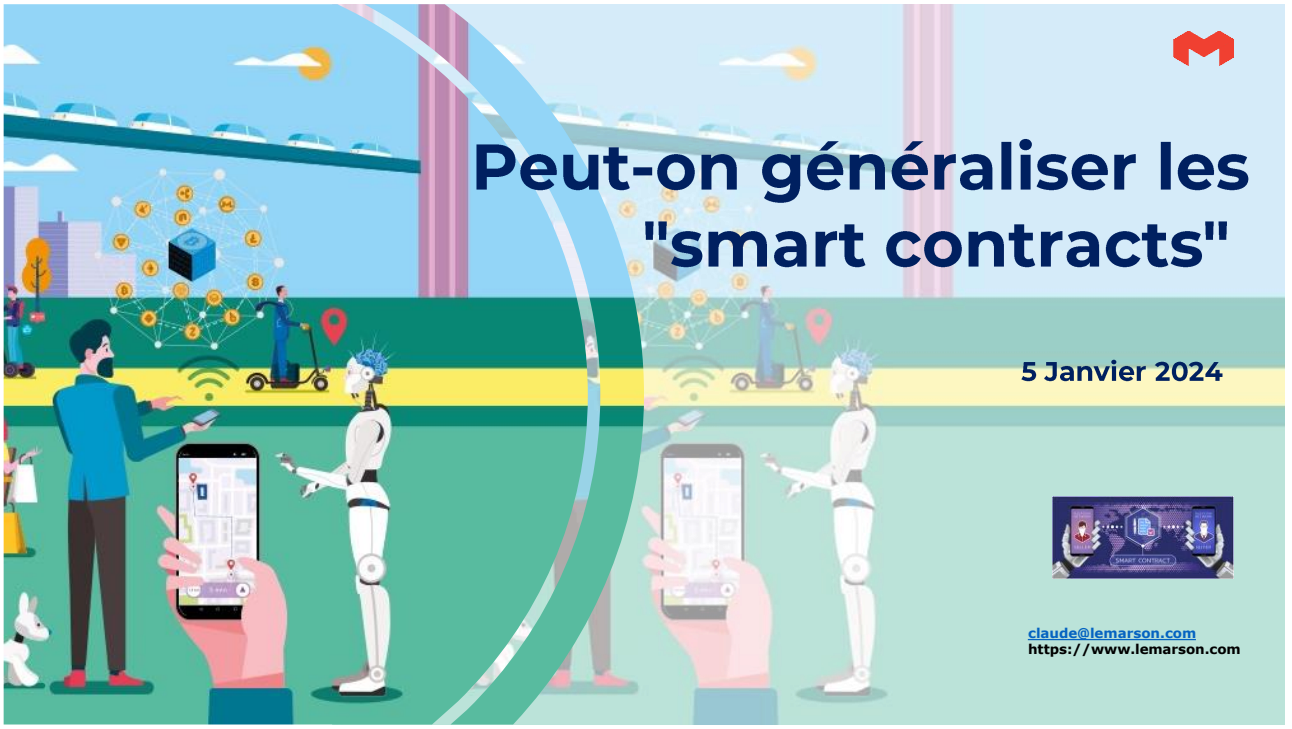


Peut-on généraliser les "smart contracts"

5 Janvier 2024

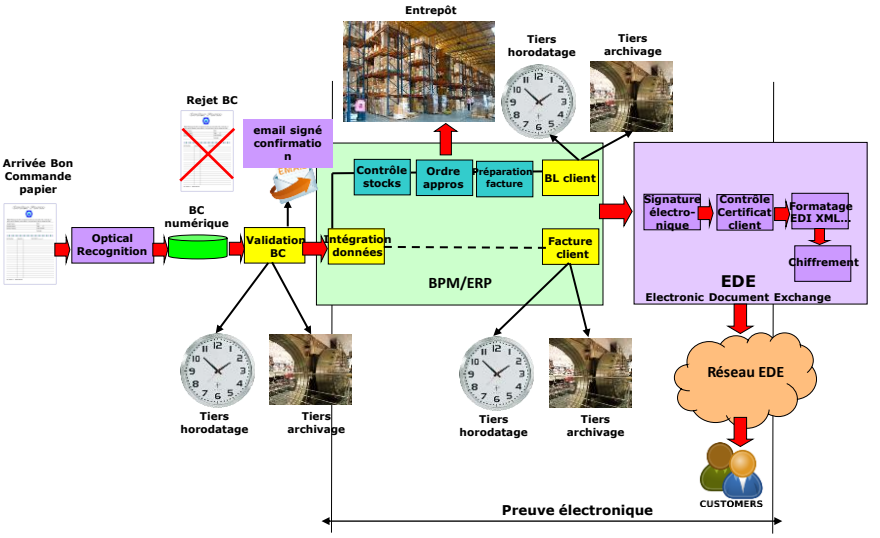
claude@lemarson.com
https://www.lemarson.com

