Blockchain

Blockchain :

- Crypto
- Enregistre des transactions
- Décentralisation
- NFT
- Sécurité
- Confiance

Pourquoi et en quoi répond la blockchain :

- Centralisation (Banques)
- Falsification
- Unique information
- Double spending problem (problème de double dépense)

Monnai virtuelle pionnière :

- DigiCash (1990) : Centraliser mais manque d'adoption et transparence
- e-Gold (1996) : or numérique mais hacker
- HashCash (1997) : hacker aussi

Satoshi Nakamoto à créer la première application de la blockchain (BitCoin) en 2008 après la crise de 2008.

Pourquoi est-ce révolutionnaire ?

- **Resolution du double dépence :** chaque transaction est vérifiée et enregistrée dans un registre public
- **Transparence :** toutes les données sont accessibles à tous les participants du réseau
- **Sécurité :** les données enregistrées ne peuvent être modifiées sans consensus

Un algo de consensus est que tous les noeuds d'un réseau se mettent d'accord pour autorisé 1 d'entre-eux pour enregistrer les transactions (le plus rapide à résoudre un calcule mathématique) et est vérifier par tous les autres (s'il les résultats correctes) puis broadcast aux autres.

WEB 1 : 1990 - 2000, Lire Statique
WEB 2 : 2000 - Now, Lire / Écriture Dynamique (plusieurs copy)
WEB 3 : 2014 - Now, Lire / Écriture OWN / transfert de valeur (1 unique file)

Les entreprises privées utilisent des blockchains privés pour évité la diffusion d'information qu'ils ne veulent pas dilvugué à un concurrent.

BDD	Blockchain
Modifiable	Immuable
Centraliser	Décentraliser
Plus rapide	Plus lent
Admin et internes de l'entreprise	tous le monde pour les publics

Hyperledger Foundation est une blockchain privée créer par IBM et Linux fondation.

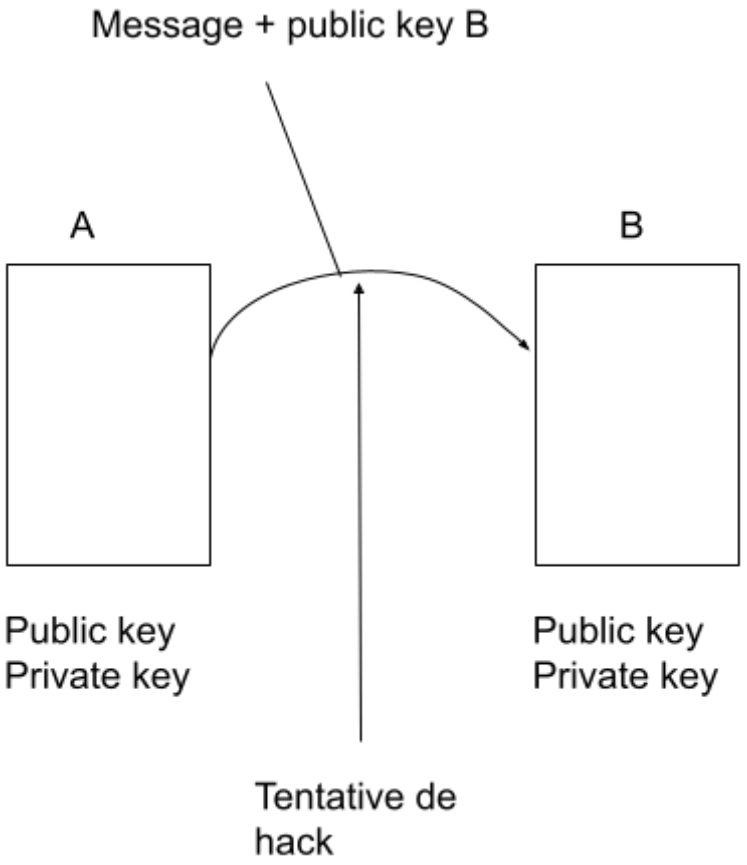
Blockchain consortium est un peu mélange de public et privée.

Hash :

- **Unidirectionnel**
- **Longueur fixe**
- **Sensible aux modifications**

SHA = Secure Hash Algorithm
256 = Une chaine de 256 bits (64 hexadecimal)

Cryptographie symétrique : 1 clé pour l'encrypt et le decrypt
Cryptographie asymétrique : 2 clés, 1 public pour encrypt et 1 privée pour decrypt



Algo de consensus les plus utilisé :

- Proof of Work (PoW) “Travail acharné”
- Proof of Stake (Pos) “Mise en garantie”

Les mineurs prennent les transactions avec le plus grand fee (un fee est généré pour chaque transaction)

Nonce est une variable de 0 à INF
Ce nonce est la solution à trouvé par rapport à la difficulté du hashage.

