BLOCKCHAIN

DE LA SÉCURITÉ AU BESOIN DE RÉGULATION

Christine Hennebert

CEA LETI – Laboratoire des Systèmes Embarqués Sécurisés

DÉFINITION

« Le plus long chemin qui relie le genesis block (root of tree) à une feuille (dernier bloc) est appelé la blockchain.
La blockchain forme un historique de transactions horodatées cohérent sur lequel tous les nœuds peuvent, en principe, s'accorder. »

1. Roger WattenHoffer, « The science of blockchain », *Published by Createspace Independent Publishing Platform*, Jan 2016, ISBN : 978-1-5227-5183-0, *en ligne*, *https://dl.acm.org/doi/book/10.5555/300270*

« un dispositif d'enregistrement électronique partagé permettant l'authentification de ces opérations »

2. Journal officiel, Ordonnance n° 2016-520 du 28 avril 2016 relative aux bons de caisse, *JORF n° 0101 du 29 avril 2016, texte n° 16*, *Legifrance*, *en ligne*, *https://www.legifrance.gouv.fr/eli/ordonnance/2016/4/28/FCPT1608300R/jo/texte*

SECTION 1

PRÉAMBULE

« Le plus long chemin qui relie le genesis block (root of tree) à une feuille (dernier bloc) est appelé la blockchain.
La blockchain forme un historique de transactions horodatées cohérent sur lequel tous les nœuds peuvent, en principe, s'accorder. »

DU TROC AUX BANQUES CENTRALES

Le **troc** peut se pratiquer au sein d'un groupe entre des personnes partageant **les mêmes conventions**

L'usage d'un **bien de référence** facilite les échanges.
L'invention de la **monnaie fiduciaire** introduit une **référence** dont la valeur est garantie par un **tiers de confiance**.
Cela implique de faire confiance aux **banques** et aux états **émetteurs** de cette monnaie.

DE LA POSSESSION À LA DÉLÉGATION

La **preuve** de **possession**

QUIZZ

Comment je prouve que ce bien m'appartient ?

1. Parce que je le porte
2. Parce qu'un ami peut témoigner qu'il est à moi
3. Parce que j'en délègue la preuve à une **institution de confiance**

banque

acte notarié

cadastre

propriété intellectuelle

La **délégation** implique la **confiance**

NOTRE SYSTÈME D'ÉCHANGE

Usage d'une **monnaie** dont la **création** et la **valeur** sont déléguées à un **tiers de confiance**

Production d'une **preuve de possession** dont la gestion est déléguée à un **tiers de confiance**

La **blockchain** est une technologie qui permet :

- La **création** de **crypto-monnaie**
- Une **unité d'échange** dont la **valeur** est déterminée par **l'offre et la demande** sur des places de marché
- L'**échange** par **transaction**
- L'**enregistrement** horodaté de **preuves**
- La **confiance** repose sur un mécanisme de **consensus**

VALEUR DU BITCOIN

LE CONSENSUS TRANSPARENT

Soit un **groupe d'utilisateurs**, chacun avec leur intérêt, souhaitant effectuer des **échanges**.

Dans un **système traditionnel**, ces personnes aurait besoin d'un **tiers de confiance** (un banquier) pour tenir un **livre de compte** qui ferait **référence**.

Dans une société où plusieurs groupes co-existent, chacun avec son banquier, les différents banquiers vont devoir **s'accorder ensemble**. Acteurs incontournables, les banquiers seront rapidement tentés d'imposer leurs propres règles.

La technologie **Blockchain** propose que **chaque utilisateur** détienne **une copie du livre de compte**.

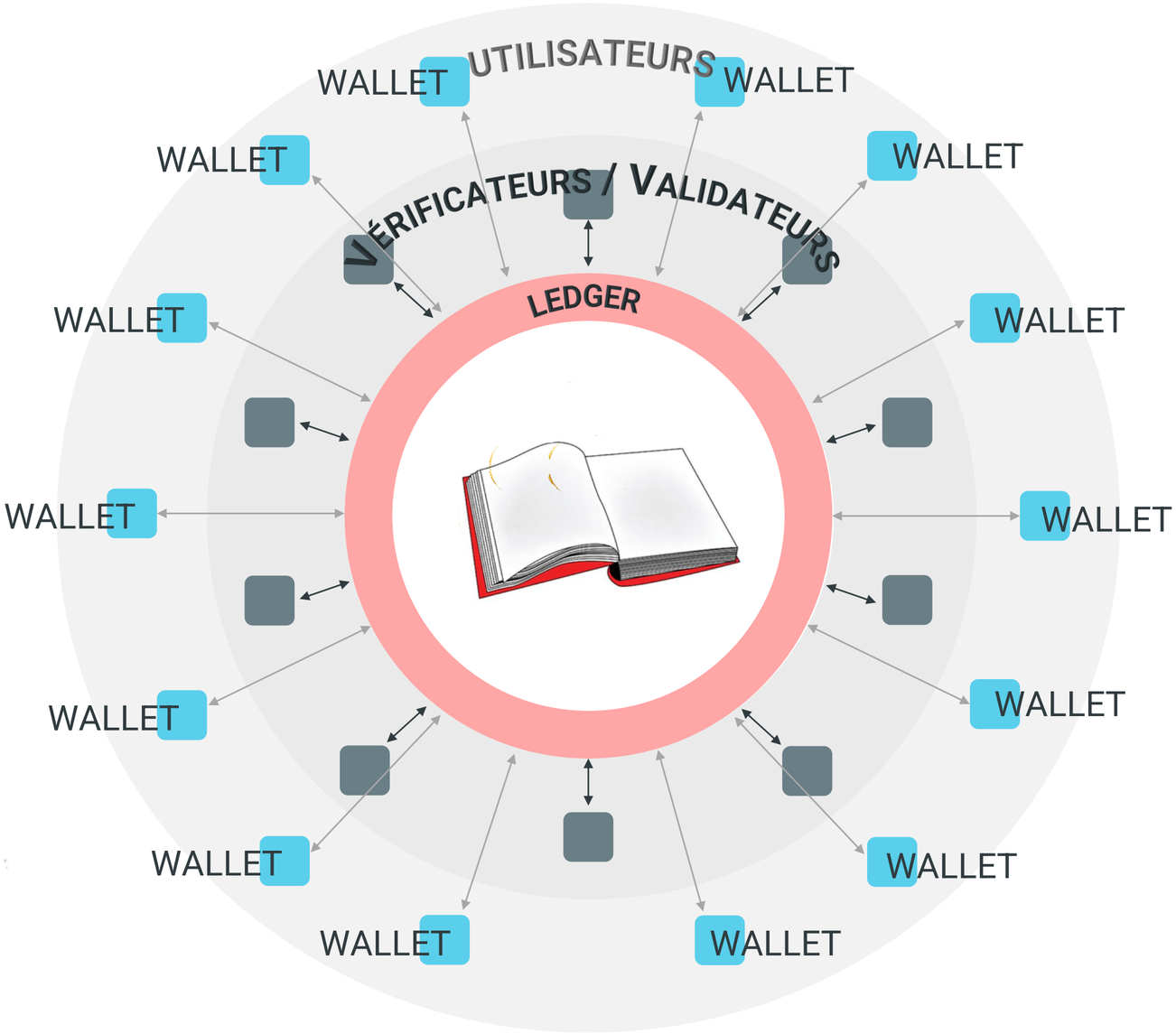
A chaque nouvel ajout, les différentes copies du livre de compte sont **(presque) immédiatement** mises à jour. Les utilisateurs ont alors besoin de **s'accorder** sur ce qui est **enregistré** dans le livre de compte, car une fois écrit, le contenu devient **infalsifiable**.

