Groupe 2

ARIZANOVA Simona
DIARSO Laye Kemo
HADI Ismaïl
JOLIMAN Iris

Spécifications fonctionnelles et techniques pour le composant n°2

Blockchain Wallet

Version	Date	Modifications	Auteur
1.0	17/04/2020	Initiation du document	Groupe 2

Table des matières

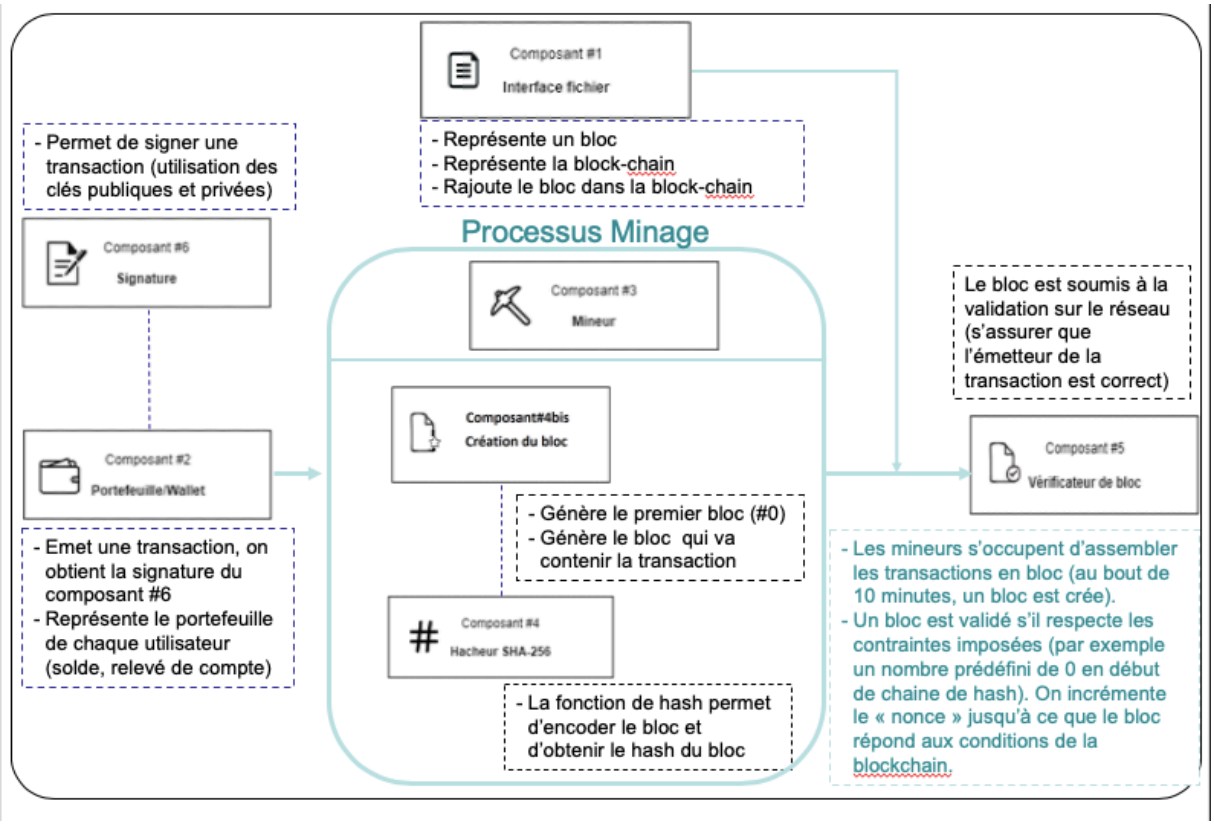
I . Contexte général

Une blockchain est une nouvelle technologie permettant de stocker et transmettre des informations de manière transparente, sécurisée et non centralisée. Sa principale application relève des cryptomonnaies telles que le bitcoin (2008) ou l'éther. Depuis, les entreprises prennent confiance en cette technologie dont la notoriété ne cesse de grandir. On assimile de façon plus simple une blockchain à un livre comptable public, anonyme et impossible à falsifier.