Si cette difficulté est de 2, il faut que la solution commence avec 2 zéros (00….) dans le HASH.

exemple :

Précédent hash Nonce : 0 Liste de transactions

Algorithme de Consensus

Algo consensus	Blockchain	Fonctionnement	Pros	Cons
Proof of Work (plublic blockchain)	Bitcoin, Litecoin, Doge	Résoudre un problème Math avec puissance de calcul	• Sécurisé +++ • Décentralisé +++	• Énergivore • min - - - Scalable (TPS) 7 tps - - • no fast - - -
Proof of Stake (public blockchain)	Ethereum, Solona *, cardano, ...	Les validateurs doivent résulter leurs crypto pour être sélectionné pour valider le bloc	• Less energy +++ • Fast ++	• Centralisation • less secure
Proof of authority (private blockchain)	VeChain (supply chain), BitGert, Palm Network, Xodex	Choix basé sur l'autorité / réputation	• less enrgy • Fast +++	• Centralisation • Private • Need to trust nodes
Practical Byzantine Fault Tolerance (PBFT) (private blockchain)	Hyperledger Fabric	système de vote	• Sécurité	• Latence / Pas utilisable pour les blockchains publics

- Proof of burn
- Delegated Proof of Stake
- Proof of reputation
- Tendermint

Bitcoin et Blocks

Adresses Bitcoin :

- Cle privrée : une suite de 256 bits
- public : issue de la clé privée via Elliptic Curve Digital Signature Algorithm (ECDSA)
- adresse : hash de la clé publique

Diffusion dans le Réseau :

- réseau bitcoin est P2P (peer-to-peer)
- Chaque noeud relaie ma transaction à ses pairs, qui la stock dans le mempool (si validé)

Validation initiale :

- Noeud vérifier
    - Pas de double dépense
    - signature correctes
    - Frais suffisant (optionnel mais incitatif)

Architecture du réseau :

- pas de server central mais noeud qui est client-server
- Propagation : message (transaction, blocs) se diffuse en mode inondation (gossip protocol)

Mempool :

- BDD local avec transaction non confirmées
- mineurs piochent dans le mempool pour construire un bloc

Fees et priorité :

- les transactions offrent des frais (satoshis / byte ou sat / vByte)
- plus les frais sont élevés, plus la proba d'être incluse rapidement est grande

dans le bloc, il y a 2 parties :

- header : version, timestamp, merkle root, ancienne hash bloc et nonce
- body : transaction avec count transaction et coinbase transaction

On hash que le header du bloc.

Le merkle root est une signature de tous les hash des transactions permettent un hash du header uniquement

en moyenne est de 2-3k de transactions par bloc et environ 1-2 Mo pour la taille d'un bloc.

version	4 octets	version du protocol
hash précédent	32 octets	
merkle root	32 octets	résume toutes les transactions (arbre de Merkle)
timestamp	4 octets	heure UNIX
bits (target / difficulté)	4 octets	Encodage compact de la difficulté
nonce	4 octets	Compteur que le mineur incrémente pour trouver un hash < cible

La difficulté est que le hash doit être inférieur à une cible.

Exercice :

Bloc #700000 :
version : 0x3fffe004
hash précédent :
merkle root :
1f8d213c864bfe9fb0098cecc3165cce407de88413741b0300d56ea0f4ec9c65
timestamp : 11 sept. 2021, 06:14:32
target : 18 415 156 832 118,24
nonce : 2 881 644 503

Smart Contracts

Un contrat intelligent est un programme informatique qui s'exécute auto lorsque des conditions prédéfinies sont remplis, sans nécessiter d'intervention humaine. il est stocké sur une blockchain, garantissant une transparence …

