


Introduction à la Blockchain



Plan

- Histoire du Bitcoin
- Qu'est-ce qu'une blockchain ?
- EVM et smart contracts
- DeFi/CeFi
- Outils de la finance décentralisée
- Miner Extractable Value
- ?

Raphael Westphal (westphal.rafael@gmail.com)

Parcours scolaire/professionnel

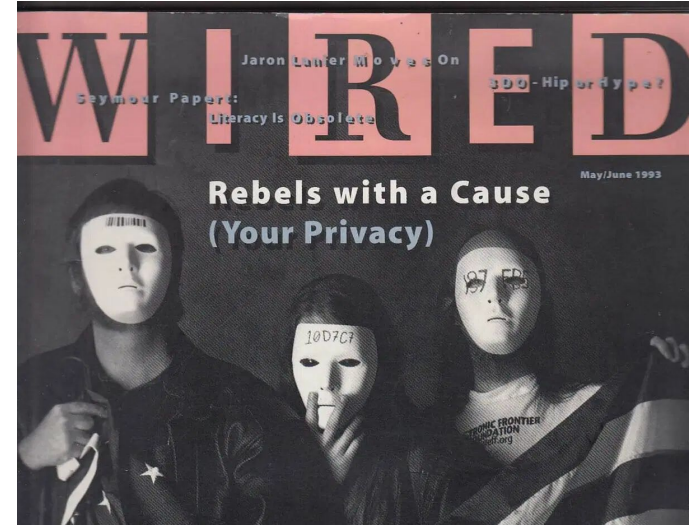
IMT Nord Europe (anciennement Télécom Lille) en alternance

Quatre années à OVH (VoIP et OverTheBox)

Histoire du bitcoin: 1992, Cypherpunks

Groupe d'individus intéressés dans la cryptographie et les technologies permettant l'amélioration et la préservation de la vie privée.

Les cypherpunks sont également connus pour leur fort engagement en faveur des droits de l'individu, leur opposition à la censure gouvernementale et au contrôle d'Internet. Ils sont souvent associés au mouvement du logiciel libre et à la lutte pour la liberté d'Internet et la vie privée.



Histoire du bitcoin: 1992, Cypherpunks

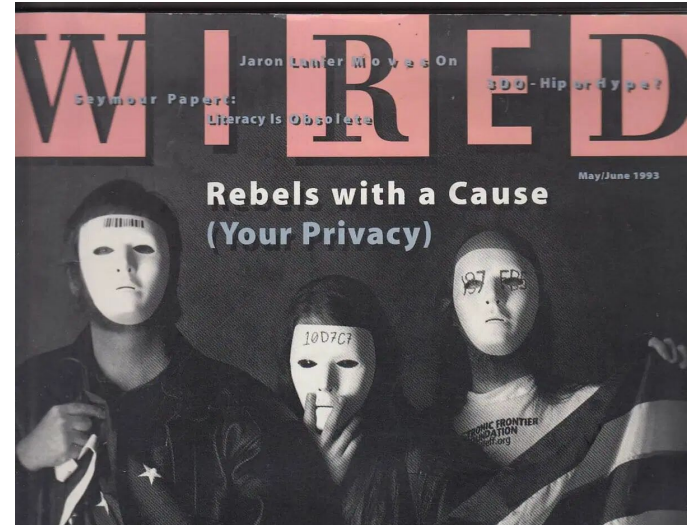
Timothy C. May (ancien ingénieur Intel)

Eric Hughes (mathématicien et programmeur)

John Gilmore (contributeur important du projet GNU et co-fondateur de l'Electronic Frontier Foundation)

Philip Zimmermann (Créateur de PGP)

Julian Assange (Rédacteur en chef et créateur de WikiLeaks)



Histoire du bitcoin: The Crypto Anarchist Manifesto

Trois constats:

- Le manifeste commence par affirmer que la vie privée est une condition nécessaire pour la liberté individuelle et que la technologie peut être utilisée pour renforcer cette vie privée contre les gouvernements et les entreprises qui cherchent à la violer.
- "Si on peut contrôler l'argent des gens, on peut contrôler tous les aspects de leur vie, si on peut surveiller l'argent des gens, alors on sait tout ce qu'il faut savoir sur eux"
- Les individus doivent avoir le contrôle sur leurs données en étant libre de les diffuser ou non

Histoire du bitcoin: 1992, Cypherpunks

Le manifeste aborde également les limites de la vie privée, en soulignant que certains renseignements peuvent être partagés volontairement avec d'autres personnes ou organisations pour des raisons légitimes. Cependant, il précise que ces choix doivent être faits de manière consciente et en toute connaissance de cause, et que les individus doivent avoir le contrôle absolu sur les informations qu'ils partagent.

Histoire du bitcoin: 1992, Cypherpunks

Pourquoi faire confiance à une monnaie ?

Les banques peuvent:

- Imprimer (diluer) votre argent
- Vous refuser un retrait (exemple de la Grèce en 2008 et le Nigéria cette semaine)
- Censurer vos transactions (exemple Wikileaks)

L'argent nous appartient-t-il vraiment ?

Histoire du bitcoin: 2008, Satoshi Nakamoto

En 2008, Satoshi Nakamoto publie son white paper sur une nouvelle monnaie électronique "Bitcoin" permettant de résoudre le problème de la double dépense.

La première version de bitcoin est publiée en 2009 et permet aux utilisateurs de créer adresses et transactions.

Bitcoin P2P e-cash paper

Satoshi Nakamoto | Sat, 01 Nov 2008 16:16:33 -0700

I've been working on a new electronic cash system that's fully peer-to-peer, with no trusted third party.

The paper is available at:

<http://www.bitcoin.org/bitcoin.pdf>

The main properties:

Double-spending is prevented with a peer-to-peer network.

No mint or other trusted parties.

Participants can be anonymous.

New coins are made from Hashcash style proof-of-work.

The proof-of-work for new coin generation also powers the network to prevent double-spending.

Histoire du bitcoin: 2008, Satoshi Nakamoto

Le projet est vite rejoint par d'autres programmeurs (notamment d'autres "Cypherpunks")

Satoshi disparaît d'internet en 2011, jusqu'à maintenant on ne sait toujours qui il(s) est

Source: Le mystère Satoshi



Qu'est-ce qu'une blockchain ?

La blockchain est une **base de données distribuée** utilisée pour stocker des informations de manière à rendre difficile ou impossible de les modifier, de les pirater ou de **tricher avec le système**. C'est un système **décentralisé** qui permet à plusieurs parties d'enregistrer et de vérifier des **transactions** sans avoir besoin d'une **autorité centrale**. Les informations dans une blockchain sont généralement stockées dans des **blocs** qui sont **liés** ensemble dans une chaîne en utilisant la **cryptographie**. Cela rend les données dans une blockchain sécurisée et transparente, et permet de les partager et d'y accéder par n'importe qui ayant les autorisations appropriées.