Ce projet consiste à créer une blockchain contenant donc un ensemble de transactions et permettant d'en enregistrer de nouvelles tout en respectant les principes de base utilisés par cette technologie (minage, cryptographie, clé privée ou publique, etc). Au sein de ce projet, différents composants interagissent entre eux. Nous avons la responsabilité du composant « wallet / portefeuille ». Un wallet peut en réalité prendre différentes formes ; il peut s'agir d'une application mobile, d'une clé usb (hardware wallet) ou encore d'un logiciel. Dans tous les cas, ce composant est vulnérable aux cyber-attaques car il contient en général des cryptomonnaies. Son fonctionnement est décrit dans la suite de ce document. Nous ne prenons pas en compte les portefeuilles multi signatures.

II . Définition du processus

Afin de permettre une meilleure compréhension du principe de blockchain, nous avons ci-après une représentation du processus général. Pour développer la blockchain, plusieurs fonctionnalités sont nécessaires ; chaque fonctionnalité majeure est appelée « composant ».



La première fonctionnalité importante sera l'architecture blockchain en général. C'est-à-dire, la représentation des blocs et de la chaîne de blocs (blockchain). Le *composant #1* « Interface fichier », s'assure donc de toute la gestion liée à la blockchain :
- Ouverture et verrouillage du fichier.
- Initialisation de la blockchain (Bloc 0 : « Genesis block »)

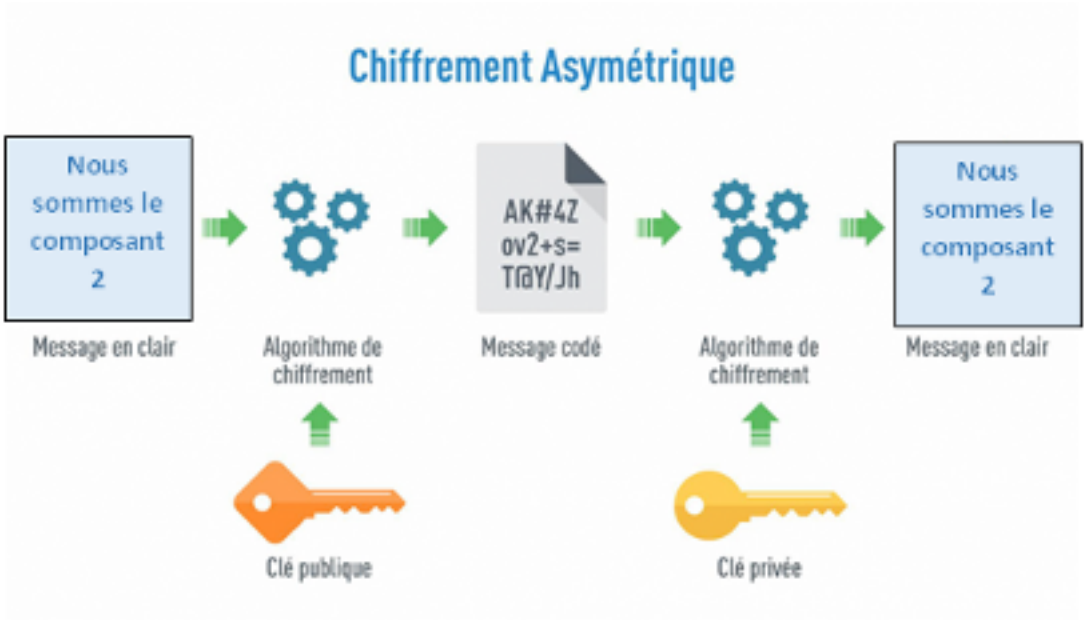
- Vérification et ajout de bloc en fin de chaîne (intervient à la fin du processus de minage).

Dès que la blockchain est définie, intervient le *composant #2*. Ce composant permet l'émission de transaction (une personne A donne à une personne B, 50 « monnaies virtuelles »). Pour valider et s'assurer qu'une transaction est approuvée par la personne B, la transaction est signée. Le composant qui permet cette action est le *composant #6.* Il fournit les clés publiques et privées, nécessaires à la signature de la transaction.

