1. Comment déployer un nœud sur une blockchain ?

Pour déployer un nœud sur une blockchain Kalima, il faut connaitre la liste des Master Nodes sur lesquels le nœud va devoir se connecter.

De plus le nœud doit être autorisé sur la blockchain (paragraphe sur les autorisations).

Le nœud doit à minima contenir un Clone. Le clone sera en charge de se connecter à la blockchain, d'établir une connexion sécurisée avec chaque Master Node et de synchroniser les données sur lesquelles il est autorisé pour établir sa base de données en mémoire.

On peut ensuite y ajouter notre métier. Pour cela deux choix s'offre à nous :

- Utilisation de callbacks directement dans le code source du nœud
- Exécutions de smart contrats selon les règles de notre choix ➔ Cependant cette option est possible uniquement avec l'API Java, via des contrats javascript ou python, et prochainement via l'API C, via des scripts LUA.

2. Gestion des autorisations

Les Master Nodes régissent les autorisations de chaque Nœud. Pour cela, les administrateurs de la blockchain peuvent définir des listes d'autorisations en PUB (Publish) et en SUB (SUBSCRIBE).

Par exemple, on peut créer une permission « demo », qui serait en autorisation SUB les adresses /alarms/fire et /sensors, et en autorisation PUB uniquement l'adresse /sensors. Cette permission permettrait donc d'autoriser des nœuds sur les données capteurs en lecture et en écriture, et sur les alarmes incendies uniquement en lecture.

Lors de la création d'un nœud, un identifiant unique appelé deviceId est créé. Ce deviceId permettra par la suite d'identifier la nœud sur la blockchain. Cependant, lors de la première connexion du nœud sur la blockchain, le nœud n'est pas encore connu, il n'est donc pas autorisé sur la blockchain. On initie donc l'identification du nœud via une autorisation temporaire d'un identifiant appelé le serialId. Le serialId sera utilisé uniquement lors de la première connexion du nœud sur la blockchain, et permettra de faire le lien entre le deviceId et une autorisation.

3. Smart contracts

Un Smart Contrat, ou contrat intelligent, est un programme informatique qui automatise l'exécution d'un ensemble d'instructions prédéfinies lorsque des conditions préalables sont réunies.

Durant son exécution, toutes les étapes de validation sont enregistrées au niveau de la blockchain. Ces enregistrements assurent la sécurisation de l'ensemble des données en empêchant leur modification ou leur suppression.

Comme expliqué plus haut, des smart contrats peuvent être exécutés par des nœud clients, grâce à l'API Java (pour des contrats javascript voir python) et prochainement grâce à l'API C (pour des contrats LUA).

La Blockchain Kalima utilise des Smart Contracts développés par des outils standards et ouverts comme JavaScript et Python, ces outils ouverts permettent sa simple interconnexion avec d'autres blockchains, d'autres bases de données et d'autres applications.

Dans ces Smart Contrats on peut faire de l'inférence de l'IA grâce à des modèles crées éventuellement à partir des données collectées de préférence par Kalima, car ce sont des données immuables, sûres et complètes.

Nous allons décrire ici le processus de publication des contrats.

a. Stockage des contrats

Les contrats sont stockés sur des répertoires git. Pour modifier un contrat, il suffit donc de pousser les modifications sur le répertoire git.

b. Déploiement des contrats

La publication des contrats vers les nœuds responsables de leurs exécutions se fait par le biais d'une chaîne devOps après chaque commit de contrat. Le déploiement se fait en 4 étapes :

- Cryptage du contrat : La chaîne devOps va crypter le contrat en AES256 avec des clés de cryptages générées aléatoirement.
- Calcul de la signature : Une signature est ensuite calculée sur le contrat crypté
- Publication sur la blockchain : La signature du contrat, ainsi que les clés permettant de le décrypter sont publiés sur la blockchain. Ainsi, seuls les nœuds autorisés sur ces données seront en mesure d'exécuter des smart contrats.
- Déploiement des contrats : La chaîne devOps se charge ensuite de déployer les contrats cryptés vers les machines cibles.

c. Chargement d'un contrat

Pour charger et exécuter un contrat, il faut installer un nœud dédié sur une machine. Une fois que les contrats ont étés correctement déployés sur la machine par la chaine devOps, le nœud va détecter l'arrivé ou la modification de contrats, et va pouvoir charger le contrat en 4 étapes :

- Récupération des informations du contrat : Si le nœud est correctement connecté à la blockchain, et bien autorisé, il aura déjà reçu les informations du nouveau contrat. Il va donc aller lire ces informations pour récupérer la signature du contrat, ainsi que les clés nécessaires à son décryptage.
- Vérification de la signature du contrat : Le nœud va calculer la signature du contrat qu'il a reçu, et vérifier qu'elle correspond à celle enregistrée en blockchain. Si les signatures de correspondent pas, il ne fera rien.

- Décryptage : Si les signatures concordent, il tentera de décrypter le contrat en mémoire, via les clés présentes en blockchain.
- Chargement : Si le décryptage se déroule correctement, il chargera le script en mémoire, et sera désormais capable de l'exécuter.

Le cryptage ainsi que la signature des contrats empêchent toute tentative de modification frauduleuse des contrats.

d. Exécution d'un contrat

Une fois que le nœud a pu charger avec succès un contrat, il va pouvoir l'exécuter selon les règles que l'on a définies. Pour exécuter un contrat on doit exécuter une fonction présente dans le script, on peut lui passer les paramètres de notre choix, et on peut attendre un retour du script.