Base de données distribuée

Une base de données distribuée est essentiellement une base de données qui n'est pas limitée à un seul système, elle est répartie sur différents sites, c'est-à-dire sur **plusieurs ordinateurs** ou sur un réseau d'ordinateurs. Un système de base de données distribuée est situé sur différents sites **qui ne partagent pas de composants physiques**. Cela peut être nécessaire lorsqu'une base de données particulière doit être consultée par différents utilisateurs à l'échelle mondiale. Elle doit être gérée de telle sorte que pour les utilisateurs, elle ressemble à une base de données unique.

Décentralisé

Le stockage décentralisé est un système qui partage entre de nombreux opérateurs indépendants, la conservation de données informatiques.

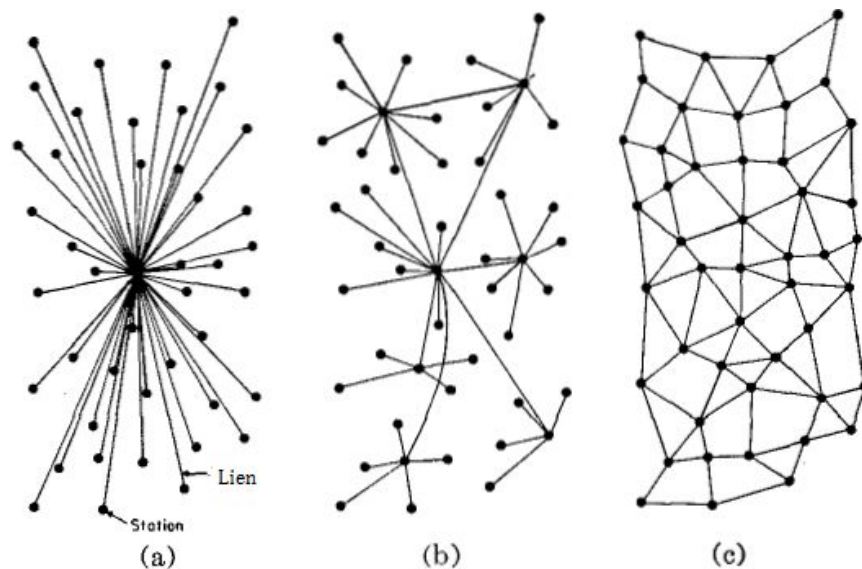


Fig. 1—(a) Centralisé (b) Décentralisé (c) Réseaux distribués

Transaction

En informatique, et particulièrement dans les bases de données, une transaction telle qu'une réservation, un achat ou un paiement est mise en œuvre via une suite d'opérations qui font passer la base de données d'un état A — antérieur à la transaction — à un état B postérieur et des mécanismes permettent d'obtenir que cette suite soit à la fois **atomique**, **cohérente**, **isolée** et **durable (ACID)**

Atomicité (A.C.I.D)

Le principe d'Atomicité garantit la bonne exécution de la transaction. Les transactions de base de données, comme les atomes, peuvent être décomposées en plus petites parties. Si une partie d'une transaction échoue, toute la transaction sera annulée.

IE: La transaction est exécutée entièrement ou ne l'est pas du tout

Cohérence (A.C.I.D)

La propriété de Cohérence signifie que seules les données qui suivent des règles prédéfinies peuvent être écrites dans la base de données.

IE: Si je ré-exécute la même transaction sur le même état A, alors je dois retomber sur B

Isolement (A.C.I.D)

L'isolement fait référence à la capacité de traiter simultanément plusieurs transactions de manière indépendante.

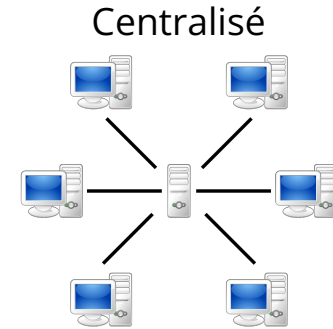
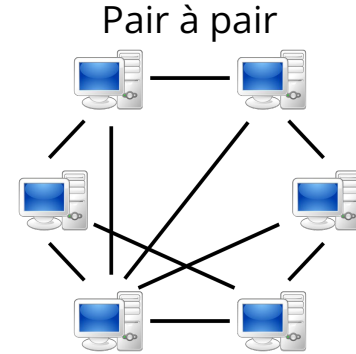
IE: Il n'est possible que d'exécuter "une seule transaction à la fois"

Durabilité (A.C.I.D)

La durabilité requiert de rendre les défaillances invisibles pour l'utilisateur final. Les données sont sauvegardées une fois la transaction terminée, même en cas de panne de courant ou de défaillance du système.

Pair à pair (p2p):

Le pair-à-pair définit un modèle de réseau informatique d'égal à égal entre ordinateurs, qui distribuent et reçoivent des données ou des fichiers. Dans ce type de réseau, comparable au réseau client-serveur, chaque client devient lui-même un serveur.



<https://blockchaindemo.westphal.fr>

Blockchain Demo

HashBlockBlockchainDistributedTokensCoinbase

SHA256 Hash

Data:

Hash:

e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855

Source: <https://github.com/anders94/blockchain-demo>

Minage de blocs

Minage: Processus durant lequel un **mineur** tente de trouver le bon **nonce** pour avoir le nombre de 0s suffisant pour créer un bloc valide



~ 300 KH/s




~1 GH/s = 100 000 KH/s



~ 110 TH/s = 110 000 000 KH/s

Minage de blocs



Calculated for
1 BTC = \$ 16,828.96

Hashing Power

TH/s ▾

Power consumption (w)

Cost per KWh (\$)

PROFIT RATIO PER DAY		PROFIT PER MONTH	
-50%		\$ -195.56	
Profit per day Day \$ -6.52 Pool Fee \$ 0.06809	Mined/day B 0.0004046	Power cost/Day \$ 13.26	
Profit per week Week \$ -45.63 Pool Fee \$ 0.4767	Mined/week B 0.002832	Power cost/Week \$ 92.82	
Profit per month Month \$ -195.56 Pool Fee \$ 2.04	Mined/month B 0.01214	Power cost/Month \$ 397.80	
Profit per year Year \$ -2,379.29 Pool Fee \$ 24.85	Mined/year B 0.1477	Power cost/Year \$ 4,839.90	

Blockchain

Hash: Chaîne de caractères (chiffre) qui permet de vérifier l'authenticité d'une donnée sans en connaître le contenu. Une fonction de hachage peut prendre en entrée une infinité de données et sortira toujours une chaîne de caractères de la même taille.

