SIGLES ET ABRÉVIATIONS

AES: Advanced Encryption Standard

API: Application Programming Interface

B3i: Blockchain Insurance Industry Initiative

BTC: Bitcoin

Dapp: decentralized application

DLT: Distributed Ledger Technology

DSA: Digital Signature Algorithm

ECDH: Elliptic curve Diffie–Hellman

ECDSA: Elliptic Curve Digital Signature Algorithm

ERC: Ethereum Request for Comments

ETH: Ethereum

EVM: Ethereum Virtual Machine

KYC: Know your customer

LADP: Lightweight Directory Access Protocol

NIST: National Institute of Standards and Technology

NISTIR: National Institute of Standards and Technology Interagency Report

RSA: Rivest–Shamir–Adleman

SHA: Secure Hash Algorithm

UML: Unified Modeling Language

LISTE DES FIGURES

Figure 1 Comportement principal entrée-sortie des fonctions de hachage 9
Figure 2 Exemple de transaction crypto-monnaie 11
Figure 3 Arbre de Merkle binaire 19
Figure 4 Chaîne générique de blocs 19
Figure 5 Grands livres en conflit 27
Figure 6 La chaîne avec block_n (B) ajoute le bloc suivant, la chaîne avec block_n (A) est maintenant orpheline. 28
Figure 7 Le cycle de vie d'un contrat intelligent: phases, acteurs et services 44
Figure 8 Ganache 66
Figure 9 Truffle 67
Figure 10 Metamask 67
Figure 11 Ganache start. 69
Figure 12 Ganache workspace 69
Figure 13 Ganache Accounts 70
Figure 14 Account information 70
Figure 15 Réseau d'ethereum 71
Figure 16 Comptes metamask 71
Figure 17 Importation de la clé privéé 72
Figure 18 Account 72
Figure 19 Edit account 73
Figure 20 Compte créé 73
Figure 21 Truffle Config 74
Figure 22 Npm installation 75
Figure 23 Créer une nouvelle politique 76
Figure 24 Payez la prime via le portefeuille Ethereum de Metamask. 76
Figure 25 Réclamez une assurance en cas de tragédies comme les inondations ou la sécheresse. 77
Figure 26 Afficher les détails de votre politique 77

LISTE DES TABLEAUX

Tableau 1 Exemples de texte d'entrée et de valeurs de synthèse SHA-256 correspondantes ... 9

Tableau 2 Impact de l'informatique quantique sur les algorithmes cryptographiques courants 31

Tableau 3 Ajouts au diagramme de classes UML (stéréotypes) 46

Tableau 4 Ajouts au diagramme de séquence UML (stéréotypes) 47

SOMMAIRE

DEDICACES i
REMERCIEMENTS ii
RESUMÉ iii
AVANT PROPOS iv
SIGLES ET ABRÉVIATIONS v
LISTE DES FIGURES vi
LISTE DES TABLEAUX vii
SOMMAIRE 1
INTRODUCTION GÉNÉRALE 2
1ère partie : Cadre de référence et méthodologique 4
Chapitre 1 : Cadre méthodologique 5
Chapitre 2 : Background 8
2ème partie : Analyse d'un smart contract 40
Chapitre 3 : Infrastructure des smart contracts 41
Chapitre 4 : Etude des plateformes 49
Chapitre 5 : Modélisation des assurances non-vie 57
3ème partie : Implémentation de la solution 62
Chapitre 6 : Présentation des applications existantes 63
Chapitre 7 : Présentation de la solution 65
Chapitre 8 : Implémentation 66
CONCLUSION 78
BIBLIOGRAPHIE 79
WEBOGRAPHIE 80
TABLES DE MATIERES 81

INTRODUCTION GÉNÉRALE

Avec l'émergence des transactions qui se font à l'échelle mondiale entres les personnes, favorisant ainsi la multiplication et la diversification des bien disponibles, il est donc nécessaire pour les utilisateurs entre eux de signer un contrat pour garantir le respect des obligations et la sécurité de leur bien. De ce fait la conception d'un contrat entre les êtres humains actuellement est extrêmement complexe dans son exécution. Beaucoup d'évènements imprévisibles au moment de la signature d'un contrat ne peuvent être incorporés dans les contrats. C'est la raison pour laquelle nous avons recours à des juges et des courts, dont le rôle est d'interpréter le contrat et essayer de déduire les intentions initiales des parties à la lumière de nouveaux éléments. De plus la mise en œuvre d'un contrat prend du temps de la signature jusqu'à l'annulation du contrat, la rédaction du contrat est lente car il faut une demande d'intervention des juristes et l'échange des originaux pour la signature et elle a une sécurité limitée.

Cependant les limites rencontrées par les contrats causent un handicap aux différents utilisateurs et donc une perte de confiance en celui-ci, c'est pour cela avec l'évolution technologique qui s'étend sur plusieurs secteurs, certaines recherches effectuées par des chercheurs tels que **Nick Szabo** qui a travaillé sur des « smart contracts »[1] peuvent apporter des solutions aux limites rencontrées par les contrats actuels, car les smart contracts contrairement au contrat traditionnel sa mise en œuvre est immédiate et irrévocable, l'archivage est facile, la sécurité est basée sur la cryptographie, extraction des données est immédiate et automatique, la rédaction est rapide. C'est en raison de ces données majeures que nous avons orienté notre étude sur : **les smart contrats pour les assurances non-vie**.

Nous pouvons dire que l'assurance non-vie est un type de couverture très répandu qui couvre les entreprises et les particuliers. Parfois appelé assurance générale ou couverture de biens et de sinistres, il assure tout dans votre vie sauf... votre vie. Si votre maison ou votre voiture est détruite, c'est votre couverture non-vie qui vient à la rescousse.

La Technologie de smart contract est définie comme un code de programme informatique capable de faciliter, d'exécuter et d'appliquer la négociation ou la performance d'un accord (c.-à-d. contrat) à l'aide de la technologie blockchain.
L'ensemble du processus est automatisé et peut agir comme un complément, ou un substitut, pour les contrats juridiques, où les termes du smart contract sont enregistrés dans un langage informatique comme un ensemble d'instructions.

Une blockchain est une liste sans cesse croissante d'enregistrements numériques dans des packages (appelés blocs) qui sont liés et sécurisés à l'aide d'une cryptographie. Ces «blocs» de données enregistrés numériquement sont stockés dans une chaîne linéaire. Chaque bloc de la chaîne contient des données (par exemple, une transaction bitcoin), est haché de manière cryptographique et horodaté. Les blocs de données hachées s'appuient sur le bloc précédent (qui

[1] *http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html*

le précédait) de la chaîne, garantissant que toutes les données de la «blockchain» globale n'ont pas été falsifiées ni modifiées.

L'avenir des contrats sera probablement un modèle hybride humain-plus-code où les contrats sont vérifiés pour l'authenticité via la blockchain, mais l'intervention humaine est encore possible pour les cas où les erreurs doivent être corrigées ou un recours judiciaire est nécessaire.
Cependant nous pouvons nous poser quelques questions :
Les contrats d'assurance traditionnelle assurent-ils un état de confiance vis-à-vis des utilisateurs ?
Les données des utilisateurs sont-elles en sécurité ?

Le smart contract permet de résoudre ce problème par un registre de données informatique qui assure l'authenticité du transfert de donnée et qui sont hachée au moyen d'un algorithme cryptographique afin de créer une empreinte numérique

Le présent mémoire se propose de vous présenter la conception et la réalisation d'un smart contract pour les assurances non-vie.
Ce mémoire se présentera sous trois parties principales :
Dans la première partie qui est vraiment théorique, il sera question d'abord de présenter l'assurance non-vie et de passer à certains rappels dont la compréhension est nécessaire pour s'imprégner de la solution proposée.
La seconde partie abordera l'analyse des smart contrats.
Puis la troisième partie portera sur la phase technique, et présentera tour à tour une étude de l'existant, et ensuite une proposition de la solution, et la phase de mise en œuvre qui se traduira par une implémentation de la solution.
Enfin une analyse critique de la solution, et des perspectives relatives à son amélioration, son implémentation ou sa gestion constitueront les principales recommandations apportées à ce travail.

1ère partie : Cadre de référence et méthodologique

Chapitre 1 : Cadre méthodologique

1.1 Présentation des assurances Non-Vie

Le contrat d'assurance est un contrat aléatoire par lequel un organisme dit "l'assureur", qui pour pratiquer l'assurance doit être autorisé par le Ministère des Finances à exercer ce type d'activité, s'engage envers une ou plusieurs personnes déterminées ou un groupe de personnes dites les "assurées", à couvrir, moyennant le paiement d'une somme d'argent dite "prime d'assurance", une catégorie de risques déterminés par le contrat que dans la pratique on appelle "police d'assurance"[2]. Nous avons 2 types de contrat d'assurance qui sont :

- Les assurances vie ;
- Les assurances non-vie ou de dommages ;

Cependant nous allons ici faire la présentation des assurances non-vie qui, regroupent à la fois des assurances de responsabilité (civile familiale, civile du conducteur, etc.) et des assurances de biens (mobilier, dommages causés au véhicule, etc.). Dans le cas d'une souscription à un contrat prévoyant ce type de risques, une indemnisation des dommages est mise en place. L'indemnisation ne peut jamais excéder la valeur des biens endommagés, c'est le principe indemnitaire[3].
En somme nous possédons tous des biens chers à nos yeux : auto, maison, meubles, moto, ordinateur, etc. L'assurance de dommages a un objectif : **les protéger**.
Vol, incendie, accident, dégât d'eau, vol de données, atteinte à la réputation, les sinistres arrivent sans prévenir, bouleversant la vie des gens et des entreprises.
Pour un individu, l'assurance de dommages protège ses biens les plus importants, soit l'habitation et l'automobile.
Pour une entreprise, elle assure la pérennité de ses affaires qui pourraient être affectées de manière catastrophique advenant un sinistre.

[2] *Voir : https://www.dictionnaire-juridique.com/definition/assurance.php*
[3] *Voir : https://www.mapa-assurances.fr/Questions-frequentes/L-assurance/Les-assurances-de-dommages.-qu-est-ce-que-c-est*

1.2 Définition du besoin

En assurance non-vie les professionnels jouent un rôle essentiel en protégeant les biens des assurés et en les aidant à retrouver rapidement une vie normale en cas de sinistre.
De ce fait les contrats d'assurance non-vie aujourd'hui font intervenir plusieurs aspects juridiques, et aussi d'évènements aléatoires qui constituent une réalité, et prennent beaucoup de temps car il faut faire signer les documents et la rédaction du contrat prend énormément de temps. En outre les données des contrats d'assurance sont centralisées, nous avons donc une sécurité aléatoire. Ainsi cette sécurité entraine une perte de confiance des utilisateurs et aussi une perte d'autonomie car nous dépendons des prestataires. La technologie peut apporter la réponse, dans le passé les réseaux étaient centralisés sur une seule source, le réseau que nous avons aujourd'hui est centralisé autour de plusieurs sources mais nous sommes toujours dépendants d'elle, le réseau du futur il s'agit d'une architecture entièrement distribué dans laquelle nous sommes tous acteurs et nœud de ce réseau, cette architecture s'appelle la blockchain.

1.3 Importance de la question

Les données sont centralisées et les processus en matière d'assurance tournent autour du téléphone et des courriers électroniques. De ce fait les données peuvent être exposées à des pertes, ou des fraudes, ou à des attaques qui exploitent les éléments vulnérables du système d'information.

La décentralisation des données en mettant en place une architecture distribuée a pour objectif d'assurer la transparence, la sécurité et ainsi la confiance.

Les attaques, les pertes, et aussi la falsification sont une réalité. Pour en citer quelques-uns nous avons les pertes incendies, catastrophes naturelles, les attaques telles que les virus ou les vers sur le système utilisé par la blockchain.
Les assurances (Absence de classification…), la faiblesse des acteurs humains (naïveté, corruption, inconscience…), sont autant de sources de vulnérabilités.
Il est donc important de se préoccuper de la mise en place d'un processus de stockage et de transmission de données non modifiable c'est-à-dire qui est gravé a vie, et aussi d'un système sans organe de contrôle.

1.4 Formulation des objectifs

Les données sont au centre de l'industrie d'assurance, les administrations des services d'assurance sont confrontées à la croissance exponentielle de la complexité et du volume de données. C'est dans ce cadre qu'on m'a confié comme sujet de mémoire, la conception et réalisation d'un smart contract pour gérer l'assurance non-vie. Il s'agit de mettre en place un système distribué avec la blockchain qui assurera le stockage, la sécurité et de transmission d'informations. L'objectif de notre travail de recherche est de concevoir un smart contract pour gérer les assurances non-vie, avec solidity qui est un langage de programmation orienté objet permettant de décrire les smart contracts, et sont basées sur le principe de la blockchain. A la fin de notre travail, le smart contract réalisé, devrait permettre à chaque personne d'avoir un accès rapide à l'assurance et avec moins de coûts administratifs. Ainsi les services effectués

seront stockés dans la blockchain, ce qui garantira la sécurité, l'autonomie, donc la confiance pourra être obtenu avec garantie.

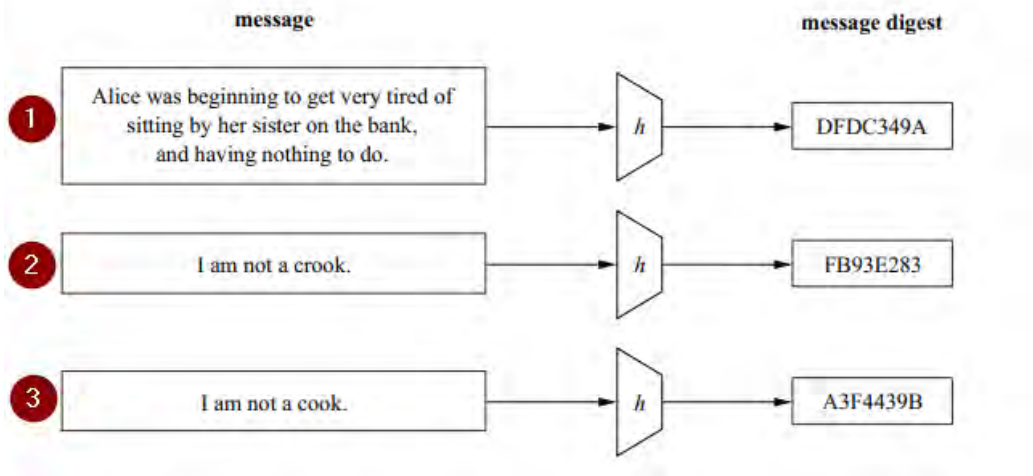
Chapitre 2 : Background

2.1 Architecture de la blockchain

La technologie des blockchain peut sembler complexe, mais elle peut être simplifiée en examinant chaque composant individuellement. À un niveau élevé, la technologie de blockchain fait appel à des mécanismes informatiques bien connus et à des primitives cryptographiques (fonctions de hachage cryptographique, signatures numériques, cryptographie à clé asymétrique) combinées à des concepts de tenue de dossiers (comme les grands livres annexés seulement). Cette section discute de chaque composante principale individuelle: les fonctions de hachage cryptographique, les transactions, la cryptographie à clé asymétrique, les adresses, les grands livres, les blocs, et comment les blocs sont enchaînés ensemble.

2.1.1 Fonctions de Hachage Cryptographique :

Un élément important de la technologie blockchain est l'utilisation de fonctions de hachage cryptographique pour de nombreuses opérations. Le hachage est une méthode d'application d'une fonction de hachage cryptographique aux données, qui calcule une sortie relativement unique (appelé un message digest, ou simplement digest) pour une entrée de presque n'importe quelle taille (par exemple, un dossier, un texte, ou une image). Il permet aux utilisateurs de prendre indépendamment les données d'entrée, de les hacher et d'obtenir le même résultat, ce qui prouve qu'il n'y a pas eu de changement dans les données. Même la plus petite modification apportée à l'entrée (par exemple, la modification d'un seul bit) entraînera un condensé de sortie complètement différent, comme indiqué dans la figure 1.



Comme vous remarquez sur figure ci-dessus, au niveau du 2 et 3 ont le même message d'entrée mais le hachage en sortie est différent, ceci est du car au niveau de la fonction de hachage 3 on a ajouté un nonce (un nombre entier utiliser une et une seule fois) au message.

Figure 1 Comportement principal entrée-sortie des fonctions de hachage [4]

Même si les fonctions de hachage ont de nombreuses applications dans la cryptographie moderne, elles sont peut-être mieux connues pour le rôle important qu'elles jouent dans l'utilisation pratique des signatures numériques.

Les fonctions de hachage cryptographique possèdent ces importantes propriétés de sécurité:

- elles sont résistantes à la pré-image. Cela signifie qu'ils sont à sens unique; il est impossible de calculer la valeur d'entrée correcte compte tenu d'une certaine valeur de sortie (par exemple, étant donné un résumé, trouver x tel que hash (x) = digest) ;

- elles sont résistantes à la deuxième pré-image, ce qui signifie que l'on ne peut pas trouver une entrée hachée sur une sortie spécifique. Plus spécifiquement, les fonctions de hachage cryptographiques sont conçues de sorte que, étant donné une entrée spécifique, il est impossible de trouver une deuxième entrée qui produise la même sortie (par exemple, étant donné x, trouver y tel que hash (x) = hash (y)). La seule approche disponible consiste à rechercher de manière exhaustive l'espace d'entrée, mais cela est impossible à faire sur le plan des calculs avec toute chance de succès ;

- elles sont résistantes aux collisions. Cela signifie que l'on ne peut pas trouver deux entrées hachées vers la même sortie. Plus précisément, il est impossible sur le plan informatique de trouver deux entrées qui produisent le même condensé (par exemple, trouver un x et un y avec hash (x) = hash (y)) ;

La fonction de hachage cryptographique spécifique utilisée dans de nombreuses implémentations de blockchain est L'algorithme de hachage sécurisé (SHA) avec une taille de sortie de 256 bits (SHA-256). De nombreux ordinateurs prennent en charge cet algorithme en matériel, ce qui le rend rapide à calculer. SHA-256 a une sortie de 32 octets (1 octet = 8 bits, 32 octets = 256 bits), généralement affichée comme une chaîne hexadécimale de 64 caractères (Voir tableau 1 ci-dessous))

Input text	SHA-256 Digest Value
1	0x6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
2	0xd4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f90da3a666eec13ab35
Hello, World !	0xdffd6021bb2bd5b0af676290809ec3a53191dd81c7f70a4b28688a362182986f

Tableau 1 Exemples de texte d'entrée et de valeurs de synthèse SHA-256 correspondantes

Puisqu'il existe un nombre infini de valeurs d'entrée possibles et un nombre fini de valeurs de résumé de sortie possibles, il est possible, mais très peu probable, d'avoir une collision où hash

[4] *Christof PAAR, Jan PELZL, Understanding cryptography*

(x) = hash (y) (c'est-à-dire que le hachage de deux entrées différentes produit le même condensé). SHA-256 est censé être résistant aux collisions, car pour trouver une collision dans SHA-256, il faudrait exécuter l'algorithme, en moyenne, environ 2^{128} fois (ce qui représente 340 milliards, ou plus précisément 340 282 366 920 938 463 463 374 607 431 738 211 456; environ 3 402 x10^{38}).

Pour mettre cela en perspective, le taux de hachage (hachages par seconde) de l'ensemble du réseau Bitcoin en 2015 était de 300 quadrillions de hachages par seconde (300 000 000 000 000 000 000 / s)[5]. À ce rythme, il faudrait à l'ensemble du réseau Bitcoin environ 35 942 991 748 521 (environ 3,6 x10^{13}) ans[6] pour fabriquer une collision (notez que l'univers est estimé à 1,37 x 10^{10} ans)[7]. Même si une telle entrée x et y produisant le même résumé, il serait également très peu probable que les deux entrées soient valides dans le contexte du réseau de blockchain (c'est-à-dire que x et y sont tous deux des transactions valides).

Au sein d'un réseau blockchain, les fonctions de hachage cryptographiques sont utilisées pour de nombreuses tâches, telles que:

- Dérivation d'adresse que nous allons voir dans la section 1.4 ;
- Créer des identifiants uniques ;
- Sécuriser les données de bloc - un nœud de publication hachera les données de bloc, créant un résumé qui seront stockés dans l'en-tête du bloc ;
- Sécuriser l'en-tête de bloc un nœud de publication hachera l'en-tête de bloc. Si le réseau de chaînes de blocs utilise un modèle de consensus de preuve de travail (voir section 2), le nœud de publication devra hacher l'en-tête de bloc avec différentes valeurs de nonce (voir le paragraphe 1.1) jusqu'à ce que les conditions requises soient remplies. Le condensé de hachage de l'en-tête du bloc actuel sera inclus dans l'en-tête du prochain bloc, où il sécurisera les données de l'en-tête du bloc actuel ;

Étant donné que l'en-tête de bloc comprend une représentation hachée des données de bloc, les données de bloc sont également sécurisées lorsque le résumé de l'en-tête de bloc est stocké dans le bloc suivant.

Il existe de nombreuses familles de fonctions de hachage cryptographiques utilisées dans la technologie de blockchain (SHA-256 n'est pas le seul), comme Keccak (sélectionné par le NIST comme le gagnant d'un concours visant à créer le standard de hachage SHA-3[8]).

[5] *https://en.bitcoin.it/wiki/Hash_per_second*
[6] *https://www.wolframalpha.com/input/?i=2%5E128%2F%28300000000000000000+*+60+*+60+*+24+*+365.25%29*
[7] *https://map.gsfc.nasa.gov/universe/uni_age.html*
[8] *https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf*

2.1.1.1 Cryptographie Nonce :

Un nonce cryptographique est un nombre arbitraire utilisé une seule fois. Un nonce cryptographique peut être combiné avec des données pour produire différents digests de hachage par nonce:

Hash (data + nonce) = digest

Seule la modification de la valeur nonce fournit un mécanisme permettant d'obtenir différentes valeurs de résumé tout en conservant les mêmes données. Cette technique est utilisée dans le modèle de consensus de preuve de travail (voir section 2.1).

2.1.2 Transaction :

Une transaction représente une interaction entre les parties. Avec les crypto-monnaies, par exemple, une transaction représente un transfert de la crypto-monnaie entre utilisateurs du réseau blockchain. Pour les scénarios d'entreprise à entreprise, une transaction peut être un moyen d'enregistrer des activités se produisant sur des actifs numériques ou physiques. La figure 5 montre un exemple théorique d'une transaction crypto-monnaie. Chaque bloc d'une blockchain peut contenir zéro transaction ou plus. Pour certaines implémentations de blockchain, un approvisionnement constant de nouveaux blocs (même sans transaction) est essentiel pour maintenir la sécurité du réseau de blockchain; en publiant en permanence de nouveaux blocs, cela empêche les utilisateurs malveillants de "rattraper" et de fabriquer une blockchain modifiée plus longue (voir section 2.7).

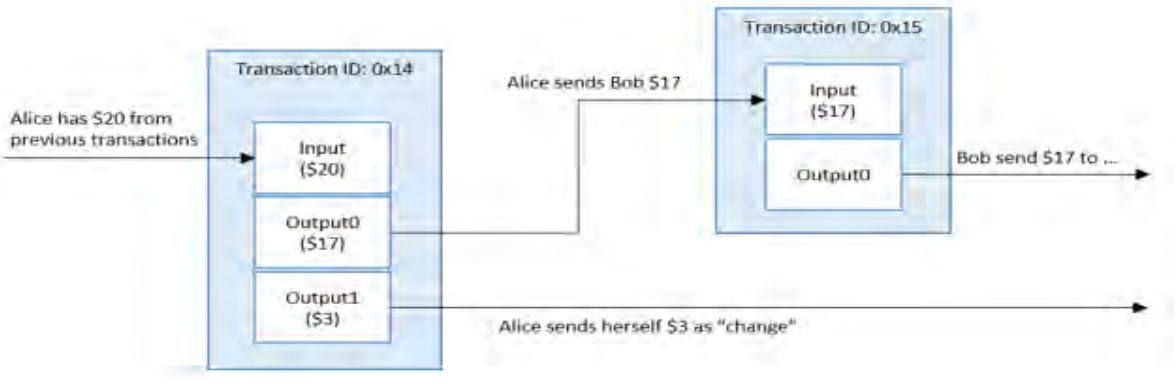


Figure 2 Exemple de transaction crypto-monnaie

Les données qui composent une transaction peuvent être différentes pour chaque implémentation de la blockchain, mais le mécanisme de transaction est en grande partie le même. Un utilisateur du réseau blockchain envoie des informations au réseau blockchain. Les informations envoyées peuvent inclure l'adresse de l'expéditeur (ou un autre identifiant pertinent), la clé publique de l'expéditeur, une signature numérique, des entrées de transaction et des sorties de transaction.

Une transaction de crypto-monnaie unique nécessite généralement au moins les informations suivantes, mais peut en contenir davantage:

- **Inputs** - Les entrées sont généralement une liste des actifs numériques à transférer. Une transaction référencera la source de l'actif numérique (fournissant la provenance), soit la transaction précédente où elle avait été transmise à l'expéditeur, soit, dans le cas de nouveaux actifs numériques, l'événement d'origine. L'entrée dans la transaction étant une référence à des événements passés, les actifs numériques ne changent pas. Dans le cas des crypto-monnaies, cela signifie que la valeur ne peut pas être ajoutée ou retirée des actifs numériques existants ;
  Au lieu de cela, un actif numérique unique peut être divisé en plusieurs nouveaux actifs numériques (chacun ayant une valeur moindre) ou plusieurs actifs numériques peuvent être combinés pour former moins de nouveaux actifs numériques (avec une valeur correspondante supérieure). Le fractionnement ou la jonction d'actifs sera spécifié dans la sortie de la transaction.
  L'expéditeur doit également fournir la preuve qu'il a accès aux entrées référencées, généralement en signant numériquement la transaction - prouvant l'accès à la clé privée ;

- **Outputs** - Les sorties sont généralement les comptes qui seront les destinataires des actifs numériques, ainsi que le montant des actifs numériques qu'ils recevront. Chaque sortie spécifie le nombre d'actifs numériques à transférer au (x) nouveau (s) propriétaire (s), l'identifiant du ou des nouveaux propriétaires et un ensemble de conditions que les nouveaux propriétaires doivent remplir pour dépenser cette valeur. Si les ressources numériques fournies sont supérieures aux besoins, les fonds supplémentaires doivent être explicitement renvoyés à l'expéditeur (il s'agit d'un mécanisme permettant de «rendre la modification») ;

Principalement utilisées pour transférer des actifs numériques, les transactions peuvent plus généralement être utilisées pour transférer des données. Dans un cas simple, une personne peut simplement vouloir publier de manière permanente et publique des données sur la blockchain. Dans le cas de systèmes de smart contract, les transactions peuvent être utilisées pour envoyer des données, les traiter et stocker certains résultats dans la blockchain. Par exemple, une transaction peut être utilisée pour modifier un attribut d'un actif numérisé, tel que l'emplacement d'une expédition dans un système de chaîne logistique basé sur la technologie blockchain.