UN SYSTÈME DISTRIBUÉ

PERMISSIONLESS PUBLIC

PERMISSIONED PUBLIC

PERMISSIONED PRIVATE



UN SYSTÈME EN COUCHES

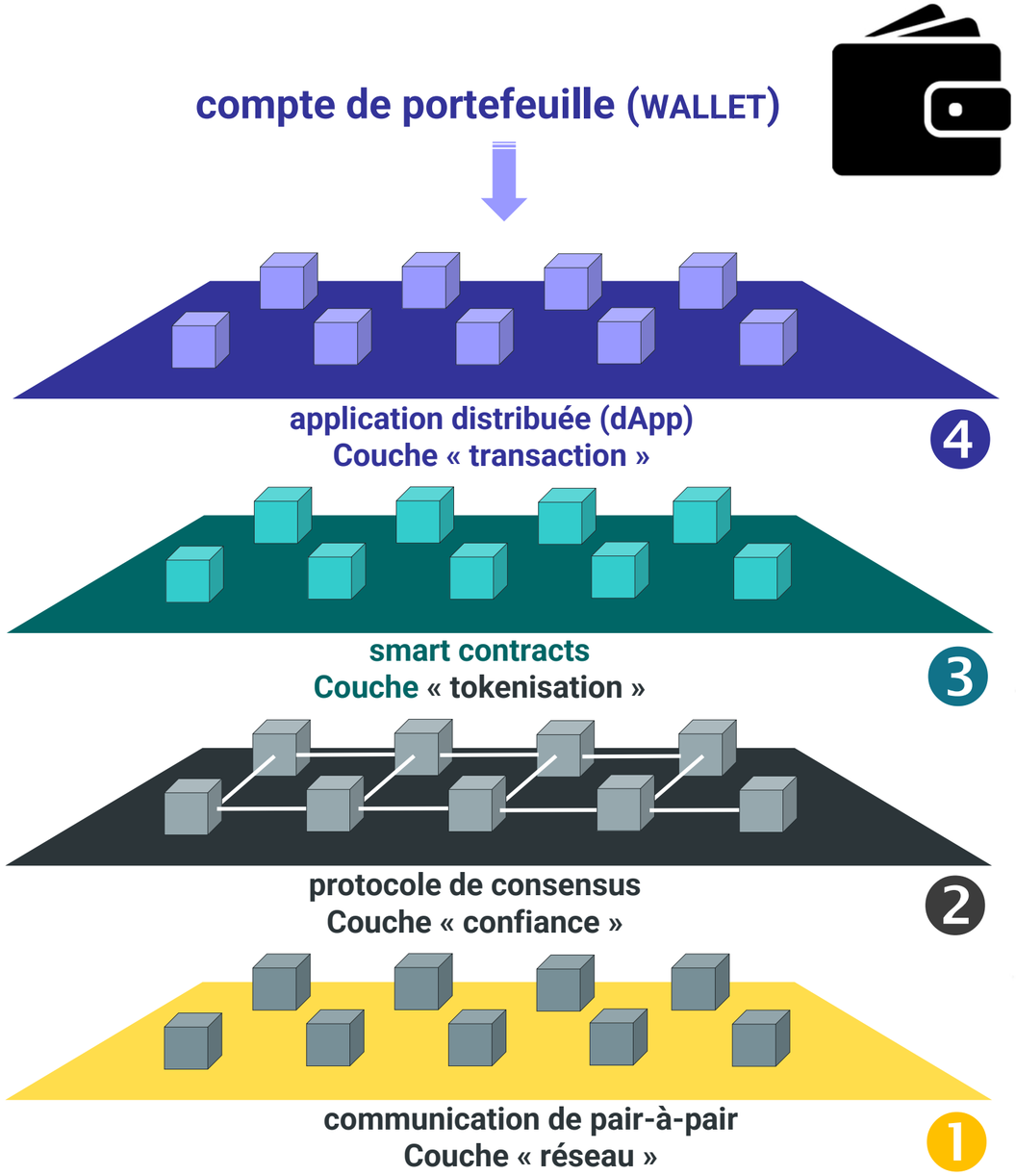
RÔLES

UTILISATEUR
DÉVELOPPEUR

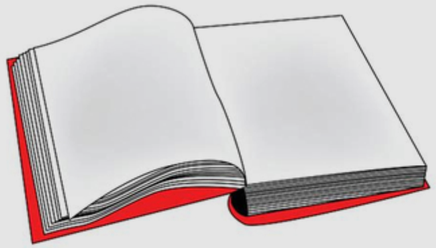
VÉRIFICATEUR
MINEUR

VALIDATEUR

ROUTEUR



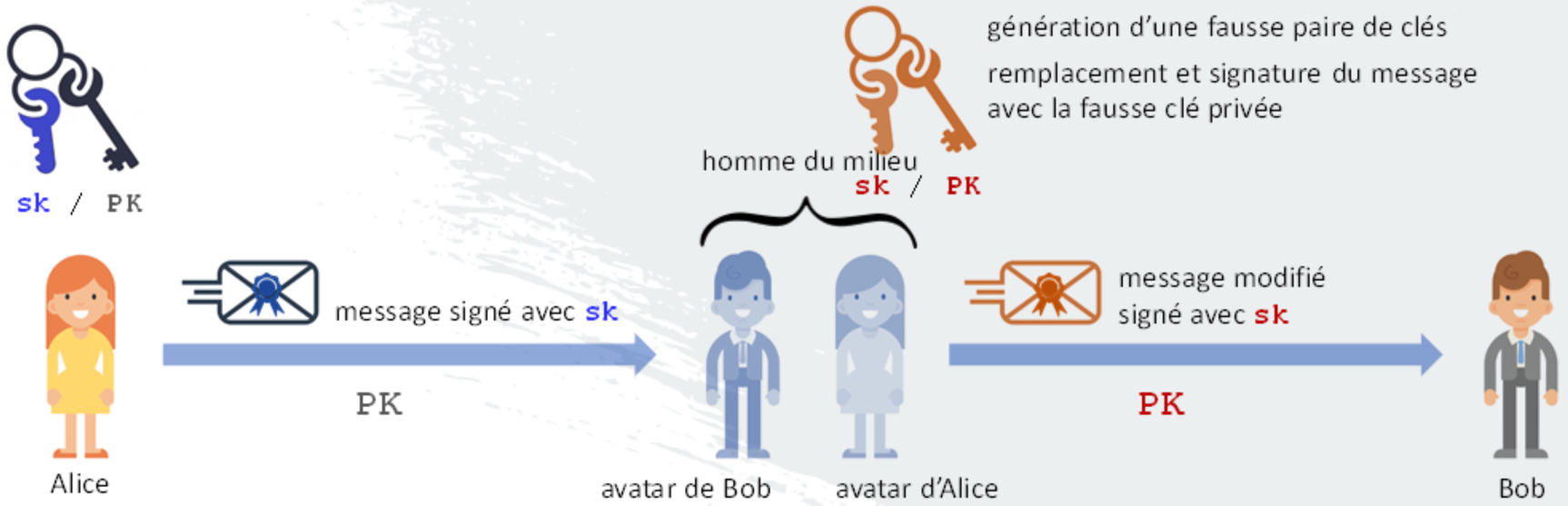
livre de comptes
(LEDGER)