SOMMAIRE

Les "smart contracts" nécessitent de repenser la "propriété"

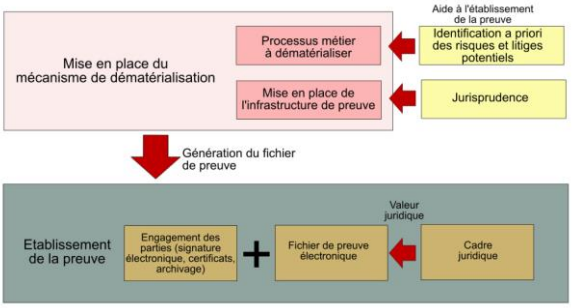
Le marché global des contrats intelligents était de 1,6 G$ en 2022. Il est attendu à 8,3 G$ en 2032, avec un CAGR de 21,4 %. Selon Market Research Future.

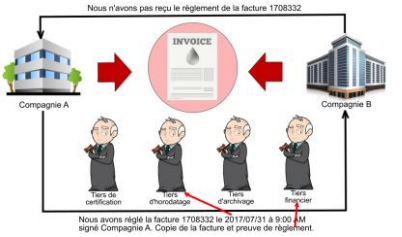
Le monde de la dématérialisation



L'échec patenté de la dématérialisation

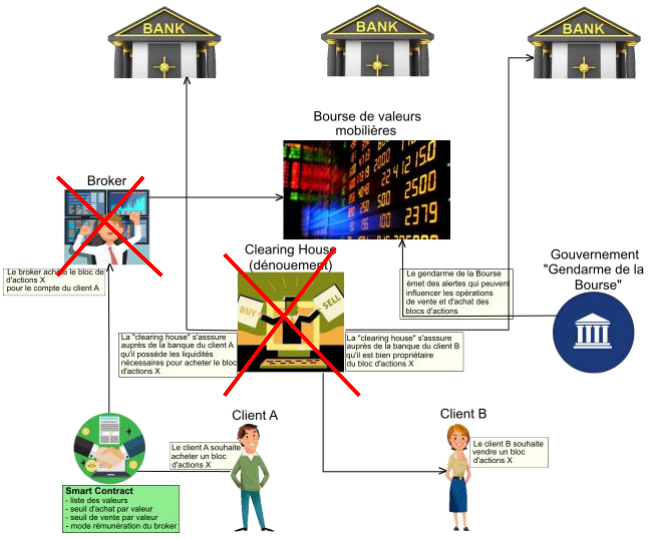
- Malgré les "rodomontades" des fournisseurs, la dématérialisation est un échec.
- En France, 36 % des processus sont dématérialisés, généralement imposés par des structures gouvernementales.
- Dans les entreprises, ce sont le plus souvent des procédures de gré à grté, qui n'ont pas de caractère générique.
- Les raisons :
  - Pour beaucoup d'usagers, le procédé n'est pas naturel.
  - Manque de fiabilité supposé (!!!)
  - Grandes difficultés pour prouver sa bonne foi, en cas de litige.
  - Pas de standards clairs et ebXML est un échec, sauf pour la transmission de fichiers de référence.





La grande question

- La dématérialisation a pour objet de remplacer les processus humains, fondés sur des supports matériels (papier), par des processus entièrement numériques.
- Le Web3 et la blockchain vont plus loin : supprimer toute autorité centralisée, pour éviter les dérives autocratiques, tout en fiabilisant les opérations.
  - De même qu'un algorithme pilotera mieux un avion qu'un commandant de bord, un noyau "blockchainisé" sera plus efficace que des entités intermédiaires, des "greffons" plus que des éléments nécessaires.
- Il faut savoir si c'est possible :
  - Techniquement avec le Web3.
  - Dans les têtes, tant les réticences sont grandes et les habitudes ancrées de ne faire confiance qu'aux êtres humains.
- Les "smart contracts" ou contrats intelligents ont pour vocation de programmer les processus à dématérialiser les contrats et les conditions dans lesquelles le processus auquel se réfère un contrat peut être numérisé de bout en bout.
- L'autre interrogation est de savoir si les contrats intelligents peuvent s'étendre au-delà du contexte d'Ethereum et des monnaies cryptographiques, dans lequel ils sont nés.
- Peuvent-ils servir à la dématérialisation.