Quelle que soit la manière dont les données sont formées et traitées, il est important de déterminer la validité et l'authenticité d'une transaction. La validité d'une transaction garantit que celle-ci répond aux exigences du protocole et à tous les formats de données formalisés ou aux exigences du contrat intelligent spécifiques à la mise en œuvre de la blockchain. L'authenticité d'une transaction est également importante, car elle détermine que l'expéditeur d'actifs numériques a accès à ces actifs numériques. Les transactions sont généralement signées numériquement par la clé privée de l'expéditeur (la cryptographie à clé asymétrique est brièvement décrite à la section 1.3) et peuvent être vérifiées à tout moment à l'aide de la clé publique associée.

2.1.3 Cryptographie à clé asymétrique :

La technologie Blockchain utilise la cryptographie à clé asymétrique[9] (également appelée cryptographie à clé publique). La cryptographie à clé asymétrique utilise une paire de clés: une clé publique et une clé privée qui sont liées mathématiquement. La clé publique est rendue publique sans réduire la sécurité du processus, mais la clé privée doit rester secrète pour que les données conservent leur protection cryptographique. Même s'il existe une relation entre les deux clés, la clé privée ne peut pas être déterminée efficacement en fonction de la connaissance de la clé publique. On peut signer avec une clé privée, puis déchiffrer avec la clé publique. Alternativement, on peut chiffrer avec une clé publique, puis déchiffrer avec une clé privée.

La cryptographie à clé asymétrique permet d'établir une relation de confiance entre les utilisateurs qui ne se connaissent pas ou ne se font pas confiance, en fournissant un mécanisme permettant de vérifier l'intégrité et l'authenticité des transactions tout en permettant aux transactions de rester publiques. Pour ce faire, les transactions sont «signées numériquement». Cela signifie qu'une clé privée est utilisée pour chiffrer une transaction de sorte que toute personne possédant la clé publique puisse la déchiffrer. Puisque la clé publique est librement disponible, le cryptage de la transaction avec la clé privée prouve que le signataire de la transaction a accès à la clé privée. Vous pouvez également chiffrer les données avec la clé publique d'un utilisateur, de sorte que seuls les utilisateurs ayant accès à la clé privée puissent la déchiffrer. Un inconvénient est que la cryptographie à clé asymétrique est souvent lente à calculer.

Cela contraste avec la cryptographie à clé symétrique[10] dans laquelle une seule clé secrète est utilisée pour crypter et décrypter. Avec la cryptographie à clé symétrique, les utilisateurs doivent déjà avoir une relation de confiance établie entre eux pour échanger la clé pré-partagée. Dans un système symétrique, toute donnée chiffrée pouvant être déchiffrée avec la clé pré-partagée confirme qu'elle a été envoyée par un autre utilisateur ayant accès à la clé pré-partagée; aucun utilisateur sans accès à la clé pré-partagée ne pourra voir les données déchiffrées. Par rapport à la cryptographie à clé asymétrique, la cryptographie à clé symétrique est très rapide à calculer. Pour cette raison, lorsque l'on prétend chiffrer quelque chose en utilisant la cryptographie à clé asymétrique, les données sont souvent chiffrées avec une cryptographie à clé symétrique, puis la clé symétrique est chiffrée à l'aide de la cryptographie à clé asymétrique. Cette «astuce» peut considérablement accélérer la cryptographie à clé asymétrique.

Voici un résumé de l'utilisation de la cryptographie à clé asymétrique dans de nombreux réseaux blockchain:

---

[9] *FIPS Publication 186-4, Digital Signature Standard specifies a common algorithm for digital signing used in blockchain technologies: Elliptic Curve Digital Signature Algorithm (ECDSA). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf*

[10] *Voir The Advanced Encryption Standard voir: Christof PAAR, Jan PELZL, Understanding cryptography*

• Les clés privées sont utilisées pour signer numériquement les transactions ;

• Les clés publiques sont utilisées pour dériver des adresses ;

• Les clés publiques permettent de vérifier les signatures générées avec des clés privées ;

• La cryptographie à clé asymétrique permet de vérifier que l'utilisateur qui transfère de la valeur à un autre utilisateur est en possession de la clé privée capable de signer la transaction ;

Certains réseaux de blockchain autorisés peuvent exploiter l'infrastructure de clé publique existante d'une entreprise pour la cryptographie à clé asymétrique afin de fournir des informations d'identification d'utilisateur au lieu de laisser chaque utilisateur du réseau de blockchain accéder à ses propres clés asymétriques. Cela se fait en utilisant les services de répertoire existants et en utilisant ces informations dans le réseau blockchain.

Certains réseaux blockchain autorisés peuvent exploiter l'infrastructure de clé publique existante d'une entreprise pour la cryptographie à clé asymétrique afin de fournir les informations d'identification de l'utilisateur - plutôt que de demander à chaque utilisateur du réseau blockchain de gérer ses propres clés asymétriques. Cela se fait en utilisant les services d'annuaire existants et en utilisant ces informations au sein du réseau blockchain. Les réseaux de chaînes de blocs qui utilisent un service d'annuaire existant peuvent y accéder via des protocoles existants, tels que le protocole LDAP (Lightweight Directory Access Protocol)[11], et utiliser les informations de l'annuaire en mode natif, ou les importer dans une autorité de certification interne au sein du réseau de chaînes de blocs

2.1.4 Adresses et dérivation d'adresses :

Certains réseaux blockchain utilisent une adresse, qui est une courte chaîne alphanumérique de caractères dérivée de la clé publique de l'utilisateur du réseau blockchain à l'aide d'une fonction de hachage cryptographique, ainsi que des données supplémentaires (numéro de version, sommes de contrôle, par exemple). La plupart des implémentations blockchain utilisent des adresses en tant que points d'extrémité "à" et "à partir de" d'une transaction. Les adresses sont plus courtes que les clés publiques et ne sont pas secrètes. Une méthode pour générer une adresse consiste à créer une clé publique en y appliquant une fonction de hachage cryptographique et en convertissant le hachage en texte:

Public key → cryptographic hash function → address

Chaque mise en œuvre de la blockchain peut mettre en œuvre un procédé différent pour dériver une adresse. Pour les réseaux blockchain sans permission, qui permettent la création anonyme de comptes, un utilisateur du réseau blockchain peut générer autant de paires de clés asymétriques, et donc d'adresses, qu'il le souhaite, permettant un degré variable de pseudo-anonymat. Les adresses peuvent servir d'identifiant public dans un réseau de chaînes de blocs pour un utilisateur. Souvent, une adresse est convertie en un QR code (code de réponse rapide,

[11] *https://ldap.com/*

code à barres bidimensionnel pouvant contenir des données arbitraires) pour une utilisation plus facile avec les téléphones mobiles.

Les utilisateurs du réseau blockchain peuvent ne pas être la seule source d'adresses dans les réseaux blockchain. Il est nécessaire de fournir une méthode permettant d'accéder à un contrat intelligent une fois qu'il a été déployé au sein d'un réseau blockchain. Pour Ethereum, les contrats intelligents sont accessibles via une adresse spéciale appelée compte de contrat. Cette adresse de compte est créée lors du déploiement d'un contrat intelligent (l'adresse d'un compte de contrat est calculée de manière déterministe à partir de l'adresse du créateur du contrat intelligent). Ce compte de contrat permet d'exécuter le contrat lorsqu'il reçoit une transaction, ainsi que de créer des contrats intelligents supplémentaires à son tour.

➢ Stockage de clé privée

Avec certains réseaux blockchain (en particulier avec les réseaux blockchain sans autorisation), les utilisateurs doivent gérer et stocker en toute sécurité leurs propres clés privées. Au lieu de les enregistrer manuellement, ils utilisent souvent un logiciel pour les stocker en toute sécurité. Ce logiciel est souvent appelé un portefeuille. Le portefeuille peut stocker des clés privées, des clés publiques et les adresses associées. Il peut également exécuter d'autres fonctions, telles que le calcul du nombre total de ressources numériques qu'un utilisateur peut posséder.

Si un utilisateur perd une clé privée, tous les actifs numériques associés à cette clé sont perdus, car il est impossible en termes de calcul de régénérer la même clé privée. Si une clé privée est volée, l'attaquant aura un accès complet à tous les actifs numériques contrôlés par cette clé privée. La sécurité des clés privées est si importante que de nombreux utilisateurs utilisent un matériel sécurisé spécial pour les stocker ; sinon, les utilisateurs peuvent tirer parti d'une industrie émergente de services de dépôt fiduciaire à clé privée.

Ces services d'en tiercement de clé peuvent également satisfaire aux lois de KYC[12] en plus de stocker des clés privées car les utilisateurs doivent fournir une preuve de leur identité lors de la création d'un compte.

Le stockage de clés privées est un aspect extrêmement important de la technologie blockchain. Quand il est rapporté dans les nouvelles que «la crypto-monnaie XYZ a été volée…», cela signifie certainement que des clés privées ont été trouvées et utilisées pour signer une transaction envoyant de l'argent à un nouveau compte, pas que le réseau de blockchain ait été compromis. Notez que, comme les données de blockchain ne peuvent généralement pas être modifiées, une fois qu'un criminel a volé une clé privée et transféré publiquement les fonds associés à un autre compte, cette transaction ne peut généralement pas être annulée.

Cependant on peut aussi stocker les clés publique avec PKCS#12 (.pfx ou.p12) et.jks* (créés par l'outil Java) sont des fichiers qui contiennent votre paire de clés publiques/privées. Contrairement aux magasins de clés des systèmes d'exploitation et des navigateurs stockés en local, ces fichiers peuvent être stockés pratiquement n'importe où, y compris sur des serveurs distants ; ils sont toujours protégés par un mot de passe (ainsi, chaque fois que vous voulez

[12] *https://www.capfi.fr/blog/it-finance/kyc-know-your-customer*

utiliser votre clé privée, vous devez saisir un mot de passe FORT). Autre avantage : comme il ne s'agit en définitive que de fichiers, vous pouvez facilement en distribuer des copies si plusieurs personnes ont besoin d'utiliser le certificat. Mais justement, comme il ne s'agit que de fichiers, ils risquent d'être distribués de manière non sécurisée. Au vu des rapides progrès des algorithmes utilisés pour craquer les mots de passe, pensez à créer un mot de passe suffisamment long et aléatoire si vous utilisez cette méthode.

2.1.5 Grands livres :

Un grand livre est un ensemble de transactions. Tout au long de l'histoire, les registres à stylos et à papier ont été utilisés pour suivre les échanges de biens et de services. À l'époque moderne, les grands livres ont été stockés sous forme numérique, souvent dans de grandes bases de données appartenant à une tierce partie de confiance centralisée (c'est-à-dire le propriétaire du grand livre) et gérées par elle pour le compte d'une communauté d'utilisateurs. Ces grands livres avec une propriété centralisée peuvent être implémentés de manière centralisée ou distribuée (c'est-à-dire un seul serveur ou un cluster de serveurs de coordination).

Il est de plus en plus intéressant d'explorer la possibilité de répartir la propriété du grand livre. La technologie Blockchain permet une telle approche utilisant à la fois une propriété distribuée et une architecture physique distribuée. L'architecture physique distribuée des réseaux blockchain implique souvent un ensemble d'ordinateurs beaucoup plus grand que celui typique d'une architecture physique distribuée gérée de manière centralisée. L'intérêt croissant pour la propriété distribuée des grands livres est dû aux problèmes de confiance, de sécurité et de fiabilité possibles liés aux grands livres avec propriété centralisée:

- Les grands livres appartenant à l'organisme central peuvent être perdus ou détruits; un utilisateur doit avoir la certitude que le propriétaire sauvegarde correctement le système ;
    - Un réseau de chaînes de blocs est distribué par conception, créant ainsi de nombreuses copies de sauvegarde, mises à jour et synchronisées avec les mêmes données de grand livre entre homologues. Un avantage clé de la technologie blockchain est que chaque utilisateur peut conserver sa propre copie du grand livre. Chaque fois que de nouveaux nœuds complets rejoignent le réseau de chaînes de blocs, ils vont découvrir d'autres nœuds complets et demandent une copie complète du journal du réseau, ce qui rend difficile la perte ou la destruction du registre. Remarque: certaines implémentations de la blockchain permettent de prendre en charge des concepts tels que les transactions privées ou les canaux privés. Les transactions privées facilitent la transmission d'informations uniquement aux nœuds participant à une transaction et non à l'ensemble du réseau ;
- Les grands livres appartenant à un centre peuvent se trouver sur un réseau homogène, où tous les logiciels, le matériel et l'infrastructure réseau peuvent être identiques. En

raison de cette caractéristique, la résilience globale du système peut être réduite car une attaque sur une partie du réseau fonctionnera partout ;

  - Un réseau blockchain est un réseau hétérogène où le logiciel, le matériel et l'infrastructure réseau sont tous différents. En raison des nombreuses différences entre les nœuds du réseau de chaînes de blocs, il n'est pas garanti qu'une attaque sur un nœud fonctionne sur d'autres nœuds ;

- Les grands livres appartenant à des centrales peuvent être situés entièrement dans des zones géographiques spécifiques (par exemple, tous dans un pays). Si des pannes de réseau se produisaient à cet emplacement, le grand livre et les services qui en dépendent pourraient ne pas être disponibles ;

  - Un réseau blockchain peut être composé de nœuds géographiquement divers pouvant être trouvés dans le monde entier. De ce fait, et le réseau de chaînes de blocs fonctionnant entre homologues, il résiste à la perte de tout nœud, voire de toute une région de nœuds ;

- Les transactions sur un grand livre appartenant à l'administration centrale ne sont pas effectuées de manière transparente et peuvent ne pas être valides; un utilisateur doit avoir la certitude que le propriétaire valide chaque transaction reçue ;
  - Un réseau de blockchain doit vérifier que toutes les transactions sont valides; si un nœud malveillant transmettait des transactions non valides, d'autres les détecteraient et les ignoreraient, empêchant ainsi les transactions non valides de se propager à travers le réseau de la blockchain ;
- La liste des transactions sur un grand livre appartenant à l'administration centrale peut ne pas être complète; un utilisateur doit avoir la certitude que le propriétaire inclut toutes les transactions valides reçues ;
  - Un réseau de blockchain contient toutes les transactions acceptées dans son grand livre distribué. Pour construire un nouveau bloc, il est nécessaire de faire référence à un bloc précédent - construisant par conséquent dessus. Si un nœud de publication n'incluait pas de référence au dernier bloc, les autres nœuds le rejetteraient ;
- Les données de transaction sur un grand livre appartenant à la centrale peuvent avoir été modifiées; un utilisateur doit avoir la certitude que le propriétaire ne modifie pas les transactions passées ;
  - Un réseau blockchain utilise des mécanismes cryptographiques tels que des signatures numériques et des fonctions de hachage cryptographique pour fournir des grands livres inviolables et résistants à la fraude ;
- Le système centralisé peut ne pas être sécurisé; un utilisateur doit avoir la certitude que les systèmes informatiques et les réseaux associés reçoivent des correctifs de sécurité critiques et a mis en œuvre les meilleures pratiques en matière de sécurité. Une violation du système et des informations personnelles peuvent avoir été volées en raison d'insécurités ;
  - Un réseau de blockchain, en raison de sa nature distribuée, ne fournit aucun point d'attaque centralisé. En règle générale, les informations sur un réseau de chaînes de blocs sont visibles publiquement et ne proposent rien à voler. Pour attaquer les utilisateurs du réseau blockchain, un

attaquant devrait les cibler individuellement. Le ciblage de la blockchain elle-même rencontrerait la résistance des nœuds honnêtes présents dans le système. Si un nœud individuel n'était pas corrigé, cela n'affecterait que ce nœud, et non le système dans son ensemble ;

2.1.6 Blocs :

Les utilisateurs du réseau de blockchain soumettent des transactions de candidats au réseau de blockchain via un logiciel (applications de bureau, applications pour smartphones, portefeuilles numériques, services Web, etc.). Le logiciel envoie ces transactions à un ou plusieurs nœuds du réseau de chaînes de blocs. Les nœuds choisis peuvent être des nœuds complets non publiés ainsi que des nœuds de publication. Les transactions soumises sont ensuite propagées aux autres nœuds du réseau, mais cela en soi ne place pas la transaction dans la chaîne de blocs. Pour de nombreuses implémentations de la blockchain, une fois qu'une transaction en attente a été distribuée aux nœuds, elle doit attendre dans une file d'attente jusqu'à ce qu'elle soit ajoutée à la blockchain par un nœud de publication.

Les transactions sont ajoutées à la blockchain lorsqu'un nœud de publication publie un bloc. Un bloc contient un en-tête de bloc et des données de bloc. L'en-tête de bloc contient des métadonnées pour ce bloc. Les données de bloc contiennent une liste de transactions validées et authentiques qui ont été soumises au réseau de chaînes de blocs. La validité et l'authenticité sont assurées en vérifiant que la transaction est correctement formatée et que les fournisseurs d'actifs numériques dans chaque transaction (répertoriés dans les valeurs de «saisie» de la transaction) ont chacun signé la transaction de manière cryptographique. Cela permet de vérifier que les fournisseurs d'actifs numériques pour une transaction avaient accès à la clé privée qui pouvait signer les actifs numériques disponibles. Les autres nœuds complets vérifieront la validité et l'authenticité de toutes les transactions d'un bloc publié et n'accepteront pas un bloc s'il contient des transactions non valides.

Il convient de noter que chaque implémentation blockchain peut définir ses propres champs de données. Cependant, de nombreuses implémentations blockchain utilisent des champs de données tels que:

- En-tête de bloc
    - Le numéro de bloc, également appelé hauteur de bloc dans certains réseaux de chaînes de blocs ;
    - La valeur de hachage de l'en-tête du bloc précédent ;
    - Une représentation en hachage des données de bloc (différentes méthodes peuvent être utilisées pour y parvenir, telles que la génération d'un arbre de Merkle[13] (Figure 10) et le stockage du hachage racine ou en utilisant un hachage de toutes les données de bloc combinées) ;
    - Un horodatage ;

[13] *Une structure de données dans laquelle les données sont hachées et combinées jusqu'à ce qu'il y ait un hachage racine unique qui représente la structure entière.*

- La taille du bloc ;
- La valeur de nonce. Pour les réseaux blockchain utilisant l'extraction, il s'agit d'un numéro manipulé par le nœud de publication pour résoudre le casse-tête (voir la section 2.1 pour plus de détails). D'autres réseaux de chaînes de blocs peuvent ou non l'inclure ou l'utiliser à d'autres fins que la résolution d'un casse-tête ;

- Données de bloc
  - Une liste des transactions et des événements du grand livre inclus dans le bloc ;
  - D'autres données peuvent être présentes ;

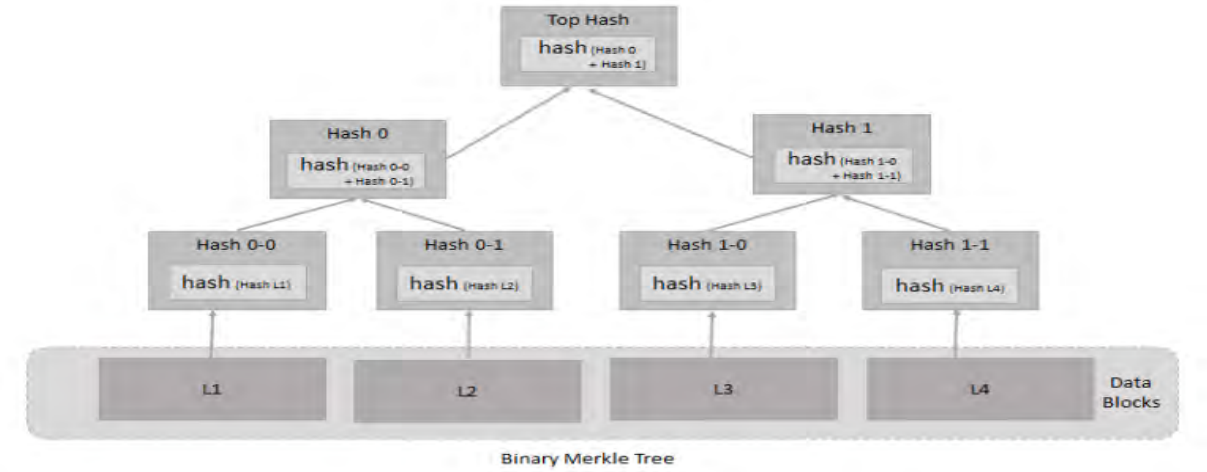


Figure 3 Arbre de Merkle binaire

2.1.7 Chaînage de blocs :

Les blocs sont enchaînés à travers chaque bloc contenant le condensé de hachage de l'en-tête du bloc précédent, formant ainsi la blockchain. Si un bloc précédemment publié était modifié, le hachage serait différent. Cela entraînerait à son tour un hachage différent dans tous les blocs suivants, car il comprend celui du bloc précédent. Cela permet de détecter et de rejeter facilement les blocs altérés. La figure 3 montre une chaîne générique de blocs

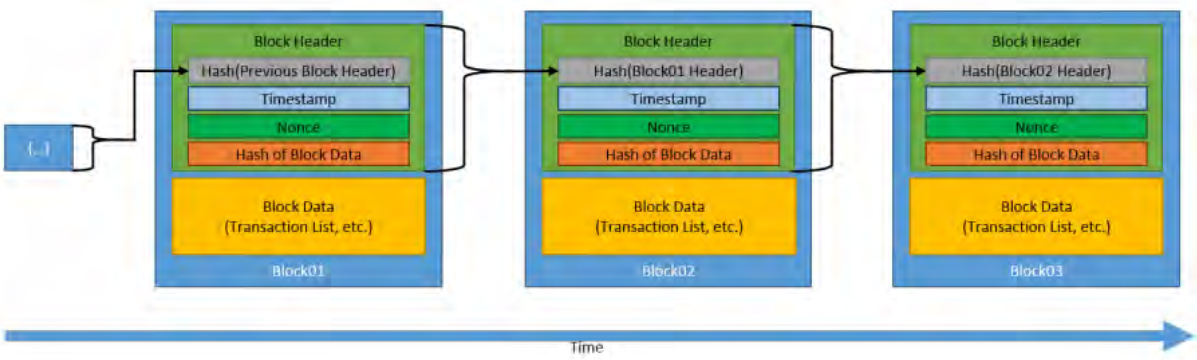


Figure 4 Chaîne générique de blocs

2.2 Consensus

Un aspect clé de la technologie de blockchain consiste à déterminer quel utilisateur publie le prochain bloc. Ce problème est résolu par la mise en œuvre de l'un des nombreux modèles de consensus possibles. Pour les réseaux de blockchain sans permission, il existe généralement de nombreux nœuds de publication en concurrence simultanée pour publier le bloc suivant. Ils le font généralement pour gagner des frais de crypto-monnaie et / ou de transaction. Ce sont généralement des utilisateurs qui se méfient mutuellement et qui ne peuvent se connaître que par leurs adresses publiques. Chaque nœud de publication est probablement motivé par un désir de gain financier et non par le bien-être des autres nœuds de publication ou même du réseau lui-même.
Dans une telle situation, pourquoi un utilisateur propagerait-il un bloc qu'un autre utilisateur tente de publier? En outre, qui résout les conflits lorsque plusieurs nœuds publient un bloc à peu près au même moment? Pour que cela fonctionne, les technologies de blockchain utilisent des modèles consensuels pour permettre à un groupe d'utilisateurs se méfiant mutuellement de travailler ensemble.
Lorsqu'un utilisateur rejoint un réseau de blockchain, il accepte l'état initial du système. Ceci est enregistré dans le seul bloc préconfiguré, le bloc de genèse. Chaque réseau de blockchain à un bloc de genèse publié et chaque bloc doit être ajouté à la blockchain après celui-ci, sur la base du modèle de consensus convenu. Cependant, quel que soit le modèle, chaque bloc doit être valide et peut donc être validé indépendamment par chaque utilisateur du réseau de la blockchain. En combinant l'état initial et la possibilité de vérifier chaque bloc depuis lors, les utilisateurs peuvent s'accorder indépendamment sur l'état actuel de la blockchain. Notez que si deux chaînes valides ont déjà été présentées à un nœud complet, le mécanisme par défaut dans la plupart des réseaux de blockchain est que la chaîne "la plus longue" est considérée comme la chaîne correcte et sera adoptée. C'est parce qu'il a eu le plus de travail mis dedans. Cela se produit fréquemment avec certains modèles consensuels et sera discuté en détail.

Les propriétés suivantes sont alors en place:

- L'état initial du système est convenu (par exemple, le bloc de genèse) ;
- Les utilisateurs acceptent le modèle de consensus selon lequel des blocs sont ajoutés au système ;
- Chaque bloc est lié au bloc précédent en incluant le condensé de hachage de l'en-tête du bloc précédent (à l'exception du premier bloc «genèse», qui n'a pas de bloc précédent et pour lequel le hachage de l'en-tête du bloc précédent est généralement mis à zéro) ;
- Les utilisateurs peuvent vérifier chaque bloc indépendamment ;

En pratique, le logiciel gère tout et les utilisateurs n'ont pas besoin de connaître ces détails.
Une caractéristique clé de la technologie blockchain est qu'il n'est pas nécessaire de faire appel à un tiers de confiance pour fournir l'état du système: chaque utilisateur du système peut vérifier l'intégrité du système. Pour ajouter un nouveau bloc à la blockchain, tous les nœuds doivent parvenir à un accord commun dans le temps. Toutefois, un certain désaccord temporaire est autorisé. Pour les réseaux de blockchain sans permission, le modèle de consensus doit fonctionner même en présence éventuellement d'utilisateurs malveillants, ces derniers pouvant tenter de perturber ou de prendre en charge la chaîne de blocs.

Notez que pour les réseaux de blockchain autorisés, des recours légaux peuvent être utilisés si un utilisateur agit par malveillance.
Dans certains réseaux blockchain, tels que ceux autorisés, il peut exister un certain niveau de confiance entre les nœuds de publication. Dans ce cas, il n'est peut-être pas nécessaire de disposer d'un modèle de consensus utilisant beaucoup de ressources (temps de calcul, investissement, etc.) pour déterminer quel participant ajoute le bloc suivant à la chaîne. En règle générale, à mesure que le niveau de confiance augmente, la nécessité d'utiliser des ressources pour mesurer la confiance créée diminue. Pour certaines implémentations blockchain autorisées, la vision du consensus va au-delà de la garantie de la validité et de l'authenticité des blocs, mais englobe tous les systèmes de contrôles et de validations allant de la proposition d'une transaction à son inclusion finale dans un bloc.

