« Comment sécuriser les smart contracts dans un environnement décentralisé de type blockchain »

Mission Smart, Partie 2 : Développement de solutions open-source

1) Eléments clés de la réponse :

Eléments clés de la réponse	
Axe de l'appel à contributions	EQUIPER : développer des composants fondamentaux pour les outils de sécurité et mettre en place un socle technique pour servir les autres axes
Licence Open-source	Apache
Accès à la solution gratuit	Utilisation de blockchain EVM POLYGON de test (Amoy)
Sujet retenu	Détection automatique des smart contracts des projets (7.a.4.1) : ● Identification et catalogage des smart contracts. ● Création d'une base de données consultable. ● Développement d'une interface utilisateur. ● Intégration de la vérification des versions et des mises à jour.
contacts	cyril.vignet@bpce.fr +33 6 22 04 08 56 jose.luu@bpce.fr +33 6 24 07 19 67

Avec l'explosion anticipée du nombre de smartContracts due à la tokenisation de l'économie et l'account abstraction, nous pensons que la première étape doit être la facilité de création de listes de références pour des accès privatifs ou publics, hors de la blockchain ou en interne de la blockchain.

2) Décrivez votre idée / projet : en quoi et comment votre proposition permet de répondre à la problématique adressée ? En quoi votre projet est-il innovant ?

Problématique : la multiplication des smartContracts

La vision autour des smartContracts a fortement évolué dans l'écosystème Blockchain, principalement Ethereum. Initialement ils étaient perçus comme des artefacts quasi-uniques dont l'utilisation est garantie par la connaissance de l'adresse du smartContract. Une telle sécurité d'utilisation nécessite que chaque utilisateur gère par ses propres moyens, la liste des adresses à laquelle il fait confiance.

Cette approche pose un réel souci dès lors que le nombre de smartContracts utilisés par chaque utilisateur augmente. Par ailleurs, ceci oblige à mettre l'adresse des smartContracts en dur dans le code d'autres smartContracts dès lors que certains processus nécessitent une vérification on-chain impliquant plusieurs smartContracts.

Depuis 2 à 3 ans,deux nouvelles orientations sont considérées comme inévitables :

- l'account abstraction, qui va permettre une meilleure interaction utilisateur (paiement de GAS par un tiers, protection contre la perte de clés privées),
- la tokenisation de l'économie qui nécessite une représentation des actifs du monde réel sous forme de token fongibles ou non fongibles.

Ces deux orientations vont nécessiter :

- la création de smartContracts en grands nombres pour les actifs et les utilisateurs,
- la création de smartContracts liés à la gestion des processus des actifs, en nombre guère plus restreint, avec un besoin de contrôles des smartContracts précédents dans toutes les étapes de leur propre exécution.