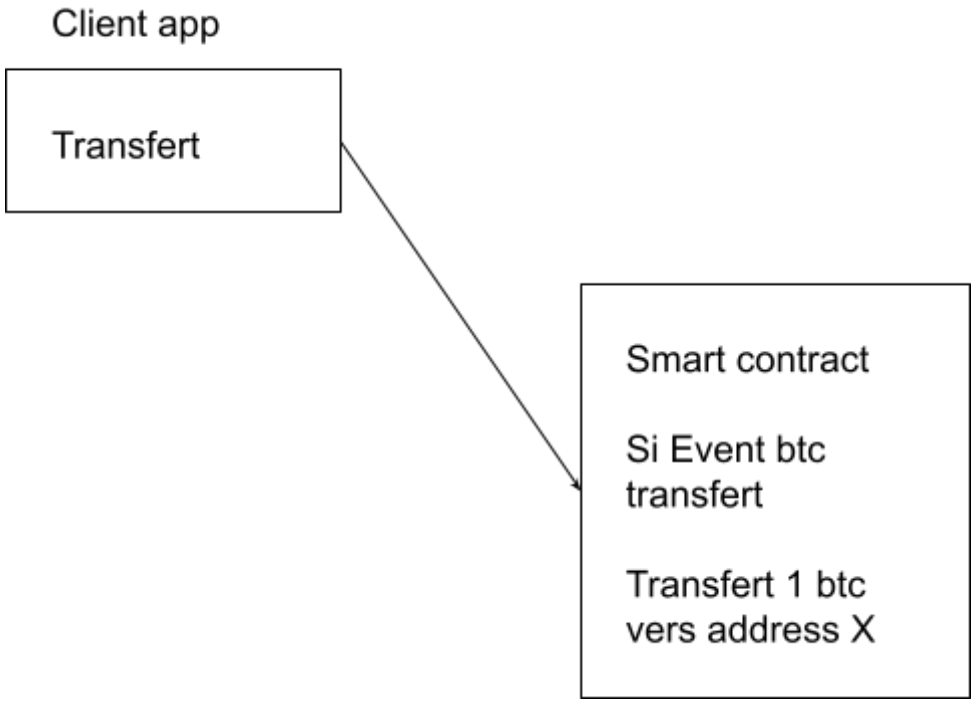
Historique :
Origine :

- Introduit par nick Szabo dans les années 1990 (1994) pour décrire des cp,trats automatisés et auto-exécutoires
- vision d'un système où les termes d'un contrat seraient codés et automatiquement appliqué

évolution :

- Avant blockchain : idées de contrat numérique et automatique
- 2009

- Remplace les contrats traditionnels dans un contexte de blockchain
- Programme autonome sans autorité, qui controle directement les valeurs numériques (actifs numériques), selon des conditions convenues d'un commun accord
- Ressemble à des instructions "si-alors" qui évaluent automatiquement des conditions prédéfinies et effectuent des transactions



Comparaison avec les smart contracts :

4.1 : Contrat traditionnels :

- Processus : Négociation, rédaction, signature et exécution manuelle
- Inconvénients : Risque d'erreurs humaines, délais, coûts liés aux intermédiaires.

4.2 : Contrat Intelligents :

- Processus : Automatisation de l'exécution selon des condition prédéfinies.
- Avantages : Transparence, efficacité, réduction des coûts et des délais.

Critère	Contrat Traditionnel	Contrat intelligents
Automatisation	Faible	Élevée
Transparence	Limitée	Totale
Coûts	Élevés (intermédiaires, frais juridiques)	Réduits (moins d'intermédiaires)
Temps d'exécution	Lent	Rapide
Sécurité	Variable	Renforcée (cryptographie, immuabilité)

☐ Architecture techniques :

- Déploiement : Le contrat est déployé sur la blockchain via une transaction
- Exécution : Déclenchée par des transactions ou appels (appel de fonction)
- Consensus : Les noeuds du réseau valident l'exécution via le mécanisme de consensus (Proof of Work / Stake, etc…)

☐ Acteurs impliqués

- Développeurs : Écrivent le code du contrat intelligent.
- Utilisateur : Interagissent avec le contrat en initiant des transactions
- Mineurs / Validateurs : Valident et enregistrent les transactions sur la blockchain

Fonctionnement Technique des Contrats Intelligents

Langages de Programmations : Les contrats intelligents sont généralement écrits en Solidity, un langage de programmation influencé par JavaScript, Python et C++, conçu spécifiquement pour la plateforme Ethereum.

Compilation et Déploiement : Le code source est compiléen bytecode, qui déployé sur la blockchain Ethereum. Chaque contrat déployé possède une adresse unique sur le réseau, permettant aux utilisateurs d'interagir avec lui.

Machine Virtuelle Ethereum (EVM) : L'EVM est responsable de l'exécution du bytecode des contrats intelligents. Elle garantit que chaque noeud du réseau exécute le contrat de manière identique, assurant ainsi la cohérence et l'intégrité du réseau.

