

BLOCKCHAIN LOCALE :

Spécification Composant 6

Réalisé par :

Groupe1

- **ENNAJIMI Yassine**
- **MOUZITA Coccley**
- **PADONOU Jimmy**
- **SOUSSOU Kamel**

Version doc	Date	Auteur(s)	Modifications
1.0	27/01/2016	Jose Luu	Version initiale
1.1	18/02/2016	Groupe 1	Objectif du composant
1.2	26/02/2016	Groupe 1	Etapes et description des erreurs
1.3	28/02/2016	Groupe 1	Rédaction du plan de test
1.4	01/03/2016	Groupe 1	Mise à jour cas d'erreur et plan de test

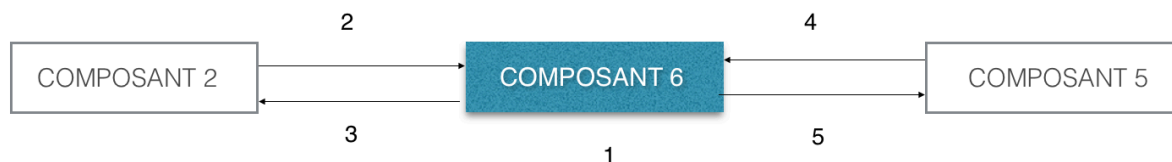
Ce document rassemble les spécifications concernant le composant 6 « Blockchain » du projet de classe.

I. Fonctionnement du composant

L'objectif du composant 6 est premièrement de créer de nouvelle adresse c'est-à-dire génère une paire de clés cryptographiques composé d'une clé privée et d'une clé publique.

Ce composant permet également de signer un message avec une clé privée (que seul l'émetteur connaît) et de vérifier la signature en utilisant la clé publique correspondante.

Voici un schéma explicatif du mode de fonctionnement du composant demandé :



- 1) Génère la paire de clés (Publique & Privée)
- 2) Réception de Appel du composant 2 pour la signature de données
- 3) Retour de la signature de données
- 4) Réception d'Appel Valider signature venant du composant 5
- 5) Retour de Valider signature

Chacune des étapes du schéma seront explicitées afin de comprendre le fonctionnement du composant et son mode de réalisation.

II. Les étapes

1) Génération de la paire de clés

Notre composant génère une paire de clés (privée et publique) nécessaire à chaque utilisateur.

La clé publique sera connue de tous et la clé privée par un seul utilisateur.

Pour créer une transaction, il est nécessaire d'avoir une clé privée pour l'émetteur et la clé publique du destinataire ainsi que le montant.

HashMap <string, int > generateKeys();

2) Réception de la demande du composant 2(Wallet)

Le « Wallet » qui compose une transaction fournit à notre composant plusieurs paramètres permettant la signature des données :

- La clé privée de l'émetteur

La signature de la transaction utilise l'algorithme ECDSA. La signature permettra d'authentifier l'émetteur de la transaction par le destinataire.

String signatureData(int privateKey);

3) Retour de la demande au composant 2

Le résultat de la fonction **signatureData** est renvoyé au « Wallet ».

4) Réception de la demande du composant 5

Le composant 5 s'occupe de la vérification des blocs et transactions.

La vérification d'une transaction implique la vérification de la signature.

Le composant 5 fait appel à la fonction ***valideSignature*** du composant 6 en passant la clé publique et la signature de la transaction en paramètres.

Ce dernier vérifie que “sign” est la signature de la transaction de clé publique “clePub” en renvoyant vrai si la signature est valide et faux sinon.

bool valideSignature(int clePub, String sign)

5) Retour de la demande du composant 5

Le résultat de la fonction ***valideSignature*** est renvoyé au composant 5.

En renvoyant “Vrai”, la transaction est vérifiée et correcte et dans le cas contraire la transaction n’est pas correcte.

III. Description des erreurs

Chacune des erreurs du composant est gérée par les exceptions « throws » ci-dessous :

Etape 2

Nom Fonction : string signatureData(int privateKey)	
Erreur : donnée négative/zéro	Not positive value
Erreur : données manquantes	Not Enough Argument
Erreur : Données en plus	Too Much Argument
Erreur: Mauvais type d'argument	Invalid Type Argument

Etape 4

Nom Fonction : bool valideSignature(String clePub, String sign)	
Erreur : Données nulles	Null Value
Erreur : Données manquantes	Not Enough Argument
Erreur : Données en plus	Too Much Argument
Erreur: Mauvais type d'argument	Invalid Type Argument

IV. Plan de tests

Les fonctions des différents composants sont testées lors de l'exécution de la fonction principale(main) afin de vérifier si le comportement du composant répond aux attentes.

Aucun message d'erreur ne sera affiché si le comportement du composant est correct et dans le cas contraire un message d'erreur sera indiqué.

- `String signatureData(int privateKey)`

Si le paramètre passé est négatif, le message suivant est retourné :

« NOT POSITIVE VALUE » ;

Si la fonction est appelée sans aucun paramètre, suivant est retourné:

« NOT ENOUGH ARGUMENT »;

Si la fonction est appelée avec un autre type que celui attendu, le message suivant est retourné :

« INVALID TYPE ARGUMENT » ;

- `bool valideSignature(String clePub, String sign)`

Si l'un des paramètres (clePub, sign) est nul alors le message d'erreur suivant doit être renvoyé :

« NULL VALUE »;

S'il manque au moins un des paramètres(clePub, sign) alors le message d'erreur suivant doit être renvoyé :

« NOT ENOUGH ARGUMENT »;

Si la fonction est appelée avec des types différents que ceux attendus, le message suivant est retourné :

« INVALID TYPE ARGUMENT »;