L'exemple du dénouement des achats/ventes d'actions sur une place de marché, avec ou sans l'aide des sociétés de bourse et d'un organisme central.

L'exemple d'ebXML

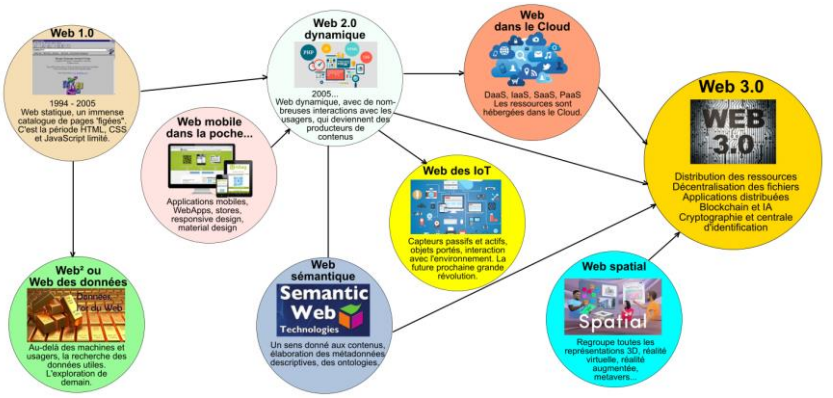
- Méthodologie et infrastructure de support des échanges électroniques.
- Standard d'échange d'informations commerciales et financières basés sur XML (OASIS et UN/CEFACT, spécialiste des échanges de données)
  - Très adapté aux petits échanges entre partenaires épisodiques comme les PME.
  - Mais aussi aux gros volumes entre partenaires stables.
    Similaire ou concurrent d'EDIFACT (United Nations Electronic Data Interchange For Administration, Commerce and Transport). Dans la pratique, il est plutôt complémentaire.
- Une entreprise X peut ainsi découvrir l'existence et les caractéristiques de son interlocuteur Y grâce au Registre (sorte de mode d'emploi).
- Les fonctions ebXML semblaient utiles et adaptées :
  - Renseigner le Registre ebXML : un vendeur potentiel apporte la description de son Profil d'affaires (capacités et contraintes du système dans le Registre.
  - Consulter le Registre (entreprise X, client potentiel). L'entreprise peut découvrir les scénarios d'affaires supportés par Y et la façon de communiquer avec lui.
  - Notifier une demande de relation.
  - Répondre à une demande : Y répond par une proposition d'échange. L'accord sous-entend que les deux sociétés s'engagent mutuellement sur les scénarios d'affaires.
  - Echanger des données (c'est le but final) : les échanges sont sécurisés et effectués suivant des formats standardisés.
- N'est pas une réussite : utilisé pour formaliser des transferts de fichiers lourds mais pas pour construire des plates-formes ouvertes.

ebXML

Le triptyque des "smart contracts"

- L'architecture des contrats intelligents induit la mise en place d'un triptyque :
- Web3 : des nœuds adressables par leur contenu
- Un transactionnel adapté à Internet, qui peut être Ethereum ou toute autre solution acceptable en termes d'API, de performances et de respect des standards
- Blockchain : portefeuille de données décentralisés, hash et arbre de Merkel…

Le Web3 : évolution logique du Web
(pour s'y retrouver)



Petite distinction sémantique
Web 3.0 : les pages sont liées par leur contenu et pas par leur adresse.
Web3 : web décentralisé, symbolise la lutte contre la toute puissance des GAFAM : il n'y a plus de structure centralisée, c'est une blockchain.

Le Web3 doit faire ses preuves

Distribution des fichiers dans une infrastructure décentralisée où chaque fichier est désigné par son contenu (caractérisé par son hash).

La distribution des applications (les DApps) accompagne celle des fichiers. Dans ce contexte, chaque noeud est susceptible d'exercer un rôle actif dans l'exécution.

La cryptographie est une composante incontournable de Web3, que l'on retrouve dans la Blockchain, les techniques d'authentification et de NFT, les actifs non fongibles.