Exemple de contrat :

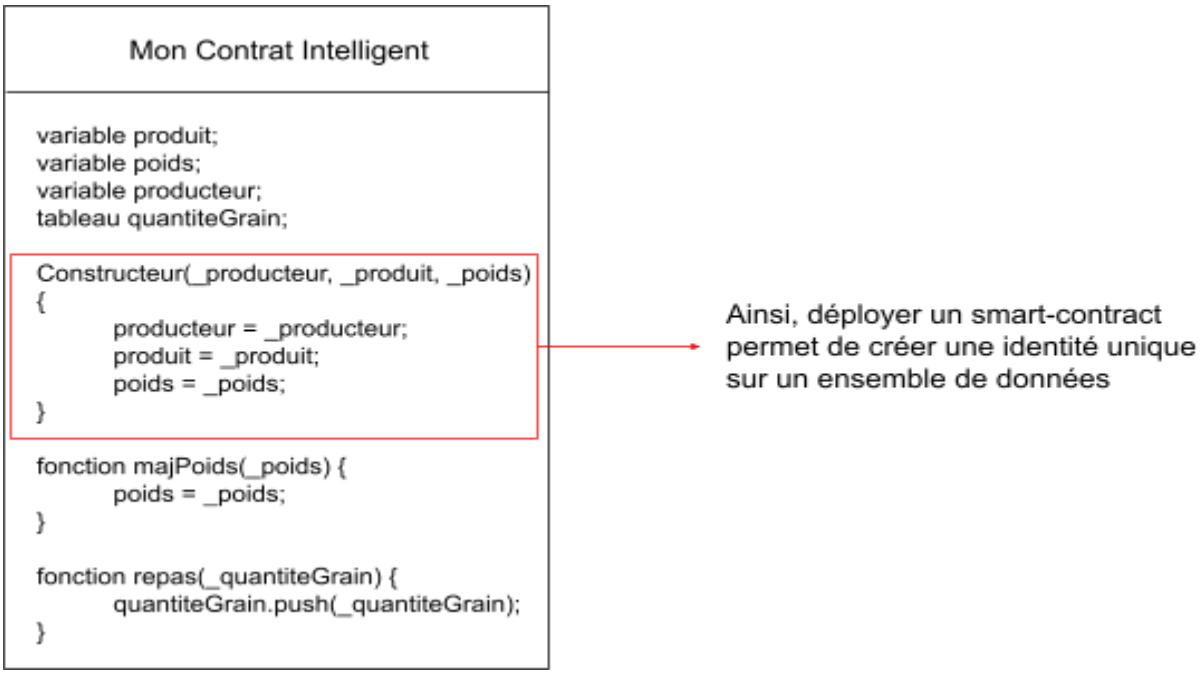
Mon Contrat Intelligent

```
variable produit;
variable poids;
variable producteur;
tableau quantiteGrain;

Constructeur(_producteur, _produit, _poids)
{
        producteur = _producteur;
        produit = _produit;
        poids = _poids;
}

fonction majPoids(_poids) {
        poids = _poids;
}

fonction repas(_quantiteGrain) {
        quantiteGrain.push(_quantiteGrain);
}
```