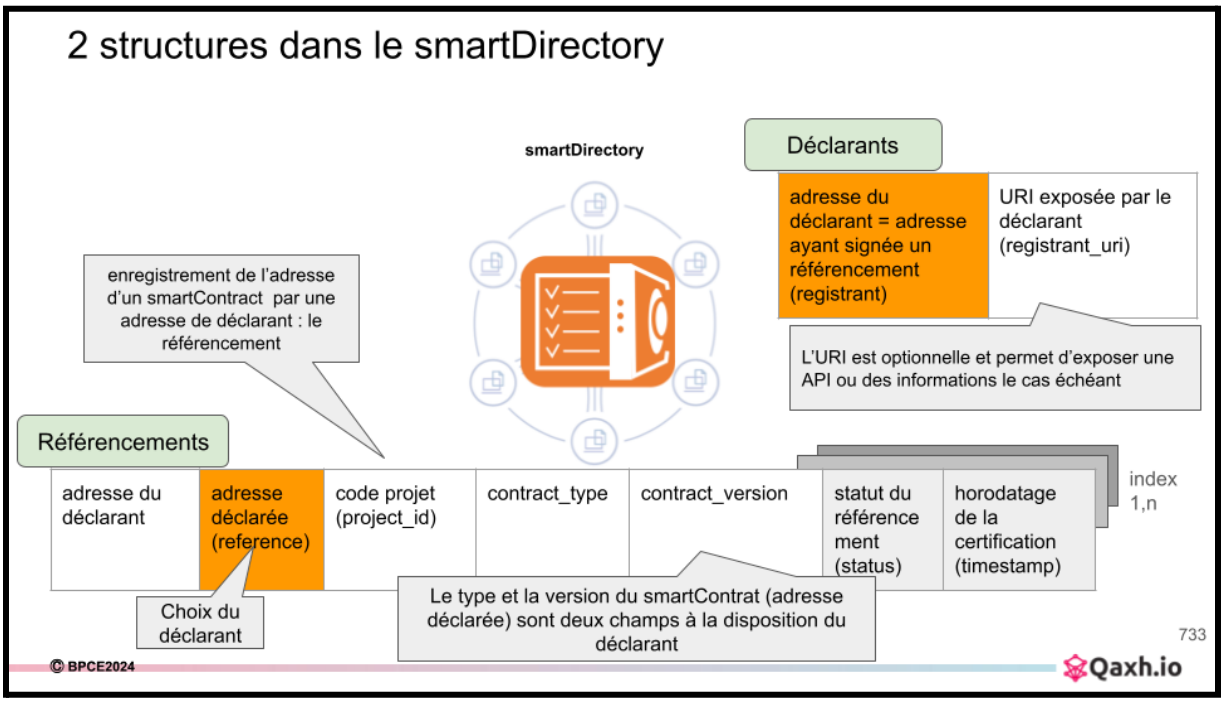
En synthèse, nous identifions les besoins génériques suivants :

- une facilité de déploiement de nouveaux smartContracts,
- une capacité de contrôle de la validité de ces smartContracts par un utilisateur externe à la blockchain,
- une capacité de contrôle de la validité de ces smartContracts par un autre smartContract,
- une nécessité d'identifier des écosystèmes de smartContracts pour en faire une analyse,
- une capacité de maintenir à jour ces écosystèmes.

Le composant central de la solution proposée : le smartDirectory

Le smartDirectory est un smartContract qui va permettre de conserver de façon partagée sur la blockchain des listes d'adresses, soit EOA soit smartContracts. Fonctionnellement, ces listes représentent les tables des matières des différents écosystèmes.

Une entité, **le déclarant**, munie d'un accès blockchain avec capacité de signature pourra écrire un ou plusieurs enregistrements dans le smartDirectory permettant la constitution de ces listes. Pour laisser de la latitude aux utilisateurs, le fonctionnement du smartDirectory reste le plus basique possible avec principalement deux structures logiques : les référencements et les déclarants.



1. Les référencements, c'est-à-dire les listes des adresses enregistrées par un déclarant.
2. Les déclarants, c'est-à-dire la liste de toutes les adresses des déclarants, constituée et mise à jour automatiquement à chaque création d'un nouvel enregistrement dans les référencements.

Liste des déclarants, fonctionnalités accessibles :

- En lecture
    - Une fonction de récupération permet d'obtenir la liste complète des déclarants.
    - Une fonction de récupération prenant en paramètre l'adresse d'un déclarant permettant la lecture de l'URI du champ complémentaire.

        Ce champ complémentaire est optionnel. Un exemple d'utilisation est d'y placer une URI d'API permettant de consulter des données sur les codes

projets (projetID), les types de contrats (type) et les versions de contrats (version).

- En écriture
    - Mise en place ou changement de l'URI, faisable uniquement par le déclarant lui-même (validé par une signature correspondant à son adresse).