Dans les sections suivantes, plusieurs modèles de consensus ainsi que l'approche de résolution de conflit la plus courante sont abordés.

2.2.1 Modèle de consensus sur la preuve de travail :

Dans le modèle de preuve de travail, un utilisateur publie le bloc suivant en étant le premier à résoudre un puzzle informatique intensif. La solution à ce puzzle est la "preuve" qu'ils ont effectué un travail. Le puzzle est conçu de telle sorte que sa résolution est difficile, mais il est facile de vérifier qu'une solution est valide. Cela permet à tous les autres nœuds complets de valider facilement les blocs suivants proposés, et tout bloc proposé ne satisfaisant pas le casse-tête serait rejeté.

Une méthode de puzzle courante consiste à exiger que le condensé de hachage d'un en-tête de bloc soit inférieur à une valeur cible. Les nœuds de publication apportent de nombreuses modifications mineures à leur en-tête de bloc (modification du nonce, par exemple) en essayant de trouver un condensé de hachage répondant à l'exigence. Pour chaque tentative, le nœud de publication doit calculer le hachage pour l'en-tête de bloc complet. Le hachage de l'en-tête de bloc à plusieurs reprises devient un processus de calcul intensif. La valeur cible peut être modifiée au fil du temps pour ajuster la difficulté (à la hausse ou à la baisse) afin d'influer sur la fréquence de publication des blocs.

Par exemple, Bitcoin, qui utilise le modèle de preuve de travail, ajuste la difficulté du puzzle tous les blocs 2016 pour que le taux de publication des blocs soit d'environ une fois toutes les dix minutes. L'ajustement est effectué en fonction du niveau de difficulté du puzzle et augmente ou diminue le nombre de zéros au début. En augmentant le nombre de zéros non significatifs, la difficulté du puzzle augmente, car toute solution doit être inférieure au niveau de difficulté, ce qui signifie que les solutions possibles sont moins nombreuses. En diminuant le nombre de zéros non significatifs, le niveau de difficulté diminue, car il existe plus de solutions possibles. Cet ajustement a pour but de maintenir la difficulté de calcul du puzzle et, par conséquent, de préserver le mécanisme de sécurité central du réseau Bitcoin. La puissance de calcul disponible augmente avec le temps, tout comme le nombre de nœuds de publication. La difficulté du puzzle augmente donc généralement.

Les ajustements apportés à l'objectif de difficulté visent à garantir qu'aucune entité ne puisse prendre en charge la production de blocs, mais les calculs de résolution de casse-tête nécessitent par conséquent une consommation de ressources importante.

En raison de la consommation importante de ressources de certains réseaux de blockchain de preuve de travail, on souhaite ajouter des nœuds de publication aux régions où il existe un excédent d'approvisionnement en électricité bon marché.

Un aspect important de ce modèle est que le travail inséré dans une énigme n'influence pas les chances de résolution des énigmes actuelles ou futures, car les énigmes sont indépendantes. Cela signifie que lorsqu'un utilisateur reçoit un bloc terminé et valide d'un autre utilisateur, il est incité à abandonner son travail actuel et à commencer à créer le bloc nouvellement reçu, car il sait que les autres nœuds de publication le construiront.

A titre d'exemple, considérons un puzzle dans lequel, à l'aide de l'algorithme SHA-256, un ordinateur doit trouver une valeur de hachage répondant aux critères cible suivants (appelé niveau de difficulté):

SHA256 ("blockchain" + Nonce) = Hash Digest starting with "000000"

Dans cet exemple, la chaîne de texte «blockchain» est ajoutée à une valeur nonce, puis le résumé de hachage est calculé. Les valeurs nonce utilisées seront uniquement des valeurs numériques. Il s'agit d'un puzzle relativement facile à résoudre. Voici un exemple de résultat:

SHA256 ("blockchain0") =
0xbd4824d8ee63fc82392a6441444166d22ed84eaa6dab11d4923075975acab938 (not solved)

SHA256 ("blockchain1") =
0xdb0b9c1cb5e9c680dfff7482f1a8efad0e786f41b6b89a758fb26d9e223e0a10 (not solved)

...

SHA256 ("blockchain10730895") =
0x000000ca1415e0bec568f6f605fcc83d18cac7a4e6c219a957c10c6879d67587 (solved)

Pour résoudre ce puzzle, il a fallu 10 730 896 hypothèses (achevées en 54 secondes sur du matériel relativement ancien, en commençant à 0 et en testant une valeur à la fois).

Dans cet exemple, chaque valeur supplémentaire du «zéro principal» augmente la difficulté. En augmentant la cible d'un zéro supplémentaire ("0000000"), le même matériel a pris 934 224 175 propositions pour résoudre le puzzle (terminé en 1 heure, 18 minutes, 12 secondes):

SHA256 ("blockchain934224174") =
0x0000000e2ae7e4240df80692b7e586ea7a977eacbd031819d0e603257edb3a81

Il n'existe actuellement aucun raccourci connu vers ce processus. Les nœuds de publication doivent déployer des efforts de calcul, du temps et des ressources pour trouver la valeur de nonce correcte pour la cible. Souvent, les nœuds de publication tentent de résoudre ce puzzle informatique complexe en réclamant une récompense (généralement sous la forme d'une crypto-monnaie offerte par le réseau blockchain). La perspective d'être récompensé pour l'extension et la maintenance de la blockchain est appelée système de récompense ou modèle d'incitation.

2.2.2 Modèle de consensus de preuve de participation :

Le modèle de preuve d'enjeu repose sur l'idée que plus un utilisateur investit dans le système, plus il a de chances de vouloir que le système réussisse et moins il est probable qu'il voudra le renverser. L'enjeu est souvent une quantité de crypto-monnaie que l'utilisateur du réseau blockchain a investi dans le système (par divers moyens, par exemple en le verrouillant via un type de transaction spécial, en l'envoyant à une adresse spécifique ou en le conservant dans un logiciel de portefeuille spécial). Une fois implanté, la crypto-monnaie ne peut généralement plus être dépensée. La preuve d'enjeu des réseaux de blockchain utilise la quantité de participation d'un utilisateur comme facteur déterminant pour la publication de nouveaux blocs. Ainsi, la probabilité qu'un utilisateur du réseau de la blockchain publie un nouveau bloc est liée au rapport de sa mise sur la quantité totale de crypto-monnaie dépendante du réseau de la blockchain.
Avec ce modèle consensuel, il n'est pas nécessaire d'effectuer des calculs nécessitant beaucoup de ressources (temps, électricité et puissance de traitement) comme dans la preuve de travail. Comme ce modèle consensuel utilise moins de ressources, certains réseaux de blockchain ont décidé de renoncer à une récompense pour la création de blocs. Ces systèmes sont conçus de manière à ce que toute la crypto-monnaie soit déjà distribuée parmi les utilisateurs plutôt qu'une nouvelle crypto-monnaie générée à un rythme constant. Dans de tels systèmes, la publication en bloc est généralement récompensée par le gain des frais de transaction supportés par l'utilisateur.

Les méthodes utilisées par le réseau blockchain peuvent varier. Nous abordons ici quatre approches: la sélection aléatoire d'utilisateurs impliqués, le vote à plusieurs tours, les systèmes de vieillissement des pièces et les systèmes de délégation. Quelle que soit l'approche choisie, les utilisateurs ayant plus de participation sont plus susceptibles de publier de nouveaux blocs.

Lorsque le choix de l'éditeur de blocs est un choix aléatoire (parfois appelé preuve de participation basée sur une chaîne), le réseau de chaînes de blocs examine tous les utilisateurs concernés et choisit parmi eux en fonction du rapport entre leur participation et la quantité totale de crypto-monnaie mise. Ainsi, si un utilisateur détient 42% de la totalité de la participation au réseau blockchain, il sera choisi 42% du temps; ceux avec 1% seraient choisis 1% du temps.

Lorsque le choix de l'éditeur de blocs est un système de vote à plusieurs tours (parfois appelé preuve de tolérance de panne byzantine[14]), la complexité est accrue. Le réseau de blockchain sélectionnera plusieurs utilisateurs impliqués pour créer les blocs proposés. Ensuite, tous les utilisateurs mis en jeu voteront pour un bloc proposé. Plusieurs tours de vote peuvent avoir lieu avant qu'un nouveau bloc ne soit décidé. Cette méthode permet à tous les utilisateurs impliqués d'avoir une voix dans le processus de sélection de bloc pour chaque nouveau bloc.
Lorsque le choix de l'éditeur de bloc se fait par l'intermédiaire d'un système de jeu de pièces appelé une preuve de jeu de pièces de monnaie, la crypto-monnaie jouée a une propriété de temps. Au bout d'un certain temps (30 jours, par exemple), la crypto-monnaie impliquée peut être prise en compte dans la sélection de l'utilisateur propriétaire pour la publication du bloc suivant. La crypto-monnaie implicite a alors son âge réinitialisé et elle ne peut plus être utilisée tant que le temps requis n'est pas écoulé.

[14] *https://ieeexplore.ieee.org/document/7161576*

Cette méthode permet aux utilisateurs ayant plus d'enjeu de publier plus de blocs, sans pour autant dominer le système, puisqu'ils ont un temps de recharge attaché à chaque pièce de monnaie crypto-comptée comptabilisée dans la création de blocs. Des pièces plus anciennes et des groupes de pièces plus importants augmenteront la probabilité d'être choisi pour publier le bloc suivant. Pour empêcher les parties prenantes d'accumuler des crypto-monnaies anciennes, il existe généralement un maximum intégré à la probabilité de gagner.

Lorsque le choix de l'éditeur de blocs se fait par l'intermédiaire d'un système délégué, les utilisateurs votent pour que les nœuds deviennent des nœuds de publication, créant ainsi des blocs pour leur compte. Le pouvoir de vote des utilisateurs du réseau Blockchain étant lié à leur enjeu, plus l'enjeu est important, plus le vote a de poids. Les nœuds qui reçoivent le plus de votes deviennent des nœuds de publication et peuvent valider et publier des blocs. Les utilisateurs du réseau Blockchain peuvent également voter contre un nœud de publication établi pour tenter de les supprimer de l'ensemble des nœuds de publication. Le vote pour les nœuds de publication est continu et rester un nœud de publication peut être très compétitif. La menace de perdre le statut du nœud de publication, et par conséquent les récompenses et la réputation étant constantes, incite les nœuds de publication à ne pas agir de manière malveillante. De plus, les utilisateurs du réseau blockchain votent pour les délégués qui participent à la gouvernance de la blockchain. Les délégués proposeront des modifications et des améliorations qui seront mises aux voix par les utilisateurs du réseau blockchain.

Il convient de noter qu'un problème connu sous le nom de «rien en jeu» peut découler de la preuve de certains algorithmes d'enjeu. Si plusieurs chaînes de blocs concurrentes devaient exister à un moment donné (en raison d'un conflit temporaire dans le grand livre, comme indiqué à la section 2.6), un utilisateur mis en jeu pourrait agir sur chacune de ces chaînes en concurrence, car il est essentiellement libre de le faire. L'utilisateur mis en jeu peut le faire pour augmenter ses chances de gagner une récompense. Cela peut faire en sorte que plusieurs branches de la chaîne de blocs continuent de croître sans être réconciliées en une branche unique pendant de longues périodes.

Avec des systèmes de preuve d'enjeu, les «riches» peuvent plus facilement engranger davantage d'actifs numériques, en gagnant plus d'actifs numériques; Cependant, obtenir la majorité des actifs numériques dans un système pour le «contrôler» est généralement d'un coût prohibitif.

2.2.3 Modèle de consensus Round Robin :

Round Robin est un modèle de consensus utilisé par certains réseaux de chaînes de blocs autorisés. Dans ce modèle de consensus, les nœuds créent à tour de rôle des blocs. Round Robin Consensus à une longue histoire fondée sur l'architecture de système distribué. Pour gérer les situations dans lesquelles un nœud de publication n'est pas disponible pour publier un bloc à son tour, ces systèmes peuvent inclure une limite de temps pour permettre aux nœuds disponibles de publier des blocs afin que les nœuds non disponibles ne provoquent pas un arrêt de la publication du bloc. Ce modèle garantit qu'aucun nœud ne crée la majorité des blocs. Il bénéficie d'une approche simple, manque de puzzles cryptographiques et a une faible consommation d'énergie.

Dans la mesure où la confiance entre les nœuds est nécessaire, round robin ne fonctionne pas bien dans les réseaux blockchain sans autorisation utilisés par la plupart des crypto-monnaies. En effet, les nœuds malveillants peuvent ajouter en permanence des nœuds supplémentaires pour augmenter leurs chances de publier de nouveaux blocs. Dans le pire des cas, ils pourraient l'utiliser pour compromettre le bon fonctionnement du réseau de chaînes de blocs.

2.2.4 Modèle consensuel de preuve d'autorité / de preuve d'identité :

Le modèle de consensus preuve d'autorité (également appelé preuve d'identité) repose sur la confiance partielle des nœuds de publication via leur lien connu aux identités du monde réel. Les identités des nœuds de publication doivent avoir une identité prouvée et vérifiable au sein du réseau de chaînes de blocs (par exemple, identifier les documents vérifiés, notariés et inclus dans la chaîne de blocs). L'idée est que le nœud de publication mise sur son identité / réputation pour publier de nouveaux blocs. Les utilisateurs du réseau blockchain affectent directement la réputation du nœud de publication en fonction du comportement du nœud de publication. Les nœuds de publication peuvent perdre leur réputation en agissant de manière à être en désaccord avec les utilisateurs du réseau de chaînes de blocs, tout comme ils peuvent acquérir une réputation en agissant de manière à ce que les utilisateurs du réseau de chaînes de blocs soient d'accord. Plus la réputation est mauvaise, moins il est possible de publier un bloc. Par conséquent, il est dans l'intérêt d'un nœud de publication de conserver une réputation élevée. Cet algorithme ne s'applique qu'aux réseaux blockchain autorisés avec des niveaux de confiance élevés.

2.2.5 Modèle de consensus sur la preuve du temps écoulé :

Dans le modèle de consensus Preuve de temps écoulé, chaque nœud de publication demande un temps d'attente à une source de temps matérielle sécurisée au sein de son système informatique. La source de temps matérielle sécurisée générera un temps d'attente aléatoire et le renverra au logiciel du nœud de publication. Les nœuds de publication prennent le temps aléatoire qui leur est attribué et deviennent inactifs pendant cette durée. Une fois qu'un nœud de publication sort de l'état inactif, il crée et publie un bloc sur le réseau de chaînes de blocs, en alertant les autres nœuds du nouveau bloc. Tout nœud de publication encore inactif cessera d'attendre et le processus entier recommencera.

Ce modèle nécessite de s'assurer qu'un temps aléatoire a été utilisé, car si le temps d'attente n'était pas sélectionné au hasard, un nœud de publication malveillant attendrait simplement la durée minimale par défaut pour dominer le système. Ce modèle nécessite également de s'assurer que le nœud de publication a attendu l'heure actuelle et qu'il n'a pas démarré tôt. Ces exigences sont résolues en exécutant les logiciels dans un environnement d'exécution approuvé, présent sur certains processeurs (tels que Software Guard Extensions d'Intel[15], Processeur de sécurité de plate-forme AMD[16] ou TrustZone d'ARM[17]).

Les logiciels vérifiés et approuvés peuvent fonctionner dans ces environnements d'exécution sécurisés et ne peuvent pas être modifiés par des programmes extérieurs. Un nœud de publication interroge les logiciels s'exécutant dans cet environnement sécurisé pendant une durée aléatoire, puis attend la fin de cette période. Après avoir attendu l'heure attribuée, le nœud de publication peut demander un certificat signé attestant que le nœud de publication a attendu l'heure assignée de manière aléatoire. Le nœud de publication publie ensuite le certificat avec le bloc.

2.2.6 Conflits et résolutions du grand livre :

Comme indiqué précédemment, pour certains réseaux de blockchain, il est possible que plusieurs blocs soient publiés à peu près au même moment. Cela peut entraîner l'existence de versions différentes d'une blockchain à un moment donné; ceux-ci doivent être résolus rapidement pour avoir une cohérence dans le réseau blockchain. Dans cette section, nous discutons de la façon dont ces situations sont généralement gérées.

Avec n'importe quel réseau distribué, certains systèmes du réseau seront en retard sur les informations ou auront des informations alternatives. Cela dépend de la latence du réseau entre les nœuds et de la proximité des groupes de nœuds. Les réseaux de blockchain sans autorisation sont plus susceptibles d'avoir des conflits en raison de leur ouverture et du nombre de nœuds d'édition concurrents. La résolution des conflits de données est un élément essentiel pour s'accorder sur l'état du réseau de chaînes de blockchain (pour en arriver à un consensus).

---

[15] *Intel SGX - https://software.intel.com/en-us/sgx*
[16] *AMD Secure Technology - https://www.amd.com/en/technologies/security*
[17] *ARM TrustZone - https://www.arm.com/products/silicon-ip-security*

Par exemple :

- Nœud_A crée block_n (A) avec les transactions n ° 1, 2 et 3. Nœud_A le distribue à certains nœuds.
- Nœud_B crée block_n (B) avec les transactions n ° 1, 2 et 4. Nœud_B le distribue à certains nœuds.
- Il y a un conflit.
    - block_n ne sera pas identique sur le réseau.
        - block_n (A) contient la transaction n ° 3, mais pas la transaction n ° 4.
        - block_n (B) contient la transaction n ° 4, mais pas la transaction n ° 3.

Les conflits génèrent temporairement différentes versions de la blockchain, comme illustré à la figure 12. Ces différentes versions ne sont pas «fausses»; ils ont plutôt été créés avec les informations disponibles par chaque nœud. Les blocs en concurrence contiendront probablement différentes transactions. Par conséquent, ceux qui ont block_n (A) peuvent voir des transferts d'actifs numériques qui ne sont pas présents dans block_n (B). Si le réseau de blockchain traite de la crypto-monnaie, il peut arriver qu'une partie de la crypto-monnaie soit dépensée ou non, en fonction de la version de la chaîne de blocs visualisée.

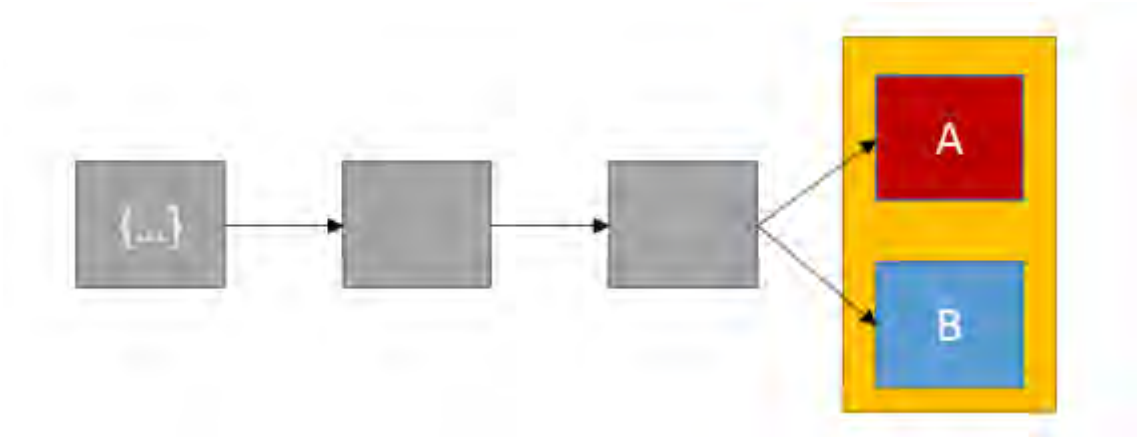


Figure 5 Grands livres en conflit

Les conflits sont généralement résolus rapidement. La plupart des réseaux de chaînes de blocs attendent la publication du bloc suivant et utilisent cette chaîne comme «chaîne de blocs officielle», adoptant ainsi la «chaîne de blocs plus longue». Comme dans la figure 13, la chaîne de blocs contenant block_n (B) devient la chaîne «officielle», car elle a obtenu le prochain bloc valide. Toute transaction présente dans block_n (A), le bloc orphelin, mais non présent dans la chaîne block_n (B), est renvoyée au pool de transactions en attente (c'est-à-dire où résident toutes les transactions non incluses dans un bloc). Notez que cet ensemble de transactions en attente est géré localement sur chaque nœud, car il n'existe aucun serveur central dans l'architecture.

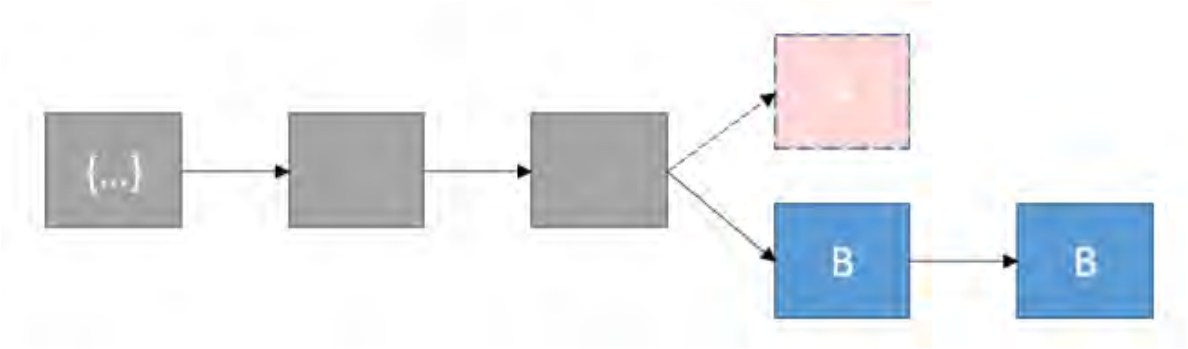


Figure 6 La chaîne avec block_n (B) ajoute le bloc suivant, la chaîne avec block_n (A) est maintenant orpheline.

En raison de la possibilité que des blocs soient écrasés, une transaction n'est généralement pas acceptée comme confirmée tant que plusieurs blocs supplémentaires n'ont pas été créés par-dessus le bloc contenant la transaction correspondante. L'acceptation d'un bloc est souvent plus probabiliste que déterministe, car les blocs peuvent être remplacés. Plus le nombre de blocs créés sur un bloc publié est élevé, plus il est probable que le bloc initial ne sera pas écrasé.

Hypothétiquement, un nœud dans un réseau de blockchain de preuve travail avec des quantités énormes de puissance de calcul pourrait commencer au bloc de genèse et créer une chaîne plus longue que la chaîne existante, effaçant ainsi toute l'histoire du blockchain. Cela ne se produit pas dans la pratique en raison de la quantité prohibitive de ressources que cela exigerait. De plus, certaines implémentations de blockchain bloquent des blocs plus anciens spécifiques dans le logiciel blockchain en créant des points de contrôle pour s'assurer que cela ne puisse jamais se produire.

2.3 Forking

Effectuer des changements et mettre à jour la technologie peut être difficile dans le meilleur des cas. Cela devient extrêmement difficile pour les réseaux de blockchain sans permission, composés de nombreux utilisateurs, répartis dans le monde entier et régis par le consensus des utilisateurs. Les modifications apportées au protocole et aux structures de données d'un réseau de blockchain sont appelées forks. Ils peuvent être divisés en deux catégories: soft forks et hard forks. Pour un soft forks, ces modifications sont rétro compatibles avec les nœuds qui n'ont pas été mis à jour. Pour un hard forks, ces modifications ne sont pas compatibles avec les versions antérieures, car les nœuds qui n'ont pas été mis à jour rejetteront les blocs après les modifications. Cela peut conduire à une scission du réseau de chaînes de blocs créant plusieurs versions de la même chaîne de blocs. Les réseaux de blockchain autorisés, en raison de la connaissance des nœuds de publication et des utilisateurs, peuvent atténuer les problèmes de forking en exigeant des mises à jour logicielles.

Notez que le terme fork est également utilisé par certains réseaux de chaînes de blocs pour décrire les conflits temporaires dans le grand livre (par exemple, deux blocs ou plus dans le réseau de chaînes de blocs avec le même numéro de bloc), comme décrit dans la section 2.7. Bien que ce soit un fork dans le grand livre, il est temporaire et ne découle pas d'un changement de logiciel.

2.3.1 Soft Forks :

Un soft fork est une modification apportée à une implémentation blockchain compatible avec les versions antérieures. Les nœuds non mis à jour peuvent continuer à effectuer des transactions avec des nœuds mis à jour. Si aucun (ou très peu) nœud n'est mis à niveau, les règles mises à jour ne seront pas suivies.

Un exemple fictif de "soft fork" serait si une blockchain décidait de réduire la taille des blocs (par exemple de 1,0 Mo à 0,5 Mo). Les nœuds mis à jour ajusteraient la taille du bloc et continueraient à effectuer leurs transactions normalement; Les nœuds non mis à jour verront ces blocs comme valides, car la modification apportée n'enfreint pas leurs règles (c'est-à-dire que la taille du bloc est inférieure à leur maximum autorisé). Toutefois, si un nœud non mis à jour créait un bloc d'une taille supérieure à 0,5 Mo, les nœuds mis à jour les rejetteraient comme non valides.

2.3.2 Hard Forks :

Un hard fork est une modification d'une implémentation blockchain qui n'est pas compatible avec les versions antérieures. À un moment donné (généralement à un numéro de bloc spécifique), tous les nœuds de publication devront passer au protocole mis à jour. De plus, tous les nœuds devront passer au nouveau protocole afin de ne pas rejeter les blocs récemment formatés. Les nœuds non mis à jour ne peuvent pas continuer à effectuer des transactions sur la blockchain mise à jour car ils sont programmés pour rejeter tout bloc qui ne suit pas leur version de la spécification de bloc.

Les nœuds de publication qui ne se mettent pas à jour continueront à publier des blocs en utilisant l'ancien format. Les nœuds utilisateur qui ne se sont pas mis à jour rejetteront les blocs nouvellement formatés et n'accepteront que les blocs avec l'ancien format. Cela se traduit par deux versions de la blockchain existantes simultanément. Notez que les utilisateurs de différentes versions de hard fork ne peuvent pas interagir les uns avec les autres. Il est important de noter que bien que la plupart des hard forks soient intentionnelles, les erreurs de logiciel peuvent produire des hard forks non intentionnelles.