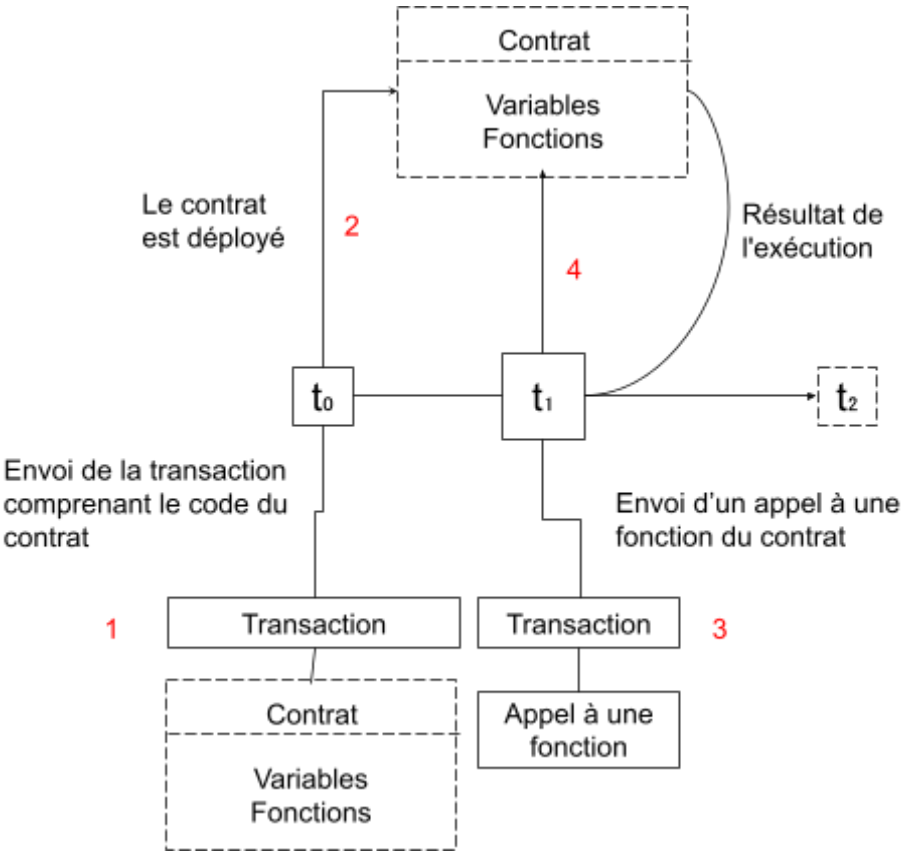
Concrètement, un smart-contract est un bloc de code pouvant embarquer :

- Variables
- Fonctions
- Règles de gestion

Les smart-contracts peuvent interagir entre eux mais sont isolés d'autre environnements informatiques. Ils fonctionnent dans un écosystème fermé embarqué sur la blockchain.