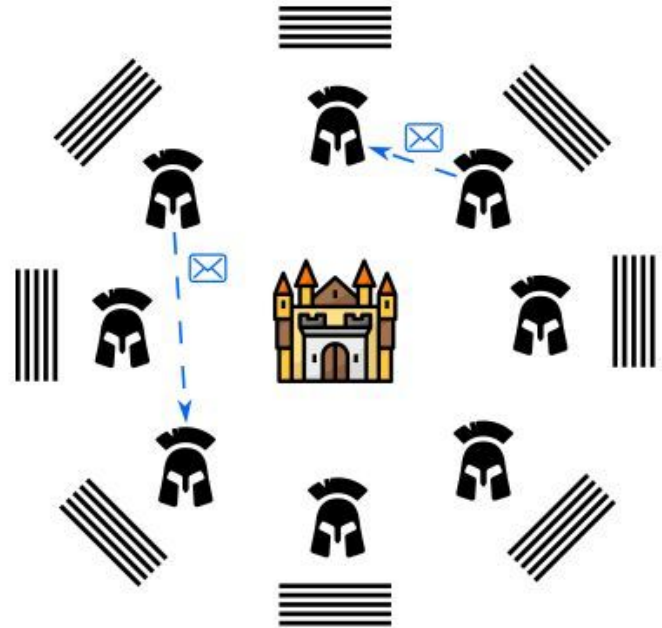
Un bloc contient:

- Un ensemble de données (dans le cas de Bitcoin, des transactions)
- Un numéro d'identification
- Une référence (ou non) à un bloc précédent
- Un nonce
- Une coinbase (identifiant de l'ordinateur ayant trouvé la solution au "problème des zéros")

Le hash du bloc n'est pas stocké dans celui-ci mais en est dérivé

Problèmes des généraux byzantins

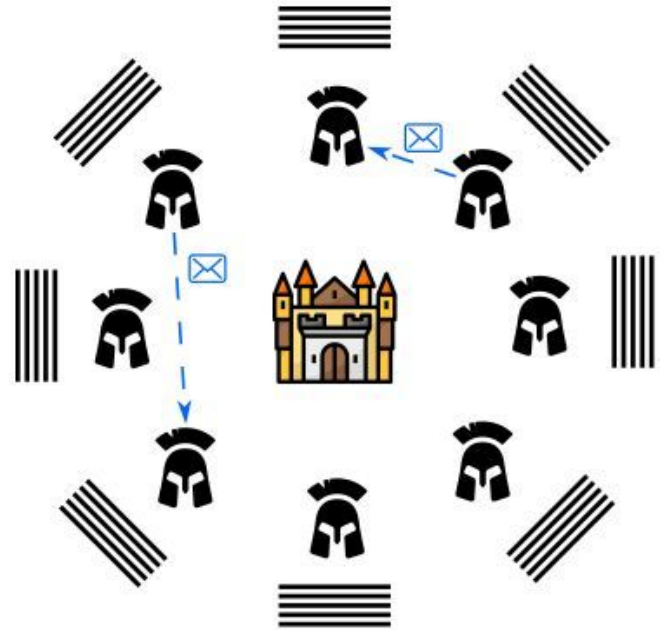
Théorisé par Leslie Lamport en 1982, il pose l'analogie la coordination d'une attaque en généraux byzantins et le bon fonctionnement d'un ordinateur ayant un ou plusieurs composants défectueux



Problèmes des généraux byzantins

Des généraux byzantin veulent coordonner une attaque sur une ville. Pour gagner la bataille, il faut que toutes les armées s'accordent pour attaquer le même jour à la même heure.

Problème: les canaux de communication ne sont pas fiables et des traîtres se sont glissés parmi les messagers

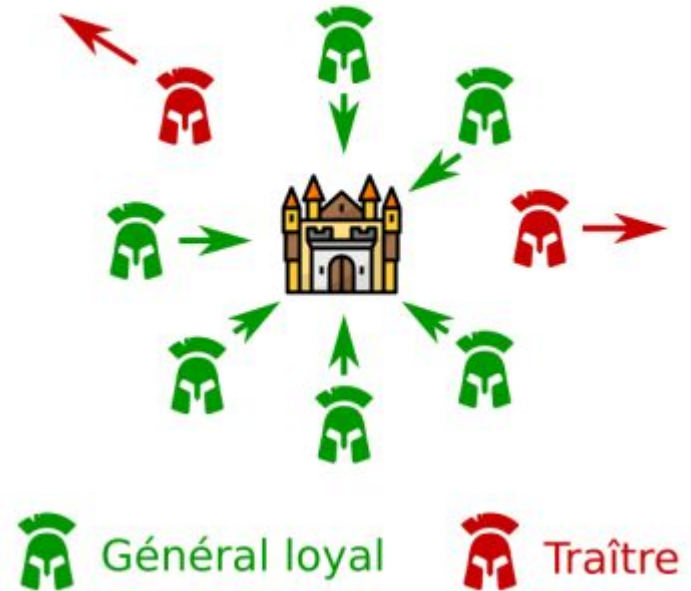


Problèmes des généraux byzantins

Comment réussir à atteindre le **consensus** sur la date et l'heure d'attaque ?

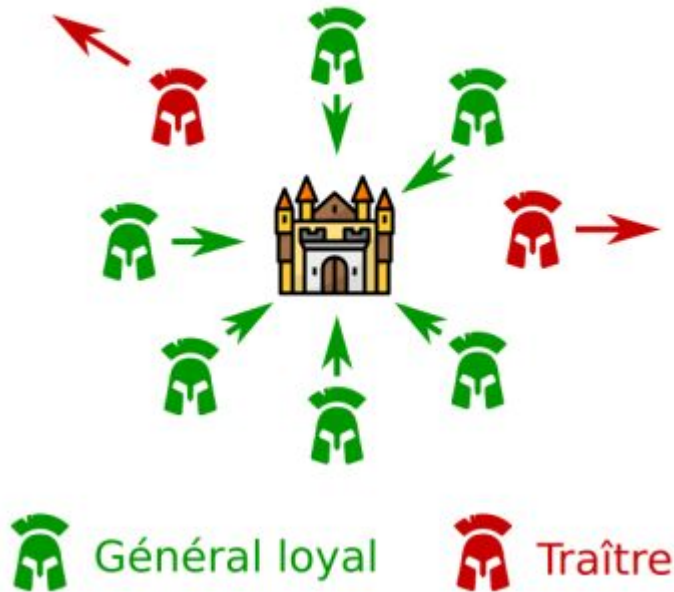
(Le jour de l'attaque importe peu, ce qui compte est que tout le monde attaque en même temps)