Ethereum est un exemple bien connu du hard fork. En 2016, un contrat intelligent a été construit sur Ethereum, appelé Decentralized Autonomous Organization (DAO). En raison de failles dans la manière dont le contrat intelligent a été construit, un attaquant a extrait Ether, la crypto-monnaie utilisée par Ethereum, entraînant un vol de 50 millions de dollars. Les détenteurs d'Ether ont voté pour une proposition d'un hard fork, et la grande majorité des utilisateurs a accepté de créer une nouvelle version de la blockchain, sans la faille, et qui a également restitué les fonds volés.

Avec les crypto-monnaies, s'il existe un hard fork et que la blockchain se divise, les utilisateurs disposeront d'une monnaie indépendante sur les deux fourchettes (le double du nombre de pièces au total). Si toute l'activité passe à la nouvelle chaîne, l'ancienne peut ne plus être utilisée car les deux chaînes ne sont pas compatibles (il s'agira de systèmes monétaires indépendants). Dans le cas du hard fork Ethereum, la grande majorité du support est passée à la nouvelle fourche. L'ancienne fourche a été renommée Ethereum Classic et a continué à fonctionner.

2.3.3 Changements cryptographiques et forks :

Si des failles sont trouvées dans les technologies de cryptographie à l'intérieur d'un réseau à blockchain, la seule solution peut être de créer un hard fork, en fonction de l'importance de la faille. Par exemple, si un défaut était trouvé dans les algorithmes sous-jacents, il pourrait y avoir un fork exigeant que tous les futurs clients utilisent un algorithme plus fort. Le passage à un nouvel algorithme de hachage pourrait poser un problème pratique important, car il pourrait invalider tout le matériel minier spécialisé existant.

Hypothétiquement, si SHA-256 étaient découverts pour avoir un défaut, les réseaux de blockchain qui utilisent SHA256 auraient besoin d'un hard fork pour migrer à un nouvel algorithme de hachage. Le bloc qui est passé au nouvel algorithme de hachage "verrouillerait" tous les blocs précédents dans SHA-256 (pour vérification), et tous les nouveaux blocs devraient utiliser le nouvel algorithme de hachage.

Il existe de nombreux algorithmes de hachage cryptographique, et les réseaux de blockchain peuvent utiliser celui qui convient à leurs besoins. Par exemple, alors que Bitcoin utilise SHA-256, Ethereum utilise Keccak-256[18].
Une possibilité de modifier les caractéristiques cryptographiques présentes dans un réseau blockchain serait la mise au point d'un système informatique quantique pratique, capable d'affaiblir considérablement (et, dans certains cas, de rendre inutile) les algorithmes cryptographiques existants. Le rapport interne NIST (NISTIR) 8105, Rapport sur la cryptographie post-quantique[19], fournit un tableau décrivant l'impact de l'informatique quantique sur des algorithmes de chiffrement courants. Le tableau 2 reproduit ce tableau.

Algorithme cryptographique	**Type**	**Objectif**	**Impact de l'ordinateur quantique à grande échelle**
AES	Symetric Key	Encryption	Grandeurs de clé nécessaires
SHA-2, SHA3	N/A	Hash Functions	Plus grande sortie nécessaire
RSA	Public Key	Signatures, Key establishment	N'est plus sécurisé
ECDSA, ECDH (Elliptic Curve Cryptography)	Public Key	Signatures, Key Exchange	N'est plus sécurisé
DSA (Finite Field Cryptography)	Public Key	Signatures, Key Exchange	N'est plus sécurisé

Tableau 2 Impact de l'informatique quantique sur les algorithmes cryptographiques courants

Les algorithmes cryptographiques utilisés dans la plupart des technologies blockchain pour les paires de clés asymétriques devront être remplacés si un ordinateur quantique puissant devient une réalité. En effet, les algorithmes reposant sur la complexité de calcul de la factorisation de nombres entiers (telle que RSA) ou travaillant à la résolution de logarithmes discrets (tels que DSA et Diffie-Hellman) sont très susceptibles d'être détruits par l'informatique quantique. Les algorithmes de hachage utilisés par les réseaux de chaînes de blocs sont beaucoup moins sensibles aux attaques informatiques quantiques mais sont encore affaiblis.

[18] *https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf*
[19] *https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf*

2.4 Catégorisation de la Blockchain

Les réseaux de blockchain peuvent être classés en fonction de leur modèle d'autorisation, qui détermine qui peut les gérer (par exemple, des blocs de publication). Si quelqu'un peut publier un nouveau bloc, il est sans permission. Si seuls des utilisateurs particuliers peuvent publier des blocs, l'autorisation est accordée. En termes simples, un réseau blockchain autorisé est comme un intranet d'entreprise contrôlé, alors qu'un réseau blockchain sans autorisation s'apparente à l'internet public, où tout le monde peut participer. Les réseaux de chaînes de blocs autorisés sont souvent déployés pour un groupe d'organisations et d'individus, généralement appelé consortium. Cette distinction est nécessaire pour comprendre car elle affecte certains des composants de la chaîne de blocs décrits plus loin dans ce document.

2.4.1 Permission :

Les réseaux de blockchain autorisés sont ceux où les utilisateurs qui publient des blocs doivent être autorisés par une autorité (centralisée ou décentralisée). Etant donné que seuls les utilisateurs autorisés gèrent la blockchain, il est possible de restreindre l'accès en lecture et de restreindre l'accès aux transactions. Les réseaux de blockchain autorisés peuvent ainsi permettre à quiconque de lire ces chaînes ou restreindre l'accès en lecture à des personnes autorisées. Ils peuvent également permettre à quiconque de soumettre des transactions à inclure dans la blockchain ou, encore une fois, restreindre cet accès uniquement aux personnes autorisées. Les réseaux de blockchain autorisés peuvent être instanciés et maintenus à l'aide de logiciels open source ou fermés.

Les réseaux de blockchain autorisés peuvent avoir la même traçabilité des actifs numériques lorsqu'ils passent dans la blockchain, ainsi que le même système de stockage de données distribué, résilient et redondant en tant que réseaux de blockchain sans autorisation. Ils utilisent également des modèles consensuels pour la publication de blocs, mais ces méthodes ne nécessitent souvent pas de dépense en ressources ni en maintenance (comme c'est le cas avec les réseaux de blockchain sans autorisation actuels). C'est parce que l'établissement de l'identité est nécessaire pour participer en tant que membre du réseau de chaînes de blocs autorisé; ceux qui gèrent la blockchain ont un niveau de confiance réciproque, puisqu'ils étaient tous autorisés à publier des blocs et que leur autorisation peut être révoquée s'ils se conduisent mal. Les modèles de consensus dans les réseaux de blockchain autorisés sont généralement plus rapides et moins onéreux en termes de calcul.

Les réseaux autorisés à utiliser une blockchain peuvent également être utilisés par les organisations qui ont besoin de contrôler et de protéger plus étroitement leur blockchain. Cependant, si une seule entité contrôle qui peut publier des blocs, les utilisateurs de la blockchain devront avoir confiance dans cette entité. Les réseaux de blockchain autorisés peuvent également être utilisés par les organisations qui souhaitent travailler ensemble mais ne se font pas entièrement confiance. Ils peuvent établir un réseau de blockchain autorisé et inviter les partenaires commerciaux à enregistrer leurs transactions sur un grand livre distribué. Ces organisations peuvent déterminer le modèle de consensus à utiliser, en fonction de leur degré de confiance mutuel. Au-delà de la confiance, les réseaux de blockchain autorisés offrent une transparence et des informations qui peuvent aider à mieux éclairer les décisions commerciales et responsabiliser les parties qui se conduisent mal.

Cela peut inclure explicitement des entités d'audit et de supervision, faisant des audits une occurrence constante par rapport à un événement périodique.

Certains réseaux de blockchain autorisés prennent en charge la possibilité de révéler de manière sélective des informations sur les transactions en fonction de l'identité ou des informations d'identification des utilisateurs du réseau de chaînes. Avec cette fonctionnalité, un certain degré de confidentialité dans les transactions peut être obtenu. Par exemple, il se peut que la blockchain enregistre qu'une transaction a eu lieu entre deux utilisateurs du réseau de blockchain, mais que le contenu réel des transactions n'est accessible qu'aux parties impliquées.

Certains réseaux blockchain autorisés exigent que tous les utilisateurs soient autorisés à envoyer et recevoir des transactions (ils ne sont ni anonymes ni même pseudo-anonymes). Dans de tels systèmes, les parties travaillent ensemble pour mettre en place un processus commercial commun avec des éléments dissuasifs naturels de commettre une fraude ou de se comporter autrement comme un mauvais acteur (car ils peuvent être identifiés). Si un mauvais comportement devait se produire, il est bien connu où les organisations sont incorporées, quels recours juridiques sont disponibles et comment exercer ces recours dans le système judiciaire compétent.

2.4.2 Sans permission :

Les réseaux de blockchain sans autorisation sont des plates-formes de grand livre décentralisées ouvertes à tous ceux qui publient des blocs, sans avoir besoin de l'autorisation d'une autorité. Les plates-formes blockchain sans autorisation sont souvent des logiciels open source, disponibles gratuitement pour tous ceux qui souhaitent les télécharger. Etant donné que tout le monde a le droit de publier des blocs, il en résulte que tout le monde peut lire la blockchain et émettre des transactions sur la blockchain (en incluant ces transactions dans des blocs publiés). Tout utilisateur du réseau blockchain au sein d'un réseau blockchain sans autorisation peut lire et écrire dans le grand livre. Étant donné que les réseaux de chaînes de blocs sans autorisation sont ouverts à la participation de tous, des utilisateurs malveillants peuvent tenter de publier des blocs de manière à sous-inverser le système (exposé en détail plus loin). Pour éviter cela, les réseaux de blockchain sans permission utilisent souvent un accord multipartite ou un système de «consensus» (voir la section 2) qui oblige les utilisateurs à dépenser ou à maintenir des ressources lorsqu'ils tentent de publier des blocs. Cela empêche les utilisateurs malveillants de renverser facilement le système. Des exemples de tels modèles consensuels incluent les méthodes de preuve de travail (voir section 2.1) et de preuve d'enjeu (voir section 2.2). Les systèmes de consensus dans les réseaux de blockchain sans permission encouragent généralement les comportements non malveillants en récompensant les éditeurs de blocs conformes au protocole avec une crypto-monnaie native.

2.5 Cryptocurrencies

De nombreuses applications des technologies blockchain sont principalement axées sur le transfert de devises d'un compte à un autre. Cette section présente plusieurs exemples d'applications de blockchain

2.5.1 Bitcoin (BTC) :

Le Bitcoin est un système de paiement numérique qui a déjà été présenté comme le pionnier de l'utilisation d'une blockchain. De nouveaux blocs sont créés environ une fois toutes les 10 minutes à l'aide du hachage SHA-256 (voir section 1.4) pour les lier ensemble. C'est un système de preuve de travail où les nœuds d'exploration de données doivent trouver un nonce à inclure dans leur bloc de telle sorte que le hachage du bloc soit inférieur à une valeur de difficulté prédéterminée. La difficulté est ajustée vers le haut ou vers le bas pour tenter d'atteindre l'objectif de 10 minutes pour la création de blocs. Au début de l'histoire de Bitcoin, des ordinateurs individuels pouvaient extraire et publier des blocs; Actuellement, Bitcoin nécessite du matériel spécialisé, de grands centres de données ou de nombreuses personnes travaillant ensemble dans un pool de minage pour gagner la compétition pour publier un bloc.

Avec Bitcoin, le paiement des frais de transaction est techniquement facultatif car les nœuds miniers obtiennent la plupart de leurs fonds via la publication de blocs. Ces frais sont conçus pour être peu élevés pour chaque transaction, mais ils peuvent et sont devenus importants en raison d'un important arriéré de transactions en attente. Payer des frais de transaction plus élevés peut donner à une transaction une plus grande priorité pour être ajouté à la blockchain. Initialement, les nœuds d'exploration de données ont obtenu 50 Bitcoin pour chaque bloc, et seulement la moitié après un certain nombre de blocs. Par exemple, la récompense pour l'extraction d'un bloc était de 12,5 Bitcoins en juillet 2016. Selon le protocole Bitcoin, cette récompense diminuera de moitié tous les 210 000 blocs (environ quatre ans) et passera à zéro une fois que 21 millions de Bitcoins auront été produits. L'extraction de Bitcoin continuera à ce stade, mais la récompense sera entièrement dérivée des frais de transaction.

2.5.2 Ethereum (ETH) :

Ethereum est une plate-forme de blockchain axée sur la fourniture de smart contracts. Les smart contracts sont des programmes qui existent sur le blockchain et auxquels les utilisateurs d'Ethereum peuvent accéder. Ils peuvent à la fois recevoir et envoyer des fonds tout en effectuant des calculs arbitraires. Un contrat correctement conçu peut agir comme un tiers de confiance dans les transactions financières puisque son code est à la fois public et immuable. Le langage de programmation des transactions Ethereum est complet. Les nœuds miniers reçoivent des fonds par le biais de droits miniers et de frais de transaction.

Ethereum a également un concept appelé " gas " utilisé pour alimenter les calculs transactionnels (et est généralement autour de 1/100 000 d'un Ether). Chaque transaction consomme du gas comme il s'exécute, et à l'origine d'une transaction particulière doit payer suffisamment de gas, ou de l'exécution de la transaction échoue.

Il y a une limite maximale de gaz par contrat smart (actuellement trois millions de gas) pour empêcher les programmes coûteux en calcul d'être soumis aux nœuds miniers Ethereum. Cela est dû au fait que tous les nœuds miniers doivent exécuter les transactions en parallèle.

La soumission d'une transaction à un contrat Ethereum fait qu'un programme est exécutée en parallèle sur les ordinateurs des nœuds miniers. L'état résultant du contrat est stocké sur le blockchain par l'utilisateur qui publie le bloc suivant.

2.5.3 Ripple (XRP) :

Ripple est le nom à la fois d'une crypto-monnaie et du réseau de paiement sur lequel elle est transférée. L'objectif de Ripple est de s'appuyer sur l'approche du Bitcoin et de connecter différents systèmes de paiement ensemble. Il dispose d'une offre fixe de 100 milliards de XRP, dont la moitié est destinée à la circulation. Les clients Ripple n'ont pas besoin de télécharger l'intégralité de la blockchain, ce qui facilite la connexion des clients en quelques secondes. De plus, il n'y a aucune récompense minière pour l'exécution d'un serveur car chaque transaction coûte une petite quantité d'ondulation, similaire au gas Ethereum. Par conséquent, il n'y a pas de nœuds d'exploration ou de pools d'exploration de données; au lieu de cela, environ un millième de cent de chaque transaction est détruit. Ripple n'est pas conçu avec des objectifs explicites d'anonymat, mais il dispose de fonctionnalités assurant la confidentialité, telles que l'utilisation de paiements par passerelle mandatés.

2.6 Limitations de Blockchain et idées fausses

Il y a une tendance à sur-typer et à abuser de la technologie naissante. De nombreux projets tenteront d'intégrer la technologie, même si elle est inutile. Cela tient au fait que la technologie est relativement nouvelle et mal comprise, entourée d'idées fausses et de la peur de passer à côté. La technologie Blockchain n'a pas été épargnée. Cette section met en évidence certaines des limitations et idées fausses de la technologie blockchain.

2.6.1 Immutabilité :

La plupart des publications sur la technologie blockchain décrivent les registres comme étant immuables. Cependant, ce n'est pas strictement vrai. Ils sont inviolables et c'est une raison de leur confiance pour les transactions financières. Ils ne peuvent pas être considérés comme totalement immuables, car il existe des situations dans lesquelles la blockchain peut être modifiée. Dans cette section, nous examinerons différentes manières de violer le concept d'immuabilité pour les registres à blocs.

La blockchain elle-même ne peut être considérée comme totalement immuable. Pour certaines implémentations de la blockchain, les blocs les plus récemment publiés, peuvent être remplacés (par une chaîne alternative plus longue, avec des blocs "taille" différents). Comme indiqué précédemment, la plupart des réseaux de blockchain utilisent la stratégie consistant à adopter la chaîne la plus longue (celle avec le plus de travail fourni) comme une vérité lorsqu'il existe plusieurs chaînes en concurrence. Si deux chaînes sont en compétition mais que chacune d'elles comprend sa propre séquence de blocs de queue, la plus longue des deux sera adoptée. Toutefois, cela ne signifie pas que les transactions au sein des blocs remplacés sont perdues elles peuvent plutôt avoir été incluses dans un bloc différent ou renvoyées au pool de transactions en attente. C'est ce degré de faible immuabilité des blocs d'extrémité qui explique pourquoi la plupart des utilisateurs du réseau blockchain attendent plusieurs créations de bloc avant de considérer qu'une transaction est valide.

2.6.2 Utilisateurs impliqués dans la gouvernance de Blockchain :

La gouvernance des réseaux de blockchain traite des règles, pratiques et processus par lesquels le réseau de blockchain est dirigé et contrôlé. Une idée fausse commune est que les réseaux blockchain sont des systèmes sans contrôle ni propriété. La phrase «personne ne contrôle une blockchain!» s'exclame souvent. Ce n'est pas strictement vrai. Les réseaux de blockchain autorisés sont généralement configurés et gérés par un propriétaire ou un consortium, qui régit le réseau de blockchain. Les réseaux blockchain sans autorisation sont souvent régis par les utilisateurs du réseau blockchain, les nœuds de publication et les développeurs de logiciels. Chaque groupe a un niveau de contrôle qui influe sur la direction de l'avancement du réseau de blockchain.

Les développeurs de logiciels créent le logiciel blockchain utilisé par un réseau blockchain. Étant donné que la plupart des technologies blockchain sont open source, il est possible d'inspecter le code source et de le compiler indépendamment. Il est même possible de créer un logiciel séparé mais compatible pour contourner les logiciels précompilés publiés par les développeurs.

Cependant, tous les utilisateurs ne seront pas en mesure de le faire, ce qui signifie que le développeur du logiciel Blockchain jouera un rôle important dans la gouvernance du réseau Blockchain. Ces développeurs peuvent agir dans l'intérêt de la communauté en général et sont tenus pour responsables. Par exemple, en 2013, les développeurs de Bitcoins ont publié une nouvelle version du client Bitcoin le plus répandu, introduisant une faille et démarrant deux blockchain en concurrence. Les développeurs devaient décider de conserver la nouvelle version (qui n'avait pas encore été adoptée par tous) ou de revenir à l'ancienne version. Dans les deux cas, une chaîne sera rejetée et certaines transactions de l'utilisateur du réseau blockchain deviendront invalides. Les développeurs ont fait leur choix, sont revenus à l'ancienne version et ont contrôlé avec succès la progression de la blockchain Bitcoin.

En résumé, les développeurs de logiciels, les nœuds de publication et les utilisateurs du réseau blockchain jouent tous un rôle dans la gouvernance du réseau blockchain.

2.6.3 Blockchain Death :

Les systèmes centralisés traditionnels sont constamment créés et démantelés, et les réseaux à chaînes multiples ne seront probablement pas différents. Cependant, parce qu'ils sont décentralisés, il y a une chance que lorsqu'un réseau de blockchain "s'arrête" il ne sera jamais complètement arrêté, et qu'il peut toujours y avoir des nœuds de blockchain en cours d'exécution.

Un blockchain obsolète ne serait pas adapté à un enregistrement historique, car sans beaucoup de nœuds d'édition, un utilisateur malveillant pourrait facilement dominer les quelques nœuds d'édition restants et refaire et remplacer n'importe quel nombre de blocs.

2.6.4 Au-delà du Digital :

Les réseaux de blockchain fonctionnent extrêmement bien avec les données au sein de leurs propres systèmes numériques. Cependant, lorsqu'ils ont besoin d'interagir avec le monde réel, certains problèmes se posent (souvent appelé le problème Oracle[20]). Un réseau blockchain peut être un endroit pour enregistrer à la fois les données d'entrée humaines et les données d'entrée de capteurs du monde réel, mais il peut ne pas y avoir de méthode pour déterminer si les données d'entrée reflètent des événements du monde réel. Un capteur peut ne pas fonctionner correctement et enregistrer des données inexactes. Les humains pourraient enregistrer de fausses informations (intentionnellement ou non). Ces problèmes ne concernent pas uniquement les réseaux blockchain, mais l'ensemble des systèmes numériques. Cependant, pour les réseaux de blockchain qui sont pseudonymes, le traitement de fausses déclarations de données en dehors du réseau numérique peut être particulièrement problématique.

Par exemple, si une transaction de crypto-monnaie a eu lieu pour acheter un article du monde réel, il est impossible de déterminer dans le réseau de blockchain si l'envoi a eu lieu, sans faire appel à un capteur extérieur ou à une intervention humaine.

[20] *https://cointelegraph.com/explained/blockchain-oracles-explained*

2.6.5 Cybersecurité :

L'utilisation de la technologie blockchain ne supprime pas les risques inhérents à la cybersécurité qui nécessitent une gestion des risques réfléchie et proactive. Bon nombre de ces risques inhérents comportent un élément humain. Par conséquent, un solide programme de cybersécurité reste essentiel pour protéger le réseau et les organisations participantes des cyber-menaces, en particulier à mesure que les pirates informatiques développent davantage de connaissances sur les réseaux blockchain et leurs vulnérabilités.

Les normes et directives existantes en matière de cybersécurité restent extrêmement pertinentes pour garantir la sécurité des systèmes qui interfacent et / ou reposent sur des réseaux blockchain. Sous réserve de certains ajustements pour tenir compte d'attributs spécifiques de la technologie de la blockchain, les normes et directives existantes constituent une base solide pour la protection des réseaux de la blockchain contre les cyberattaques.

En plus des principes généraux et des contrôles, il existe des normes spécifiques de cybersécurité qui s'appliquent à la technologie blockchain et qui existent déjà et sont largement utilisées par de nombreuses industries. Par exemple, le cadre de cybersécurité du NIST stipule expressément qu'il ne s'agit pas d'une approche universelle de gestion du risque de cybersécurité, car « les organisations continueront d'avoir des risques uniques, différentes menaces, différentes vulnérabilités, différentes tolérances au risque et la façon dont elles mettent en œuvre les pratiques du [Framework] variera. » Cela dit, même si le Framework n'a pas été conçu spécifiquement pour la technologie blockchain, ses normes sont suffisamment générales pour couvrir la technologie blockchain et pour aider les institutions à élaborer des politiques et des processus qui permettent de cerner et de contrôler les risques qui touchent la technologie des blockchain.

2.6.6 Cyberattaques et attaques réseau :

Les technologies de blockchain sont considérées comme extrêmement sécurisées en raison de leur conception inviolable et résistante à la falsification une fois qu'une transaction est validée dans la chaîne de blocs, elle ne peut généralement pas être modifiée. Toutefois, cela n'est vrai que pour les transactions qui ont été incluses dans un bloc publié. Les transactions qui n'ont pas encore été incluses dans un bloc publié dans la blockchain sont vulnérables à plusieurs types d'attaques. Pour les réseaux blockchain qui ont des horodatages transactionnels, le temps d'usurpation d'identité ou le réglage de l'horloge d'un membre d'un service de commande peut avoir des effets positifs ou négatifs sur une transaction, faisant du temps et de la communication du temps un vecteur d'attaque. Les attaques par déni de service peuvent être menées sur la plate-forme blockchain ou sur le contrat intelligent implémenté sur la plate-forme.

Les réseaux de blockchain et leurs applications ne sont pas à l'abri des acteurs malveillants qui peuvent effectuer une analyse et une reconnaissance de réseau afin de découvrir et d'exploiter des vulnérabilités et de lancer des attaques «zero day». Dans la hâte de déployer des services basés sur des blockchain, les applications nouvellement codées (telles que les contrats intelligents) peuvent contenir des vulnérabilités nouvelles et connues ainsi que des faiblesses de déploiement qui seront découvertes puis attaquées via le réseau, tout comme les sites Web ou les applications sont attaqués aujourd'hui.

2.6.7 Infrastructure à clé publique et identité :

En entendant que la technologie blockchain intègre une infrastructure à clé publique, certaines personnes pensent immédiatement qu'elle supporte intrinsèquement l'identité. Ce n'est pas le cas, car il peut ne pas y avoir de relation un à un de paires de clés privées avec les utilisateurs (un utilisateur peut avoir plusieurs clés privées), et il n'existe pas non plus de relation un à un entre les adresses en chaîne et les clés publiques (plusieurs adresses peuvent être dérivées d'une seule clé publique).

Les signatures numériques sont souvent utilisées pour prouver l'identité dans le monde de la cybersécurité, ce qui peut entraîner une confusion quant à l'application potentielle d'une blockchain à la gestion des identités. Le processus de vérification de la signature d'une transaction de la blockchain relie les transactions aux propriétaires de clés privées, mais ne permet pas d'associer des identités réelles à ces propriétaires. Dans certains cas, il est possible de connecter des identités du monde réel à des clés privées, mais ces connexions sont établies via des processus extérieurs et non explicitement pris en charge par la blockchain. Par exemple, un organisme chargé de l'application de la loi pourrait demander des enregistrements à un échange qui relierait des transactions à des individus spécifiques. Un autre exemple est une personne qui affiche une adresse de crypto-monnaie sur son site Web personnel ou sa page de média social pour les dons, ce qui fournirait un lien d'adresse à l'identité du monde réel.

Bien qu'il soit possible d'utiliser la technologie blockchain dans les cadres de gestion des identités nécessitant un composant de grand livre distribué, il est important de comprendre que les implémentations classiques de blockchain ne sont pas conçues pour servir de systèmes de gestion d'identités autonomes. La sécurité des identités numériques ne se limite pas à la simple mise en place d'une blockchain.

2ème partie : Analyse d'un smart contract

Chapitre 3 : Infrastructure des smart contracts

3.1 Smart contacts

Le terme smart contract date de 1994, défini par Nick Szabo comme «un protocole de transaction informatisé qui exécute les termes d'un contrat. Les objectifs généraux de la conception intelligente de contrats sont de satisfaire les conditions contractuelles courantes (telles que les conditions de paiement, les privilèges, la confidentialité et même l'application), de minimiser les exceptions malveillantes et accidentelles et de minimiser le besoin d'intermédiaires de confiance. »

Les smart contracts étendent et exploitent la technologie blockchain. Un smart contract est une collection de codes et de données (parfois appelés fonctions et état) qui est déployée à l'aide de transactions signées cryptographiquement sur le réseau de la blockchain (par exemple, les contrats intelligents d'Ethereum, le code de chaîne d'Hyperledger Fabric). Le smart contract est exécuté par des nœuds au sein du réseau blockchain; tous les nœuds qui exécutent le smart contract doivent dériver les mêmes résultats de l'exécution, et les résultats de l'exécution sont enregistrés sur la blockchain

Les utilisateurs du réseau Blockchain peuvent créer des transactions qui envoient des données à des fonctions publiques offertes par un smart contract. Le smart contract exécute la méthode appropriée avec les données fournies par l'utilisateur pour effectuer un service. Le code, étant sur la blockchain, est également inviolable et résistant aux altérations et peut donc être utilisé (entre autres) en tant que tiers de confiance. Un smart contract peut effectuer des calculs, stocker des informations, exposer des propriétés pour refléter un état exposé publiquement et, le cas échéant, envoyer automatiquement des fonds à d'autres comptes. Il n'a même pas nécessairement à remplir une fonction financière. Par exemple, les auteurs de ce document ont créé un contrat intelligent Ethereum qui génère publiquement des nombres aléatoires fiables. Il est important de noter que toutes les blockchain ne peuvent pas exécuter de contrats intelligents.