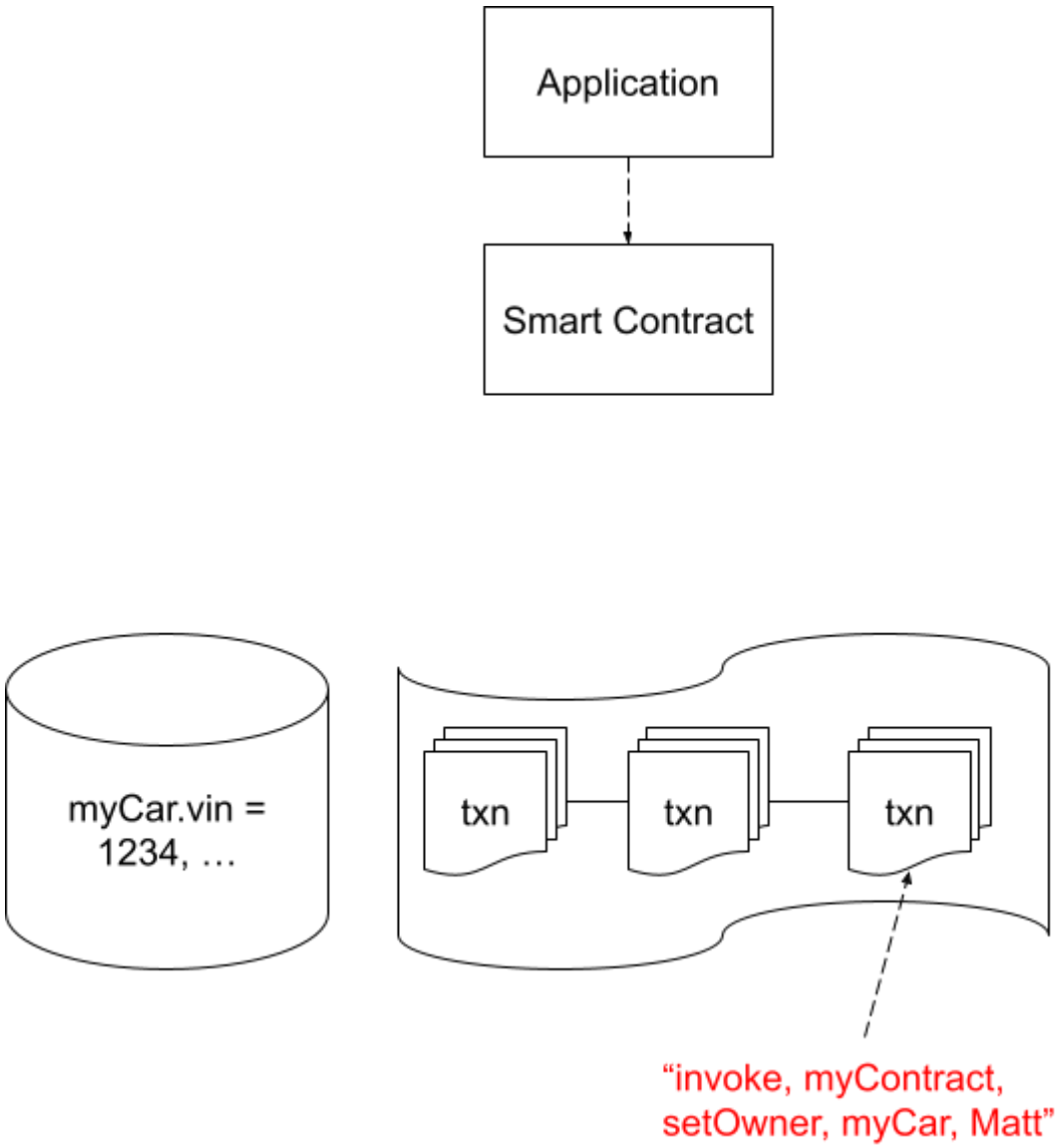


Smart contract fonctionnement



1. Un contrat est déployé au sein d'une blockchain via une transaction
2. Le contrat devient accessible à une adresse précise de la chaine
3. Les fonctionnalités du contrat peuvent être appelées ultérieurement à l'adresse donnée via une autre transaction
4. La transaction est traitée, les instructions de l'application exécutées
5. Le résultat de l'exécution est enregistré dans le registre

Exemple :



Transaction input → Sent from application

```
invoke(myCar, setOwner, myCar, Matt)
```

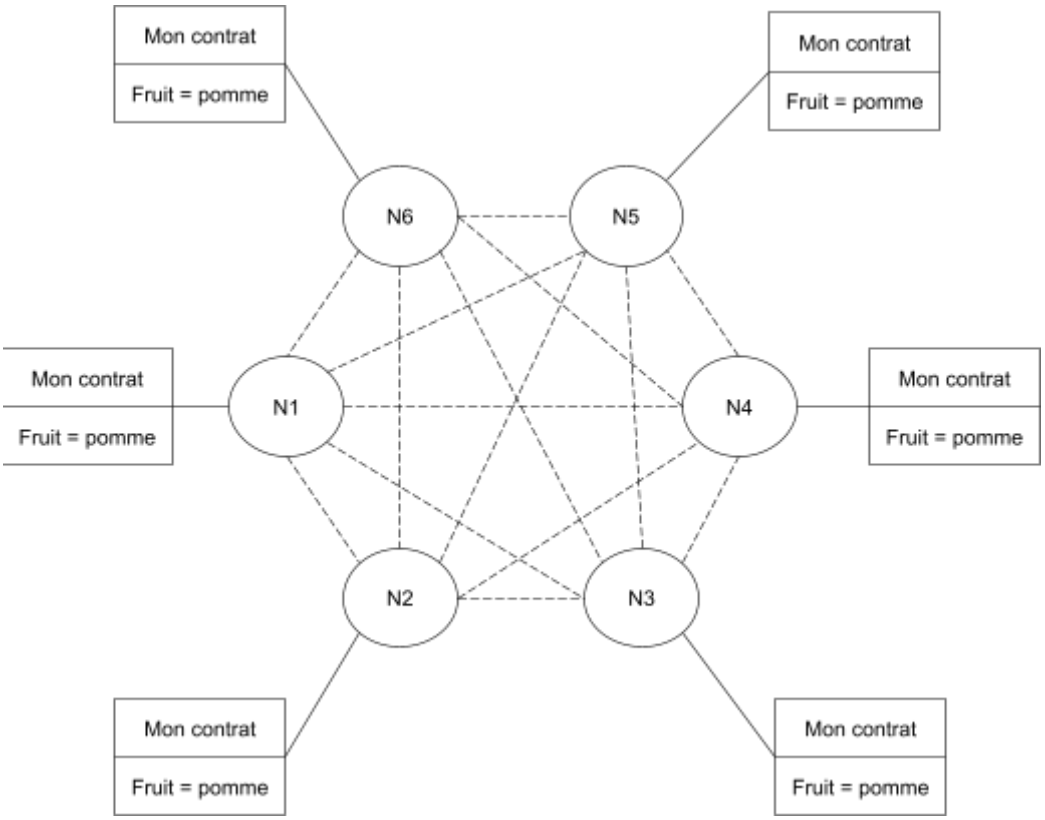
Smart contract implementation

```
setOwner(Car, newOwner) {
     set car.owner = newOwner
}
```