Victoire

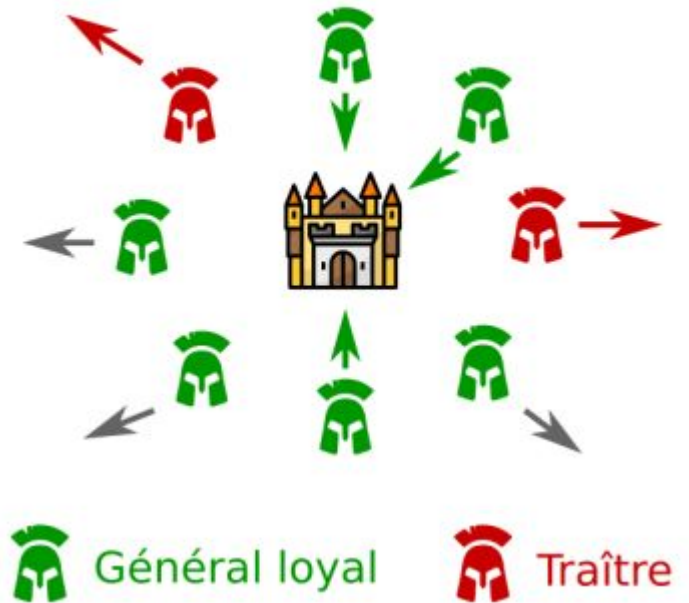


Problèmes des généraux byzantins

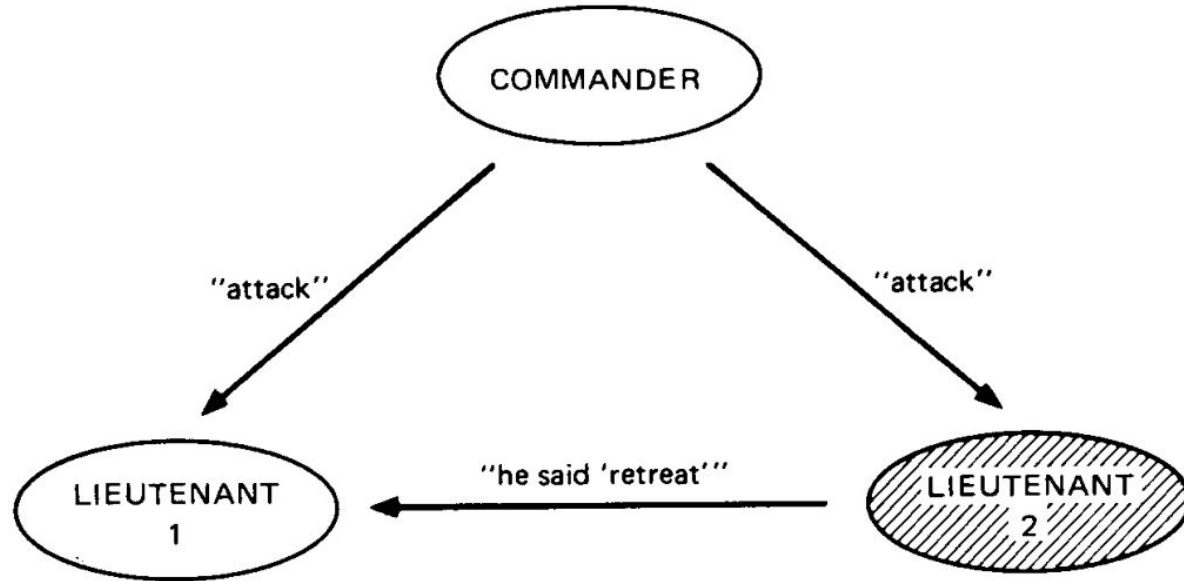
Victoire



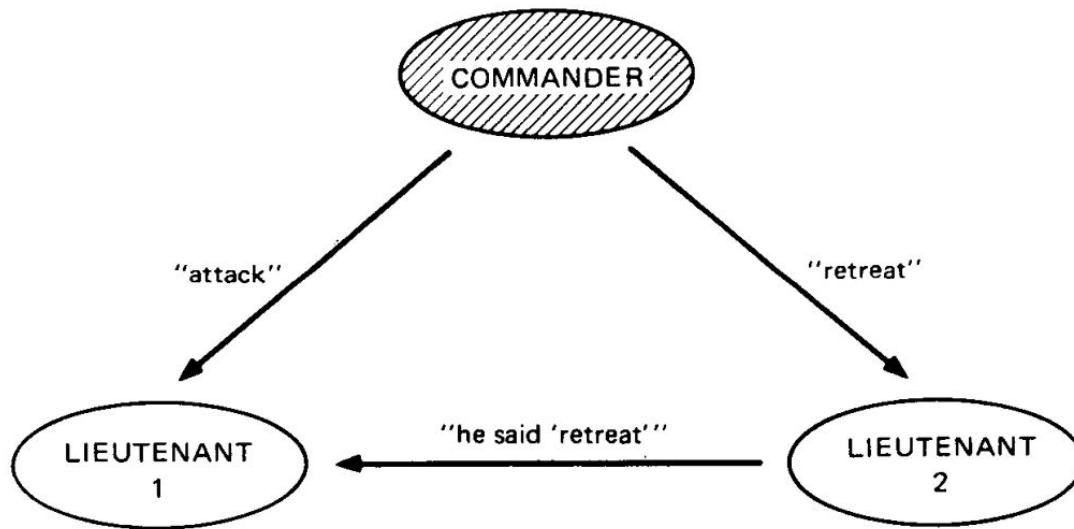
Défaite



Problèmes des généraux byzantins



Problèmes des généraux byzantins



Problèmes des généraux byzantins

[illegible]

Block:

2

Nonce:

22810

Data:

Je suis un général et je vais attaquer Lundi

Prev:

000099fbb7bdcf37249d096a0b917f61c319aecb5114762d03fet

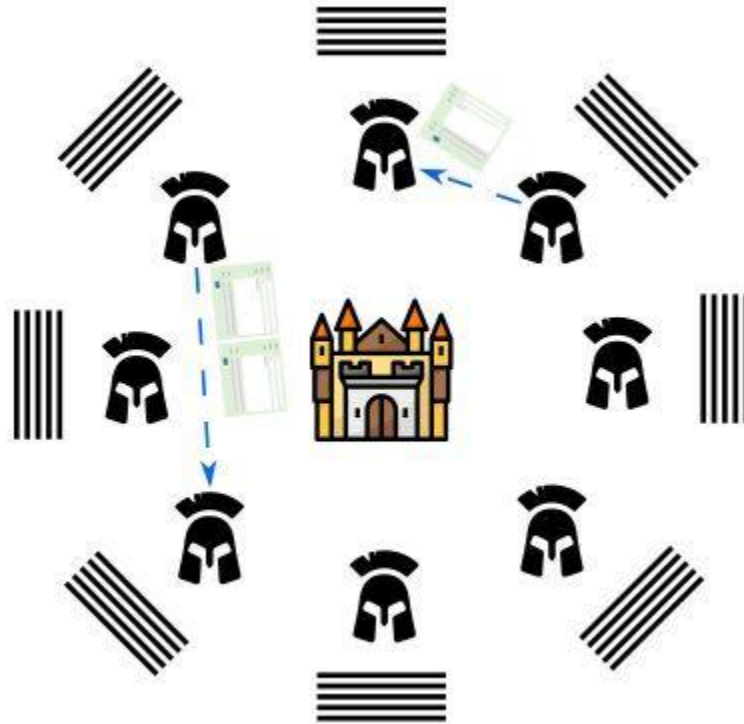
Hash:

