beAchain : cryptage et preuve d'ipséité

beAchain, blockchain orientée en objets, est un réseau de machines sociales qui engagent des transactions cryptées et non-anonymes. Explications.

L'encryptage beAchain

L'encryptage **beAchain** est particulier. Ce n'est pas, ou alors de façon très marginale, un encryptage destiné à protéger les données contre des utilisateurs humains malveillants - fraudeurs, hackers - mais plutôt un mode de cryptage optimisant ces données pour les rendre plus légères à transporter, plus rapides à computer et plus sûres à gérer.

Imaginons un smart-contract censé décrire la transaction suivante : un restaurateur veut acheter 100 kilos de pommes à un producteur à un certain prix et souhaite se les faire livrer directement au restaurant.

En langage humain le dialogue donnerait en gros ceci :
- Combien coûtent vos pommes au kilo ?
- Deux euros quarante.
- Si vous me les faites à deux euros je vous en prend cent kilos.
- D'accord pour deux euros.
- Vous me les livrez ?
- Oui, à quelle adresse ?
- Au restaurant La Bonne Table, 32 rue des Acacias.
soit environ 350 signes à traiter.

beAchain serait plutôt en mode SMS :
- ? kg
- 2.4
- si OK 2 > 100 kg
- OK où ?
- 48.539960, 7.735490
soit 48 signes transités et traités. 83% d'économie en kilo-octets.

L'encryptage **beAchain** est un *cryptage réductionniste* dont l'objectif est de réduire à son minimum la taille des datas distribuées. Par exemple l'information « *Le 30 avril 2016 à 14h30'22'' la machine 365109 détient 2350 unités approuvées par les machines 130274, 761430, 985532 et 452867* » donnera par exemple « *eR_7jB*n06F(E48h1-Gf§nsŒ* », soit seulement 24 signes.

Un exemple simple pour comprendre la *logique réductionniste* issue des principes de la sémantique. Si je dis « *je veux acheter une automobile* » ou si je dis « *je souhaite acquérir une auto* » ou si je dis « *je vais me payer une voiture* » j'énonce, une fois réduit, un schéma de type [je] [acheter] [voiture]. Si dans le cryptage algorithmique [je/moi] vaut A, [payer/acheter/acquérir] vaut 5 et [voiture/auto] vaut Z, ce que je veux faire est « A5Z » : 3 signes suffisent pour réduire la formulation initiale, quelle qu'ait pu être son énonciation originelle*.

Contrairement aux blockchains «classiques» écrites en base 58 (58 signes distincts facilement différenciés par des utilisateurs humains) ces 24 signes seront codés sur plus de 100 signes distincts mixant signes accentués et signes clavier. Ce mode d'encodage permet de transférer des informations beaucoup plus courtes, donc plus rapides à computer. Sur un ensemble transactionnel complexe type smart-contract, gagner quelques kilo-octets par information peut représenter un gain de quelques millièmes de secondes par transaction, ce qui, multiplié à la fois par le nombre de transactions, de machines et de contrats, peut représenter plusieurs kilooctets et plusieurs secondes de calcul économisés.

Aucun humain n'ayant jamais accès à ces datas cryptées constituant les blocks de la chain, n'importe que encryptage conviendrait pourvu qu'il soit optimisant et raccourcissant. Mais parmi toutes les machines potentiellement connectées à la chain il n'y a pas que des objets honnêtes : on peut imaginer y trouver des bots dont le travail est de délivrer les datas de façon lisible à des hackers mal intentionnés. Un cryptage basique permettrait à n'importe quel bot de traduire ces informations. On doit donc marginalement protéger les datas partagées contre une possible attaque.

beAchain comporte deux dispositifs de sécurité :
1. les datas sont retraitées avant d'être encodées : « *la machine 123456 appartient à Alain Brégy* » devient « *123456:Alain_Brégy* » qui, hashée, devient « *12AlBr34aiég56:n_y* » qui une fois saltée devient « *e12AlBre34aiég56e:ne_ye* » pour finir cryptée publiquement par l'algorithme en « *ëbZKdy[nOzM4-sB9* »
2. chaque machine connectée a une clé privée utilisée comme signature dans les process de validation consensuelle. Cette clé privée réencrypte les datas détenues par chaque machine : notre code « *ëbZKdy[nOzM4-sB9* » obtenu juste avant est algorithmiquement produit en clé publique ; sur la machine 967431 il deviendra « *u5där9]_Kd34zEÔ* » et « *Fj!8m_dEëbçVn(65* » sur la machine 345280.