« un dispositif d'enregistrement électronique partagé permettant l'authentification de ces opérations »

AUTHENTIFICATION PAR CRYPTOGRAPHIE ASYMÉTRIQUE

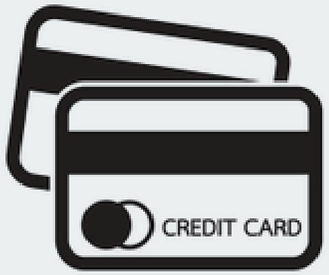
Quand Alice s'authentifie auprès de Bob…

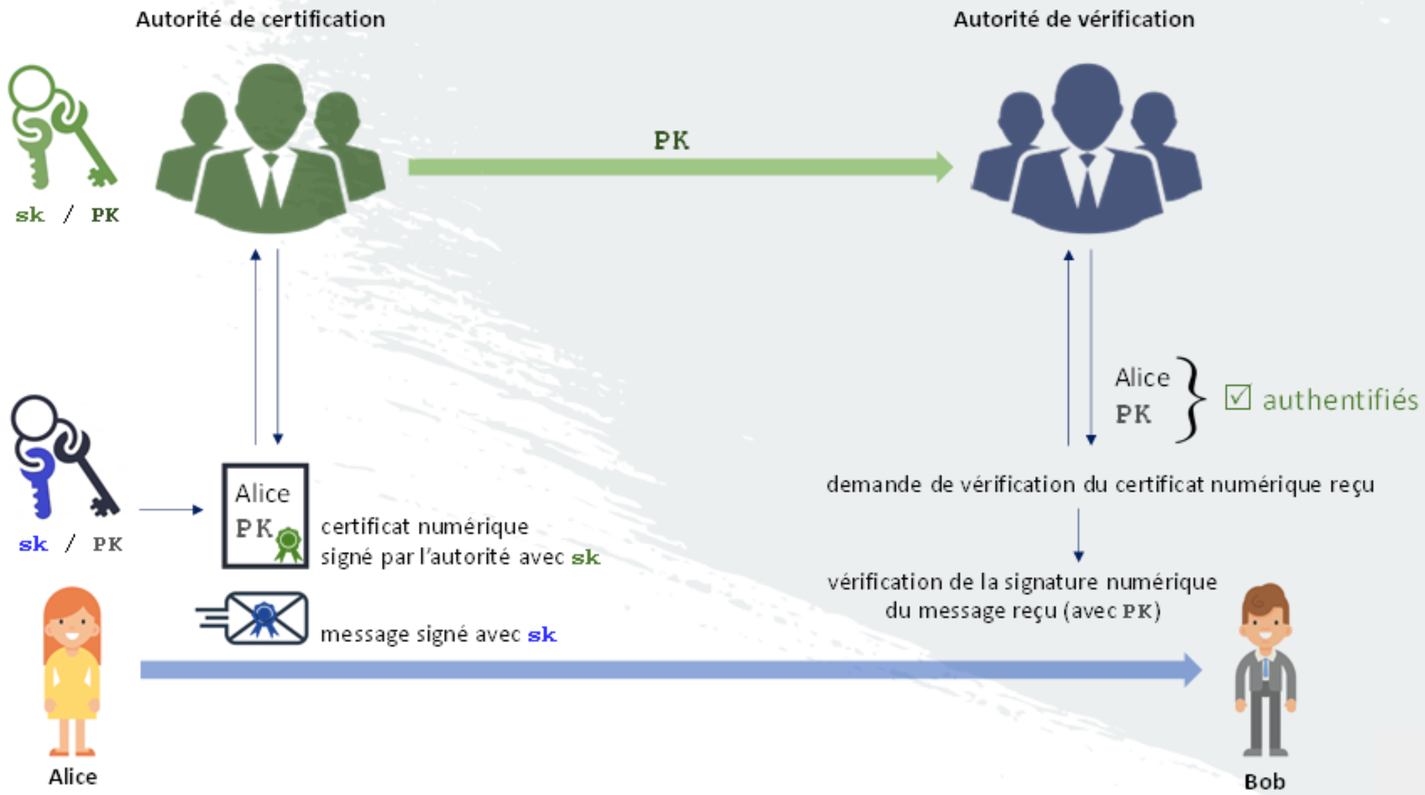


Sans identification des parties, une attaque « *man-in-the-middle* » est possible

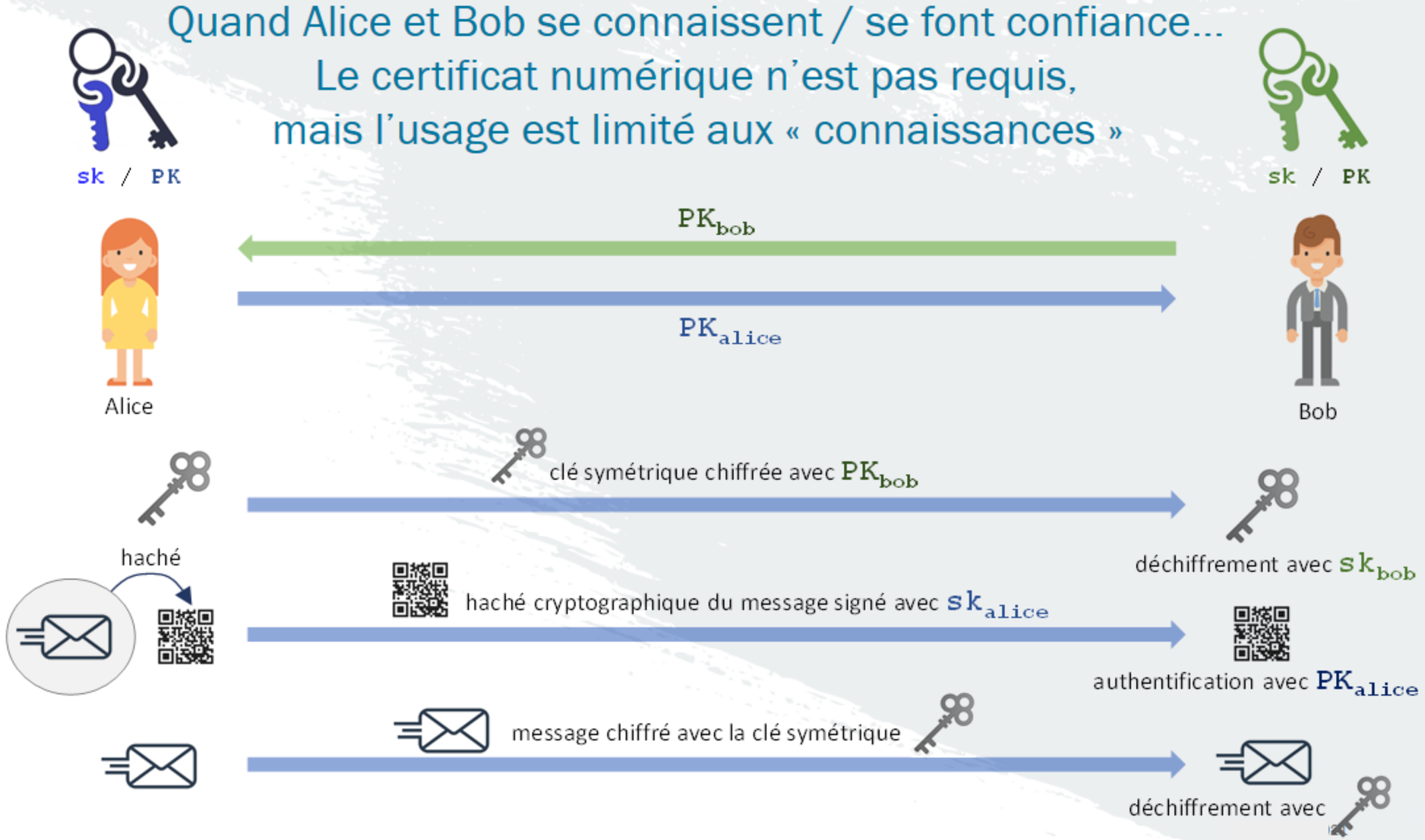
CERTIFICAT NUMÉRIQUE ET PKI

Quand Alice et Bob ne se connaissent pas / ne se font pas confiance...
Une autorité certifie à Bob que son interlocuteur, authentifié par PK, est Alice
Le certificat numérique identifie les parties