0000d2a84e2fd428ad08163bf98c53a09f0bd579dc9237ffd93b4

Mine

[illegible]

Problèmes des généraux byzantins



Théorie des jeux

La théorie des jeux regroupe plusieurs dispositifs qui permettent de déterminer ce qu'un acteur, peut avoir comme influence sur un autre afin de retirer le meilleur de chaque situation

Consensus, comment choisir un bloc valide

Le choix de la chaîne valide doit suivre deux composantes:

- Les règles (taille du bloc, transactions) doivent être valides (exclusif)
- On choisit toujours la chaîne la plus longue (non exclusif)

3 types of blockchain forks

temp.



soft

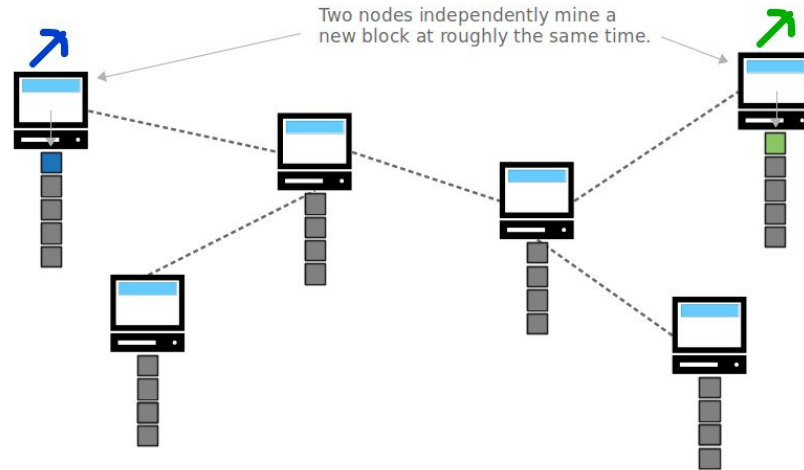


hard



Consensus, réorganisation de blocs

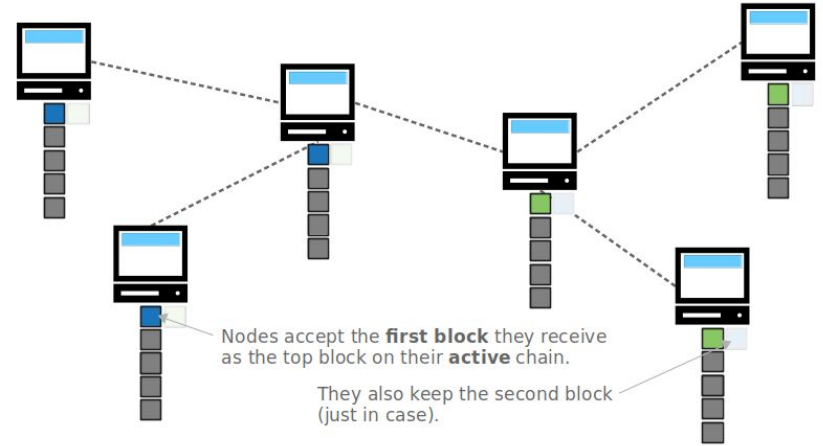
Que se passe-t-il lorsque deux blocs sont créés au même moment par deux mineurs différents ?



Consensus, réorganisation de blocs

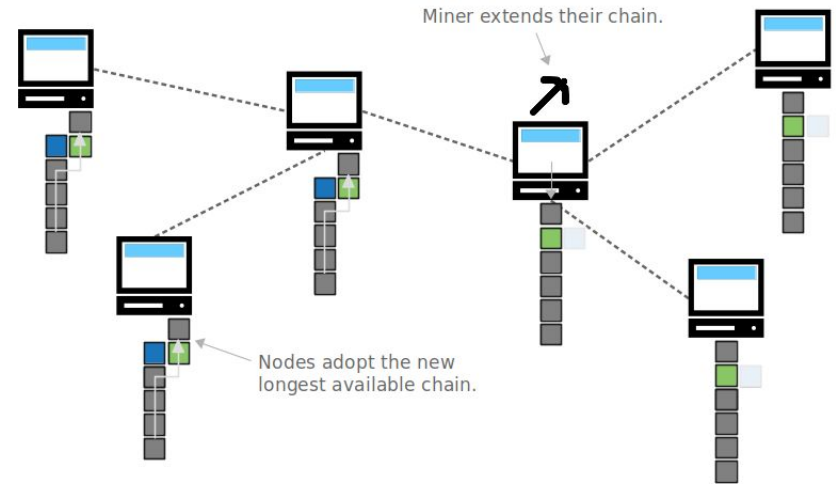
Dans un réseau distribué, chaque node ne connaît que l'état de ses voisins, il n'a pas forcément connaissance des autres blocs.

Si deux (ou plus) blocs sont vus au même moment, les deux règles sont respectées mais on ne sait pas qui choisir



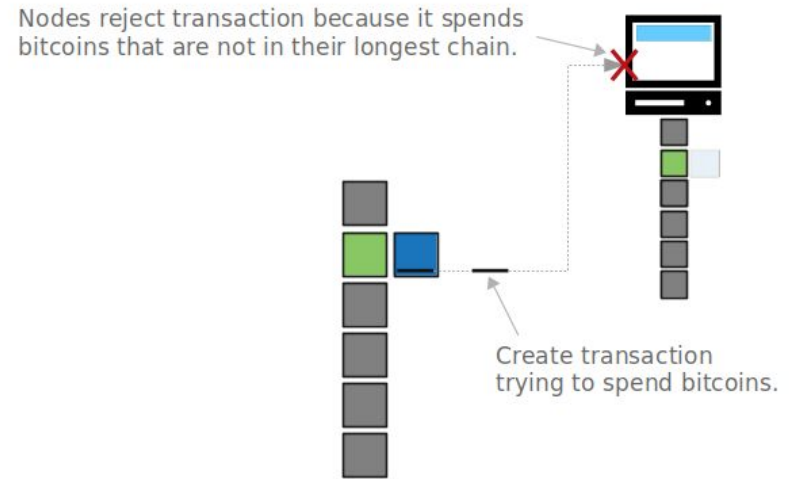
Consensus, réorganisation de blocs

Les mineurs vont choisir un des deux blocs (généralement le premier vu) et essayer de construire un autre bloc par dessus



Consensus, réorganisation de blocs

Les mineurs vont choisir un des deux blocs (généralement le premier vu) et essayer de construire un autre bloc par dessus



Block: # 2

Nonce: 178843

Tx:

\$ 100	From: Ripley	->	Lambert
\$	From:	->	
\$	From:	->	
\$	From:	->	
\$	From:	->	
\$	From:	->	
\$	From:	->	