Le code de smart contract peut représenter une transaction multipartite, généralement dans le contexte d'un processus métier. Dans un scénario multipartite, l'avantage est que cela peut fournir des données attestables et une transparence qui peuvent favoriser la confiance, fournir des informations qui peuvent permettre de meilleures décisions commerciales, réduire les coûts de la réconciliation qui existe dans les applications commerciales traditionnelles et réduire le temps de terminer une transaction.

Les smart contracts doivent être déterministes, dans la mesure où, étant donné une entrée, ils produiront toujours la même sortie en fonction de cette entrée. De plus, tous les nœuds exécutant le smart contract doivent se mettre d'accord sur le nouvel état obtenu après l'exécution. Pour y parvenir, les smart contracts ne peuvent pas fonctionner sur des données en dehors de ce qui leur est directement transmis (par exemple, les smart contracts ne peuvent pas obtenir de données de services Web à partir du smart contract - elles devraient être transmises en tant que paramètre). Tout contrat intelligent qui utilise des données en dehors du contexte de son propre système est censé utiliser un «Oracle». (Le problème d'Oracle est décrit dans la section 6.4 de la partie 1).

Pour de nombreuses implémentations de blockchain, les nœuds de publication exécutent le code de smart contract simultanément lors de la publication de nouveaux blocs. Il existe certaines implémentations de blockchain dans lesquelles il existe des nœuds de publication qui n'exécutent pas de code de smart contract, mais valident à la place les résultats des nœuds qui le font. Pour les réseaux blockchain sans autorisation activés par smart contract (tels que Ethereum), l'utilisateur émettant une transaction vers un smart contract devra payer le coût de l'exécution du code. Il existe une limite au temps d'exécution pouvant être consommé par un appel à un smart contract, en fonction de la complexité du code. Si cette limite est dépassée, l'exécution s'arrête et la transaction est rejetée. Ce mécanisme récompense non seulement les éditeurs pour l'exécution du code de smart contract, mais empêche également les utilisateurs malveillants de déployer puis d'accéder à des contrats intelligents qui effectueront un déni de service sur les nœuds de publication en consommant toutes les ressources (par exemple, en utilisant des boucles infinies).

Pour les réseaux blockchain autorisés par smart contract, tels que ceux utilisant le code de chaîne d'Hyperledger Fabric, les utilisateurs peuvent ne pas être tenus de payer pour l'exécution du code de smart contract. Ces réseaux sont conçus pour avoir des participants connus et d'autres méthodes de prévention des mauvais comportements peuvent être utilisées (par exemple, la révocation de l'accès).

3.2 Cycle de vie d'un smart contract

Le cycle de vie d'un smart contract se compose généralement de quatre grandes phases: création du contrat intelligent, gel du contrat intelligent, exécution du contrat intelligent et finalisation du contrat intelligent.

3.2.1 Création :

La phase de création peut être divisée en une négociation de contrat itérative et une phase de mise en œuvre. Premièrement, les parties doivent convenir du contenu et des objectifs généraux du contrat. Cela peut être fait en ligne ou hors ligne et est similaire aux négociations contractuelles classiques. Toutes les parties participantes doivent avoir un portefeuille sur la plate-forme de comptabilité sous-jacente. Son identifiant est dans la plupart des cas pseudonyme, et il est utilisé pour l'identification des parties ainsi que pour le transfert de fonds.

Après s'être mis d'accord sur les objectifs et le contenu du contrat, le l'accord doit être transformé en code. La codification du contrat est limitée par l'expressivité du langage de codage du smart contract sous-jacent. Pour valider l'exécution d'un smart contract comportement et contenu, la plupart des environnements de smart contract fournissent l'infrastructure pour créer, maintenir et tester le contrat. Comme on peut le voir dans les langages de programmation classiques, la transformation des exigences en code nécessite plusieurs itérations entre les parties prenantes et le (s) programmeur (s). Les smart contracts ne seront pas différents et nécessiteront probablement de nombreuses itérations entre la phase de négociation et de mise en œuvre.

Une fois que les parties ont accepté la version codifiée du contrat, il est soumis au grand livre distribué lors de la phase de publication. Au cours de cette phase, les nœuds participant au grand livre distribué reçoivent le contrat dans le cadre d'un bloc de transaction et une fois le bloc confirmé par la majorité des nœuds, le contrat est prêt à être exécuté.

Étant donné que les smart contracts décentralisés ne peuvent pas être modifiés après avoir été acceptés par la blockchain, les modifications du smart contrat ne sont pas possibles et nécessitent la création d'un nouveau contrat. Bien qu'un smart contrat ait été stocké sur la blockchain, ce fait à lui seul ne devrait pas être considéré comme un accord de la partie pour conclure le contrat car tout le monde peut soumettre un smart contract à la blockchain indiquant une obligation pour tout propriétaire de portefeuille aléatoire. De même, les smart contracts décentralisés peuvent bénéficier à tout participant de la blockchain, qu'il ait accepté ou non de bénéficier des avantages au préalable.

3.2.2 Gel :

Une fois le smart contrat soumis à la blockchain, il est persisté par une confirmation majoritaire des nœuds participants. En échange de ce service, et pour éviter une inondation de l'écosystème avec des smarts contrats, une redevance doit être payée aux mineurs. À partir de ce moment, le contrat et toutes les parties sont publics et accessibles via le grand livre public. Au cours de la phase de gel, tous les transferts effectués vers l'adresse portefeuille du smart contrat sont gelés et les nœuds jouent le rôle d'un conseil de gouvernance, garantissant que les conditions préalables à l'exécution du contrat sont remplies.

3.2.4 Exécution :

Les contrats stockés sur le grand livre distribué sont lus par les nœuds participants. L'intégrité du contrat est validée et le moteur d'inférence de l'environnement de smart contract (compilateur, interprète) exécute le code. Les entrées pour l'exécution sont collectées auprès des oracles intelligents et des parties impliquées (engagement envers les marchandises par le biais de pièces) et les fonctions du smart contract sont exécutées. L'exécution du smart contract entraîne un ensemble de nouvelles transactions ainsi qu'un nouvel état du smart contract. L'ensemble des résultats ainsi que les nouvelles informations d'état sont soumis au grand livre distribué et sont validés par le protocole de consensus.

3.2.5 Finalisation :

Une fois le smart contrat exécuté, les transactions résultantes et les nouvelles informations d'état sont stockées dans le grand livre distribué et confirmées selon le protocole de consensus. Les actifs numériques précédemment engagés sont transférés (dégel des actifs) et avec la confirmation de toutes les transactions, le contrat a été exécuté.

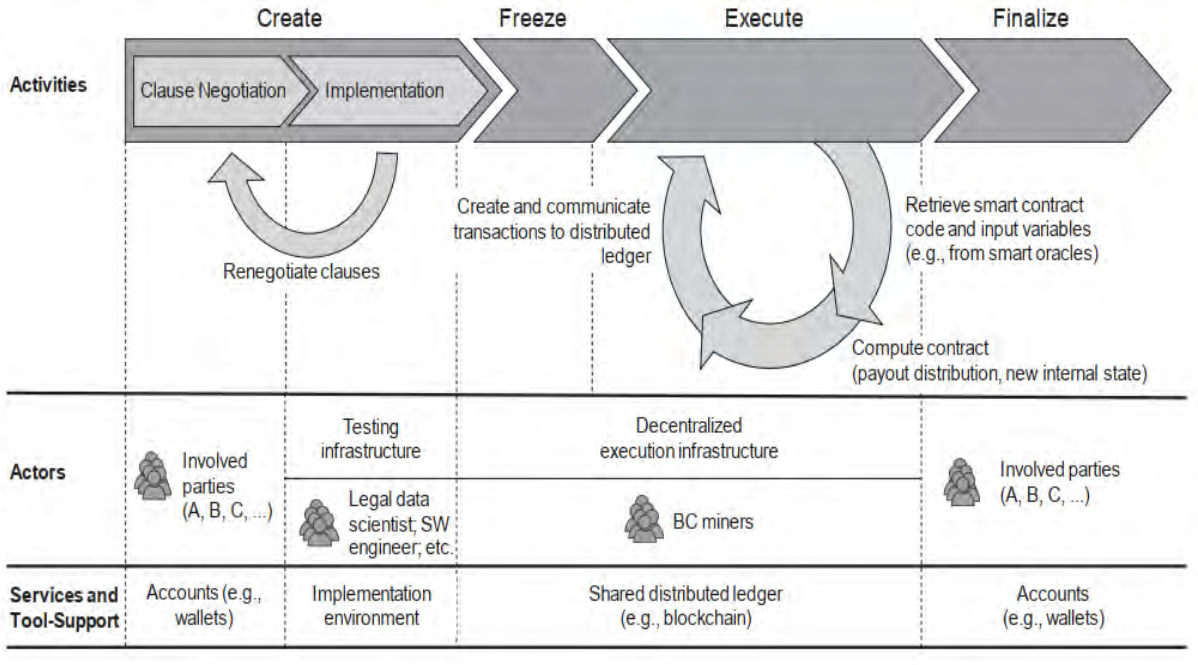


Figure 7 Le cycle de vie d'un contrat intelligent: phases, acteurs et services

3.3 Modélisation UML d'un smart contract

Les smart contracts pour Ethereum sont généralement écrits en utilisant le langage Solidity. Solidity est un langage orienté objet, et les contrats y sont définis comme des classes ils ont une structure de données, des fonctions publiques et privées, et peuvent hériter d'autres contrats. Les smart contracts ont également des concepts spécifiques comme les événements et les modificateurs.

Pour aider à la modélisation des smart contracts, nous utilisons des diagrammes UML. Cependant, comme les smart contracts ont des caractéristiques très spécifiques, nous avons introduit de nouveaux concepts dans ces diagrammes, pour pouvoir mieux modéliser et spécifier les smart contracts. Dans la mesure du possible, ces concepts sont simplement introduits sous forme de stéréotypes UML, qui sont des balises qui peuvent être utilisées dans les diagrammes UML lorsque cela est nécessaire. Dans quelques autres cas, nous avons dû introduire une notation spécifique, comme le transfert des Ethers dans les diagrammes de séquence. Les diagrammes UML que nous trouvons utiles pour modéliser les smart contracts sont:

- Diagrammes de classes, pour représenter la structure et les relations des smart contracts; nous avons introduit divers stéréotypes dans ce type de diagramme.
- Diagramme de d'états, pour représenter les différents états d'un smart contract; ce schéma n'a pas besoin de nouveau concept.
- Diagrammes de séquence, pour représenter les messages envoyés à un smart contract, et d'un smart contract à un autre smart contract; ce diagramme a besoin de nouveau type de messages - le transfert des Ethers.

Les diagrammes de classes UML sont utilisés pour représenter les smart contracts et les structures. Dans Solidity, il n'y a pas le concept de classe, mais les smart contracts sont très similaires aux classes. Comme une classe, un smart contract peut avoir une structure de données, des fonctions publiques et privées, et peut hériter d'un ou plusieurs smart contracts. Cependant, les SC ont une nature spécifique; ils sont créés par des transactions, mais une transaction peut créer au plus un seul smart contract. Les smart contracts, cependant, peuvent envoyer des messages à d'autres smart contract, résidant dans la même Blockchain. Dans Solidity, il est également possible de définir des structures, c'est-à-dire des structures de données complexes, qui ne sont pas pourvues de fonctions. Par conséquent, le modèle d'un smart contract créé par une transaction peut inclure d'autres smart contract dont il hérite, les structures utilisées et les smart contracts externes auxquels sont envoyés des messages.

D'autres concepts spécifiques des smart contracts Ethereum sont des événements, des drapeaux qui sont levés quand quelque chose de pertinent se produit, et qui le signalent au monde extérieur (qui doit observer le smart contract de manière autonome et agir en conséquence), et des modificateurs, fonctions spéciales qui sont appelées avant une fonction, vérifiant ses contraintes et éventuellement arrêtant l'exécution. Le tableau 2 montre les stéréotypes que nous avons introduits pour permettre de représenter les concepts smart contract dans les diagrammes de classes UML. Les événements pourraient être représentés dans un autre compartiment, en plus de ceux contenant le nom, l'attribut et les fonctions (opérations).

Stéréotype	Position	Description
Contrat	Symbole de classe - compartiment supérieur	Désigne un smart contract.
«Contrat de bibliothèque»	comme ci-dessus	Un contrat tiré d'une bibliothèque (standard)
«Structure»	comme ci-dessus	Une structure, contenant des données mais aucune opération, définie et utilisée dans la structure de données d'un contrat
«énumération»	comme ci-dessus	Une énumération contenant uniquement une liste de valeurs possibles
«interface»	comme ci-dessus	Un contrat contenant uniquement des déclarations de fonction
«modifier»	Symbole de classe - compartiment inférieur	Un type particulier de fonction, défini dans Solidity
«Tableau»	Rôle d'une association	La relation 1: n est implémentée à l'aide d'un tableau
« Map »	comme ci-dessus	La relation 1: n est implémentée à l'aide d'un mappage
«Map [uint]»	comme ci-dessus	La relation 1: n est implémentée à l'aide d'un mappage de l'entier à la valeur

Tableau 3 Ajouts au diagramme de classes UML (stéréotypes)

Les trois derniers stéréotypes définissent la mise en œuvre de relations 1: n dans la structure de données d'un smart contract. Dans Solidity, les seules collections prises en charge pour gérer le stockage (les données stockées en permanence dans la Blockchain) sont le tableau et le mappage. Le premier est un tableau de base classique de tous les langages informatiques, avec l'ajout que de nouveaux éléments peuvent y être ajoutés (mais pas supprimés). Le mappage est une collection capable de stocker des paires clé-valeur et de récupérer efficacement une valeur, compte tenu de sa clé, mais pas capable d'itérer sur ses éléments. Le dernier stéréotype fait référence à un modèle commun de programmation Solidity en utilisant un mappage avec des entiers positifs comme clés, de sorte qu'il est possible d'itérer dessus.

Les diagrammes de séquence UML sont utilisés pour modéliser la messagerie. Dans Ethereum, les messages sont associés à des transactions envoyées à la Blockchain depuis des utilisateurs

ou des systèmes externes ou depuis des smart contracts. Comme dans le jargon orienté objet, les messages sont synonymes d '«appels de fonction publique». Si une fonction n'écrit pas ou ne modifie pas la Blockchain (on l'appelle une fonction «voir»), le message correspondant peut être envoyé sans frais. Les autres messages doivent être payés en GAS[21] pour être exécutés.

Les spécificités d'Ethereum concernant la messagerie sont les types de participants (identifiés par leurs comptes) et les types de messages. Les participants peuvent être spécifiés à l'aide de stéréotypes, comme indiqué dans le tableau 3. Les messages nécessitent une notation spécifique.

Stéréotype	**Description**
« Personne »	Un rôle humain, l'envoi de messages à l'aide d'un portefeuille ou d'une autre application
«Système»	Un système externe, capable d'envoyer des messages à la Blockchain
«Device»	Un appareil (généralement IoT), capable d'envoyer des messages
«Contrat»	Un smart contract, faisant partie du système ou externe à celui-ci
«Oracle»	Un type particulier de smart contract, dont la date est écrite par un tiers de confiance, et permet d'accéder à des informations sur le monde extérieur
« Compte »	Un compte Ethereum, détenant simplement Ethers. Il ne peut recevoir des Ethers, ou envoyer des Ethers vers un autre compte ou smart contract que si le propriétaire active le transfert

Tableau 4 Ajouts au diagramme de séquence UML (stéréotypes)

Les différents types de messages pertinents pour la conception sont:

- Création de smart contract : il est envoyé par un participant externe ou par un autre smart contract; dans un diagramme de séquence, une création est représentée en dessinant le nouveau participant au niveau temporel de sa création.
- Appel de fonction : une transaction entraînant une modification de la Blockchain, et donc un paiement GAS; c'est le message «synchrone» ou «asynchrone» classique.
- Appel de fonction View / pure : une transaction n'entraînant aucune modification de la Blockchain et aucun paiement GAS; il peut être modélisé en ajoutant le stéréotype «vue» ou «pur» au nom du message.

[21] *https://www.investopedia.com/terms/g/gas-ethereum.asp*

- Transferts ETH : une transaction spéciale qui transfère des Ethers d'un compte, ou d'un smart contract, vers un autre compte ou smart contract. Ceci est modélisé à l'aide d'une flèche spéciale, similaire à la flèche d'héritage des diagrammes de classes UML.

Chapitre 4 : Etude des plateformes

4.1 Plateformes d'un smart contract

Les smart contracts sont des moyens d'échanger de l'argent, des actions, des propriétés ou toute forme d'actif de manière transparente, sécurisée et sans conflit tout en omettant la nécessité d'une intermédiation par un intermédiaire. Tout au long de cette section, nous passerons en revue les plates-formes de smart contracts les plus largement utilisées qui se sont avérées efficaces et fiables dans diverses applications commerciales.

4.1.1 Ethereum :

Ethereum est une plate-forme décentralisée basée sur la blockchain qui exécute des smart contracts, qui a également ouvert la porte à des applications décentralisées (DApp). La machine virtuelle Ethereum (EVM) est une machine virtuelle qui exécute tous les contrats intelligents. EVM est une machine virtuelle 256 bits Turing Complete[22]. Les smart contracts basés sur Ethereum sont codés à l'aide de Solidity, qui est un langage de programmation Turing Complete qui permet le codage des instructions de code en boucle et en dérivation. La «complétude de Turing» de Solidity rend Ethereum idéal pour coder des smart contracts avec une logique sophistiquée.

Le «GAS» est le carburant des smart contracts d'Ethereum. Il quantifie la quantité de puissance de calcul nécessaire pour exécuter des smart contracts via l'EVM. Lorsque vous soumettez un smart contracts, vous devez déterminer sa valeur en gas. Chaque étape du code du smart contracts nécessite l'exécution d'une quantité de gas prédéterminée.

Les contrats intelligents Ethereum peuvent :

- Agir comme des comptes ethereum "multi-signatures", de sorte que les pièces ne sont dépensées que si un nombre prédéterminé d'utilisateurs sont d'accord
- Offrir l'utilité à d'autres contrats intelligents sur la blockchain d'Ethereum
- Enregistrez des informations sur la propriété des actifs, l'enregistrement de domaine, les privilèges d'adhésion, les droits d'application, et plus encore
- Gérez les accords entre plusieurs parties, tels que les locations, la collaboration commerciale et l'assurance
- Être codé pour émettre des jetons tels que les jetons ICO utilisés pour le financement participatif. Il existe plusieurs normes de jetons utilisées pour émettre des jetons sur la plate-forme Ethereum, y compris les normes ERC-20, ERC223 et ERC77. ERC-20 est la norme la plus couramment utilisée pour émettre des jetons à des fins d'ICO, malgré ses graves bugs qui ont déjà entraîné des millions de dollars de pertes dans l'industrie de la cryptographie.

[22] *https://fr.wikipedia.org/wiki/Turing-complet*

ERC-20 effectue une transaction de jeton via l'une des deux méthodes suivantes:

- Transfer () cette fonction déclenche l'envoi de jetons à l'adresse d'un utilisateur spécifique.
- Approve () + transferFrom () : cette fonction déclenche le dépôt de jetons dans un smart contract prédéfini.

Cependant, si la fonction transfer () est accidentellement utilisée pour envoyer des jetons à un smart contract, la transaction sera exécutée avec succès, mais cette transaction ne sera jamais reconnue par l'adresse du smart contract du destinataire. Ce bug a inspiré les développeurs à créer les standards ERC223 et ERC77.

- ERC223 : cette norme atténue le bogue critique de l'ERC-20 en modifiant la fonction transfer () afin qu'elle déclenche une erreur en réponse à des transferts invalides et annule la transaction afin qu'aucuns fonds ne soient perdus.
- ERC777 : Cette norme résout le problème de l'ERC20 de manque d'opérations de gestion des transactions.

Ethereum est un choix populaire pour créer des smart contracts, mais les problèmes d'évolutivité de la plate-forme la rendent inadaptée à de nombreuses applications du monde réel. Solidity manque de flexibilité de codage fournie par les langages de programmation plus récents. Solidity ne prend pas en charge les tableaux multidimensionnels dans les paramètres d'entrée ainsi que les paramètres de sortie. De plus, Solidity ne prend en charge que 16 paramètres dans une fonction de smart contracts.

Malgré cela, les smart contracts basés sur Ethereum sont utilisés dans diverses applications. Par exemple, PCHAIN[23] a été le premier projet de blockchain à créer un système natif à chaînes multiples qui prend entièrement en charge la machine virtuelle Ethereum (EVM) - l'environnement d'exécution pour les smart contracts Ethereum.

4.1.2 EOS :

EOS devient de plus en plus l'une des plateformes de smart contracts les plus populaires. La plate-forme a attiré l'attention de la communauté cryptographique pour une multitude de raisons, à savoir que les transactions sur la plate-forme nécessitent des frais presque nuls et la capacité de la plate-forme à gérer des millions de transactions par seconde.

Les smart contracts sont programmés à l'aide de C ++, ce qui augmente la flexibilité de programmation. Les smart contracts EOS sont mis en œuvre sur la blockchain sous la forme d'un assemblage Web précompilé (WASM), qui favorise une exécution plus rapide des contrats par rapport aux smart contracts basés sur Ethereum. WASM est compilé avec C / C ++ via clang et LLVM. Les développeurs doivent avoir des connaissances en C / C ++ afin de pouvoir coder des smart contracts sur la blockchain d'EOS. Même si C peut être utilisé pour créer des contrats, il est fortement recommandé d'utiliser l'API EOS.IO C ++, qui renforce la sécurité des contrats et rend son code facilement lisible.

[23] *https://www.cointelligence.com/coins/pchain/*

EOS utilise le mécanisme de consensus de preuve de participation déléguée, qui agit avec une évaluation partielle et une exécution parallèle pour offrir une plate-forme de contrat intelligente avec des niveaux élevés d'évolutivité et des frais de transaction presque nuls.

Même si EOS est beaucoup moins populaire qu'Ethereum, il a établi le modèle de "parachutage" en tant que concurrent du modèle de financement participatif ICO d'Ethereum.

4.1.3 AION :

Aion est une plate-forme de smart contract qui permet le routage des transactions et des messages entre différentes blockchain via ses protocoles innovants de «pontage». Aion est un réseau à plusieurs niveaux comprenant les composants suivants :

- Ponts
- Réseaux de connexion
- Transactions inter-chaînes
- Réseaux participants

Les ponts d'Aion permettront d'effectuer des transactions sur plusieurs blockchain (transactions inter-chaînes) via l'écosystème de blockchain AION. Les transactions entre chaînes sont exécutées via des ponts et des réseaux de connexion. Les réseaux de connexion représentent les protocoles par lesquels toutes les blockchain publiques et privées peuvent communiquer avec l'écosystème de blockchain d'AION. Les réseaux participants sont des réseaux qui ont rempli un ensemble spécial d'exigences pour faire partie de l'écosystème blockchain d'AION. Les réseaux participants doivent prendre en charge la diffusion des transactions atomiques et mettre en œuvre un temps de verrouillage qui leur permettra de geler les transactions qui entrent dans un état "Oh Hold".

La machine virtuelle Aion (AVM) permet l'exécution de smart contracts. AVM est une implémentation JVM conçue pour exécuter la logique de chaîne. Le langage Aion est le langage de script utilisé pour programmer des contrats intelligents dans AVM. Actuellement, le noyau d'Aion fonctionne sur Java, les développeurs doivent donc utiliser des langages comme Python ou Groovy pour coder des smart contracts sur la blockchain d'Aion. Cependant, la plate-forme s'appuiera à terme sur le langage Aion pour l'écriture de smart contract.

Aion-1 est la plate-forme autonome d'Aion qui permet l'exécution de smart contracts créés sur d'autres blockchain. À l'heure actuelle, Aion s'appuie sur l'EVM d'Ethereum, mais finalement Aion-1 sera activé et permettra aux développeurs d'exécuter leurs smart contracts et DApp basés sur Ethereum beaucoup moins cher et plus rapidement que sur l'EVM.

4.1.4 NEM :

NEM est une plate-forme de smart contract plus évolutive qu'Ethereum. Là où Ethereum peut gérer 15 transactions par seconde, NEM peut gérer des centaines de transactions par seconde. NEM est plus rapide, plus sécurisé et fournit une technologie de smart contract simple. NEM utilise du code hors blockchain pour programmer des smart contracts, ce qui rend la blockchain de NEM moins décentralisée que celle d'Ethereum, tout en promouvant des niveaux de sécurité plus élevés, une confirmation plus rapide des transactions et un code de programmation plus léger. Les fonctionnalités de sécurité en ligne de NEM telles que les signatures multiples et les actifs intelligents résolvent ce problème.

Les actifs intelligents sont des applications de gestion de données uniques qui peuvent être utilisées pour créer des enregistrements de données, des jetons, des systèmes de vote et de nouvelles pièces en utilisant un code de programmation simple. L'extrême fonctionnalité de la blockchain de NEM est fournie via sa puissante API, qui permet l'utilisation de tout langage de programmation (comme JS, Python et autres) pour coder des smart contracts. L'API de NEM est utilisée pour développer des "contrats hors chaîne", qui peuvent être mis à jour à tout moment, sans communiquer avec la blockchain de NEM.

4.1.5 Stellar :

Stellar est une plate-forme de smart contract où les transactions sont plus sécurisées, plus rapides et moins chères que les transactions sur la blockchain d'Ethereum. Les smart contracts stellaires (SSC) ne sont pas Turing complete et sont déployés sous la forme d'accords programmés entre plusieurs parties qui sont appliqués par des transactions. Alors qu'il faut environ 3,5 minutes pour qu'une transaction soit confirmée sur la blockchain d'Ethereum, une transaction sur la blockchain de Stellar ne nécessite que 5 secondes environ pour être confirmée. Les frais de transaction sont négligeables, en moyenne autour de (0,0001 XLM ~ = 0,0000002 $). SSC peut être codé en utilisant n'importe quel langage de programmation tel que Python, JS, PHP, Golang et autres via l'API Stellar. Un SSC est composé de transactions qui sont interconnectées et exécutées au moyen de multiples contraintes, notamment les signatures multiples, le traitement par lots / atomicité, la séquence et les limites de temps. Le traitement par lots permet d'inclure plusieurs opérations dans une même transaction. L'atomicité garantit qu'à la soumission d'une série d'opérations au réseau de Stellar, toutes les opérations d'une transaction échoueront si une seule opération ne s'exécute pas. La séquence est un concept unique présenté sur la blockchain de Stellar via le "numéro de séquence". Avec les numéros de séquence, des transactions spécifiques échoueraient si une transaction alternative était exécutée avec succès. Les limites de temps représentent des limitations sur la période de validité d'une transaction. L'utilisation de limites de temps permet la représentation de périodes dans un SSC.