Chaque transaction modifiant aussi bien les datas que la structure des blocks concernés, la chain est alors intégralement réécrite soit directement dans le cycle de la transaction pour les machines approbatrices, soit par capillarité pour le reste du réseau. Hacker **beAchain** consiste à prendre le contrôle de chaque machine, récupérer sa chaine de bloc (le *ledger* distribué) et sa clé privée, décoder/décrypter/déhasher/désalter les blocks, modifier le block-cible à falsifier puis le réencoder-réencrypter selon chaque clé de chaque machine avant de réinjecter la chain piratée, le tout avant qu'une nouvelle transaction ne vienne changer les datas de chain de chaque machine. Si on a une transaction par minute on a une minute pour le faire. Si on a 1000 transactions/seconde, on a un millième de seconde.

* c'est cette même logique réductionniste en ULS (unités lexicales sémantiques de type [je] [acheter] [voiture]) qui sera mise en oeuvre dans l'interface utilisateur de *rédaction de smart-contracts en langage naturel*. Un algorithme d'IA (intelligence artificielle) analyse la structure de chaque condition contractuelle (si ceci alors cela : *if / then*) pour la réduire en ULS qui seront elles-mêmes cryptées puis encodées dans le blocks de la chain de sorte que si la condition est remplie, la suite du contrat se réalisera automatiquement. L'exemple cité plus haut (les pommes du producteur) ressemblerait à ceci :

1. Le restaurateur écrit directement dans l'interface.
[case IF 1] : *si les pommes coûtent 2 euros/kg* => ULS : [objet] [valoir] [2/k]
[case THEN 1] : *je vous en commande 100 kg* => ULS : [je] [acheter] [100k]
[case IF 2] : *si vous les livrez 32 rue des Acacias* => ULS : [vendeur] [transporter] [coord. GPS]
[case THEN 2] : *paiement 200 euros à livraison* => ULS : [je] [verser] [vendeur] [200]

2. Le producteur accepte le contrat en le signant d'un clic. Sa machine (authentifiée, reconnue) signale qu'elle accepte le contrat et envoie les datas sur le réseau pour son approbation/validation consensuelle puis son écriture dans un block rajouté à la chain. Quand les 100 kilos de pommes seront géolockées au point GPS du restaurant, la machine 897654 (n° ID machine restaurateur) versera 200 unités à la machine 123456 (n° ID machine producteur).

La preuve d'ipséité (PoI)

Pour qu'une machine soit reconnue apte à participer à une validation consensuelle de transaction elle doit d'abord démontrer qu'elle est bien qui elle prétend être. Si l'utilisateur est toujours anonyme, sa (ou ses) machine(s) ne l'est (le sont) pas.

beAchain n'utilisant ni PoW (*Proof of work*) ni PoS (*Proof of stack*) la confiance n'est ni dans le travail produit ni dans ce qui est détenu par la machine mais dans une caractéristique interne à chaque machine : son ipséité. Ce qui fait qu'elle est elle-même et pas une autre. En termes conceptuels, l'identité est ce que nous partageons avec d'autres (même groupe, même famille, même nationalité, même origine, mêmes valeurs) alors que l'*ipséité* est ce qui nous distingue des tous les autres par nos *caractéristiques non-communes*.

S'authentifier/s'identifier est une transaction comme une autre : si une machine est connectée à **beAchain** c'est qu'elle a été reconnue et acceptée par consensus/validation lors de sa connexion. La *preuve d'ipséité* (*Proof of Ipseity*) est une épreuve supplémentaire. Au moment où une transaction est lancée (A verse 100 à B) un certain nombre de machines (Frères-Cousins) vont être appelées pour évaluer les termes de cette transaction et décider consensuellement si elle est est acceptée ou si elle est rejetée. Ces machines devront préalablement démontrer qu'elles sont bien qui elles disent être avant de donner leur avis sur la transaction : c'est la *preuve d'ipséité* attendue par les autres machines. Il s'agit d'une data cryptée envoyée par la machine au réseau. Produire cette data utilise le numéro de la machine, la date de sa première apparition/approbation (inscription), la date à laquelle sa dernière connexion s'est faite, le nombre de validations qu'elle a réalisées et quelques autres données. Le résultat est directement lié à ce que *chaque machine a d'absolument spécifique et qui fait qu'aucune autre ne peut obtenir le même*.

Ce qui est évalué par les autres machines à ce moment-là, ce n'est pas ce que chacune d'elles sait de la machine requérante (elle n'est ni A ni B, elle n'a pas à être consensualisée) mais si la data d'ipséité qu'elle passe est *possible et cohérente* : comme un PoW, cette data est compliquée à produire mais quand même très rapide (quelques micro-secondes) ; comme un PoW elle est très simple à vérifier. Le meilleur exemple est la grille de sudoku qui peut être très compliquée à remplir mais ultra-simple à vérifier : le total de chaque ligne/colonne vaut toujours 45.

Les fonctions de cryptage/encodage et les preuves d'ipséité forment l'ossature de sécurité de **beAchain**, garantissant fiabilité et confiance dans les transactions.