Prev: 0000c52990ee86de55ec4b9b32beefd745d71675dc8eddfbc7b88336e2e296b

Hash: 0000d24136989b4a4f5fedb8622f9a760b7cc7514834d6ab755230cc2169e637

Mine

Block: # 3

Nonce: 24926

Tx:

\$ 10.00	From: Ripley	->	Jackson
\$	From:	->	
\$	From:	->	

Prev: 0000d24136989b4a4f5fedb8622f9a760b7cc7514834d6ab755230cc2169e637

Hash: 0000fa001da10dd8a621ba5ef5cb9f59d8d29908aebc597bc483cb0bfeeb1c8c

Mine

Block: # 2

Nonce: 96771

Tx:

\$ 1	From: Ripley	->	Lambert
\$	From:	->	
\$	From:	->	
\$	From:	->	
\$	From:	->	
\$	From:	->	
\$	From:	->	

Prev: 0000c52990ee86de55ec4b9b32beefd745d71675dc8eddfbc7b88336e2e296b

Hash: 00001b8dbce1a8887dce2be02316963d7aa33ebbl1ebf7accc55083d7b3073c39

Mine

Block: # 3

Nonce: 13804

Tx:

\$ 100	From: Ripley	->	Jackson
\$	From:	->	
\$	From:	->	

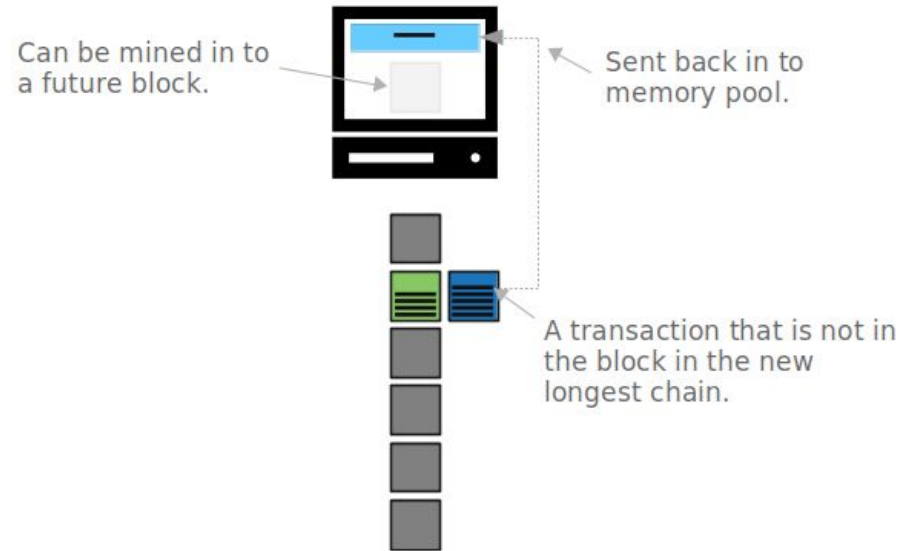
Prev: 00001b8dbce1a8887dce2be02316963d7aa33ebbl1ebf7accc55083d7b3073c39

Hash: 0a0a6159ec0e1ae1b96b326f7735a6ad677d38252d45bca74441257e008dd35

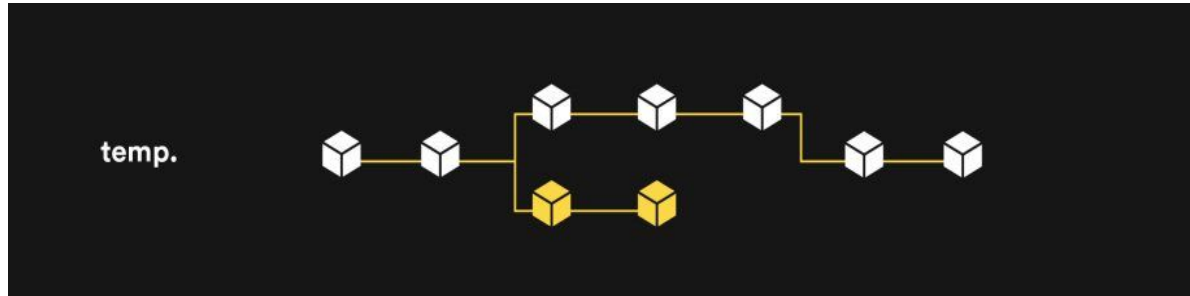
Mine

Consensus, réorganisation de blocs

Lors d'une réorganisation, les transactions du bloc orphelin sont réinsérées dans la "mempool" et peuvent être minées dans les blocs suivants



Consensus, comment choisir un bloc valide



Les autres noeuds du réseau ont tout intérêt à choisir la chaîne la plus longue, ils vont donc choisir la chaine du haut et abandonner la chaine du bas qui deviendra alors **orpheline**

Consensus, comment choisir un bloc valide

Un **hard fork** est un embranchement de la chaîne de blocs causé par une divergence des règles de consensus.

3 types of blockchain forks

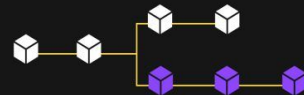
temp.



soft



hard



Consensus, comment choisir un bloc valide

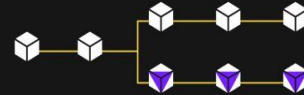
Un **soft fork** est une modification rétrocompatible (ou post compatible à proprement parler) des règles de consensus, dans le sens où les nœuds suivant les anciennes règles continuent de voir les blocs produits comme valides et restent donc connectés au réseau. Il s'agit essentiellement d'une restriction du protocole : des transactions et des blocs anciennement valides deviennent invalides. Une illustration typique de soft fork est la réduction de la taille des blocs (de 1 Mo à 300 ko par exemple) : les anciens nœuds voient les petits blocs comme valides alors même que la règle qu'ils appliquent (limite à 1 Mo) est plus large.

3 types of blockchain forks

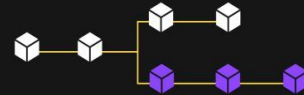
temp.



soft



hard



Chiffrement clé publique/privée et signatures

<https://blockchainedemo2.westphal.fr>

Public / Private Key Pairs

Private Key

32051498339452478494859112526972400482152160531055882381917805645804794210600



Random

Public Key

043714e8beb71bcc24854561198a6723d7716e756ce600e906ef3d88ef0892ca9f4b3c3d5db0cbc945133c8f0b7a3372fd06097dea88e

Chiffrement clé publique/privée et signatures

Clé privée: Très grand nombre aléatoire souvent représenté sous forme de caractères. À garder secret

Clé publique: Nombre dérivé de la clé privée, peut être partagé publiquement

Signature: Chaîne de caractères qui, joint à une clé publique, permet d'identifier un message