Liste des référencements, fonctionnalités accessibles :

- En lecture
    - Une fonction de récupération permet d'obtenir la liste des smartContracts enregistrés(adresses déclarées) par un déclarant sous un code projet (projectID).
    - Une fonction de récupération permet de demander les informations label déclarant, version de smartContract, statut du référencement, timestamp de ce statut) avec en entrée le triplet (adresse de déclarant, code projet, adresse déclarée).
- En écriture
    - L'écriture initiale pour l'ajout d'une ligne est ouverte à tout participant volontaire (ou bien contrôlée par un mécanisme non décrit) :
        - L'adresse du déclarant est décidée par le smartDirectory lui-même comme étant l'adresse correspondant à la signature de la transaction demandant l'écriture.
        - La première écriture issue d'un adresse non encore répertoriée dans la liste des déclarants ajoute une entrée dans cette dernière dont le champ URI devra si nécessaire être rempli ultérieurement. La clé est constituée des trois champs (adresse déclarée, code projet,).
    - L'écriture en modification n'est accessible qu'au déclarant lui-même, l'usage prévu étant de signaler qu'une version de contrat est devenue obsolète. Les champs initiaux ne sont pas modifiables, mais un "statut" (chaine de caractères) horodaté peut être ajouté. La liste de tous ces statuts est disponible en lecture.

Code projet (projectID) : Celui-ci permet à un déclarant d'organiser ses smart contracts à sa guise. En particulier, un smartContract peut être référencé dans plusieurs codes projet par un même déclarant.

Ce code projet permet ainsi de faire des listes dans lesquelles le déclarant donne une visibilité à des smart contracts qui peuvent ne pas être les siens.

Exemples :

- liste des smartContracts représentants des Third Party Providers,
- liste des smartcontracts de confiance pour le service XYZ),
- listes pour l'usage propre du déclarant (sans que cela soit pertinent d'en faciliter une transparence quelconque).

Les cas d’usages

1. Usage privatif (ou de supervision) par exemple *un annuaire de partenaires commerciaux*:

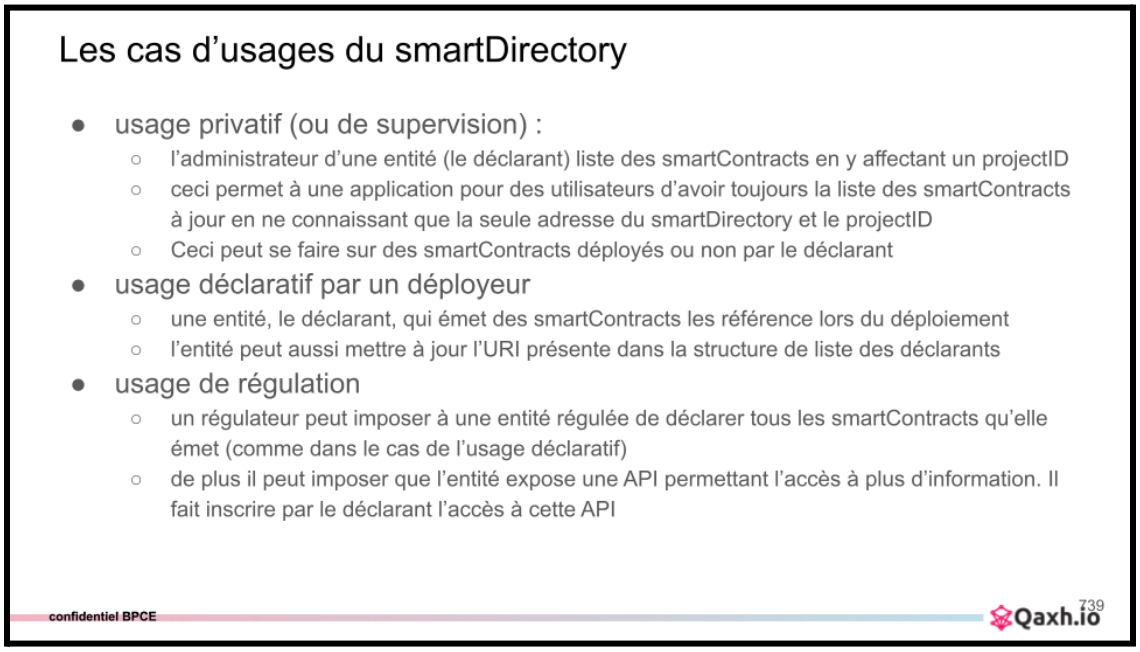
   Une entreprise veut lister les adresses de ses partenaires commerciaux avec sécurité. Dans son App, elle ne référence que l’adresse du smartDirectory, son adresse de déclarant ainsi que le ou les codes projet nécessaires. Son App peut donc facilement lire la blockchain pour mettre à jour la liste des partenaires, sous forme d’une liste d’adresses de blockchain (EOA ou smartContracts).