4.1.6 Hyperledger Fabric (HLF) :

Hyperledger Fabric (HLF) est une blockchain autorisée conçue avec une flexibilité avancée. Les smart contracts de HLF sont appelés «chaincode». HLF est écrit en langage Go, le langage de programmation open source de Google, donc chaincode prend également très bien en charge ce langage.

4.1.7 Corda :

Corda est une plate-forme de smart contract idéale pour créer des accords financiers. Les smart contracts de Corda sont des transactions valides qui doivent être acceptées par le smart contract de chacun de ses états d'entrée et de sortie. Les smart contracts sont codés à l'aide d'un langage de programmation JVM tel que Java ou Kotlin. L'exécution d'un smart contract est déterministe et son acceptation d'une transaction repose uniquement sur le contenu de la transaction. Parfois, la validité d'une transaction dépend d'une information externe, comme un prix symbolique. Dans ce cas, un oracle est nécessaire. Un fait peut être codé pour faire partie de la commande d'une transaction. Un oracle représente un service qui ne confirmera une transaction que si le fait de la commande est vrai.

Les DApp de Corda, ou CorDapps, sont installés au niveau des nœuds de réseau, plutôt que sur le réseau de blockchain lui-même. Les CorDapp sont codés en Java ou Kotlin. Les CorDapp sont codés pour fonctionner sur la plate-forme de Corda. Ceci est réalisé via la définition des flux que les opérateurs de nœuds Corda peuvent invoquer via des appels RPC.

4.1.8 NEO :

NEO est une plate-forme de smart contract qui propose des smart contracts efficaces et peu coûteux. Les smart contracts peuvent être codés à l'aide d'une myriade de langages de programmation, notamment C #, F #, Java, Python, VB.Net et Kotlin. NEO propose des plug-ins et des compilateurs pour toutes ces langues. À l'avenir, la prise en charge de JS, du langage Go, C et C ++ sera implémentée.

Les smart contracts de NEO sont exécutés via la machine virtuelle légère NEO (NeoVM). L'exécution de smart contract via NeoVM consomme un minimum de ressources. La compilation statique des smart contracts et la mise en cache des smart contracts de hotspot peuvent être considérablement améliorées via le compilateur en temps réel JIT. Actuellement, la blockchain de NEO comprend Smart Contract 2.0 qui prend en charge les structures de données et les tableaux complexes. De plus, Smart Contract 2.0 offre une approche évolutive via un partitionnement dynamique et une concurrence élevée, en combinaison avec une conception à faible couplage. La procédure de faible couplage des smart contracts est exécutée dans NeoVM et interagit avec les systèmes hors chaîne via une couche de service interactive. À ce titre, la plupart des mises à niveau de la fonction de contrat intelligent peuvent être réalisées via l'API spéciale de la couche de service interactif.

Dans notre travail nous allons choisir la plate-forme ethereum car elle fournit plus d'outil avec son environnement virtuel EVM, son langage de programmation solidity qui est orienté objet et dédié à l'implémentation des smart contracts, ainsi que les différentes framework.

4.2 Framework Ethereum

4.2.1 Langages des smart contracts :

Tout programme exécuté sur la machine virtuelle Ethereum (EVM) est généralement appelé un smart contract ou contrat autonome. Les langages les plus utilisés pour la rédaction de smart contracts sur Ethereum sont **Solidity** et **Vyper**, bien que d'autres soient en développement.

- **Solidity :** Le langage le plus populaire sur Ethereum, inspiré de C++, Python et JavaScript.
- **Vyper :** Langage axé sur la sécurité pour Ethereum, basé sur Python.

4.2.2 Outils de développement :

Ethereum dispose d'un nombre important et croissant d'outils pour aider les développeurs à créer, tester et déployer leurs applications.

- **Truffle :** Un environnement de développement, une structure de test, un pipeline de construction et d'autres outils ;
- **Embark :** Environnement de développement, Framework de test et autres outils intégrés à Ethereum, IPFS et Whisper ;
- **Waffle :** Un Framework pour le développement et les tests de smart contracts avancés (fondés sur ethers.js) ;
- **Etherlime :** Framework fondé sur Ethers.js pour le développement de Dapps (Solidity & Vyper), déploiement, débogage, tests, entre autres ;
- **Buidler :** Un gestionnaire de tâches pour les développeurs de smart contracts Ethereum ;
- **ZeppelinOS :** Un Framework de développement pour la création de smart contracts évolutifs, et de gestion sécurisée des applications de type smart contract ;

4.2.3 Environnements de développement intégrés (IDE) :

- **Visual Studio Code :** IDE professionnel multiplateformes avec un support officiel d'Ethereum ;
- **Remix :** IDE basé sur le Web avec analyse statique intégrée et une machine virtuelle de test de blockchain ;
- **Superblocs :** IDE basé sur le Web avec une machine virtuelle dans le navigateur, l'intégration de MetaMask, un logger de transactions et d'autres fonctionnalités ;
- **EthFiddle** IDE Web qui vous permet d'écrire, de compiler et de lancer votre smart contract ;

4.2.4 API front-end en JavaScript :

- **Web3.js :** API JavaScript pour Ethereum ;
- **Ethers.js *:*** Implémentation complète d'un portefeuille Ethereum, et utilitaires en JavaScript et TypeScript ;
- **light.js :** Une librairie JS réactive de haut niveau optimisée pour les clients légers ;
- **Web3-wrapper *:*** Alternative Typescript à Web3.js ;

4.2.5 API de back-end :

Infura *:* L'API Ethereum « as a service »

4.2.6 Outils de sécurité :

- **Slither :** Framework d'analyse statique Solidity écrit en Python 3 ;
- **MythX :** API d'analyse de sécurité pour les smart contracts Ethereum ;
- **Manticore :** Une interface en ligne de commande qui utilise un outil d'exécution symbolique sur les smart contracts et les fichiers binaires ;
- **Securify :** Scanner de sécurité pour les smart contracts Ethereum ;

4.2.7 Outils de test :

- **Solidity-Coverage :** Outil alternatif de couverture de code Solidity.
- **Hevm :** Implémentation de l'EVM spécialement conçue pour le test unitaire et le débogage de smart contracts.
- **Whiteblock Genesis :** Une sandbox de test et de développement de bout en bout pour la blockchain.

4.2.8 Explorateurs de block :

Les explorateurs de blocs sont des services qui vous permettent de parcourir la blockchain Ethereum (et ses testnets) en recherchant des informations sur les transactions, les blocs, les contrats et toute autre activité sur la chaîne. Tels qu'Etherscan, Blocskscout, Etherchain.

4.2.9 Testnets et faucets :

La communauté Ethereum gère plusieurs testnets ou réseaux de test. Ceux-ci sont utilisés par les développeurs pour tester leurs applications dans différentes conditions avant de les déployer sur le mainnet, le réseau principal d'Ethereum.

- **Ropsten :** Blockchain en preuve de travail, ether de test pouvant être miné.
- **Rinkeby :** Blockchain en preuve d'autorité (PoA), maintenue par l'équipe de développement de Geth.
- **Goerli :** Blockchain en preuve d'autorité (PoA) multi-clients, construite et maintenue par la communauté Goerli.

4.2.10 Clients (pour faire tourner le nœud) :

Le réseau Ethereum est composé de nombreux nœuds exécutant un logiciel client compatible. La majorité de ces nœuds exécutent Geth ou Parity, chacun pouvant être configuré de différentes manières en fonction de vos besoins.

- **Geth :** Client Ethereum écrits en Go.
- **Parity :** Client Ethereum écrit en Rust.
- **Ethnode :** Faire tourner un nœud Ethereum (Geth ou Parity) pour du développement en local.

4.2.11 Patterns et anti-pattern :

- **DappSys :** Construction de blocs sûrs, simples et flexibles pour smart-contracts.
- **OpenZeppelin :** Librairie pour le développement sécurisé des smart contracts.
- **aragonOS :** Patterns pour l'évolutivité et le contrôle des permissions.

Chapitre 5 : Modélisation des assurances non-vie

5.1 Etape d'une assurance non-vie

Pour souscrire à une assurance non-vie il y a différentes étapes à suivre pour l'assurer auprès de l'assureur qui sont :

- **Informations :**

Ici dans les informations, le client doit fournir des informations sur la situation et les détails de l'assurance auquel il veut souscrire, ensuite ses informations personnelles et le contrat actuel auquel il a souscrit.

- Tarif :

Le montant a payé par l'assuré pour le type de contrat qu'il veut souscrire.

- Paiement :

L'assuré effectue le paiement du contrat.

- Confirmation :

L'assurance enregistre la validation du contrat.

En cas de sinistre l'assuré réclame des indemnisations, voici les étapes à suivre :

- **L'enquête sur le sinistre commence.** Après la déclaration, l'expert en sinistres doit faire enquête afin d'établir le montant des pertes ou des dommages couverts par votre police. Il déterminera aussi qui est responsable du sinistre. Vous pouvez faciliter le processus en lui fournissant des renseignements sur les témoins ou les coordonnées d'autres parties.
- **Votre police est examinée.** Une fois l'enquête terminée, l'expert en sinistres examinera attentivement votre police pour déterminer ce qui est couvert par votre assurance, puis vous informera de toute franchise à payer.
- **Les dommages sont évalués.** Afin d'évaluer l'étendue des dommages avec précision, l'expert en sinistres peut recourir aux services professionnels d'estimateurs, d'ingénieurs ou d'entrepreneurs. Une fois l'évaluation terminée, il vous remettra une liste de fournisseurs privilégiés pour les réparations. Rien ne vous oblige à vous y limiter, mais elle pourrait vous faire économiser du temps de recherche.
- **Les dispositions relatives au paiement sont prises.** Une fois les réparations achevées et les articles endommagés ou perdus remplacés, l'expert en sinistres communiquera avec vous pour parler du règlement de votre demande et du paiement. Le délai de versement du paiement dépendra de la complexité et de la gravité de votre situation.

5.2 Analyse des situations en assurance où la blockchain peut être un outil indispensable

Les contrats intelligents stimuleront un jour l'automatisation du processus d'assurance. Le marché de détail y travaille déjà, mais la technologie changera-t-elle également le fonctionnement de l'assurance commerciale?

5.2.1 Plus rapide et moins cher :

En bref, les contrats intelligents sont des accords entre les parties où les mots sont remplacés par du code, afin qu'ils puissent être automatiquement lus par les ordinateurs. En conséquence, les obligations énoncées dans le contrat sont automatisées et peuvent être auto-exécutées.

Pour les acheteurs d'assurance, cela signifie dépendre moins de l'interprétation des clauses par les experts en sinistres et les avocats, car les indemnités seront payées une fois que des critères objectifs seront remplis. L'ensemble du processus sera ainsi plus rapide et moins cher, ce qui entraînera peut-être une baisse des taux de prime. Il pourrait également aider les compagnies d'assurance à combler les lacunes de couverture en en apprenant davantage sur les risques auxquels sont confrontés les acheteurs d'entreprises.

5.2.2 Souscription améliorée :

Le rapport du Lloyd's[24] souligne l'importance des données collectées par les entreprises qui mettent en œuvre des contrats d'assurance intelligents pour améliorer les processus de souscription.

Les contrats intelligents pourraient également ajuster automatiquement les taux de prime si les conditions rencontrées par l'actif assuré changent.

Par exemple, les navires qui souhaitent réduire les temps de voyage peuvent choisir de payer des tarifs plus élevés pour leur couverture d'assurance maritime et de choisir un itinéraire avec des risques de piratage plus élevés. Le contrat intelligent, dans ce cas, ajusterait automatiquement le taux et ferait le nouveau débit directement dans le compte du client. Et les informations sur les acheteurs eux-mêmes seront plus nombreuses, accélérant l'analyse effectuée par les transporteurs pendant le processus de souscription.

5.2.3 Moins de douleur avec les réclamations :

Pour les acheteurs d'assurance, cependant, la caractéristique la plus attrayante des contrats intelligents est la possibilité alléchante qu'un jour, leurs réclamations soient payées automatiquement sans tous les tracas qu'ils doivent endurer aujourd'hui.

Les contrats intelligents peuvent être liés à des couvertures d'assurance paramétriques, qui promettent déjà de payer les réclamations si certains déclencheurs convenus à l'avance sont respectés.

[24] *https://riskandinsurance.com/4-ways-smart-contracts-can-make-insurance-purchases-easier/*

L'idée est que, une fois la couverture déclenchée, le contrat intelligent procède déjà au paiement de l'indemnisation, sans qu'il soit nécessaire pour les experts en sinistres de se mêler ou même de contacter le transporteur.

5.2.4 Lutter contre la fraude :

Un autre avantage des contrats intelligents pourrait être la réduction de la fraude à l'assurance.

En théorie, ils peuvent être mis en œuvre avec différentes technologies informatiques. Mais la tendance est qu'ils sont intégrés dans des systèmes de registres distribués tels que la blockchain, dont le principe principal est que chaque modification du contrat est enregistrée en permanence. Cela réduit le risque que les participants au contrat tentent quelque chose de drôle et évitent la détection.

De plus, les données des revendications antérieures seront facilement disponibles pour aider à identifier la non-conformité dans le processus de traitement des réclamations qui évolue rapidement.

5.2.5 Analyse de décision spécifique pour l'assurance non-vie :

Cette analyse permet de savoir si la blockchain est une solution appropriée pour un problème défini. Elle ne vise pas à fournir une réponse finale faisant autorité mais à aider les décideurs de haut niveau à évaluer l'opportunité de déployer des ressources dans l'exploration d'une solution basée sur la blockchain à un espace de problème donné et, si oui, à quelle échelle. L'espoir est que le fait de se concentrer sur le problème commercial, et loin d'une solution particulière, atténuera les effets du battage médiatique entourant cette technologie et encouragera une approche pratique tout en réduisant le risque d'expérimentation mal avisée.

L'analyse est composée d'un certain nombre de questions qui aident à définir si une blockchain est la bonne approche pour une entreprise particulière ou non.

- ✓ Pour qu'une blockchain soit une solution appropriée, il est important de comprendre le contexte commercial - le problème commercial nécessite-t-il la suppression d'un intermédiaire? Par exemple, serait-il moins coûteux de collaborer directement avec des fournisseurs / concurrents plutôt que d'utiliser un centre d'échange? Un exemple de ceci est le secteur bancaire utilisant une solution telle que CORDA pour gérer les envois de fonds entre eux; cela leur permet de fournir des services plus rapidement, en toute sécurité et à moindre coût qu'avec les technologies existantes. Pour ce faire, ils redéfinissent la manière dont les processus métier sont mis en œuvre dans leur secteur. Un autre exemple peut être la suppression de courtiers d'une industrie - comme un courtier en technologie ou un courtier d'assurance.

- ✓ Pour que la blockchain soit appliquée avec succès, elle doit fonctionner avec des actifs «natifs numériquement», c'est-à-dire des actifs qui peuvent être représentés avec succès dans un format numérique. Bien que cela puisse sembler complexe, c'est en fait relativement simple. Si un actif a une représentation physique qui peut changer de forme, il est difficile de gérer efficacement cet actif sur une blockchain. Un exemple de cela est le suivi et le traçage des aliments sur la blockchain - si une entreprise souhaite suivre et tracer le blé sur toute la chaîne d'approvisionnement au fur et à mesure qu'il devient du pain, il est difficile d'utiliser la blockchain pour gérer sa transition du blé, de la farine, du pain.

- ✓ Un enregistrement permanent peut-il être créé pour l'actif numérique en question? C'est peut-être la question la plus critique, car une blockchain doit être la source de confiance. S'il existe plusieurs sources de confiance concernant l'état d'un objet, l'objet ne peut pas être stocké efficacement sur la blockchain. Dans les cas où un enregistrement permanent peut être créé, il est important que toutes les parties responsables de l'état de l'actif numérique en question conviennent de la manière dont cet état sera géré / géré dans le nouveau processus opérationnel avant tout développement. Séparément, un dossier permanent est-il souhaitable? Si un enregistrement inaltérable est superflu ou contre-productif, par exemple, dans une situation où la nécessité de supprimer des informations est critique, alors la blockchain / DLT (Distributed Ledger Technology) n'est pas une solution appropriée. Il ne serait pas logique, par exemple, de stocker une liste d'épicerie ordinaire sur une blockchain.

- ✓ À ce stade, il convient également d'évaluer la vitesse requise pour le processus opérationnel en question. Si cela nécessite des performances en millisecondes sur les transactions, les chaînes de blocs ne sont pas encore en mesure de gérer cela efficacement et il est conseillé de travailler avec les technologies existantes ou d'attendre que les chaînes de blocs puissent gérer ces vitesses de transaction. Depuis avril 2018, diverses formes de DLT ont un temps de traitement de deux à 10 minutes.

- ✓ Il n'est actuellement pas conseillé de stocker des données non transactionnelles sur une blockchain. Si cela est nécessaire pour un cas d'utilisation spécifique, il n'est pas conseillé d'utiliser une blockchain. Si, cependant, la confiance en question est liée aux enregistrements de transaction (plutôt qu'aux données sous-jacentes elles-mêmes), alors une blockchain peut être applicable. Dans tous les cas, toute information privée ou toute donnée pouvant être en conflit avec les réglementations locales et mondiales de protection des données, telles que le RGPD[25], ne doivent pas être stockées sur la blockchain.

25

https://fr.wikipedia.org/wiki/R%C3%A8glement_g%C3%A9n%C3%A9ral_sur_la_protection_des_donn%C3%A9es

- ✓ Si une industrie a des exigences spécifiques concernant l'utilisation d'intermédiaires ou de partenaires de confiance, il peut être compliqué de déployer la blockchain, même si d'autres avantages de son utilisation sont facilement apparents. Dans les cas d'utilisation où la réglementation joue un rôle important, il peut être nécessaire d'inclure les régulateurs dans le projet et de fournir des moyens par lesquels les régulateurs peuvent garantir le respect des lois, telles que la législation antitrust et la concurrence. Cet engagement sera un élément essentiel qui doit être abordé dans de nombreuses industries. Un exemple est un secteur qui a des exigences strictes de la part de plusieurs régulateurs, tels que les lois antitrust et environnementales, dont chacune requiert une visibilité sur un aspect différent des données de transaction, et où l'émetteur ne cherche pas à afficher l'intégralité des données de transaction à aucun un régulateur pour des raisons juridiques ou autres. Il pourrait être assez difficile de déployer une blockchain pour cette situation sans engagement réglementaire.

- ✓ Pour que la blockchain aide à réduire les coûts et à fournir une valeur commerciale réelle, il est important qu'une blockchain se penche sur la gestion des transactions autour des actifs numériques - si un problème commercial ne concerne pas vraiment la gestion des relations contractuelles et de l'échange de valeur, alors il n'y a guère besoin d'une blockchain - une technologie différente pourrait probablement résoudre ce problème plus efficacement.

- ✓ Le cas d'utilisation nécessite-t-il un accès en écriture partagé? En d'autres termes, certains / tous les membres du réseau en question doivent-ils pouvoir écrire des transactions dans la blockchain? Si le cas d'utilisation ne nécessite pas un tel accès en écriture partagé, une autre technologie fournira probablement une meilleure solution.

- ✓ Si les acteurs / entités se connaissent déjà et se font confiance, il n'y a probablement pas besoin de blockchain. S'ils ne se connaissent pas ou ne se font pas confiance et / ou ont des intérêts mal alignés, il peut y avoir une bonne raison d'utiliser la blockchain.

- ✓ Si la possibilité de modifier les fonctionnalités d'une blockchain (par exemple, la distribution des nœuds, les autorisations, les règles d'engagement, etc.) sans avoir une discussion détaillée sur les grands forums open source pour la blockchain est souhaitable, alors vous devez sélectionner une blockchain privée et autorisée.

- ✓ Si les transactions doivent rester privées, une blockchain privée autorisée est appropriée. Sinon, une blockchain publique sans autorisation peut être utilisée.[26]

[26] *http://www3.weforum.org/docs/48423_Whether_Blockchain_WP.pdf*

3ème partie : Implémentation de la solution

Chapitre 6 : Présentation des applications existantes

6.1 The Blockchain Insurance Industry Initiative (B3i):

Construit sur l'infrastructure existante fournie par la société de technologie blockchain R3[27], qui repose elle-même sur la plateforme open source Distributed Ledger Technologie (DLT) Corda[28]. Le logiciel B3i permet aux compagnies d'assurance, aux courtiers et aux réassureurs d'établir des accords contraignants visibles par toutes les parties. Il permet également à chaque partie d'entamer un dialogue et de négocier le prix de la réassurance, donnant à chacune la possibilité de décider quelles informations elles rendent publiques à toutes les parties avant la décision du contrat final. Contrairement à une blockchain traditionnelle, il n'y a pas de crypto-monnaie liée au DLT de B3i, et au lieu d'un énorme réseau d'ordinateurs validant chaque transaction - les seules parties impliquées sont celles qui transfèrent activement ou prennent des risques. Selon B3i, les avantages de ceci sont la «certitude du contrat», les «coûts administratifs» et le risque d'erreurs dans le processus de transfert des risques. "Il s'agit d'une orchestration DLT de l'interaction entre les cédantes, qui sont les principaux assureurs cédant leurs risques à un panel de réassureurs via des courtiers", a ajouté Carolin. «Il enregistre chaque étape de la négociation jusqu'à la consolidation des contrats.»

Le principal client de B3i sera les compagnies d'assurance, mais la société souhaite également la commercialiser auprès d'autres acteurs du secteur, comme les start-ups insurtech, qui, selon elle, seraient en mesure de commercialiser leurs propres produits sur la base de la pile technologique de B3i. Carolin a déclaré que le produit Cat XoL est une «mise en œuvre» de la vision de l'entreprise qui montre ce qui est possible, mais pas la vision elle-même. «Nous ne sommes pas une entreprise de produits, nous construisons un système d'exploitation d'assurance conçu de manière à devenir un catalyseur pour permettre à d'autres personnes de s'appuyer sur notre pile technologique», a-t-il ajouté. «C'est une preuve pour nous de montrer que nous sommes capables de fournir ce dont nous avons besoin.» Les plates-formes existantes, notamment Synergy2 d'Eurobase et WebXL d'Effisoft, offrent déjà aux courtiers et aux réassureurs des outils pour suivre leurs activités de transfert de risques. Mais Carolin a déclaré que B3i n'avait pas l'intention de concurrencer des produits comme ceux-ci, mais après avoir prouvé les capacités de sa plate-forme DLT, il espère qu'ils deviendront des clients et s'appuieront sur elle pour améliorer le processus de réassurance.[29]

[27] *https://www.r3.com/*
[28] *https://www.corda.net/*
[29] *https://www.nsinsurance.com/news/b3i-launches-blockchain-based-technology-to-slash-reinsurance-admin-fees/*

6.2 Fizzy:

Fizzy est une police d'assurance retard de vol entièrement automatisée qui s'exécute sur la blockchain Ethereum et permet aux clients d'être indemnisés dès leur arrivée à destination. Le processus est entièrement automatisé, avec un contrat intelligent déterminant si les clients sont éligibles à l'indemnisation. Cela signifie qu'aucune action n'est requise de la part des clients éligibles pour réclamer leur indemnité.

On peut soutenir que Fizzy aurait pu se lancer en utilisant des technologies existantes, telles qu'une base de données centrale fonctionnant en combinaison avec une série d'API. Une telle implémentation fonctionnerait avec la plupart des aspects de la disposition centrée sur le client de Fizzy, tels que les réclamations et les paiements automatiques. Cependant, il manquerait l'occasion de renforcer la confiance dans la relation entre l'assureur et le preneur d'assurance en faisant en sorte que le contrat intelligent, au lieu de l'assureur, décide si le client est indemnisé.

Lorsque les clients paient en Ether, le contrat intelligent décide non seulement qu'un client est indemnisé, mais il effectue également le paiement sur le portefeuille Ether du client. Cela rendra Fizzy encore plus rapide et ajoutera davantage de confiance à la transaction.

Chapitre 7 : Présentation de la solution

Ce référentiel contient le code d'un Smart contrat pour les assurances non vie, codé à l'aide de **Truffle** et **Solidity**. La blockchain backend est une configuration de réseau Ethereum et les interactions sont rendues possibles par la bibliothèque javascript **Web3**.

7.1 Quels problèmes résolvons-nous?

Les agriculteurs sont souvent confrontés aux problèmes de dommages aux cultures en raison de causes naturelles telles que les inondations ou les sécheresses. Les agriculteurs assurés ont du mal à obtenir leurs montants de couverture auprès des compagnies d'assurance. De plus, les compagnies d'assurances jouent le rôle d'intermédiaire dans tout ce processus. Pour supprimer l'intermédiaire et régler immédiatement les réclamations d'assurance de l'utilisateur, j'ai eu l'inspiration de développer cette plateforme décentralisée.

7.2 Notre solution au problème:

Agriculchain est un DApp qui donne le pouvoir de réclamer légitimement l'assurance aux agriculteurs. Les utilisateurs peuvent créer leur propre politique en spécifiant le type de culture, la superficie et l'emplacement de leur champ. Ils peuvent opter pour une assurance contre les inondations ou la sécheresse. Ils paient la prime requise au contrat intelligent. En cas de sinistre spécifié, les utilisateurs peuvent accéder à la page de réclamation, saisir leur identifiant de stratégie ainsi que la date du sinistre et lancer la transaction à partir de la même adresse Ethereum avec laquelle la stratégie a été créée. Si la réclamation est correcte, un paiement égal au montant de la couverture est initié à leur adresse. Tout cela est géré par le contrat intelligent qui vérifie si l'affirmation est vraie ou non en analysant les niveaux de précipitations à cette date via l'API fournie via le marché Honeycomb et les nœuds oracle hébergés par le réseau Chainlink. Afin de calculer le montant de la couverture, certains paramètres ont été définis dans le contrat. Selon les niveaux d'eau de la région, un paiement de 50% ou 100% est effectué en fonction de l'ampleur des dommages survenus. Les frais de paiement et de prime varient également selon le type de culture sélectionnée et augmentent linéairement avec la superficie du champ. Les utilisateurs peuvent afficher le statut des politiques ainsi que le temps jusqu'à leur validité.