Partie 2

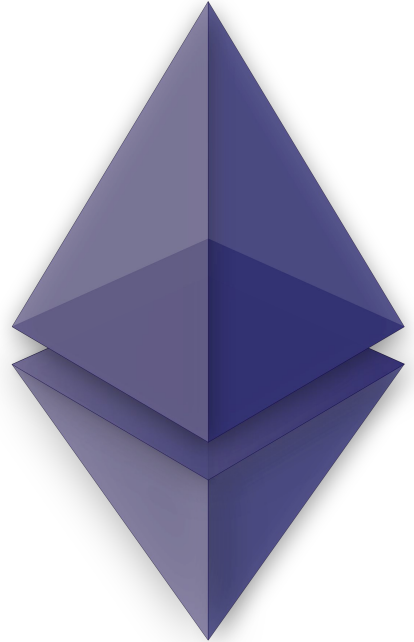
Plan

- Revue de l'histoire d'Ethereum
- Présentation de l'EVM
- Développement de votre propre token
- Déploiement du token sur le testnet BSC

Histoire: Création et lancement d'Ethereum

En décembre 2013, **Vitalik Buterin (19 ans)** publie une description de son projet Ethereum sur le forum BitcoinTalk sous la forme d'un livre blanc dans le but de lancer des applications décentralisées.

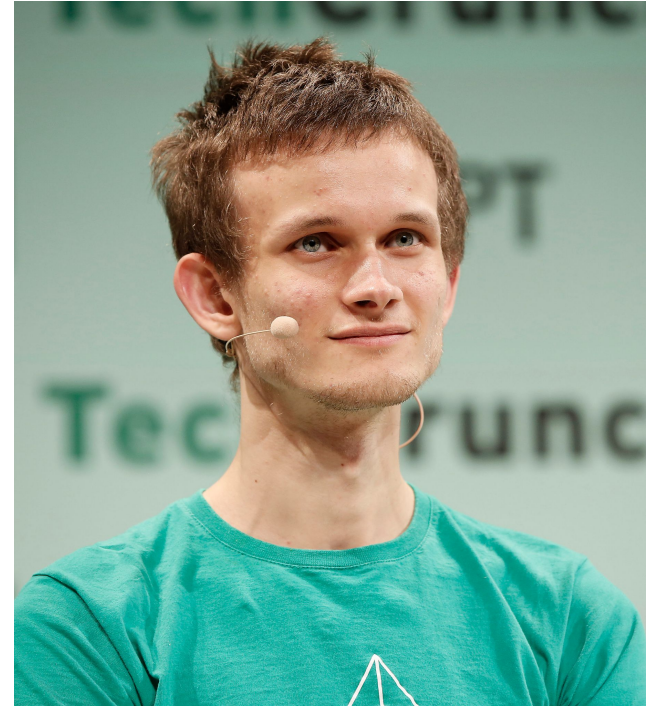
Début 2014, il met en prévente les premiers Ethers pour financer le développement du projet.



Histoire: Création et lancement d'Ethereum

La vente lui permet de rassembler 31 591 bitcoins d'une valeur de plus de 18 millions de dollars à l'époque, pour 60 millions d'Ethers vendus.

Avec cet argent il fonde Ethereum Switzerland GmbH (développement de la chaîne) et The Ethereum Foundation (association à but non lucratif) pour promouvoir le développement de cette nouvelle monnaie



Histoire: Création et lancement d'Ethereum

Bitcoin: Utilise uniquement les quatres premières couches

Ethereum: rajoute deux couches applicatives

Application Layer	Internet of Things, Smart City, Market Security, Health Records
Contact Layer	Algorithm, Smart Contract, Script Code
Incentive Layer	Issuance Mechanism, Allocation Mechanism
Consensus Layer	PoW, DAG, PoS, PoE, PoX, BFT, PoL, PoET etc.,
Network Layer	P2P Network, Communication Mechanism, Verification Mechanism
Data Layer	Data Block, Chain Structure, Time Stamp

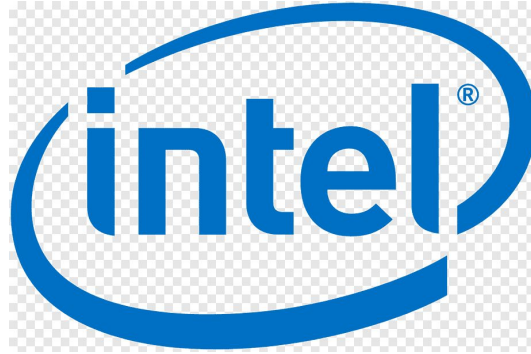
EVM: qu'est-ce qu'une machine virtuelle ?



Un compilateur est un programme informatique qui traduit le code source écrit dans un langage de programmation en une forme lisible par la machine et pouvant être exécutée par un ordinateur (**assembleur**).

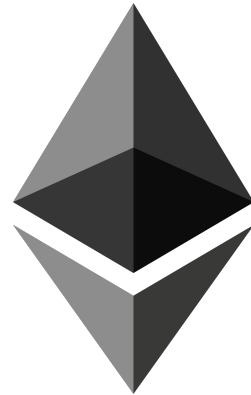
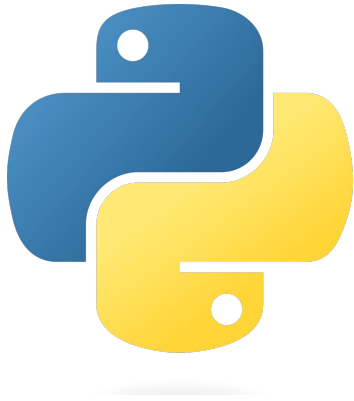
Différents types d'assembleur: natif

ARM



L'assembleur natif est destiné à être lu par un processeur physique.

Différents types d'assembleur: bytecode



Le bytecode est un code intermédiaire entre les instructions machines et le code source, qui n'est pas directement exécutable.

EVM: comment faire ?

Créer une machine virtuelle décentralisée pose plusieurs problèmes:

- Où sont stockés les programmes ?
- Qui exécute les programmes ?
- Comment éviter les abus ? (ex: trop de calculs/stockage)
- Comment être sûr que les programmes soient déterministes ? (ex: Jeux de hasard)


EVM: comment créer un contrat ?

Un programme stocké sur une blockchain s'appelle un **smart contract**

Pour créer un contrat, une transaction est émise contenant tout le code stocké. Une fois validée, le code est **immutable**



? Transaction Hash:

0x9b28f893c322350f9425bfaf6841dce7d1e50c45676cba78cc2a7b7fde3f2405 

? Status:

✓ Success

? Block:

✓ 16363398

423 Block Confirmations

? Timestamp:

🕒 1 hr 25 mins ago (Jan-08-2023 04:55:35 PM +UTC) | ⌚ Confirmed within 4 secs

? Sponsored:

? From:

<> [yannickcrypto.eth](#) 

? To:

[Contract [0x2843dd740e1f2e0662fcde14e7203485da34f4ad](#) Created] ✓ 

↳ TRANSFER 0.00000000000001 Ether From [0x2843dd740e1f2e0662fcde14...](#) To → [0xfc69f5418d69b5699115e516...](#)

? Value:

0.00000000000001 Ether (< \$0.000001)

? Transaction Fee:

0.00945949925761497 Ether (\$11.97)

? Gas Price:

0.000000017015630129 Ether (17.015630129 Gwei)

EVM: qui exécute les contrats ?

