire** | | | | |
| 1 | Vérifier la composition du bloc | Vérifier le previousHash du premier bloc | Le previousHash du bloc est non nul | Bloc n’est pas validé |
| 2 | Vérifier le hash du bloc | La règle du hash n’est pas vérifié | Bloc n’est pas validé |
| 3 | Vérifier l’index du bloc | L’index du bloc est plus grand que celui du bloc précédent | Bloc n’est pas validé |
| 4 | Vérifier le previousHash du bloc | Le previousHash du bloc est faux | Bloc n’est pas validé |
| 5 | Vérifier les transactions | Vérifier les entrées TXI | La règle de TXI n’est pas vérifié | Bloc n’est pas validé |
| 6 | Vérifier la référence de UTXO | La UTXO est référencée par une TXI dans un bloc de numéro supérieur | Bloc n’est pas validé |
| 7 | Retrouver le hash de UTXO | Si le hash de UTXO n’est pas retrouvé | Invalide |
| **Test d’intégration** | | | | |
| 8 | Vérifier la composition du bloc | -L’index du bloc est plus grand que celui du bloc précédent,  -Le previousHash du bloc correspond au hash du bloc précédent,  -Le hash du bloc lui-même est valide | Si tous les trois conditions sont validées | Bloc est validé |
| 9 | Vérifier la composition du bloc | -L’index du bloc est plus grand que celui du bloc précédent,  -Le previousHash du bloc correspond au hash du bloc précédent,  -Le hash du bloc lui-même est valide | Si une des trois conditions n’est pas validée | Bloc n’est pas validé |
|  |  |  |  |  |

## Programme de test