Chapitre 8 : Implémentation

Nous configurons notre environnement de développement pour commencer à coder. Nous installerons toutes les dépendances dont nous avons besoin pour construire notre projet.

8.1 Prérequis

8.1.1 Node Js :

La première dépendance dont vous aurez besoin est Node.js, qui vous donnera Node Package Manager (NPM). Nous utiliserons NPM pour installer d'autres dépendances dans ce tutoriel. Vous pouvez vérifier si vous avez déjà installé Node en exécutant cette commande à partir de votre terminal: **$ node -v** pour voir la version.

On peut l'installer directement depuis son site *https://nodejs.org/en/*

8.1.2 Ganache :

La dépendance suivante est une blockchain de développement, qui peut être utilisée pour imiter le comportement d'une blockchain de production. Nous utiliserons Ganache comme chaîne de développement pour notre travail. Nous pouvons l'utiliser pour développer des smart contracts, créer des applications et exécuter des tests. Trouvez la dernière version de votre système d'exploitation sur *https://www.trufflesuite.com/ganache*.

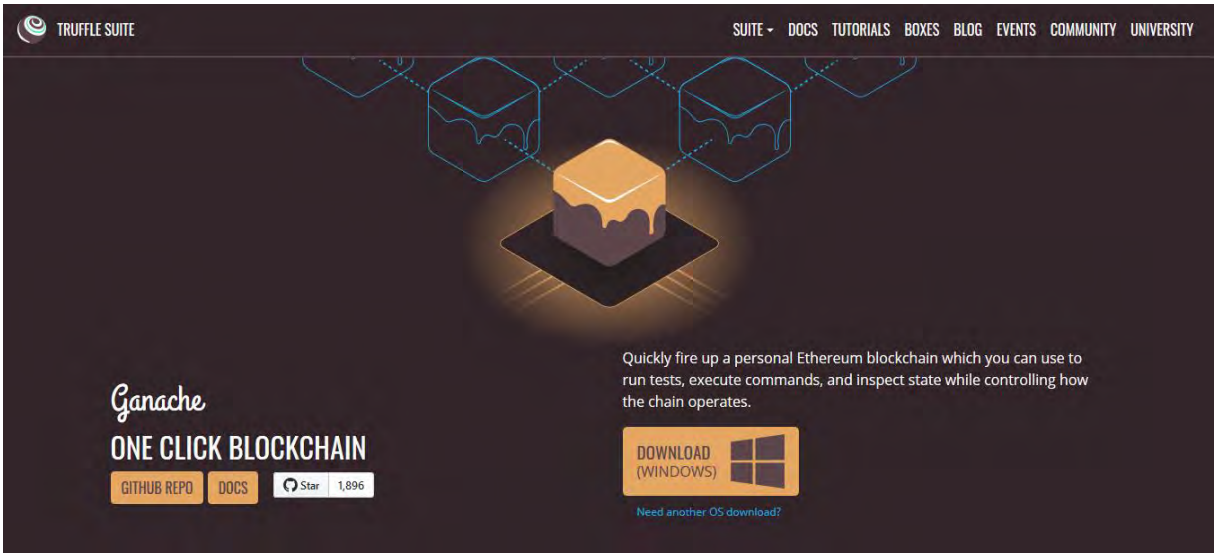


Figure 8 Ganache

8.1.3 Truffle Framework :

La prochaine dépendance est le Truffle Framework, qui nous donne une suite d'outils pour développer des applications blockchain. Cela nous permettra de développer des smart contrats, d'écrire des tests contre eux et de les déployer sur la blockchain.

Installez Truffle à partir de la ligne de commande avec NPM comme ceci (REMARQUE: vous devez utiliser cette version exacte de Truffle pour suivre ce guide):
$ npm install -g truffle@5.0.2

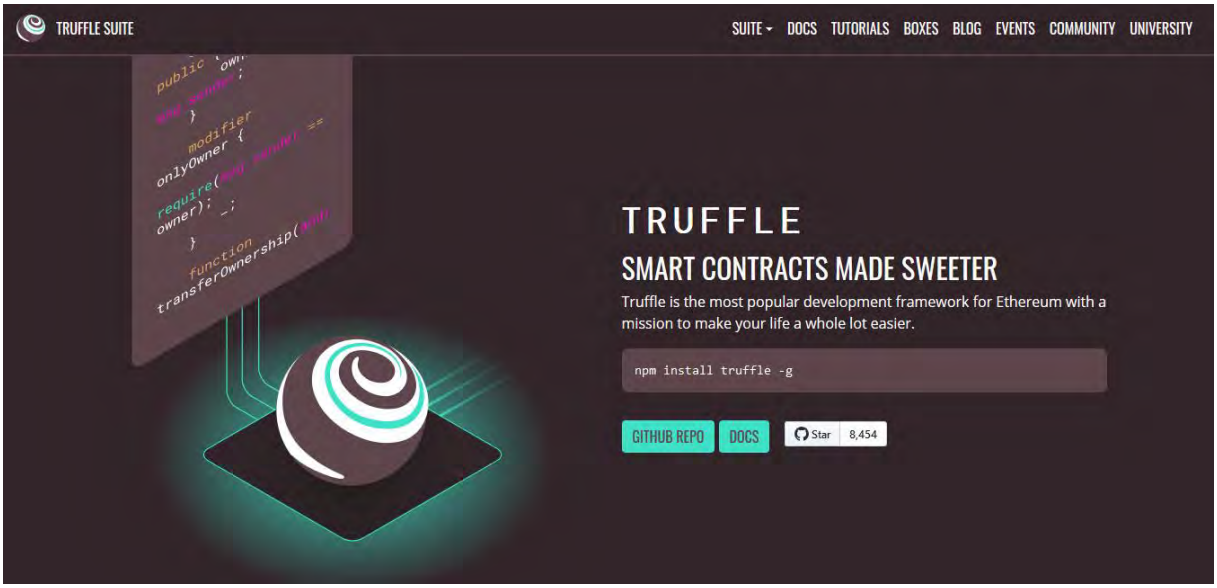


Figure 9 Truffle

8.1.4 Metamask Ethereum Wallet :

Maintenant, installons le portefeuille Metamask Ethereum afin de transformer notre navigateur Web en navigateur de blockchain. Votre navigateur Web actuel ne le prend probablement pas en charge de manière native, nous avons donc besoin de l'extension Metamask pour Mozilla (ou Google chrome) pour vous connecter à la blockchain.

Recherchez-le simplement dans la boutique en ligne de Mozilla. Lors de l'installation, assurez-vous que vous l'avez activé dans votre liste d'extensions Mozilla (il doit être "vérifié"). Lorsqu'il est actif, vous verrez une icône de renard dans le coin supérieur droit de votre navigateur.

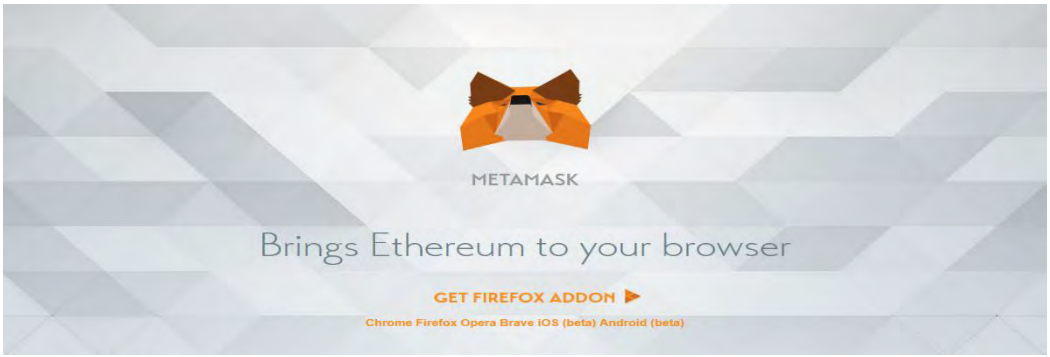


Figure 10 Metamask

8.1.5 Template :

Au lieu de commencer le code à partir de zéro, je vais utiliser une template existante pour m'aider à démarrer. La template est DRIZZLE, on peut cloner ce modèle depuis github comme ceci:

$ git clone https://github.com/truffle-box/drizzle-box.git

Cette boîte contient tout ce dont vous avez besoin pour commencer à utiliser des contrats intelligents à partir d'une application react avec Drizzle. Il comprend des composants drizzle, drizzle-react et drizzle-react-components pour vous donner un aperçu complet des capacités de Drizzle.

8.2 Configuration du projet

Maintenant que nous avons installé toutes les dépendances dont nous avons besoin, construisons notre application blockchain!

8.2.1 Configuration de Ganache :

Après avoir installé Ganache, ouvrez-le. Une fois qu'il est chargé, vous avez une blockchain en cours d'exécution sur votre ordinateur!

- Cliquer sur new workspace

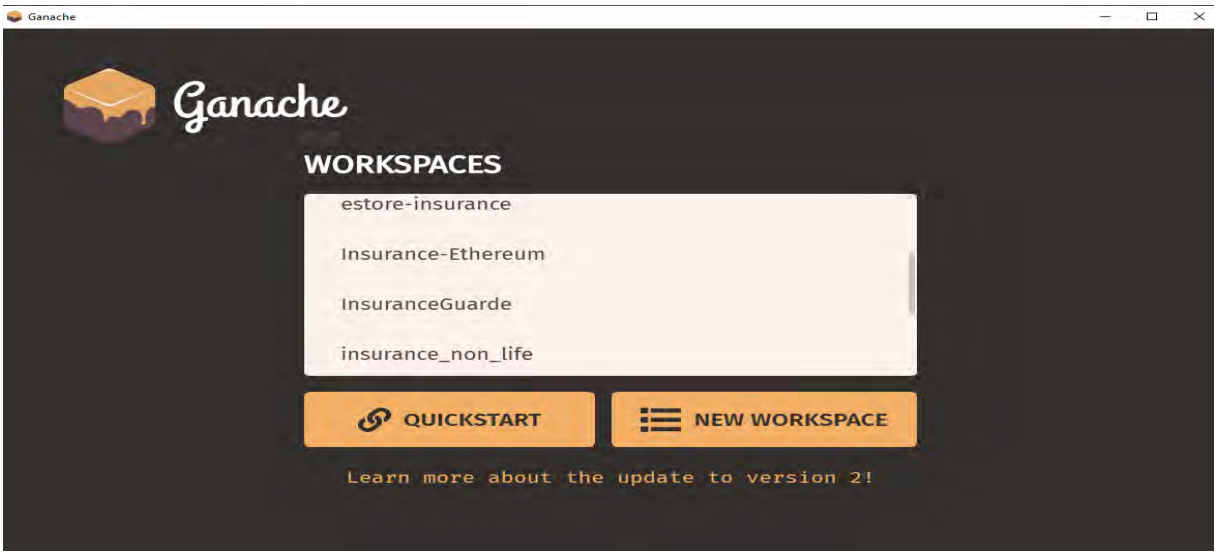


Figure 11 Ganache start.

- Mettre un nom et sélectionner le fichier truffle-config.js dans le projet

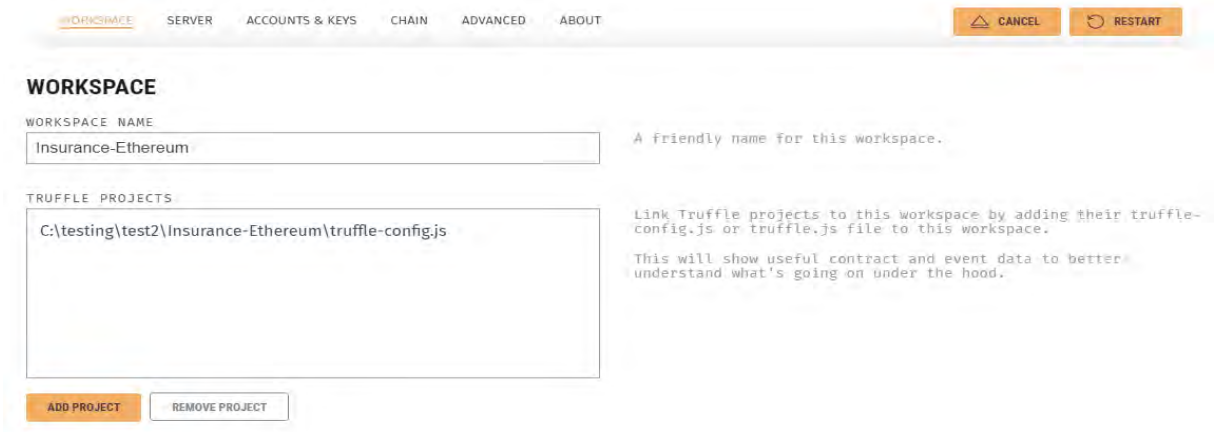
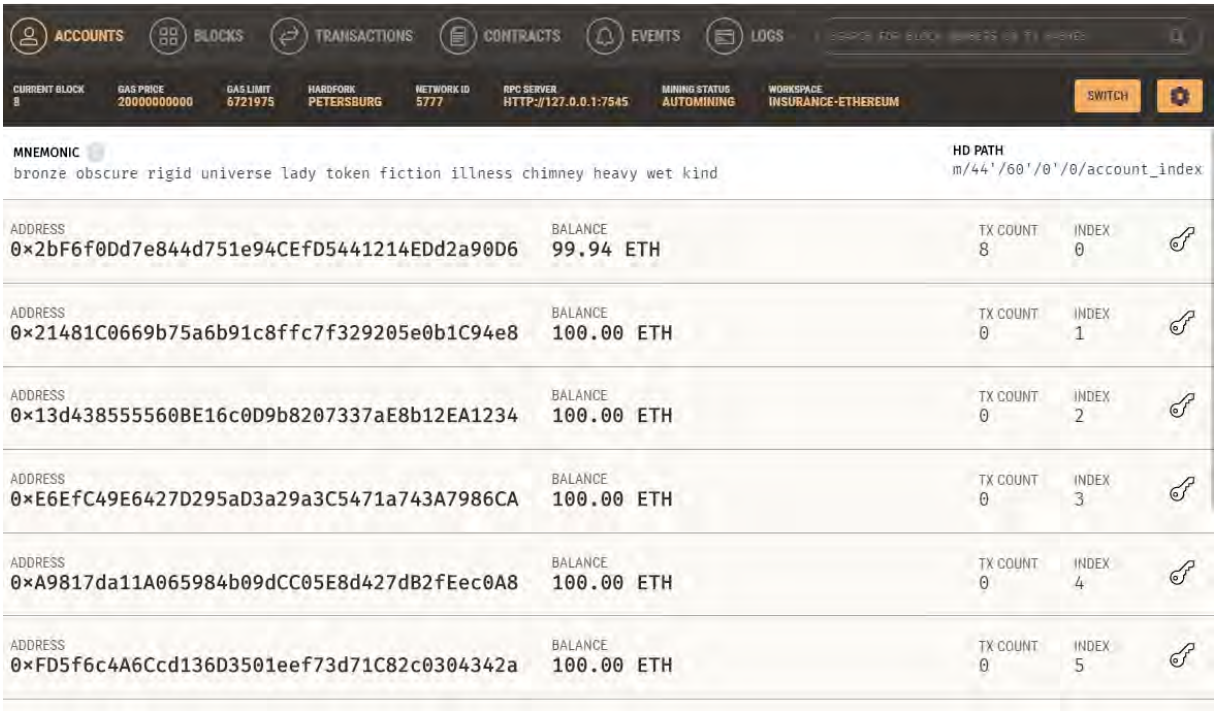
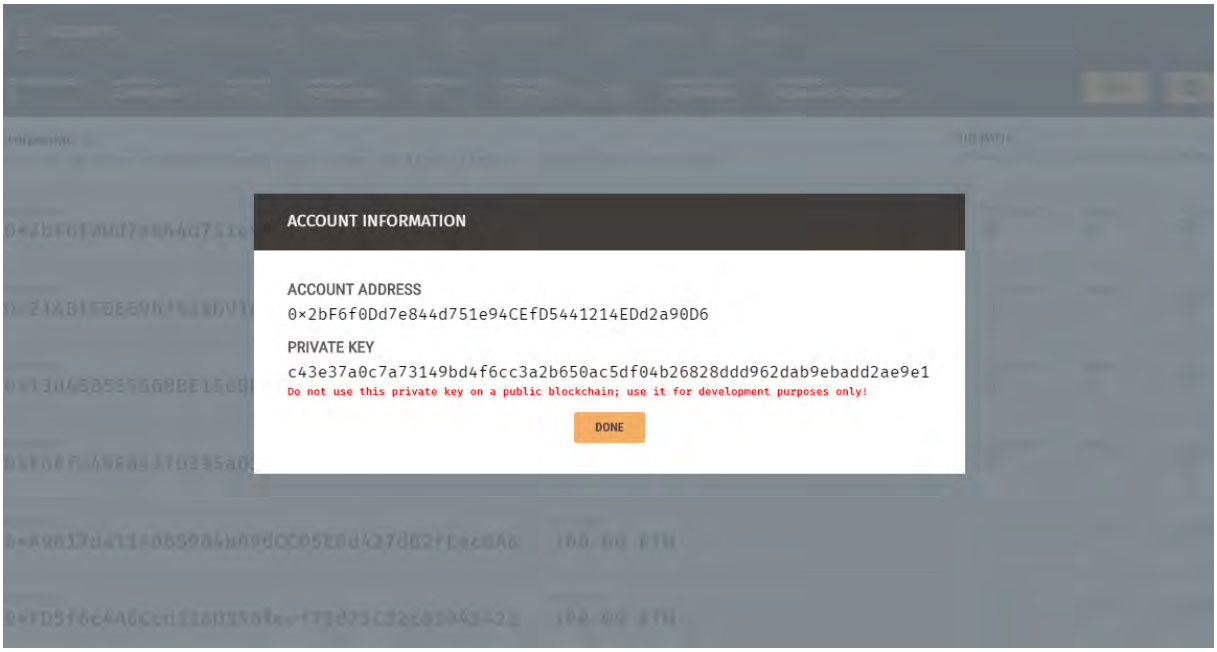


Figure 12 Ganache workspace



Nous avons 10 comptes fournis par Ganache, chacun pré-extrait avec 100 faux Ether (Cet Ether ne vaut pas de l'argent réel).

Figure 13 Ganache Accounts



Chaque compte possède une paire de clés publique et privée unique. Vous pouvez voir l'adresse de chaque compte sur l'écran principal ici. Les comptes ressemblent beaucoup à des "noms d'utilisateur" sur la blockchain et qu'ils représentent chaque utilisateur qui peut publier sur notre réseau social.

Figure 14 Account information

8.2.2 Configuration de Metamask :

Ensuite, nous importons nos comptes de Ganache dans Metamask.

- Pour commencer nous allons connecter Metamask au réseau de Ganache HTTP://127.0.0.1:7545

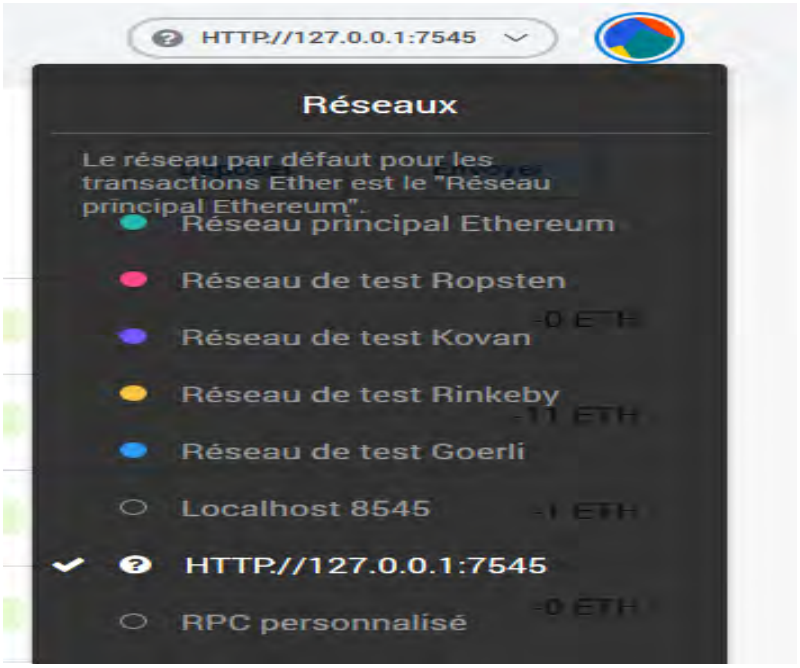


Figure 15 Réseau d'ethereum

- Ensuite Nous allons importer 2 comptes sur Metamask

Pour commencer on clique sur importer un compte

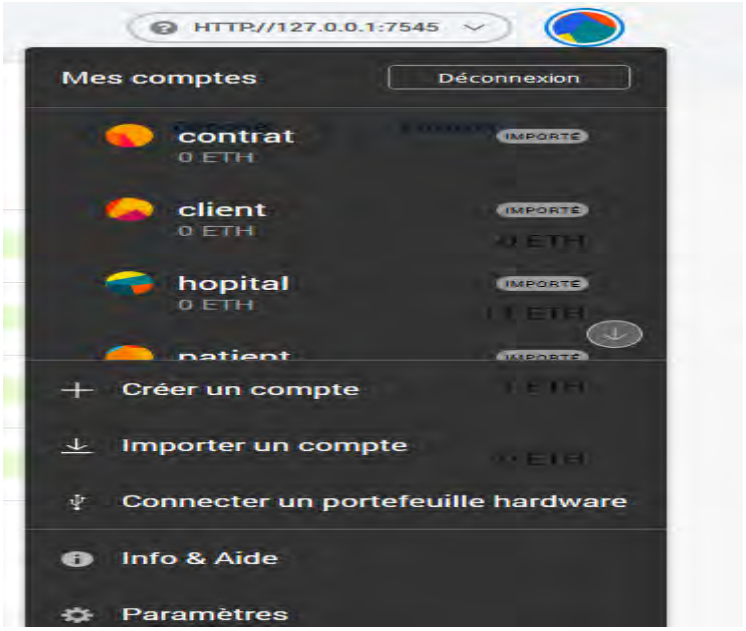


Figure 16 Comptes metamask

Ensuite récupérer la clé privée sur ganache, le coller sur Metamask, et cliquer sur importer