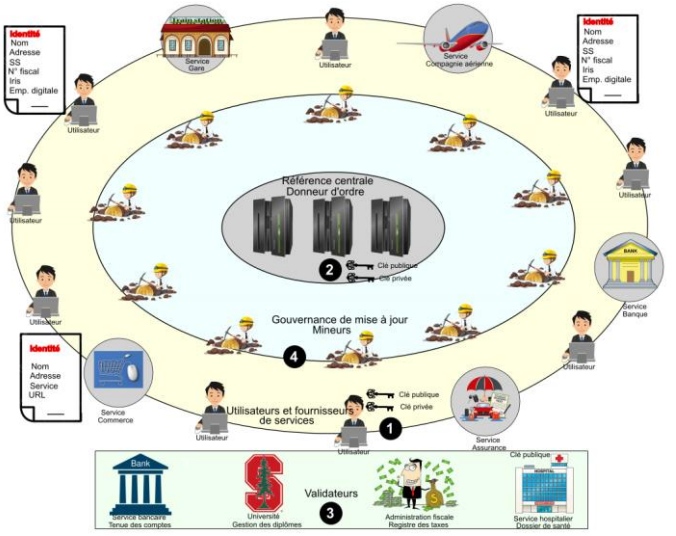
Web3 a généralement besoin d'une centrale d'identification qui puisse attribuer une identité aux différents acteurs susceptibles d'intervenir.

Les Blockchains sont très présentes dans le périmètre des Web3, le plus souvent à travers les "smart contracts" qui régissent les conditions de fonctionnement des acteurs de l'infrastructure.

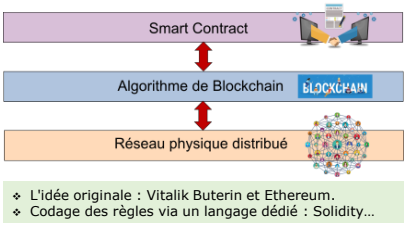
Les constituants de Web3

- La distribution : fichiers et applications (DApp).
- Pour les fichiers, le Web 3.0 s'appuie sur des technologies telles qu'IPFS, qui permettent de les distribuer dans une infrastructure de cluster, puis de les retrouver en se basant non pas sur leur adresse IP, mais sur leur nom (un hash).
  - Les DApps sont exécutées par un ensemble d' "ayants droit", sur un pied d'égalité.
  - Les DApps s'appuient sur une base Ethereum, une sorte de transactionnel privilégié du monde Internet.
  - Ces applications sont réparties sur un grand nombre de serveurs et postes de travail et toute modification doit être approuvée par l'ensemble de la communauté, qui en est "propriétaire".
  - **Une DApp est fondée sur un ou plusieurs "smart contracts", qui décrivent les mécanismes fonctionnels de l'application, avec une interface d'usage, un modèle distribué de stockage, un protocole de communication "peer to peer" et un système décentralisé de résolution de noms.**
  - La gestion des identités est également transversale, indispensable pour les autres briques.
  - On devra identifier un nœud de stockage, un acteur dans une DApp, etc, ce qui nécessite une centrale d'identification, elle-même cliente des techniques cryptographiques.
- La Blockchain est l'un de ses piliers, qui participe à l'ensemble avec certaines de ses briques les plus importantes, les monnaies cryptographiques et les "smart contracts".
- Parmi les applications phares figurent les NFT ("Non Fungible Token"), des actifs numériques non reproductibles, des fichiers, des images, des événements, n'importe quel objet.

Le lien entre Blockchain et "smart contracts



Les "smart contracts" avec la Blockchain

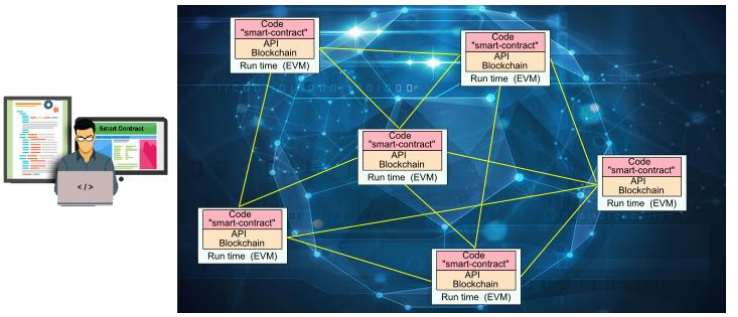


- L'idée originale : Vitalik Buterin et Ethereum.
- Codage des règles via un langage dédié : Solidity…