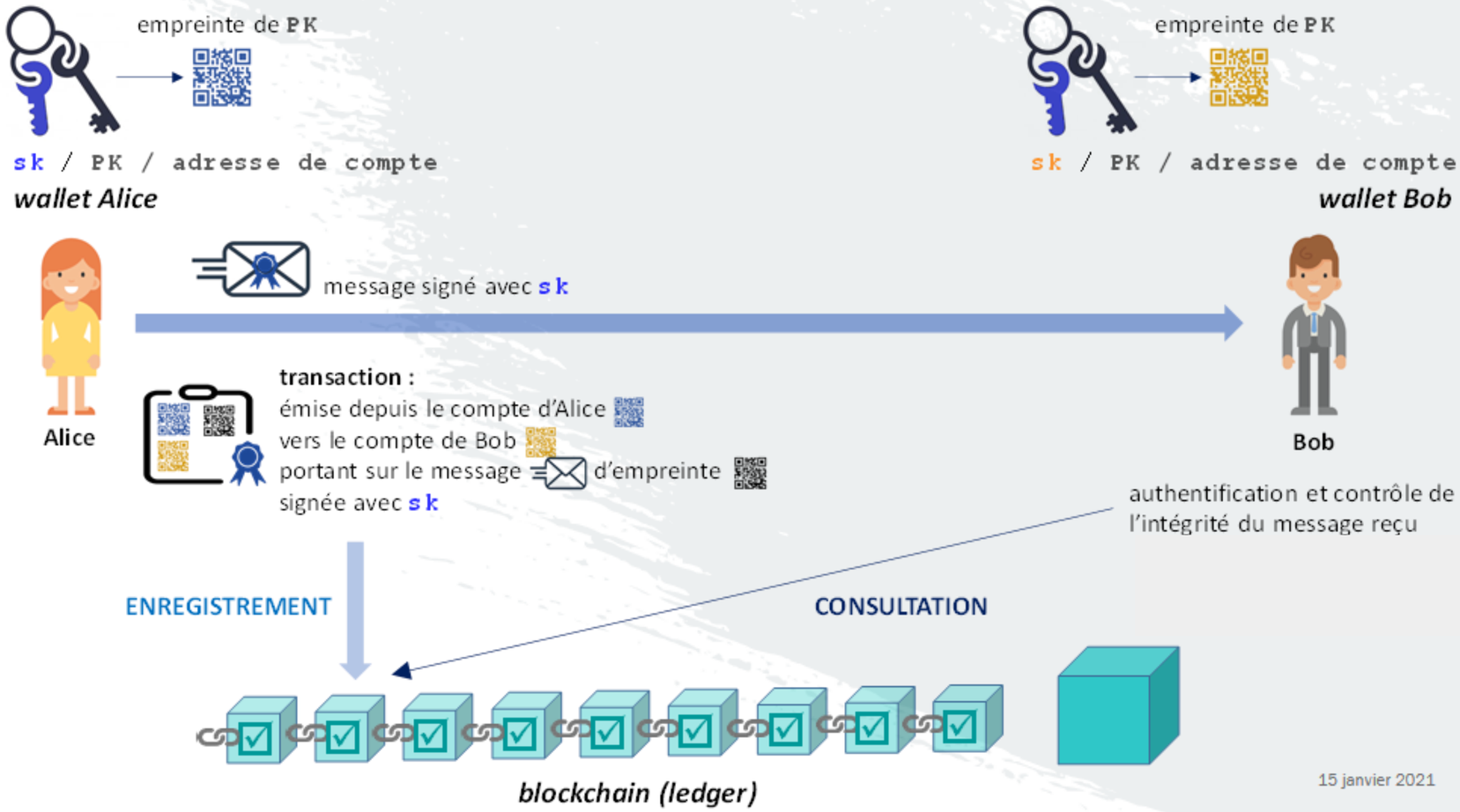


PRETTY GOOD PRIVACY



AUTHENTIFICATION SUR UNE BLOCKCHAIN

ouverte au « world wide ledger » et robuste aux attaques par « man-in-the-middle »



15 janvier 2021

NANOELEC.

https://www.interieur.gouv.fr/Actualites/L-actu-du-Ministere/La-technologie-Blockchain-une-revolution-pour-l-identification