World state : new contents

```
myCar.vin = 1234
myCar.owner = Matt
myCar.make = Audi
…
```

Smart contract fonctionnement



une fois déployée, l'instance du contrat est synchronisée sur chaque noeud du réseau

si une transaction fait évoluer l'état du contrat, cenouvel état sera répliqué sur chaque noeud du réseau

Chaque noeud du réseau vérifie l'intégrité de la transaction faite auprès du contrat

Cas d'utilisation des Contrats Intelligents

- Finance Décentralisée (DeFi) : Les contrats intelligents permettent la création de plateformes financières décentralisées offrant des services tels que les prêts, les emprunts et les échanges sans intermédiaires.

- Assurance : Automatisation des processus de réclamation et de paiement, réduisant ainsi les délais et les coûts administratifs.

- Gestion de la Chaîne d'Approvisionnement : Suivi transparent et immuable des produits tout au long de la chaîne d'approvisionnement, garantissant l'authenticité et la qualité des produits.

Tokens et Oracle

Tokens dans la blockchain :

- Actifs numériques créés à partir de smart contracts
- Stockés dans des wallets (portefeuilles numériques)
- Types principaux :
    - Fungible Tokens (FT) selon la norme ERC-20 (est qu'on peut changer / échanger)
    - Non-Fungible Tokens (NFT) selon la norme ERC-721 / ERC-1155 (est inchangeable)

Fungible Tokens (FT) :

- Interchangeables et divisibles
- Identiques en termes de valeur, de caractéristiques et d'utilité
- Norme ERC-20 sur Ethereum : fonctions standards (transfer(), balance(), …)
- Exemple : Bitcoin (BTC), Ethereum (ETH), stablecoins (USDT, USDC)

Cas d'utilisation des FT :

- Transaction financières et paiement
- Staking et yield farming (DeFi)
- Monnaies internes aux écosystèmes blockchain

yield farming est comme des intérêts (en stockant des jetons)

Non-Fungible Tokens (NFT) :

- Unicité garantie par un identifiant unique enregistré sur la blockchain
- Non interchangeables et indivisibles
- Normes techniques : ERC-721 (unique) et ERC-1155 (semi-fongibles)
- Métadonnées stockées généralement hors-chaîne (IPFS, Arweave)
- Exemple : Oeuvre d'art, objets de collection numériques (CryptoKitties, CryptoPunks)

Cas d'utilisation NFT :

- Authentification d'oeuvre d'art et propriété intellectuelle
- Actifs dans les mondes virtuels (terrain virtuel, équipements en jeu)
- Certificats d'authenticité pour objets physiques (lié par identifiant unique)
- Tickets d'événements sécurisés et vérifiables

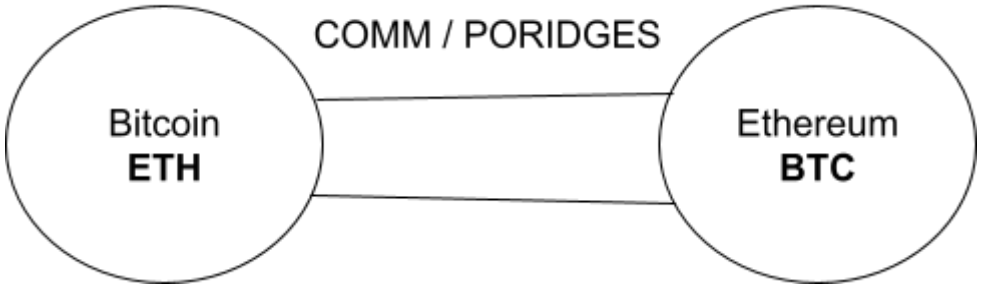
Comparaison technique FT vs NFT

Critère technique	Fungible Token (ERC-20)	Non-Fungible Token (ERC-721 / ERC-1155)
Standard Ethereum	ERC-20	ERC-721 / ERC-1155
Structure	Simple balances comptables	Identifiants uniques avec métadonnées
Interactions	transfer(), approve(), allowance()	transferFrom(), ownerOf(), tokenURI()
Fractionnement	Oui	Généralement non (possibilité via ERC-1155)

Avantages techniques des tokens :

- Traçabilité complète via blockchain
- Élimination d'intermédiaires grâce aux smart contracts
- Intégration simplifiée avec applications décentralisées (dApps)
- Interopérabilité au sein de l'écosystème blockchain