2. Usage déclaratif par exemple *un écosystème de smartContracts:*

   Une entreprise veut indiquer l’écosystème de smartContracts qui lui permet d’assurer ses obligations de gestion de consentement, d’authentification par clés EOA. Ainsi ses partenaires digitaux peuvent concevoir leur propre App et par simple connaissance du code projet récupérer la liste des smartContracts et des informations nécessaires complémentaires avec l’URI d’API présente dans la liste des déclarants.

3. Usage de régulation par exemple *une piste d’audit:*

   Un organisme de contrôle, de régulation ou d’audit, veut interroger régulièrement une liste de smartContract. Celle-ci est définie directement dans le smartDirectory par les administrateurs des déclarants. L’application de surveillance se met à jour automatiquement en allant lire la liste des adresses à surveiller sur le smartDirectory. L’organisme de régulation impose ses règles d’usage aux déclarants.



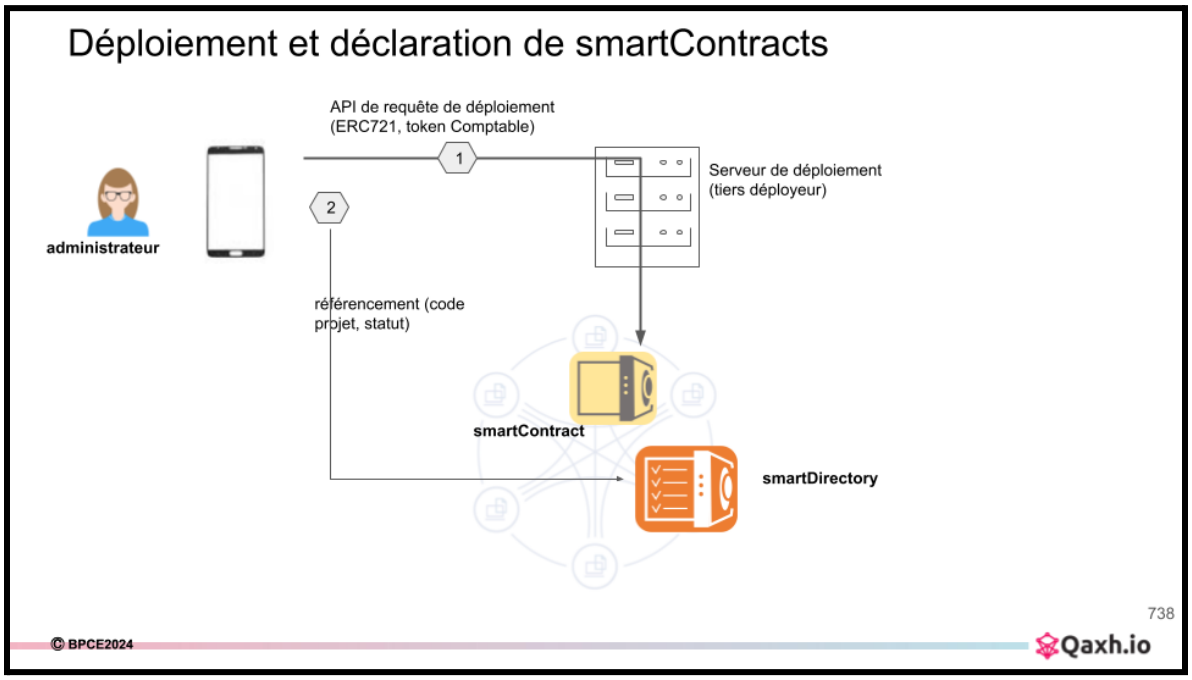
Un second composant d'aide au déploiement : le tiers déployeur

Le smartDirectory permet de créer des listes d'adresses qui peuvent être lues tant par des acteurs externes que directement par d'autres smartContracts.