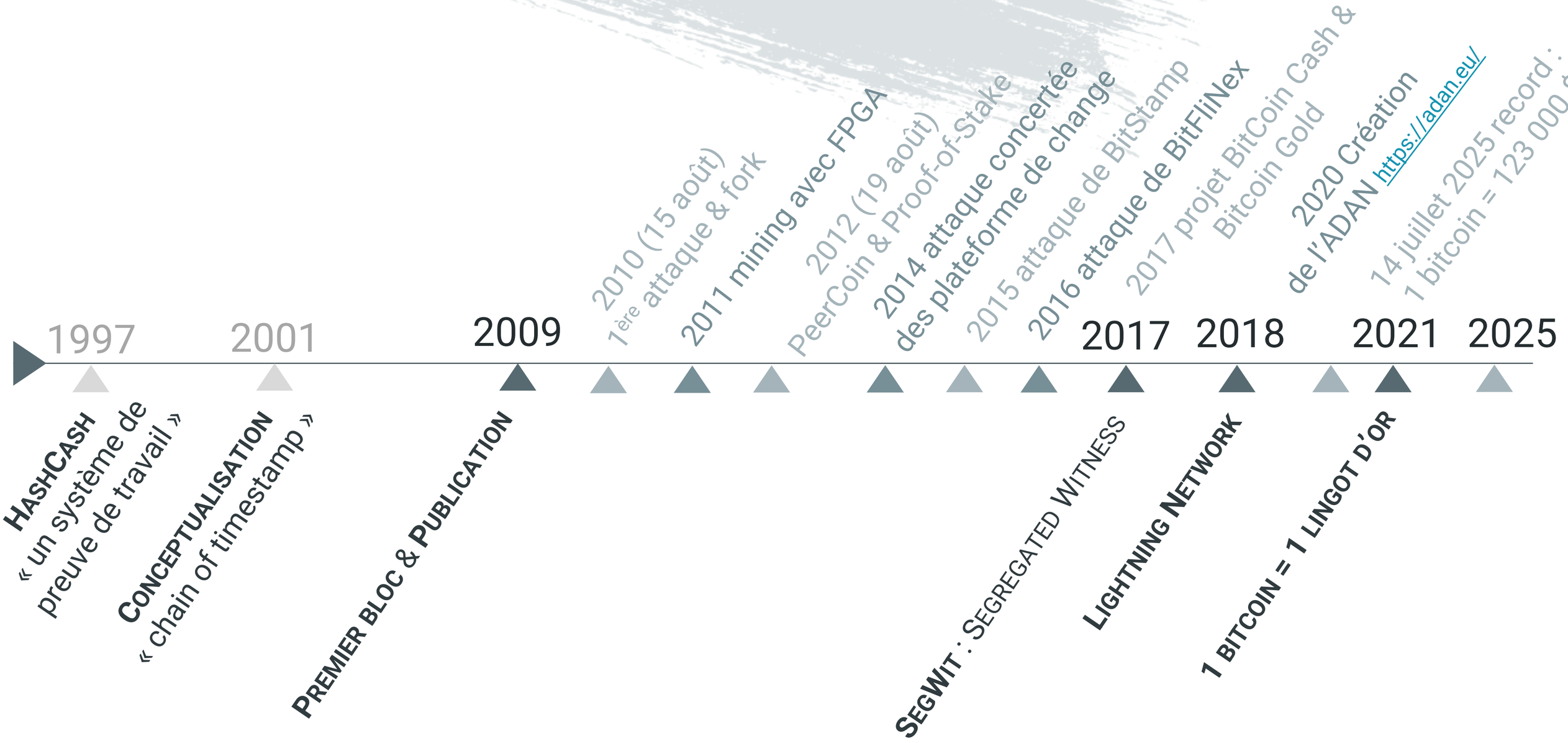
QUELQUES RÉFÉRENCES

1. « sec2, elliptic curve cryptography », Certicom Research, 2009
2. https://e-ducat.fr/links/ecdsa/
3. Pascal Lafourcade, Jean-Guillaume Dumas, Ariane Tichit, Sébastien Varette, « Les blockchains en 50 questions », Eyrolles, 2019, EAN13 : 9782100800896

SECTION 2

BITCOIN

UN PROJET OPEN-SOURCE



PROCESSUS D'UNE TRANSACTION

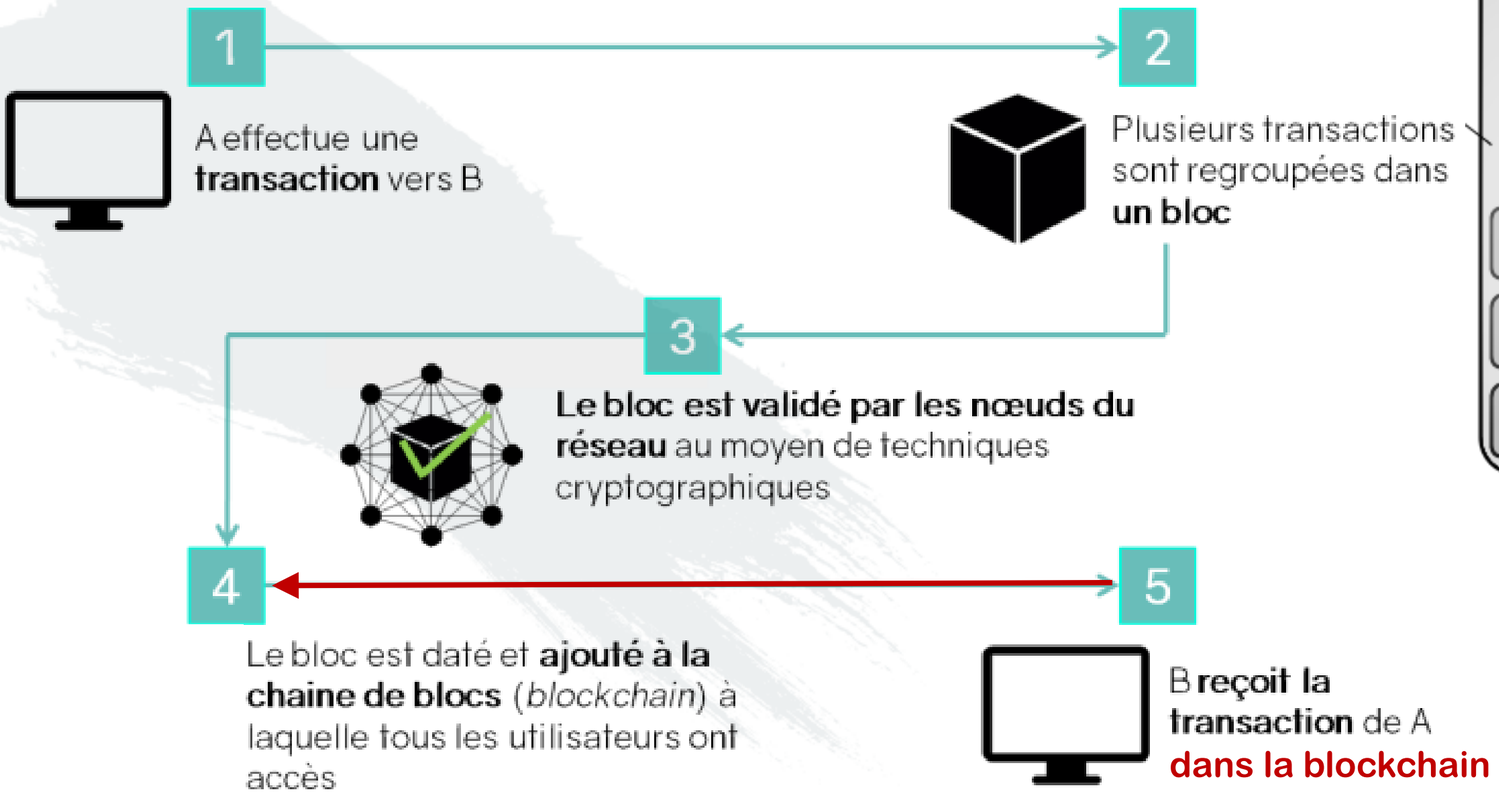
Alice et **Bob** possèdent un portefeuille de valeur (crypto-monnaie).
Alice achète un bien (ou service) à **Bob** et procède à son règlement via une transaction BitCoin

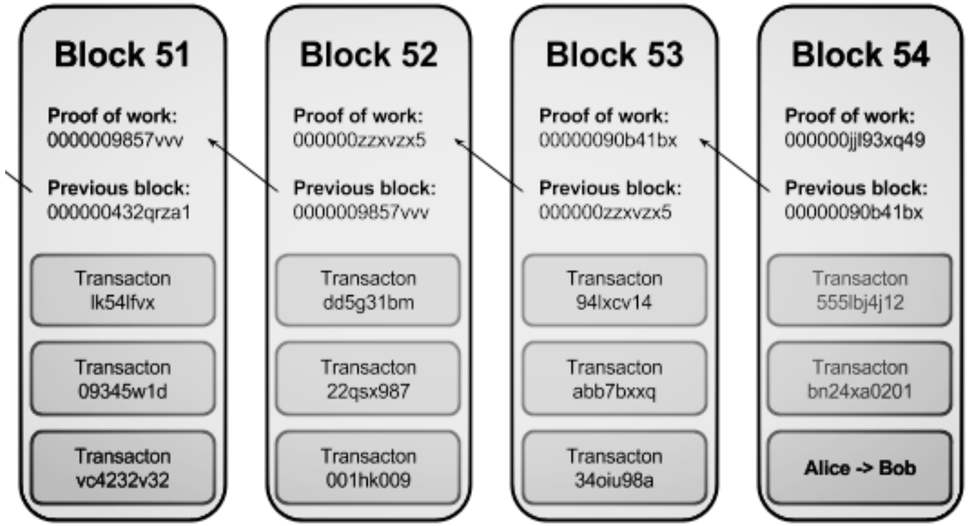
Le **processus de paiement** se déroule en 5 étapes :

