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

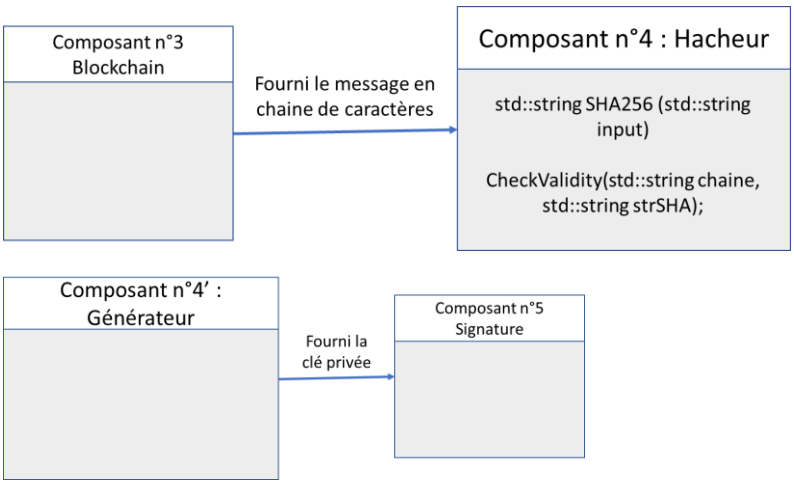
Date	Version	Ajouts

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

@Mohamed Houssem HAMROUN + @Ahmed HORRI :

Commenté [MH1]: @Mohamed Houssem HAMROUN

Commenté [MH2]: @Ahmed HORRI

- 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 :

Voir dernière vidéo - le prof explique ça a la fin du dernier cours @Samira KARIMOU

Commenté [DK3]: @Samira KARIMOU

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 test ?

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.

3.2. Description du programme de tests :

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 [AH4]: @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 ou 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 verification :

```
# 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
>>>
```