Contrats classiques	Contrats intelligents
Durée moyenne de plusieurs jours	Durée moyenne en mnutes ou secondes
Traitement manuel des paiements	Traitement automatique des paiements
Obligation de tiers de confiance : horodatage, stockage, conformité, notaires…	Deux cas possibles : avec ou sans tiers de confiance : horodatage, stockage…
Coûts très élevés	Coûts élevés ou réduits
Signature en mode présentiel	Signature numérique

- Le code d'un « smart contract » s'exécute de manière décentralisée dans les nœuds d'une Blockchain, dans un run time, tel que la machine virtuelle EVM d'Ethereum.
- Chaque nœud peut participer à un ou plusieurs contrats et reçoit le code qui les symbolise, « posé » sur la Blockchain.
- L'application est développée via un serveur Web indépendant, sur lequel on effectue les opérations d'édition et de compilation, puis de diffusion.
- Une fois le code écrit et testé sur une plate-forme concrète, il est déployé sur les nœuds concernés par le "smart contract".
  - Opération qui se fait par une simple commande sur la plate-forme de développement.
- Ex de l'outil Truffle, une suite complète, qui se charge, entre-autre, de la compilation solc, l'utilitaire le plus employé dans le monde Ethereum
  - commande *truffle compile*.

Les "smart contracts" avec la Blockchain



- Les "smart contracts" sont une sorte de moteur de règles, auxquelles les acteurs vont se référer dans leur activité.
- La localisation d'un nœud passe par un identifiant : 0xba0b295669a9fd37d5f28d9ec85e40f4cb697bae, par exemple.
- Le code lui-même possède un identifiant.
- Une application est constituée de deux blocs : le code proprement dit et l'API Blockchain, sur laquelle il est posé. C'est cette API qui s'assure de la validité des données véhiculées et donc de la "vérité" portée par les nœuds.
- Deux familles de langages orientés contrats :
  - Dédiés à une blockchain spécifique (Solidity)
  - Ceux dont le code est destiné à une ou plusieurs autres Blockchain.

Le codage d'un "smart contract"

- Solidity : DApp (Distributed Application) sur une plate-forme Ethereum, mais pas limité à la seule monnaie « maison » Ether.
- Le source Solidity est compilé en « byte-code »(solc ou un autre) et présente de grandes similitudes avec un langage objet.
- Le « byte code » Solidity est constitué de deux éléments, le code binaire proprement dit et une ABI (Application Binary Interface), qui décrit l'interface du contrat, avec ses méthodes.
- Le code Solidity s'exécute dans une EVM (Ethereum Virtual Machine). Statiquement typé, héritage, librairies, types complexes définis par l'utilisateur.
- Les functions embarquent la logique du contrat et peuvent être appelés de n'importe quelle partie du code (urbanisation)
- Les modifiers sont des fonctions spéciales qui modifient le contenu d'autres fonctions

```
contract Will { //nom du contrat, comme celui d'une classe
                address owner; //le type address correspond à une adresse de wallet Ethereum
                uint fortune; //fortune est le montant de l'héritage, un entier non signé
                bool isDeceased; //booléen qui indique si le propriétaire est vivant ou décédé
                constructor() public payable { // construction de la classe, instanciation en objet
                // payable est une spécificité de Solidity, permet d'émettre et de recevoir des Ether
 owner = msg.sender; // variable qui indique l'adresse de l'appelant de la fonction
                                fortune = msg.value; // variable qui indique le montant à envoyer
                                isDeceased = false; // par défaut à false
                }
modifier onlyOwner { // les modifiers contiennent la logique conditionnelle
                                require (msg.sender == owner); // exige que la condition soit vraie
                                _; // sorte de branchement
                }
modifier mustBeDeceased { //les modifiers conditionnent l'exécution des fonctions
                                require (isDeceased == true); // la fonction ne peut être appelée qu'à vrai
                                _;
                }
```

Les langages de programmation dédiés

(le paysage change très vite)