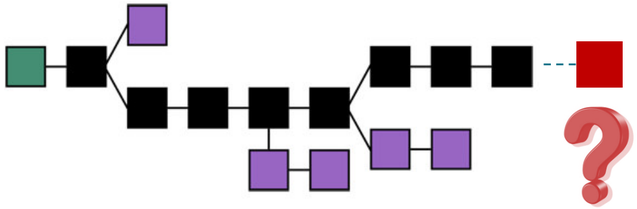
1. Bob présente la facture à Alice et une demande de paiement
2. Alice prépare la transaction et la signe
3. La transaction signée est diffusée sur le réseau BitCoin
4. Un validateur la vérifie et l'intègre à la blockchain
5. Le paiement est effectif, Bob reçoit la valeur à son adresse de compte

Ce processus est **transparent** et chacun peut vérifier l'état de la transaction dans le registre
Le bien (ou service) est (généralement) transmis hors blockchain

COMMENT FONCTIONNE BITCOIN ?



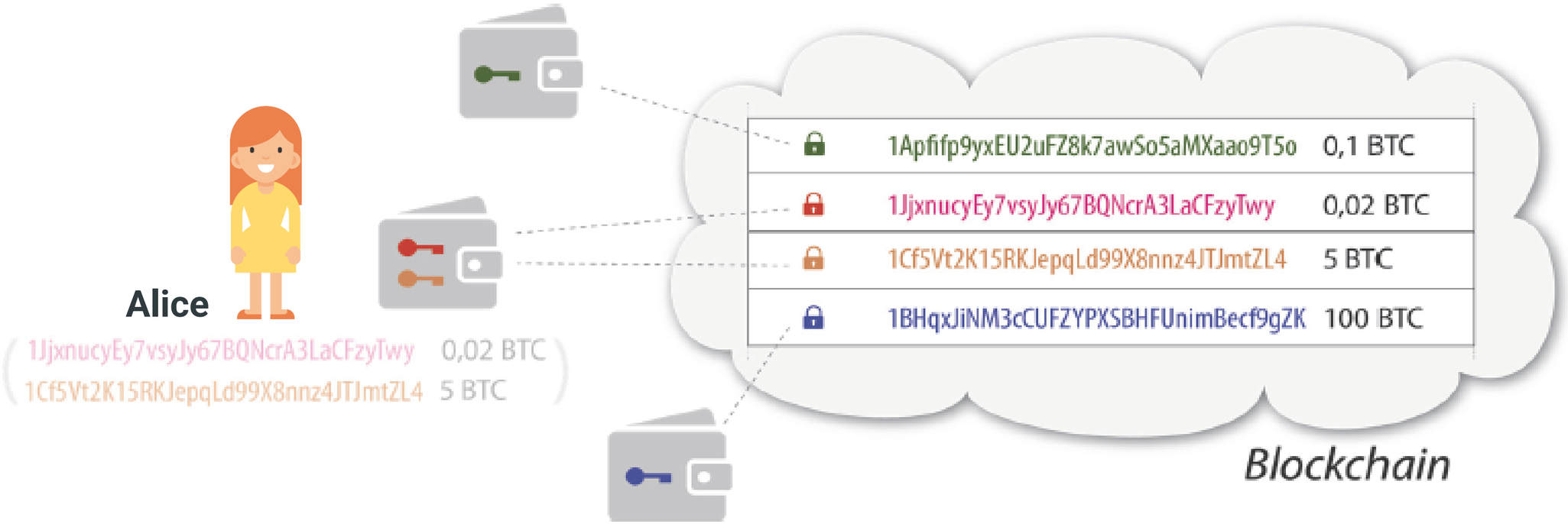




BLOCKCHAIN FRANCE

QU'EST-CE QU'UN COIN ?

La **crypto-monnaie** est gérée via un **portefeuille électronique**.
Un **Coin** est rattaché à une **adresse** gérée par le portefeuille.
Il est **enregistré** dans la blockchain et est **protégé** par un **verrou**.
Pour être dépensé dans une transaction, il faut posséder la **clé** de son verrou.
Le **portefeuille** d'Alice ne contient pas de Coin, mais des **clés** permettant de les dépenser.



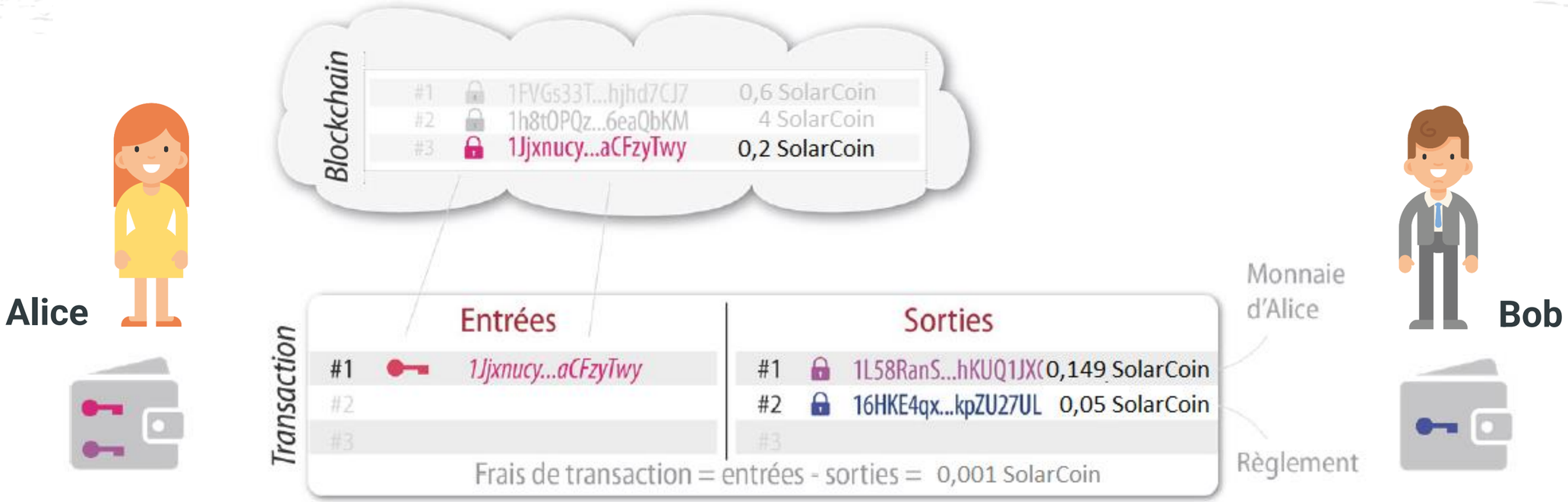
PRÉPARATION D'UNE TRANSACTION

Bob génère une **adresse** et une **clé de déverrouillage**.
L'adresse est codée et présentée à **Alice**.
La **clé de déverrouillage** est conservée **secrète** par **Bob**.
Pour des raisons de **sécurité**, une adresse n'a pas vocation à être utilisée plusieurs fois.