- Les appels aux contrats (exécution) sont effectués une première fois par le mineur/validateur
- Pour vérifier un bloc, les autres noeuds du réseau ré-exécutent la transaction (l'appel au contrat) dans leur propre EVM et vérifient si le résultat est identique
- Le résultat (et les modifications) ne sont pas stockés directement dans les blocs, uniquement les transactions

Block: # 1

Nonce: 139358

Tx:	\$	25.00	From:	Darcy	->	Bingley
	\$	4.27	From:	Elizabe	->	Jane
	\$	19.22	From:	Wickham	->	Lydia
	\$	106.44	From:	Lady Ca	->	Collins
	\$	6.42	From:	Charlot	->	Elizabe

Prev: 000

Hash: 00000c52990ee86de55ec4b9b32beefd745d71675dc

Mine

Block: # 2

Nonce: 39207

Tx:	\$	97.67	From:	Ripley	->	Lambert
	\$	48.61	From:	Kane	->	Ash
	\$	6.15	From:	Parker	->	Dallas
	\$	10.44	From:	Hicks	->	Newt
	\$	88.32	From:	Bishop	->	Burke
	\$	45.00	From:	Hudson	->	Gorman
	\$	92.00	From:	Vasquez	->	Apone

Prev: 00000c52990ee86de55ec4b9b32beefd745d71675dc

Hash:	000078be183417844c14a9251ca246fb15df1074019
-------	---

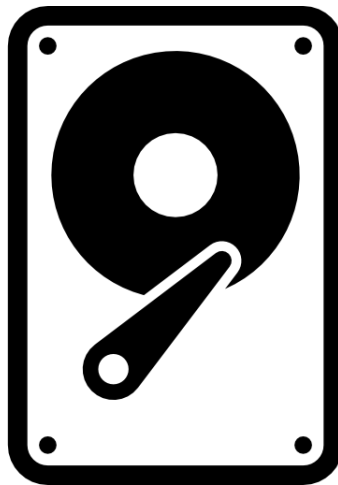
Mine

EVM: Qu'est ce que le "state" ?

Bloc 0		Bloc 1		Bloc 2	
Coinbase:	Alice	Alice -> Martin	10	Bob -> Alex	10
		Coinbase	Bob	Coinbase	Thomas
State		State		State	
Alice	25	Alice	15	Bob	15
		Martin	10	Alex	10
		Bob	25	Alice	15
				Thomas	25

EVM: Qu'est ce que le "state" ?

- Le state est comme le disque dur de l'EVM
- Il stocke:
 - La balance de chaque adresse (contrat ou EOA)
 - Le code d'exécution des smart contracts
 - Les variables modifiées par les contrats



EVM: Comment éviter les abus ?

Que se passe-t-il si quelqu'un s'amuse à:

- stocker tout son disque dur dans le state ?
- l'utiliser pour calculer tout et n'importe quoi ?

Solution: faire payer chaque opération de calcul ou de stockage via le "gas" (ou gaz)



EVM: Le gas

Sur Ethereum (et les chaînes compatibles EVM), chaque transaction est associée à une “limite de gaz” (gas limit)

Le gaz est consommé au fur et à mesure que la transaction effectue des calculs ou des opérations de stockage



EVM: Le gas

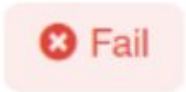
Pour éviter les abus, chaque transaction doit payer le gas consommé

② Transaction Fee:	0.0012629 BNB (\$0.34)
② Gas Limit:	277,044
② Gas Used by Transaction:	252,580 (91.17%)
② Gas Price:	0.000000005 BNB (5 Gwei)

Transaction Fee = Gas Price * Gas used by Transaction

EVM: Le gas

Que se passe-t-il si une transaction n'a pas assez de gas ?

Transaction Fee:	0.00011 BNB (\$0.03)
Gas Limit:	22,000
Gas Used by Transaction:	22,000 (100%)
Gas Price:	0.000000005 BNB (5 Gwei)
Status:	 Fail

EVM: Le revert

Lorsqu'une transaction "revert", toutes les modifications du state par la transaction sont annulées

Seul les frais de transactions sont effectivement payés

Transaction Fee:	0.00011 BNB (\$0.03)
Gas Limit:	22,000
Gas Used by Transaction:	22,000 (100%)
Gas Price:	0.000000005 BNB (5 Gwei)
Status:	✖ Fail

Projet: programmation de smart contracts

Étapes du projet:

- Programmation d'un premier smart contract simple
- Création d'un ERC-20 (token)
- Déploiement d'un token sur le testnet BSC
- Ajout du token sur Uniswap
- ?

src/Counter.sol

```
1 // SPDX-License-Identifier: UNLICENSED
2 pragma solidity ^0.8.13;
3
4 contract Counter {
5     uint256 public number;
6
7     function setNumber(uint256 newNumber) public {
8         number = newNumber;
9     }
10
11     function increment() public {
12         number++;
13     }
14 }
```

test/Counter.t.sol

```
1  // SPDX-License-Identifier: UNLICENSED
2  pragma solidity ^0.8.13;
3
4  import "forge-std/Test.sol";
5  import "../src/Counter.sol";
6
7  contract CounterTest is Test {
8      Counter public counter;
9
10     function setUp() public {
11         counter = new Counter();
12         counter.setNumber(0);
13     }
14
15     function testIncrement() public {
16         counter.increment();
17         assertEq(counter.number(), 1);
18     }
19
20     function testSetNumber(uint256 x) public {
21         counter.setNumber(x);
22         assertEq(counter.number(), x);
23     }
24 }
```

<https://github.com/wesraph/td-blockchain-esgi>

Le standard ERC-20

Un token est un sous ensemble des crypto monnaies qui vit directement sur une chaîne

Chaque token doit respecter la norme ERC-20 définie par les standards Ethereum.

La norme est définie au lien suivant:

<https://eips.ethereum.org/EIPS/eip-20>



Le standard ERC-20

Le standard ERC-20 demande à ce que le contrat implémente **au moins** les fonctions suivantes:

```
1 function name() public view returns (string)
2 function symbol() public view returns (string)
3 function decimals() public view returns (uint8)
4 function totalSupply() public view returns (uint256)
5 function balanceOf(address _owner) public view returns (uint256 balance)
6 function transfer(address _to, uint256 _value) public returns (bool success)
7 function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
8 function approve(address _spender, uint256 _value) public returns (bool success)
9 function allowance(address _owner, address _spender) public view returns (uint256 remaining)
```