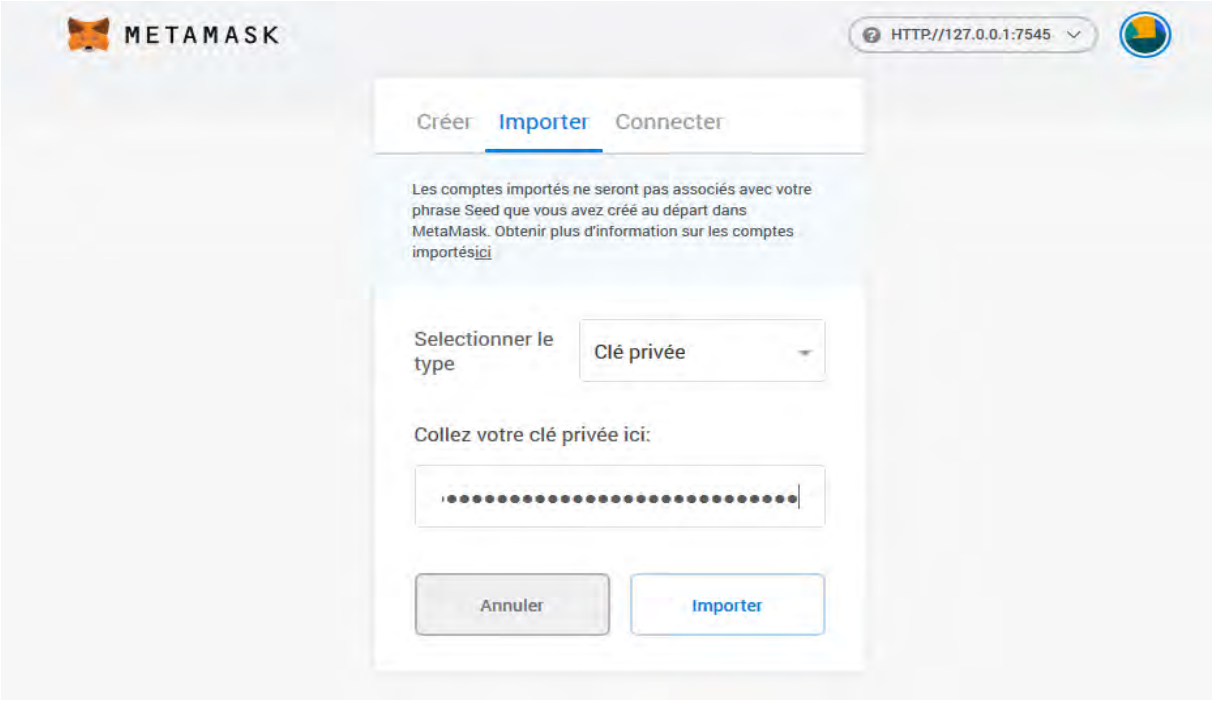


Figure 17 Importation de la clé privéé

Cliquer sur détails

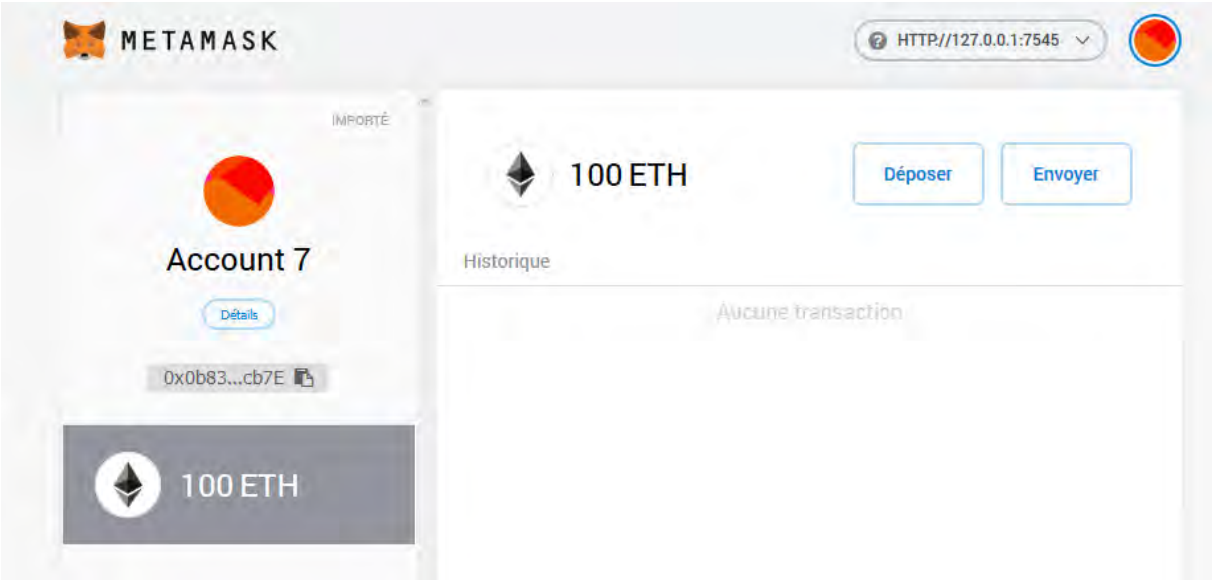


Figure 18 Account

Ensuite cliquer sur account 7 pour changer le nom

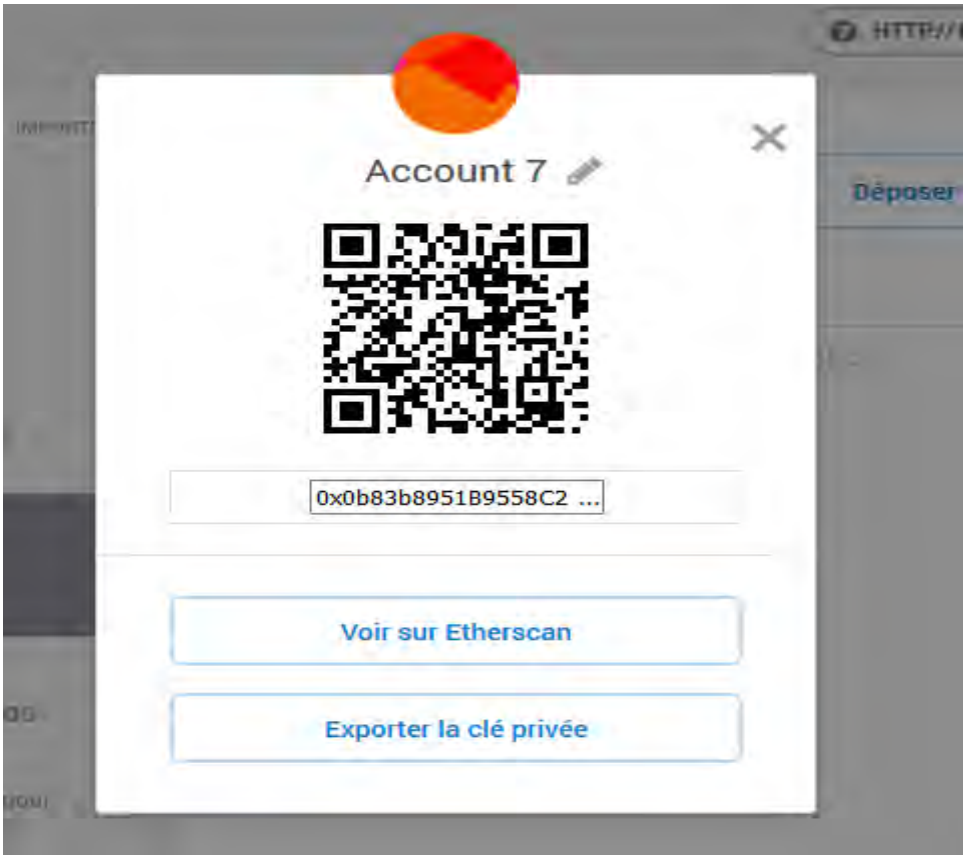


Figure 19 Edit account

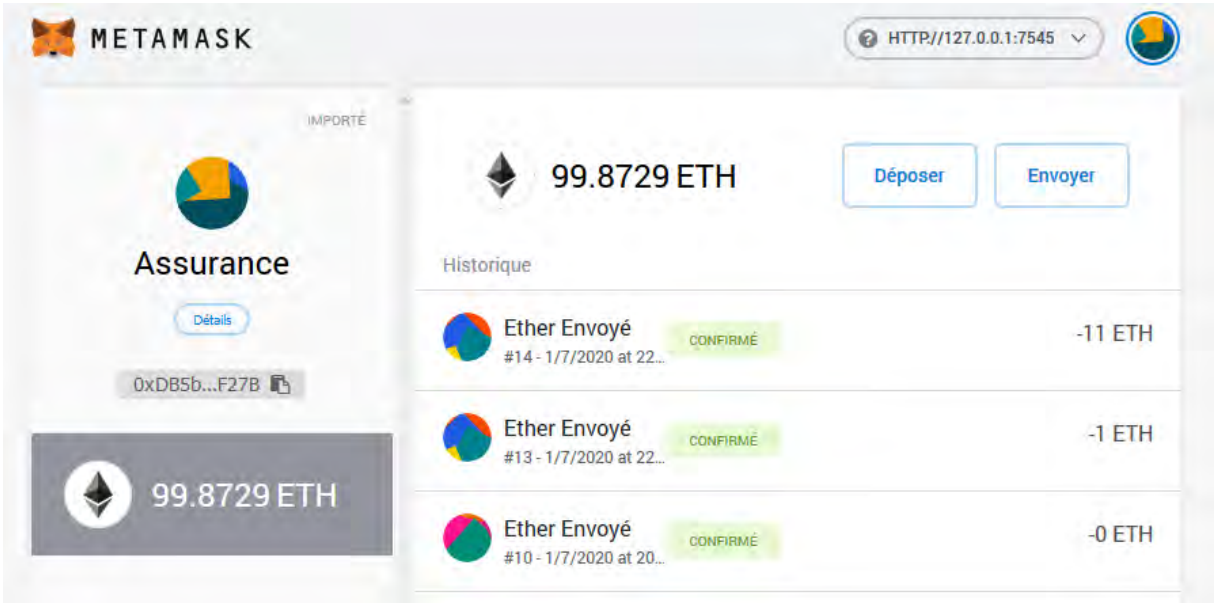


Figure 20 Compte créé

8.2.3 Configuration :

Créons maintenant un nouveau projet pour tout notre code d'application. Au lieu de le faire à partir de zéro, je vais utiliser un modèle que j'ai créé pour nous aider à démarrer rapidement. Vous pouvez cloner ce modèle depuis github comme ceci:

$ git clone https://github.com/truffle-box/webpack-box.git

Maintenant, vous pouvez entrer dans le dossier du projet nouvellement créé comme ceci:

$ Cd social-network (sur PowerShell ou invite de commande)

On ouvre le projet dans l'éditeur de texte de votre choix et recherchez le fichier truffle-config.js. C'est là que nous allons stocker tous les paramètres de notre projet Truffle.

Il est connecté à ganache par le port 7545

Figure 21 Truffle Config

Voici une liste complète de tous les fichiers de notre projet:

- Répertoire des migrations: c'est là que seront les fichiers de migration qui nous permettront de mettre de nouveaux contrats intelligents sur la blockchain.
- Répertoire node_modules: c'est là que toutes nos dépendances sont installées pour le projet.
- Répertoire app: il s'agit du dossier principal de notre site Web côté client
- Répertoire ./app/src: c'est ici que nous développerons tous les composants React.js qui alimentent notre site Web côté client.
- Répertoire contracts: c'est le dossier où nous développerons le code source de nos contrats intelligents avec Solidity.

Ensuite, regardons le fichier package.json.

Ce fichier contient toutes les dépendances du projet dont nous avons besoin pour créer l'application. J'ai inclus toutes les dépendances dont nous avons besoin dans ce modèle de kit de démarrage.

Allons de l'avant et installons ces dépendances comme ceci:

$ npm install

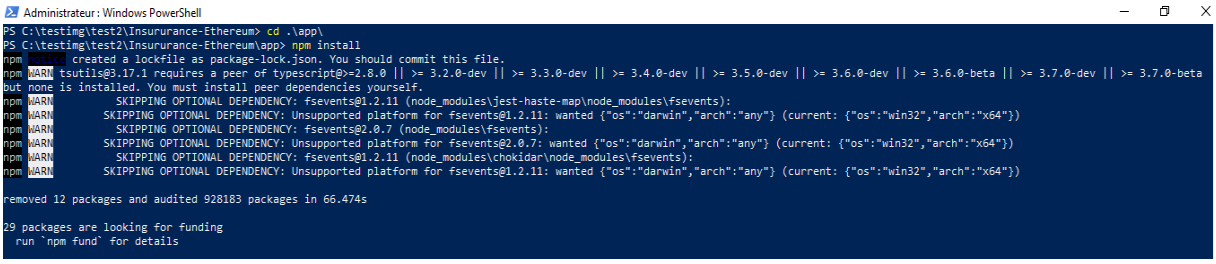


Figure 22 Npm installation

8.3 Smart contract

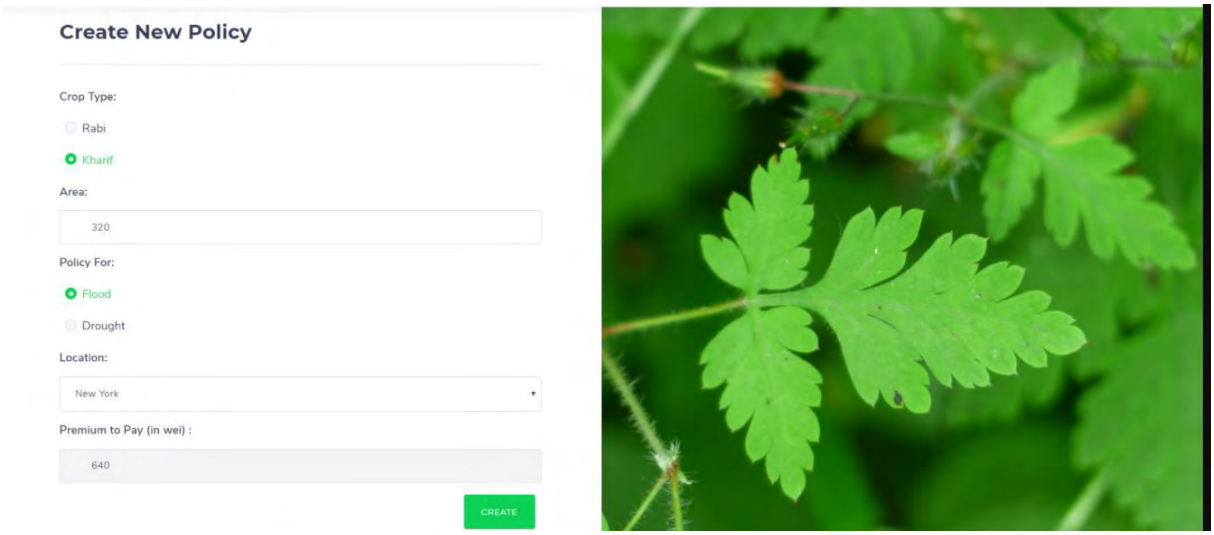


Figure 23 Créer une nouvelle politique

La figure ci-dessus montre comment créez une nouvelle police en payant le montant de prime requis

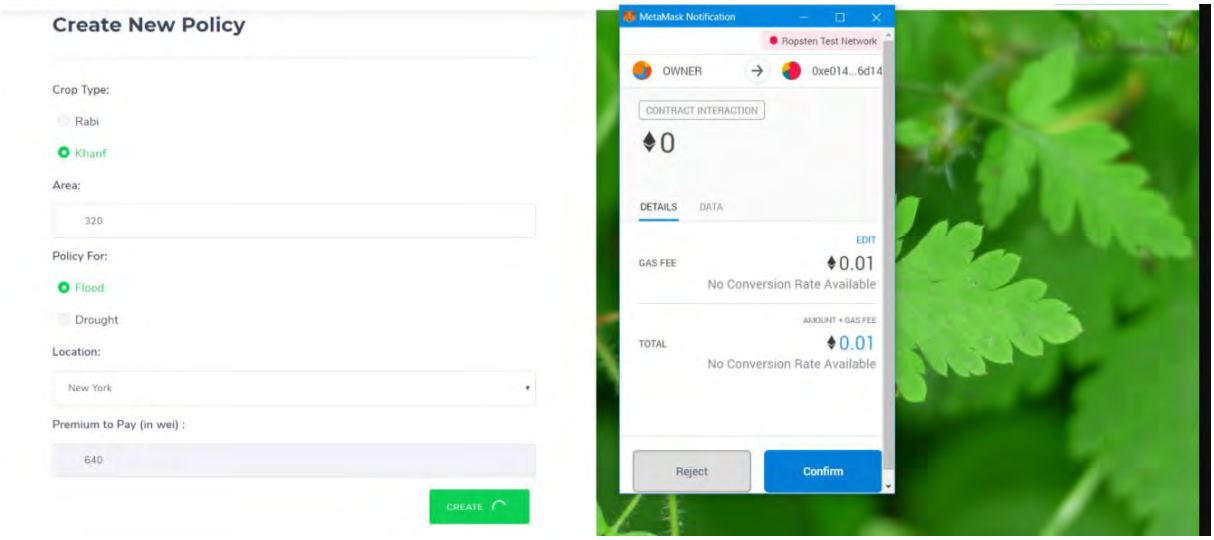


Figure 24 Payez la prime via le portefeuille Ethereum de Metamask.

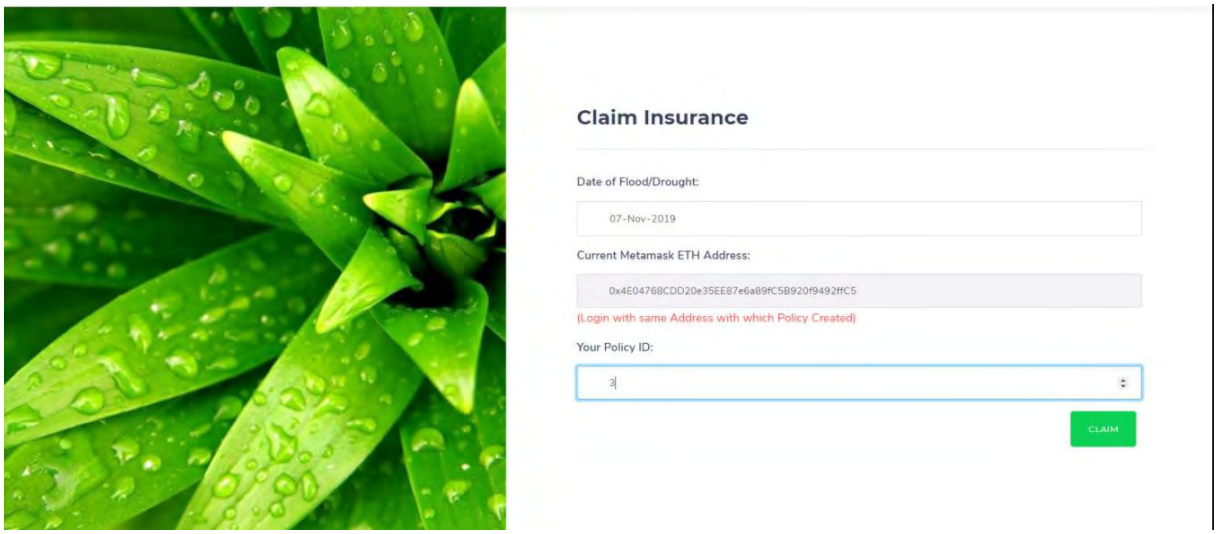


Figure 25 Réclamez une assurance en cas de tragédies comme les inondations ou la sécheresse.

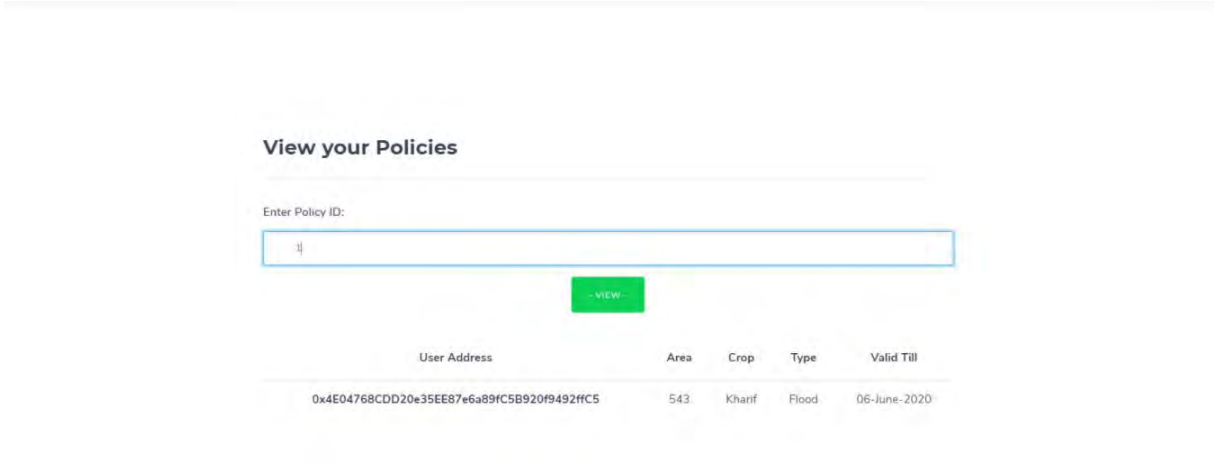


Figure 26 Afficher les détails de votre politique

CONCLUSION

Au terme de ce mémoire, il ressort que la transparence au sein d'un contrat et la sécurité des données au sein de l'assurance sont essentielles pour établir la confiance des utilisateurs. De ce fait, pour assurer le rapport de confiance contrats d'assurance, nous avons mis en place un smart contrat pour les assurances non-vie. Nous savons qu'à l'heure actuelle, la technologie blockchain utilisée par les smart contracts permet de garantir le stockage et de transmission de l'information, la transparence, la sécurité et l'autonomie.

Dans un monde plein de nouvelle technologie et en proie à diverses menaces, fraudes, il est plus sage de se mettre en un niveau de sécurité très élevé, permettant au moins de bénéficier de toutes les possibilités offertes en toute confiance.
La solution implémentée offre la possibilité de disposer d'une assurance non-vie sécurité via un smart contract. Sachant que les smart contracts sont caractérisés par une architecture distribuée et par leur côté de juridiction très important, alors dire que le travail est terminé est une incompréhension. Cependant on peut se poser la question comment vérifier s'il y a eu vraiment sinistre comme l'indique l'utilisateur ? Pour répondre à cette question, dans un futur proche nous allons ajouter de l'intelligence artificielle qui utilisera des cartographies en temps réel pour vérifier s'il y a eu des catastrophes ou non.

Ce travail a été très enrichissant pour nous. Le choix du sujet nous a poussés à multiplier nos recherches. Nous en sommes ressortis fort d'une conséquente plus-value professionnelle.

BIBLIOGRAPHIE

PAAR Christof, PELZL Jan, *Understanding Cryptography: A Textbook for Students and Practitioners*, Allemagne, Springer, 2010, 372 Pages.

SCHNEIER Bruce, *Applied cryptography: protocols, algorithms and source code in C*, Indianapolis, wiley, 1995, 792 Pages.

AUMASSON Jean-Philippe, *Serious cryptography: a practical introduction to modern encryption*, USA, 2017, 312 pages.

Min Surp Rhee (Editor), *Information security and cryptology - ICISC 2006: 9th international conference, Busan, Korea, November 30 - December 1, 2006, proceedings (course notes in computer science)*, Springer, 2006, 376 Pages.

XU Xiwei, WEBER Ingo, STAPLES Mark, *Architecture for Blockchain Applications*, Swiss, Springer, 2019, 3017 Pages

WEBOGRAPHIE

https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf National Institute of Standards and Technology Internal Report 8202 66 pages (October 2018). Consulté le 16 Novembre 2019

https://bitcoin.org/bitcoin.pdf Nakamoto, S., "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. Consulté le 6 Décembre 2019

https://doi.org/10.6028/NIST.FIPS.202 National Institute of Standards and Technology, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, Federal Information Processing Standards (FIPS) Publication 202, August 2015. Consulté le 15 Septembre 2019

https://doi.org/10.6028/NIST.FIPS.186-4 National Institute of Standards and Technology (NIST), Digital Signature Standard, Federal Information Processing Standards (FIPS) Publication 186-4, July 2013. Consulté le 28 Septembre 2019

https://www.dappuniversity.com/articles/blockchain-tutorial#part2 Gregory McCubbin · January 09, 2020. Consulté le 30 Septembre 2019

www.blockchain.com/charts/blocks-size, "BlockchainSize."Blockchain.com. Accessed July 19, 2018. Consulté le 10 Novembre 2019

https://doi.org/10.6028/NIST.IR.8105 Chen, L., Jordan, S., Liu, Y.-K., Moody, D., Peralta, R., Perlner, R., and Smith-Tone, D., Report on Post-Quantum Cryptography, National Institute of Standards and Technology Internal Report (NISTIR) 8105, April 2016. Consulté le 11 Novembre 2019

http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html, Szabo, N. "Smart Contracts," 1994. Consulté le 27 Septembre 2019

https://en.bitcoin.it/wiki/Majority_attack, "Majority Attack."Bitcoin Wiki. Consulté le 01 Janvier 2020

https://medium.com/quillhash/life-cycle-of-smart-contract-development-8929fa073b7f Abhishek Sharma Apr 2, 2019. Consulté le 05 Janvier 2020

https://medium.com/@nilscambreleng/comment-cr%C3%A9er-un-smart-contract-ethereum-simplement-6ec8eeb031f0 nils cambreleng Dec 18, 2018. Consulté le 01 Décembre 2019

https://medium.com/coinmonks/developing-ethereum-smart-contracts-ef36ee4574c0 Daniel Amores Oct 19, 2018. Consulté le 10 Décembre 2019

TABLES DE MATIERES

DEDICACES ..... **i**
REMERCIEMENTS ..... **ii**
RESUMÉ ..... **iii**
AVANT PROPOS ..... **iv**
SIGLES ET ABRÉVIATIONS ..... **v**
LISTE DES FIGURES ..... **vi**
LISTE DES TABLEAUX ..... **vii**
SOMMAIRE **1**
INTRODUCTION GÉNÉRALE **2**
1ère partie : Cadre de référence et méthodologique **4**
Chapitre 1 : Cadre méthodologique **5**
1.1 Présentation des assurances Non-Vie 5
1.2 Définition du besoin 6
1.3 Importance de la question 6
1.4 Formulation des objectifs 6
Chapitre 2 : Background **8**
2.1 Architecture de la blockchain 8
2.1.1 Fonctions de Hachage Cryptographique : 8
2.1.2 Transaction : 11
2.1.3 Cryptographie à clé asymétrique : 13
2.1.4 Adresses et dérivation d'adresses : 14
2.1.5 Grands livres : 16
2.1.6 Blocs : 18
2.1.7 Chaînage de blocs : 19
2.2 Consensus 20
2.2.1 Modèle de consensus sur la preuve de travail : 21
2.2.2 Modèle de consensus de preuve de participation : 23
2.2.3 Modèle de consensus Round Robin : 25
2.2.4 Modèle consensuel de preuve d'autorité / de preuve d'identité : 25
2.2.5 Modèle de consensus sur la preuve du temps écoulé : 26
2.2.6 Conflits et résolutions du grand livre : 26
2.3 Forking 29
2.3.1 Soft Forks : 29

2.3.2 Hard Forks : 30
2.3.3 Changements cryptographiques et forks : 30
2.4 Catégorisation de la Blockchain 32
2.4.1 Permission : 32
2.4.2 Sans permission : 33
2.5 Cryptocurrencies 34
2.5.1 Bitcoin (BTC) : 34
2.5.2 Ethereum (ETH) : 34
2.5.3 Ripple (XRP) : 35
2.6 Limitations de Blockchain et idées fausses 36
2.6.1 Immutabilité : 36
2.6.2 Utilisateurs impliqués dans la gouvernance de Blockchain : 36
2.6.3 Blockchain Death : 37
2.6.4 Au-delà du Digital : 37
2.6.5 Cybersecurité : 38
2.6.6 Cyberattaques et attaques réseau : 38
2.6.7 Infrastructure à clé publique et identité : 39
2ème partie : Analyse d'un smart contract 40
Chapitre 3 : Infrastructure des smart contracts 41
3.1 Smart contacts 41
3.2 Cycle de vie d'un smart contract 43
3.2.1 Création : 43
3.2.2 Gel : 44
3.2.4 Exécution : 44
3.2.5 Finalisation : 44
3.3 Modélisation UML d'un smart contract 45
Chapitre 4 : Etude des plateformes 49
4.1 Plateformes d'un smart contract 49
4.1.1 Ethereum : 49
4.1.2 EOS : 50
4.1.3 AION : 51
4.1.4 NEM : 52
4.1.5 Stellar : 52
4.1.6 Hyperledger Fabric (HLF) : 52
4.1.7 Corda : 53

4.1.8 NEO : 53
4.2 Framework Ethereum 54
4.2.1 Langages des smart contracts : 54
4.2.2 Outils de développement : 54
4.2.3 Environnements de développement intégrés (IDE) : 54
4.2.4 API front-end en JavaScript : 55
4.2.5 API de back-end : 55
4.2.6 Outils de sécurité : 55
4.2.7 Outils de test : 55
4.2.8 Explorateurs de block : 55
4.2.9 Testnets et faucets : 56
4.2.10 Clients (pour faire tourner le nœud) : 56
4.2.11 Patterns et anti-pattern : 56
Chapitre 5 : Modélisation des assurances non-vie 57
5.1 Etape d'une assurance non-vie 57
5.2 Analyse des situations en assurance où la blockchain peut être un outil indispensable 58
5.2.1 Plus rapide et moins cher : 58
5.2.2 Souscription améliorée : 58
5.2.3 Moins de douleur avec les réclamations : 58
5.2.4 Lutter contre la fraude : 59
5.2.5 Analyse de décision spécifique pour l'assurance non-vie : 59
3ème partie : Implémentation de la solution 62
Chapitre 6 : Présentation des applications existantes 63
6.1 The Blockchain Insurance Industry Initiative (B3i): 63
6.2 Fizzy: 64
Chapitre 7 : Présentation de la solution 65
7.1 Quels problèmes résolvons-nous? 65
7.2 Notre solution au problème: 65
Chapitre 8 : Implémentation 66
8.1 Prérequis 66
8.1.1 Node Js : 66
8.1.2 Ganache : 66
8.1.3 Truffle Framework : 67
8.1.4 Metamask Ethereum Wallet : 67

8.1.5 Template : 68
8.2 Configuration du projet 69
8.2.1 Configuration de Ganache : 69
8.2.2 Configuration de Metamask : 71
8.2.3 Configuration : 74
8.3 Smart contract 76
CONCLUSION 78
BIBLIOGRAPHIE 79
WEBOGRAPHIE 80