Document de Spécification du Composant n°4

Projet de Programmation par Composants

1. Présentation du Projet

Auteurs du document

Jérémie Facquet	Douha Karim	Samira Karimou
Mohamed Hamroun Houssem		Ahmed Horri

Historique des versions

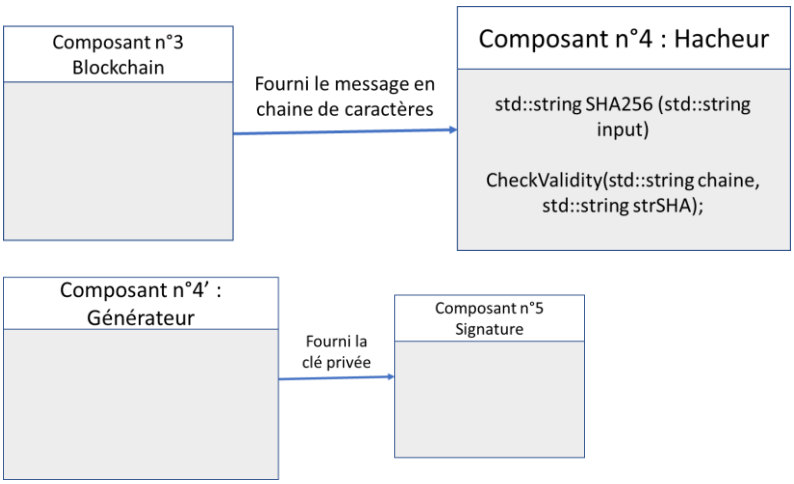
Date	Version	Ajouts
30/05/2021	1.0	
05/07/2021	2.0	Ajout du composant 0

2. Contenu du Projet :

2.1. Contexte :

Il s'agit ici de développer deux composants : un hacheur SHA-256 et un générateur de clé privée aléatoire de 256 bits

2.2. Schéma du Bloc :



2.3. Interface et interactions avec chaque autre composant :

Interaction 1 :

Le composant hachage reçoit de la part du composant Blockchain un bloc sous forme de chaine de caractère correspondant à la sauvegarde Json :

Composant Blockchain ----------Chaine de Caractère----------> **Composant Hachage**

Interaction 2 :

Composant Hachage ----------Clé privée----------> **Composant signature**

Code en C++ et interface en Python pour la fonction de génération de clé privée aléatoire

Conventions entre les blocs : Toutes les clés et signatures seront des chaines de caractères en hexadécimales

2.4. Résumé :

Déclaration des fonctions python d'interface : toutes les fonctions python et leurs arguments

- Fonction de hachage : std::string SHA256 (std::string input);

Cette fonction reçoit un block qui représente une chaine de caractère correspondant à la sauvegarde JOSN, et renvoie la chaine de caractères hachée en SHA

- Fonction de vérification du block : bool CheckValidity(std::string str , std::string strSHA);

2.5. Cas d'erreurs :

Pour le composant numéro 4, chacune des erreurs sera gérée par les exceptions dans des fonctions séparées :

Fonction de vérification des données en entrée	
Erreur 1 : Entrée NULL	Aucune chaîne de caractère trouvé
Erreur 2 : Len(chaine de caractère) >55	Input contient plus de 55 caractères, veuillez vérifier votre saisie

Fonction de transformation binaire	
Erreur 1 : Sortie non binaire	Vérifier que la transformation s'est bien passée

Fonction de remplissage	
Erreur 1 : Sortie n'est pas un incrément de 512 bits	Vérifier que la sortie est un incrément de 512 bits

Fonction de test_chaine_de_caractere_simple	
Erreur 1 : Sortie != résultat attendu	Test Failed

3. Test

3.1. Qu'est-ce qu'on teste ?

On va tester si le composant reçoit bien le bloc ou plutôt la chaine de caractère renvoyé par le composant Blockchain. Ensuite il faudra tester si le programme qui fait le hachage renvoi bien la chaine SHA. Le test se fera sur la base du hachage d'une chaîne de test dont le résultat est connu et vérifier que c'est le résultat attendu.

3.2. Description du programme de tests :

Le fichier de test est stocké dans le fichier « test.py »

Le programme de test va recevoir le bloc renvoyé par le composant blockchain, ainsi que la chaine SHA renvoyé par le composant hachage, puis retourner VRAI si ces deux paramètres ont bien été reçues et retourner FAUX sinon.

@Ahmed HORRI

Commenté [AH1]: @Ahmed HORRI

3.2. *Exemple d'utilisation du composant Hachage* :

```
# python3.7
>>> from component_Hachage import component_Hachage
>>> h=component_Hachage()
>>> h.SHA256("abc")
'ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad'
>>>
```

Dans le cas où la chaine donnée en entrée est vide :

```
# python3.7
>>> from component_Hachage import component_Hachage
>>> h=component_Hachage()
>>> h.SHA256("")
ERROR input is empty
''
>>>
```

Fonction de vérification :

```
# python3.7
>>> from component_Hachage import component_Hachage
>>> h=component_Hachage()
>>> h.SHA256("abc")
>>>
h.checkValidity("abc","ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad")
True
>>>
h.checkValidity("ab","ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20
```

015ad")
False

4. Gestion des transactions : Composant 0

4.1. Contexte

Il s'agit à présent d'implémenter un système de gestion qui permettra au composant d'interagir avec les autres du projet. Pour cela, il s'agit de créer d'implémenter les fonctions « to_json() »des classes.


```
{
  "hash": "000123456789A123456789A123456789A123456789A123456789A123456789A",
  "difficulty": 3,
  "nonce": 42,
  "num": 0,
  "previous_hash": "0000000000000000000000000000000000000000000000000000000000000000",
  "transactions": [],
  "txm": []
}
```


Le bloc_0 a été écrit manuellement en format JSON. Toute blockchain se construit au-dessus d'un bloc 0 qui est arbitraire. Le hash du bloc_0 contient les transactions (liste vide pour l'instant car il n y a pas eu de transaction) et le nonce (entier arbitraire inclut à la fois dans le bloc et dans le calcul du hash).

Le hachage du bloc_0 produit le hash_0 qu'on retrouve dans le bloc1. Comme le bloc_0 n'ayant pas de bloc précèdent, il contient un champ pour le hash du bloc précèdent, on a choisi de mettre la valeur '000..00' comme hash du bloc précèdent.

Ce fichier JSON permet d'initialiser la blockchain. Il est passé en paramètre dans le constructeur du composant_bloc, comme suit Bloc(Bloc_0.json).