Pour les smartContract déjà existants, le déclarant devra réaliser les listes de manière déclarative.

Pour les nouveaux smartContracts, il nous semble important de proposer un déploiement et un référencement "synchronisés" tant pour le confort de l'administrateur des déploiements que pour la conformité à des pratiques facilitant la supervision par les régulateurs, superviseurs et auditeurs.

Comme dit dans l'introduction, pour la création d'un écosystème important de tokenisation d'actifs du monde réel, il peut être nécessaire de déployer un grand nombre de smartContrats de type ERC20, ERC721 et assimilés.



De façon synthétique, ce second composant consiste en :

- un serveur comprenant un noeud de la blockchain et une liste de smartContracts réutilisables en provenance d'une bibliothèque interne,
- ce serveur offre des API permettant de demander le déploiement des smartContracts de la bibliothèque avec des paramétrages définis par une API et dépendant du type de smartContract,

Le protocole est découpé en deux échanges :

1. La demande de déploiement proprement dite par l'administrateur,
    a. suivi par le déploiement sur la blockchain,
    b. et, en retour d'API, la transaction de déploiement.

2. La validation du déploiement par l'administrateur et référencement (le serveur a-t-il bien déployé ce qui a été demandé ?) :
    a. Vérification du minage effectif du déploiement.
    b. Vérification de la prise en compte des paramètres du smartContract.
    c. Une transaction dans le smartDirectory (label déclarant et le code projet).

Cette mise en séquence est nécessaire car le déploiement se fait par un tiers (le serveur) et donc le consentement de l'administrateur doit être enregistré. De plus, cette séquence permet de ne pas utiliser pour le référencement les mêmes clés que celles du serveur, car autant le déploiement doit se faire par EOA, autant il peut être pratique d'utiliser un autre compte EOA ou abstrait pour le référencement.

Une telle architecture est compatible avec un administrateur humain voulant valider tous les déploiements, par analogie à un système "4 yeux" mais peut être simplifiée avec un serveur qui auto-valide et déclare ses déploiements.

Avantage de la solution

L'avantage d'une telle architecture réside dans :

1. une banalisation des déploiements et de leur référencement par l'utilisation d'API,
2. une capacité d'insérer un déploiement et une référencement dans un processus métier,
3. une capacité des smartContracts de processus de vérifier un groupe d'utilisateurs (représentés par des adresses) sans en connaître a priori la liste,
4. une facilité de déploiement des account abstraction smartContracts ce qui permet un enrôlement automatique d'un utilisateur.

3) Au terme de ce challenge, sur quoi souhaiteriez-vous aboutir ? Que souhaitez-vous développer durant ce challenge ?

Les objectifs du projet sont :

1. La réalisation en solidity du smartDirectory et son déploiement sur une blockchain de test, par exemple polygon AMOY.
2. La réalisation d'une APP sur Android pour déclarant à des fins d'UI de démonstration pour écrire (copier-coller), scanner sous forme de QRcode des adresses et créer une liste en tant que déclarant.

   N.B. : la réalisation d'une app android est proposée ici pour démontrer le concept dans des délais courts. Une interface de type web sur PC est bien évidemment un frontend possible.
3. La mise en place d'un serveur de déploiement avec 3 types de smartContracts (ERC20, ERC721, smartDirectory) en déploiement par API.
4. La réalisation d'une APP sur Android pour superviseurs permettant la consultation sélective et par critères d'un smartDirectory.

Synthèse des fonctionnalités proposées :

Fonctionnalité	Description
Identification et catalogage des smart contracts*	En utilisant l'adresse du déclarant et le code projet il est possible de créer différents écosystèmes d'adresses.
Création d'une base de données consultable*	Le smartContract "smartDirectory" enregistre les adresses des smartContracts et un code projet ainsi que l'adresse déclarante. Le smartDirectory expose des "getters" permettant la lecture directe de chaque smartContract déclaré ou bien des listes.
Développement d'une interface utilisateur- déclarant*	L'interface déclarant permet : - de choisir le type de smartContract à déployer, - de choisir les paramètres par défaut pour chaque type, - de remplacer les paramètres par défaut lors de la demande de création d'un smartContract, - de valider par transaction blockchain toute création de smartContract, - de lister les smartContracts déclarés,