CONTENU D'UNE TRANSACTION

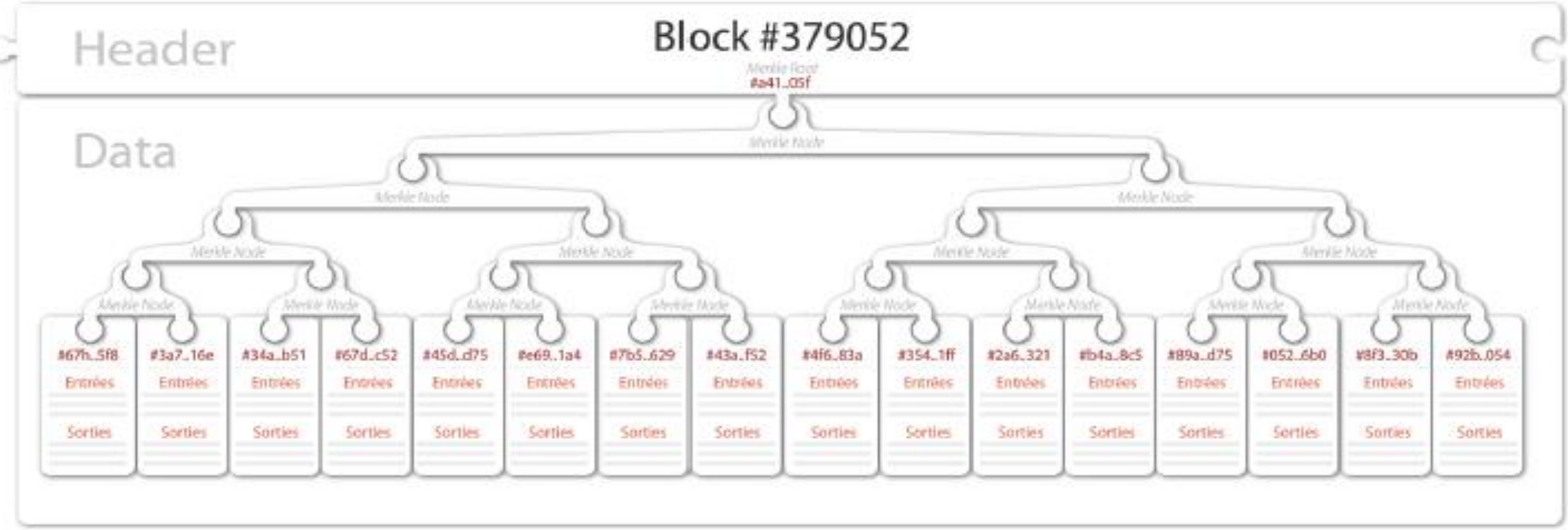
La **transaction** est composée :

- une **adresse « entrée »** qui fait référence à des Coin détenus par **Alice** enregistrés dans la blockchain
- une **adresse « sortie UTXO »** qui est l'adresse envoyée par **Bob**
- une **adresse « sortie UTXO »** pour que **Alice** récupère la monnaie
- éventuellement de **frais de transaction** pour les **mineurs**



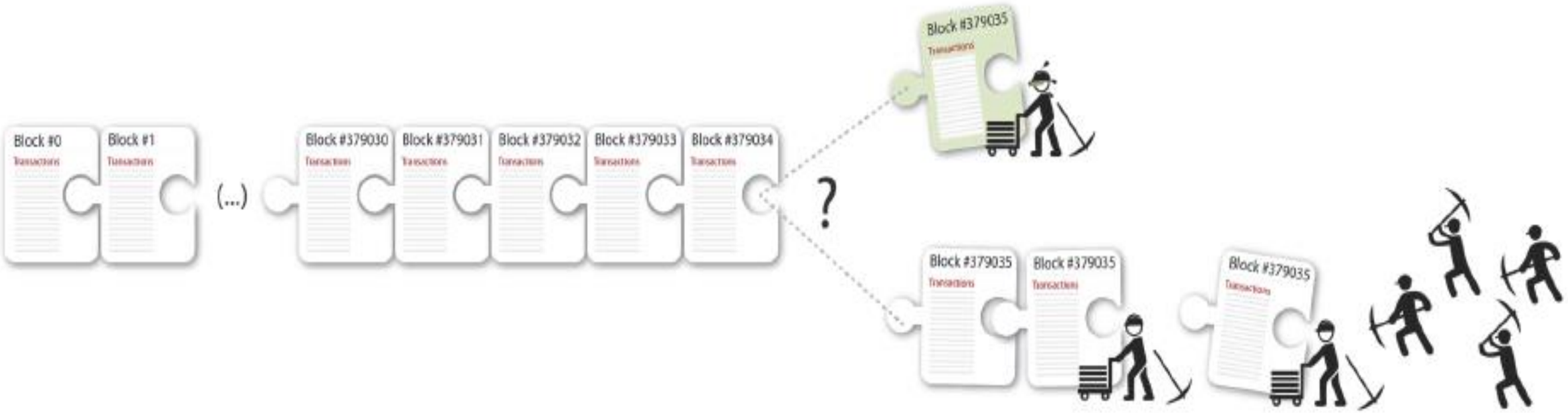
ENREGISTREMENT D'UNE TRANSACTION

La **transaction** est ajoutée au **bloc** des transactions courantes.
Toutes les transactions du bloc sont **vérifiées**.
Chaque **« entrée »** d'une transaction est nécessairement la **« sortie UTXO »** non dépensée d'une autre transaction.
Une transaction n'est **acceptée** que si ses entrées contiennent les **clés de déverrouillage** des sorties qu'elle utilise.
La **vérification** est basée sur l'usage de la **cryptographie**.



ENREGISTREMENT D'UN BLOC

Le bloc est **proposé au consensus** pour être accroché à la blockchain.
Ce travail est réalisé par des **mineurs** et fait appel à la **cryptographie**.
Les mineurs touchent les **frais de transaction**.



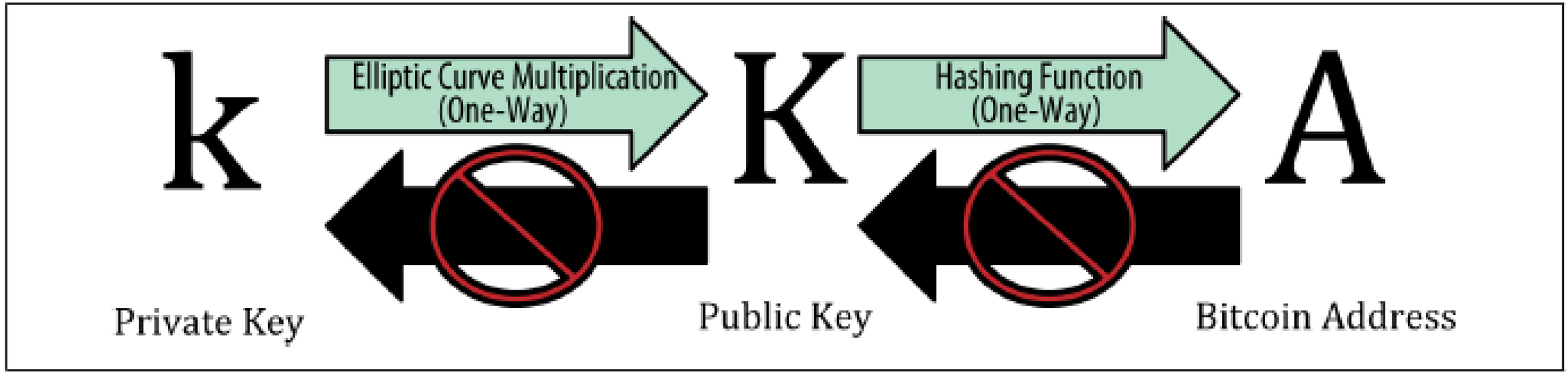
CONFIRMATION D'UNE TRANSACTION

Le **bloc** doit être **validé** par les validateurs.
Un **validateur** vote par le **choix du bloc** qu'il accroche à **sa copie locale du registre**.
Le **bloc gagnant** doit recueillir une **majorité** d'adhésion.
Le **transaction** est **confirmée** lorsque son **bloc** est **recouvert** par au moins 6 autres blocs.