Dès lors que la transaction est signée, celle-ci rentre dans le « processus de minage » (*composant #3*).
Chaque 10 minutes (dans le cas du bitcoin), l'ensemble des transactions sont rassemblées. Un bloc est généré (via le *composant #4bis*) en comportant toutes les informations qui constituent un bloc : le numéro du bloc, l'ensemble des transactions, le hash du bloc (qui est calculé en en faisant usage du *composant #4*).
Les mineurs s'assurent de respecter certaines conditions imposées par la blockchain. Un exemple serait de s'assurer que le hash d'un bloc commence nécessairement par un nombre prédéfini de zéros. Ainsi, tant que la valeur du hash ne respecte pas ces règles, un identifiant appelé « nonce » est incrémenté à chaque fois. Une simple modification de ce nombre permet de modifier le contenu du bloc et de changer ainsi le hash correspondant. Dès lors que les conditions sont respectées, le bloc est validé et passé au *composant #1* pour le rajouter dans la blockchain.
Une dernière vérification des transactions importante est effectuée par le *composant #5* afin de valider la cohérence du bloc.

La partie suivante permet d'introduire plus en détails les aspects de sécurisation de transaction ; le concept de cryptographie asymétrique.

Ainsi, la cryptographie asymétrique est un procédé qui intègre deux clés de chiffrement, une *clé publique* et une *clé privée*. Par convention, la clé de chiffrement du message est appelée clé publique (et peut-être communiquée sans restriction aucune), et la clé de déchiffrement du message est appelée clé privée.



Les fonctions de hachages permettent de chiffrer des données, parmi elles, on retrouve Secure Hash Algorithm (SHA) du NIST.

Pour prévenir les tentatives de fraude, cet intermédiaire tient seul le registre des transactions. La blockchain réalise cet échange sur un réseau de pair à pair, donc sans intermédiaire. La transaction entre deux internautes est enregistrée dans un registre (ledger) qui garde trace de toutes les opérations effectuées. Ce registre n'est pas détenu dans un lieu centralisé mais « distribué » dans les ordinateurs de tous les participants, appelés « nœuds ». À chaque transaction, les membres du réseau interrogent l'historique pour s'assurer que la personne possède bien les actifs qu'elle souhaite échanger. Les transactions sont ensuite groupées et validées par blocs – qui forment une « chaîne de blocs » ou « blockchain ». Chaque nouveau bloc de transactions vient s'ajouter à la chaîne, lié au

précédent par un procédé cryptographique. La blockchain contient ainsi l'ensemble des opérations validées depuis la création de la chaîne jusqu'à aujourd'hui.

Ces différentes caractéristiques seront mises en place au sein des composants 1, 3, 4, 5 et 6 afin de vérifier la bonne architecture d'ensemble.

III. Définition du besoin

Dans la gestion des cryptomonnaies, la notion de « wallet » ou portefeuille est une notion très importante. Il désigne une solution de stockage sécurisée des cryptomonnaies. Cette appellation recouvre différents modes de stockage, il pourrait s'agir d'un logiciel client, d'une application, d'un fichier crypté et sécurisé ou d'un simple morceau de papier.

Pour le bitcoin par exemple, le wallet ne contient pas les bitcoins, mais seulement les clés informatiques donnant accès aux transactions enregistrées dans la chaîne de blocs.

Le wallet est principalement composé :

- D'une clé publique connue de tous les correspondants. Elle peut être par exemple une adresse bitcoin ou de toutes autres cryptomonnaies.
- D'une clé privée connue du seul propriétaire du wallet. Cette clé permet de signer des transactions et de prouver à l'ensemble du réseau qu'on est bien propriétaire de tel ou tel montant d'une cryptomonnaie.
- Et en dernier le montant des cryptomonnaies contenue dans le wallet.

Afin d'obtenir ces différentes informations, il nous faut mettre en place des méthodes permettant d'exploiter les informations contenues dans chaque bloc. Pour cela, il est tout d'abord important d'être en possession des différents blocs à exploiter.

L'accès à ces blocs permettra principalement d'avoir accès à toutes les transactions valides disponibles dans la blockchain.

IV. Principe du composant

Le composant #2 explore les différents blocs de la blockchain et émet des transactions.