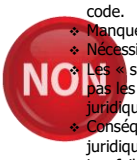
Langages	Spécificités	Compatibilité Blockchains	Similitudes
Solidity	Le plus populaire, spcécifique du monde Ethereum. Le langage est compatible Turing et dispose de ses propres mesures de protection intégrées. Langage objet un peu compliqué, ressemble à JavaScript.	Arbitrum, Avalanche, C-Chain, BNB Chain, Ethereum, Harmony, Hedera Hashgraph, Klaytn, Metis, Moonbeam, Moonriver, Optimism, Polygon, Tron	JavaScript
Vyper	Plutôt destiné aux développeurs qui manquent d'expérience. Ressemble à Python dans l'écriture et compatible avec EVM d'Ethereum. Pas compatible Turing, pas d'héritage. Faible communauté.	Les mêmes que Solidity	Python
Yul	Langage intermédiaire qui supporte EVM (bytecode). Sorte de transpiler. Excellent pour l'optimisation et l'apprentissage.	Les mêmes que Solidity	Solidity
Cairo	Turing complet. Utilisé par la blockchain Starknet (level-2 au-dessus d'Ethereum). Peu répandu en dehors de la communauté.	StarkNet/StarkEx	Python
Rust	Pas compatible EVM. Destiné à d'autres blockchains. N'est pas réservé à Web3. C'est le langage qui monte.	Solana, Polkadot, Cosmos	C, C++
Move	Langage fondé sur Rust pour la blockchain Diem de Meta (Facebook).	Aptos, Sui	Rust

- D'autres langages : Ivy de Chain, un transpiler qui à la compilation génère un script Bitcoin.
- Scilla langage de la plate-forme Zillica, une Blockchain capable de traiter des volumes transactionnels élevés, basée sur le concept de « sharding ».
- D'autres encore : Pact, pour « fabriquer » des contrats sur la plate-forme Kadena, Spedn, pour des contrats Bitcoins, Liquidity pour la Blockchain Tezos, Lity pour la plate-forme CyberMiles.
- Les langages dépréciés : Mutan, qui ressemble à Golang de Google, LLL pour la MV EVM d'Ethereum, Serpent, etc.

Langages conformes Turing : sont capables de formaliser toutes les fonctions prévues par la machine de Turing.

Avantages et Inconvénients

- Un minimum de sécurisation des transactions.
- Amélioration de la confiance, moins d'erreurs humaines et moins de contentieux.
- On se débarrasse de l'ombre des GAFAM…
- Gain de temps avec la suppression des intermédiaires humains et las workflows.
  Réduction des coûts.
- Meilleure gestion des prescriptions légales.
- Respect strict des engagements il n'y a pas d'interprétations, sauf celles prévues par le code.
- Traçabilité des opérations.
- Moins de ruptures injustifiées.
- Archivage plus pratique.



- Caractère spécifique : il peut y avoir autant de « smart contracts » que de contrats concrets.
- La diversité des contrats induit des plates-formes distinctes, des techniques de chiffrement et de signature électronique éventuellement différentes et un codage adapté au problème, via des langages qui ne seront pas toujours les mêmes.
- Il manque un socle générique d'API pour traiter les problèmes en mode « cross-platform », l'idéal étant de ne personnaliser que 20 à 30 % du code.
- Manque de standards clairs.
- Nécessité de formation sur des langages spécifiques… Qui va coder ?
- Les « smart contracts » n'ont pas de valeur juridique et ne remplacent pas les contrats traditionnels. Aucun pays n'a encore actualisé son arsenal juridique.
- Conséquence : l'implémentation d'un « smart contract » ne sera pas juridiquement exécutoire…
- Les faibles protections des contrats intelligents, vis-à-vis des malversations et tentatives de piratage. Ex de "The DAO", avec 3 millions d'Ethers volés.
- Il est difficile d'empêcher des partenaires malveillants d'injecter de fausses informations et évènements, le « smart contract » agissant alors intelligemment… sur des données fausses.
- Difficulté pour arrêter le déroulement d'un contrat actif.
- Droit à l'oubli.

Les applications possibles

Transferts d'argent entre comptes, prêts, traitement des paiements

Assurances

Gestion des polices, accélération des règlements, limitation des litiges

Immobilier

Transparence des transactions immobilières, transferts de propriétés, gestion des locations.

Logistique industrielle

Chaînes d'approvisionnement, traçabilité des produits, automatisation des paiements fournisseurs

Ressources humaines

Les Ressources Humaines sont des clients privilégiés des "smart contracts" pour la paie des employés, le remboursement des frais, le règlement des heures supplémentaires, le respect de critères de performances, qui tous sont fondés sur des règles immuables et contractuelles.

Santé

Sécurisation des échanges de données médicales, validation des contenus de dossiers patients, paiement des prestations

DATA MARKETPLACE

De plus en plus de places de marché de données exploitent les "smart contracts" et la Blochain pour assurer les liens avec les producteurs de données et les usagers. C'est une garantie de sécurité sans perte de propriété.