Interopérabilité



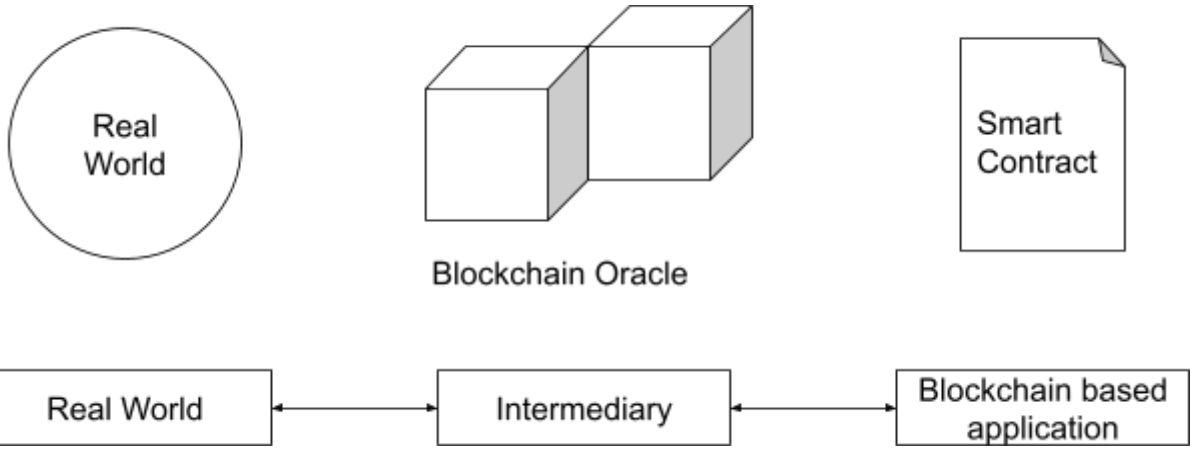
On essaye de ramener le BTC sur Ethereum et être compatible.

Défis Techniques et Enjeux :

- Scalabilité des réseaux blockchain
- Coût et rapidité des transactions (gas fees)
- Stockage des métadonnées (solutions décentralisées vs centralisées)
- Sécurité contre les attaques et failles dans les smart contracts

Les Oracles :

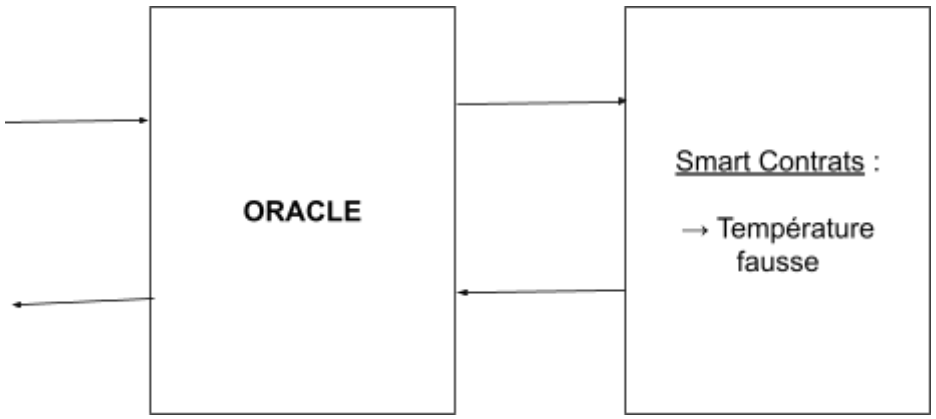
Limitations des Smart Contracts :



Oracles :

- Service qui collecte une donnée dont un contrat intelligent a besoin à une heure prédéfinie et la stocke dans la blockchain à un emplacement prédéfini
- C'est une tierce partie, ce qui va à l'encontre du principe de Blockchain
- Qui a un grand pouvoir sur le fonctionnement des contrats intelligents qui sont imparables
- Deux cas de dysfonctionnement :
    - L'oracle échoue et ne collecte aucune donnée
    - L'oracleintroduit une fausse information dans la blockchain de manière volontaire ou involontaire
- Solutions : Oracles prouvables-honnêtes, oracle basé sur le consensus et oracle physique

Exemple :



Oracle est tous les informations venant de l’extérieur.

Oracle de consensus, c’est de joindre de plusieurs site pour avoir une information, si tous sont d’accord et que le système de consensus dit oui alors on le valide sinon refuser.