	c'est-à-dire enregistrés sur la smartDIrectory,
Développement d'une interface utilisateur- superviseur*	Cette interface superviseur permet : - de filtrer la liste des smartContracts déclarés par des libellés (projectID), - de filtrer cette liste par version, - de filtrer cette liste par adresse de déclarant, - d'exporter la liste des adresses ainsi sélectionnées.
Intégration de la vérification des versions et des mises à jour*	Chaque smartContract intègre un numéro de version à l'origine. Ce numéro d'origine ne peut être modifié mais l'ajout de status horodatés et historisés permet les mises à jour le cas échéant.

* fonctionnalité identifiée dans le cahier des charge

Potentiellement on pourrait réaliser un EIP (Ethereum Improvement Proposal) qui propose une approche standard sur :

- les interactions nécessaire au déploiement et à la déclaration d'un smartContract (API et protocole),
- l'accès au smartDirectory,
- le paramétrage générique minimal que tout smartContract doit mettre en oeuvre pour être déclarés et être vérifiable par d'autres smartContracts,
- des exemples.

4) Quel est la maturité de votre projet ? Quelles sont les actions que vous avez déjà entreprises ?

Ce projet tire parti de fonctionnalités déjà développées par l'équipe de R&D blockchain du Groupe BPCE et qui seront adaptées pour le projet par cette même équipe. Cette équipe existe depuis début 2018 et possède tous les éléments et outils de développement sur des blockchains EVM, en particulier, un serveur avec des API et un noeud d'accès sur plusieurs blockchains de test : Polygon-AMOY, ZAMADEV, EVM-XRPL.

Différents tests et proofs of concept, comprenant des applications sous Android, ont été réalisés avec des concepts similaires ce qui permettra de tenir les délais en incorporant des fonctions en partie déjà existantes. En particulier, un smartContract (le schemeDirectory) est utilisé pour la structure de confiance des smartWallets d'identité du projet Qaxh.io. Ce smartContract offre un service analogue mais dans un contexte privatif et avec des éléments de sécurité spécifiques.

5) Quelles ressources comptez-vous mobiliser durant ce challenge (nombre de personnes, expertises etc) ?

Quatre personnes de l'équipe R&D blockchain vont participer au projet regroupant les expertises nécessaires : Solidity sur EVM, outils de développement, API python, node Ethereum, fonctionnel et interface smartphone.

José LUU	Tech lead du projet Qaxh.io depuis 2017 co-auteur de l'article "Blockchain and the nature of money" dans "Banque et stratégie" September 2016 https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2939042 Précédemment responsable du logiciel de valorisation des dérivés actions à Natixis.
Hamza MEKHANEG	Elève Ingénieur Alternant en dernière année de Mastère de l'Ingénierie de la Blockchain de l'Ecole Supérieure de Génie Informatique, Lauréat de plusieurs Hackathon (Hackathon Hackin'dau, Hackathon HEC x Tezos, Hackathon ETH Global Istanbul)
Cyril VIGNET	Ingénieur, animateur de la coordination blockchain du Groupe BPCE depuis 2015. Co-leader du projet de R&D Qaxh.io visant à concilier utilisateur de banque de détail et blockchain publique. Co-auteur de 10 brevets dans le domaine de la blockchain Design et interface utilisateur
Vincent GRIFFAULT	Directeur de projets Transformation Digitale & Data Caisse d'Epargne Rhône Alpes Certificate of Advanced Studies Blockchain & DLT@University of Geneva Contributeur du projet de R&D Qaxh.io

6) Quels sont les prérequis dont vous auriez besoin pour mener à bien cette mission (données, ressources, avis d’experts…) ?

A ce stade, l’équipe possède l’ensemble des ressources pour mener à bien le projet proposé. Cependant, toutes les propositions sont les bienvenues pour compléter le projet et offrir une richesse fonctionnelle plus importante. Ceci peut intervenir en début ou en milieu de projet.

Lien vers spécifications