La première phase consiste à accéder à la base où seront stockées les différents blocs afin de lire toutes les informations par bloc.

Pour la deuxième phase, il permettra d'exploiter plus en profondeur chaque bloc, en étudiant plus en détail ces trois composants principaux :

- *L'en-tête :* qui contient les métadonnées à propos du bloc.
- *L'identificateur :* qui permet de l'identifier par rapport aux autres blocs. Celui-ci possède une clé digitale qui représente le hash.
- *L'arbre de données*, qui représente toutes les transactions.

Enfin, le composant permettra d'émettre une transaction en identifiant l'émetteur, le récepteur ainsi que le montant associé. Cela nous permettra également, d'analyser les différentes transactions en faisant un résumé de chaque transaction disponible par clé publique.

V. Définition des fonctionnalités

a. Lecture des blocs et recensement des UTXO

Pour permettre la visualisation des éléments de la blockchain, il est nécessaire de créer des fonctions qui nous permettent de récupérer les informations de ces derniers.

Ainsi, nous aurons les fonctions suivantes :

- *Lecture de bloc :*

  En input : La blockchain.

  En output : La liste des blocs ainsi que leur contenu.

- *Lecture des UTXO :*

  En input : La liste des blocs (fonction « *Lecture de bloc* »).

  En output : La liste des UTXO.

Pour l'instant, nous ne listons que ces fonctions mais certainement d'autres fonctions de récupération de données seront nécessaires pour le développement du composant, voir du projet (récupération d'une transaction spécifique, obtention des TXI d'un bloc, visualisation du hash d'un bloc ...)

b. Fonctions de consultation

La fonction de consultation consistera à explorer et à analyser les différentes transactions contenues dans les différents blocs.

Pour cela, il faut mettre en place une stratégie permettant de lire toutes les transactions par bloc et d'établir le lien entre chacune d'elles. Cela consiste à consolider par clés publiques afin de déterminer les montants correspondants.

Afin d'obtenir tout cela, nous aurons besoin :

D'une fonction python *takeTX*, permettant de lire bloc par bloc toutes les transactions en fonction de leur identifiant. Cette fonction aura :

En input : l'identifiant (la clé publique qui identifiera la transaction) et la liste des UTXO (fonction « *Lecture de UTXO* »).

En output : La somme de tous les montants correspondants à la clé publique entrée en paramètre. La clé publique étant un attribut de la Classe UTXO, on pourra facilement comparer les clés et récupérer les montants correspondants.

c. Fonctions d'émission de transaction

Pour qu'une transaction soit valide, il est nécessaire qu'elle vérifie les trois conditions suivantes :

- Au cours d'une transaction, il est nécessaire que les deux parties soient en possession d'un portefeuille.
- L'acheteur doit effectivement posséder la somme qu'il souhaite envoyer.
- L'acheteur n'a pas envoyé cette somme à une autre personne avec les mêmes UTXO.

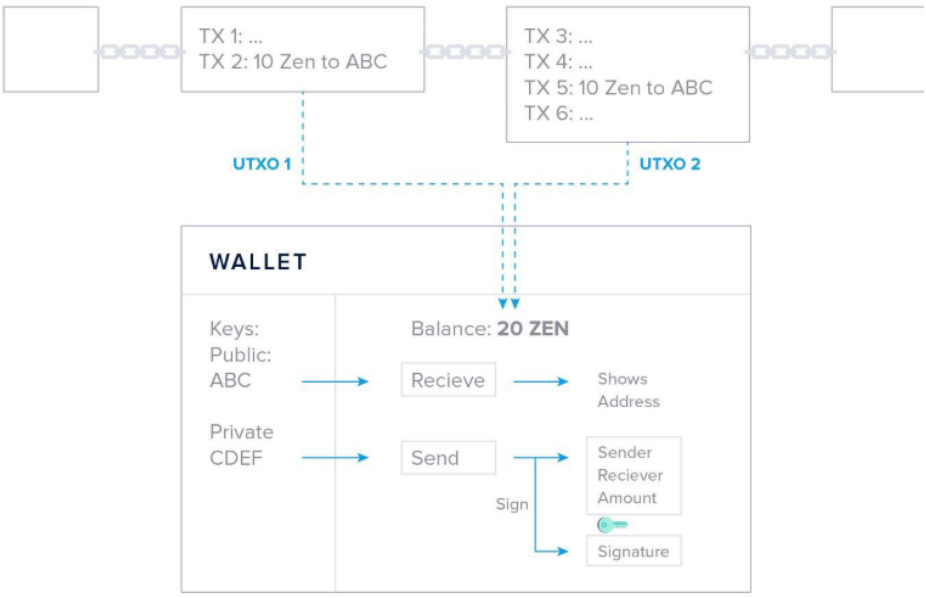
Ces conditions seront vérifiées ultérieurement grâce au *composant #3 « Mineur »*.