Vote électronique

Tend vers des élections sécurisées, transparentes et sans fraude.

Propriété intellectuelle

Automatisation des licences, paiement aux "ayant droits". Respect des copyrights et brevets.

identity

La gestion d'identités se dirige vers un usage important des contrats intelligents et des Blockchains. L'exemple d'une identité unique universelle, telle que suggérée par LeMarson est significatif de l'évolution.

Gestion des hypothèques. Les banques exploitent les Blockchains et les "smart contracts" pour valider, enregistrer, suivre et fiabiliser les hypothèques prises sur les emprunts clients.

Avec les "smart contracts", le **"retail"** accélère les transactions, les rend plus fiables et transparentes : paiements, conditions de livraison, remboursements, retours, litiges. Un complément nécessaire dans la course à la numérisation totale (Amazon Go...).

Quelques exemples

IBM Food Trust est une plate-forme de collaboration entre producteurs, grossistes, distributeurs, fabricants et détaillants, construite sur la Blockchain IBM et dédiée à la chaîne alimentaire.

Encrypgen (Indy Gene US AI) a créé une banque de données génétiques anonymes destinée aux chercheurs. 7 000 "donneurs" sont rémunérés via une monnaie cryptographique $DNA propriétaire Encrypgen. Les données sont susceptibles d'être associées à une autre base EHR de données de santé, pour étendre le périmètre de recherche.

Ocean Protocol est une plate-forme de partage de données pour des usagers qui en conservent la propriété. Permet de créer des places de marchés spécifiques pour un type de données et un usage particulier. C'est la couche au-dessus des places de marché, entièrement fondée sur une Blockchain et les "smart contracts".

Arbol, une "Fintech" qui propose sur une base Blockchain Ethereum d'assurer certaines compagnies contre les problèmes climatiques : agriculteurs, compagnies maritimes, producteurs d'énergie, hôpitaux, etc.

Home Depot est un grand distributeur qui s'est basé sur les "smart contracts" pour formaliser ses liens avec les fournisseurs : coût de fabrication des produits, délai entre la réception d'une commande et la livraison, clauses de pénalité et bonus, etc.

Dans le monde immobilier, **Propy** utilise les "smart contracts" depuis 2017 (il a été le premier). Il est spécialisé dans la vente de biens, les contrats étant exécutés automatiquement. Il assure, en particulier, l'aspect administratif du changement de propriétaire.

Mais aussi des échecs...
L'application phare **Fizzi** de la compagnie française AXA, lancée en 2017, a été arrêtée. Elle permettait d'indemniser automatiquement un assuré en cas de vol retardé. Elle n'a convaincu que quelques centaines de voyageurs...

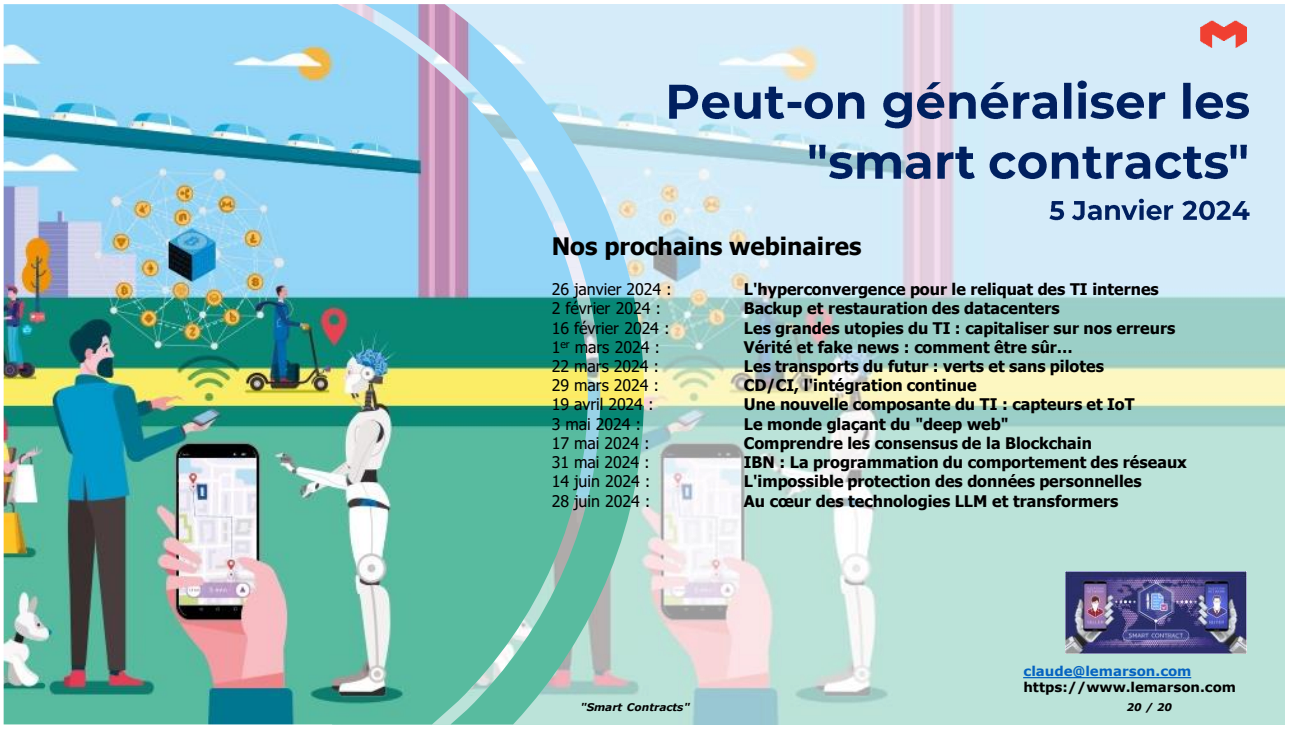
Peut-on sortir du cadre actuel

(oui, à certaines conditions)

- Les "smart contracts" sont intéressants, mais c'est loin d'être gagné.
- Les autres acteurs : Web3, Ethereum, etc, ne font pas l'unanimité.
- Ne pas chercher à faire des blockchains publiques. C'est trop tôt.
- Ne viser que des communautés structurées et limitées, ayant un objectif précis.
- Mais on risque de créer une multitude de "smart contracts", sans généricité.
- Se départir des cryptomonnaies qui sont une monnaie parmi d'autres : l'architecture n'est pas là pour les promouvoir.
- Au minimum, se méfier de l'évaluation "a priori" de certaines cryptomonnaies : bitcoins…
- Ne doit concerner que des données "partageables" et accepter qu'elles soient perçues comme telles :
  - Données financières, y compris comptes courants (ça risque de coincer…)
  - Données de santé
  - Données juridiques
  - Données "personnelles" en général, qui ne le sont plus…
  - C'est un état d'esprit qu'il faut changer, même si les données sont chiffrées.
- Il faut que le sujet en vaille la peine, car ce sont toujours des projets lourds et structurants.
- Ne pas être ébloui par des gains incertains et choisir les bonnes métriques.
- Nécessité d'avoir l'appui du management : ce sont des projets "métiers".
- Compétence particulière qui existe rarement dans les entreprises et chez les consultants.
  - La direction de projet doit être assurée par un utilisateur.
- Investissements élevés.
- Complexité technique.
- Avancer prudemment : POC puis projet à périmètre limité pour commencer.

- Ethereum joue un rôle moteur et il n'y a pas de contre-pouvoir.
- Il est dangereux de confier la maîtrise des contrats au seul Ethereum, même s'il est le plus avancé en, termes de transactionnel sur Internet.
- Ce ne serait plus GAFAM mais GAFAME…

Clés du succès : choix des sujets, prudence, gestion par les utilisateurs…



Peut-on généraliser les "smart contracts"

5 Janvier 2024

Nos prochains webinaires

26 janvier 2024 :	**L'hyperconvergence pour le reliquat des TI internes**
2 février 2024 :	**Backup et restauration des datacenters**
16 février 2024 :	**Les grandes utopies du TI : capitaliser sur nos erreurs**
1er mars 2024 :	**Vérité et fake news : comment être sûr…**
22 mars 2024 :	**Les transports du futur : verts et sans pilotes**
29 mars 2024 :	**CD/CI, l'intégration continue**
19 avril 2024 :	**Une nouvelle composante du TI : capteurs et IoT**
3 mai 2024 :	**Le monde glaçant du "deep web"**
17 mai 2024 :	**Comprendre les consensus de la Blockchain**
31 mai 2024 :	**IBN : La programmation du comportement des réseaux**
14 juin 2024 :	**L'impossible protection des données personnelles**
28 juin 2024 :	**Au cœur des technologies LLM et transformers**

claude@lemarson.com
https://www.lemarson.com