Pour émettre une transaction, valide ou non, nous devons avoir connaissance d'un montant et de l'adresse publique du récepteur. L'émetteur doit « signer » la transaction avec son adresse privée.

Une transaction contient des inputs et des outputs :

En Input : Les Unspent Transaction Output (UTXO) sont en réalité les outputs des transactions précédentes, le numéro de bloc ainsi que la signature des UTXOs.

En Output : Le montant et la clé publique du destinataire.



Il sera donc nécessaire d'implémenter la fonction python suivante :

Send() : qui prendra en entrée la clé publique du récepteur et le montant et créera une transaction tout en gérant les UTXO comme défini précédemment.

VI. Plan de test

Dans cette partie, il s'agira de recenser quelques cas de tests ainsi que la manière permettant de vérifier la cohérence de ces derniers. Chaque point constituera une idée globale regroupant des sous-points matérialisant leur recette :

- **Bonne constitution du bloc**
  - Établir un bloc avec des informations cohérentes et vérifier son acceptation dans le système
  - Établir un bloc avec des transactions d'utilisateurs inexistants et vérifier le refus
  - Établir un bloc avec des montants indisponibles pour un utilisateur et vérifier le refus
  - Établir un bloc avec un hash incohérent et vérifier le refus

- **Hash cohérent résultant du minage : nombre de zéros cohérent**
  - Entrer en dur un hash cohérent et vérifier la validation (uniquement pour confirmer la bonne implémentation)
  - Entrer en dur un hash incohérent (nombre de zéro attendu inférieur à celui entré) et constater le refus
  - Entrer en dur un hash incohérent (nombre de zéro attendu supérieur à celui entré) et constater le refus
  - Faire tourner le programme et vérifier que la validation des hash est cohérente avec le nombre de zéro attendu

- **Hash de l'UTXO correspondant au décryptage de la signature du TXI**

    - Exécuter le programme avec un UTXO inexistant et vérifier l'échec
    - Exécuter le programme avec un UTXO existant et vérifier la validation
    - Exécuter le programme sans UTXO et vérifier le refus

- **Inputs de TXI valides**

    - Entrer un numéro de bloc valide, un numéro d'UTXO valide et une signature de l'UTXO invalide et vérifier le refus
    - Entrer un numéro de bloc invalide, un numéro d'UTXO valide et une signature de l'UTXO valide et vérifier le refus
    - Entrer un numéro de bloc valide, un numéro d'UTXO invalide et une signature de l'UTXO valide et vérifier le refus
    - Entrer un numéro de bloc valide, un numéro d'UTXO valide et une signature de l'UTXO valide et vérifier la validation

- **Vérifier que l'UTXO n'est pas un TXI**

    - Exécuter le programme avec une TXI référencée par une TXI et vérifier l'erreur

- **La rémunération unique du mineur par bloc dont le montant est de 10**

    - Ajouter une transaction mineure au programme de montant 10 et vérifier la validation
    - Ajouter une transaction mineure au programme de montant différent de 10 et vérifier l'échec
    - Ajouter deux transactions mineures au sein d'un bloc de montant 10 et vérifier l'échec

- **Montant disponible d'un utilisateur supérieur à la somme payée lors d'une transaction**

    - Effectuer une transaction d'un utilisateur X à un utilisateur Y dont le montant émis est supérieur au montant disponible par X et constater l'échec
    - Effectuer une transaction d'un utilisateur X à un utilisateur Y dont le montant émis est inférieur au montant disponible par X et constater la validation