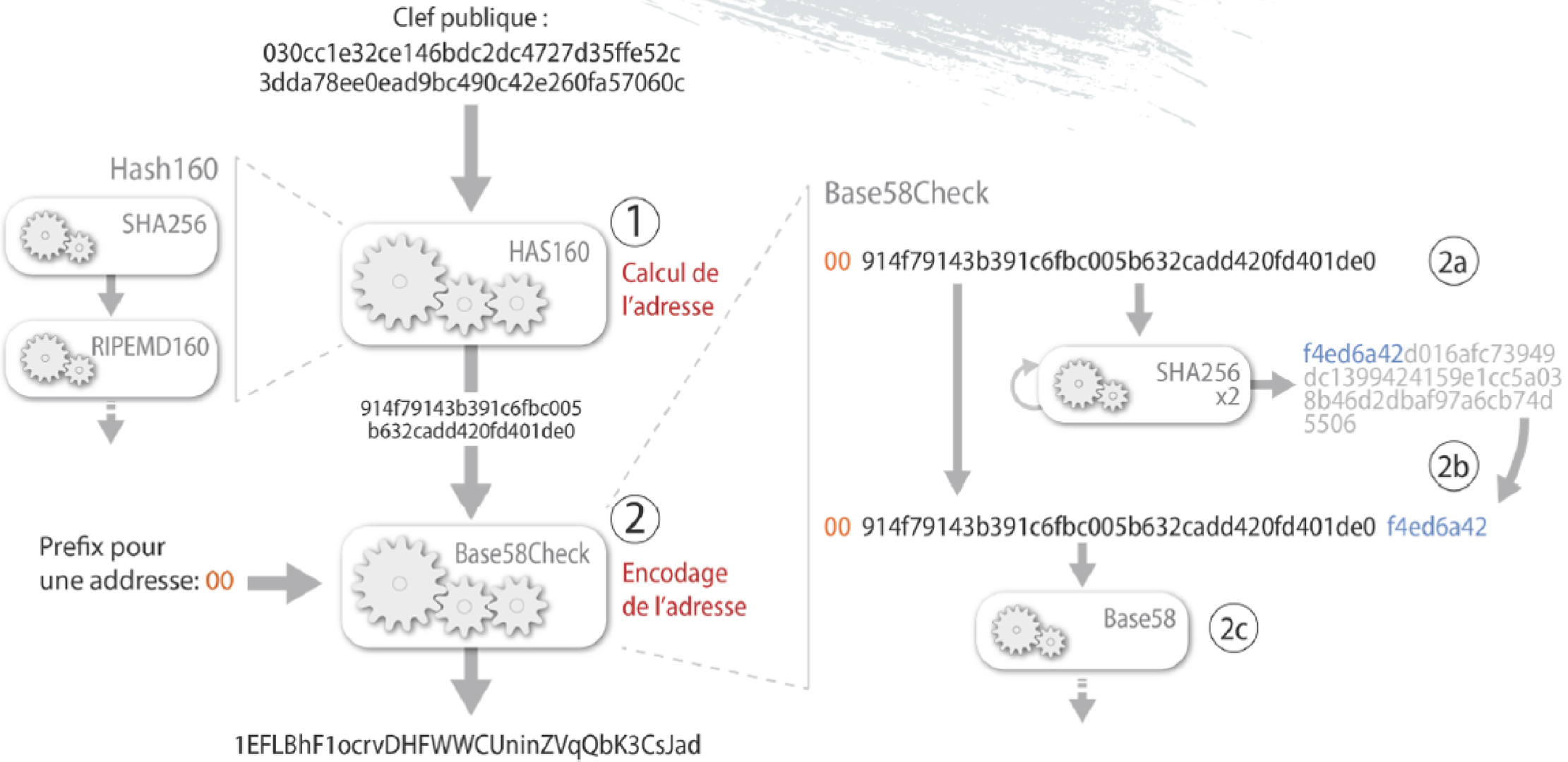
ÉLÉMENTS TECHNIQUES & CRYPTOGRAPHIQUES

CONSTRUCTION DES ADRESSES DE COMPTE

cryptographie sur courbe elliptique **secp256k1**
algorithme de signature **ECDSA**



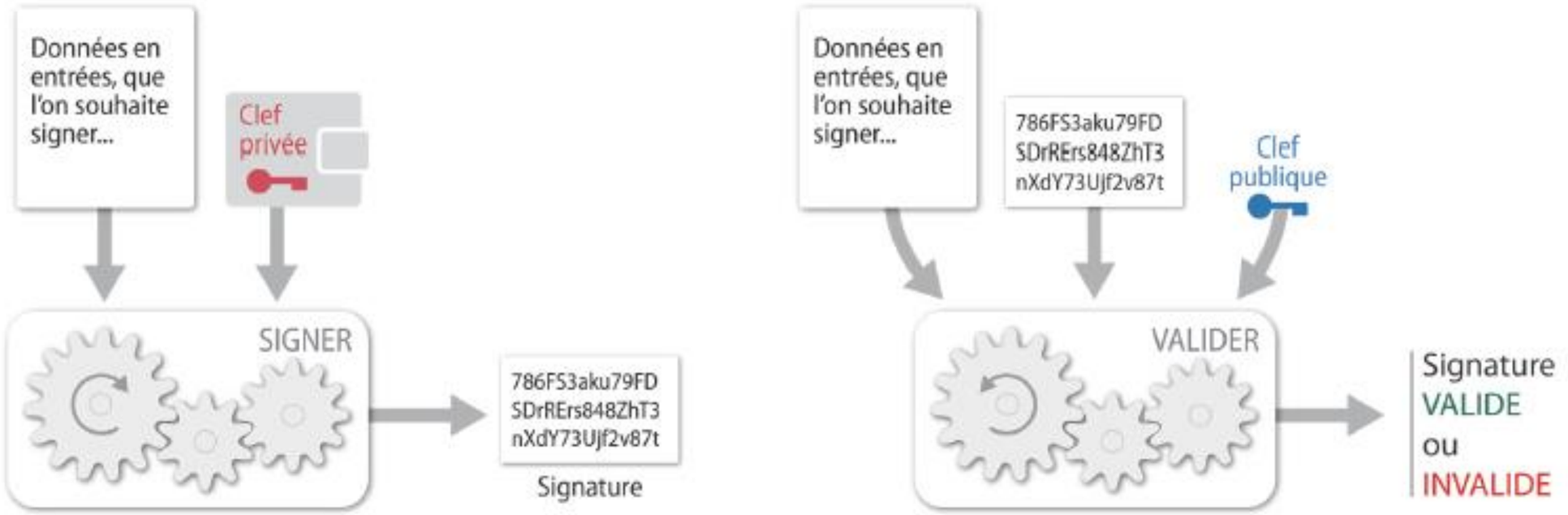
ADRESSE DE COMPTE BITCOIN



SIGNATURE NUMÉRIQUE ECDSA

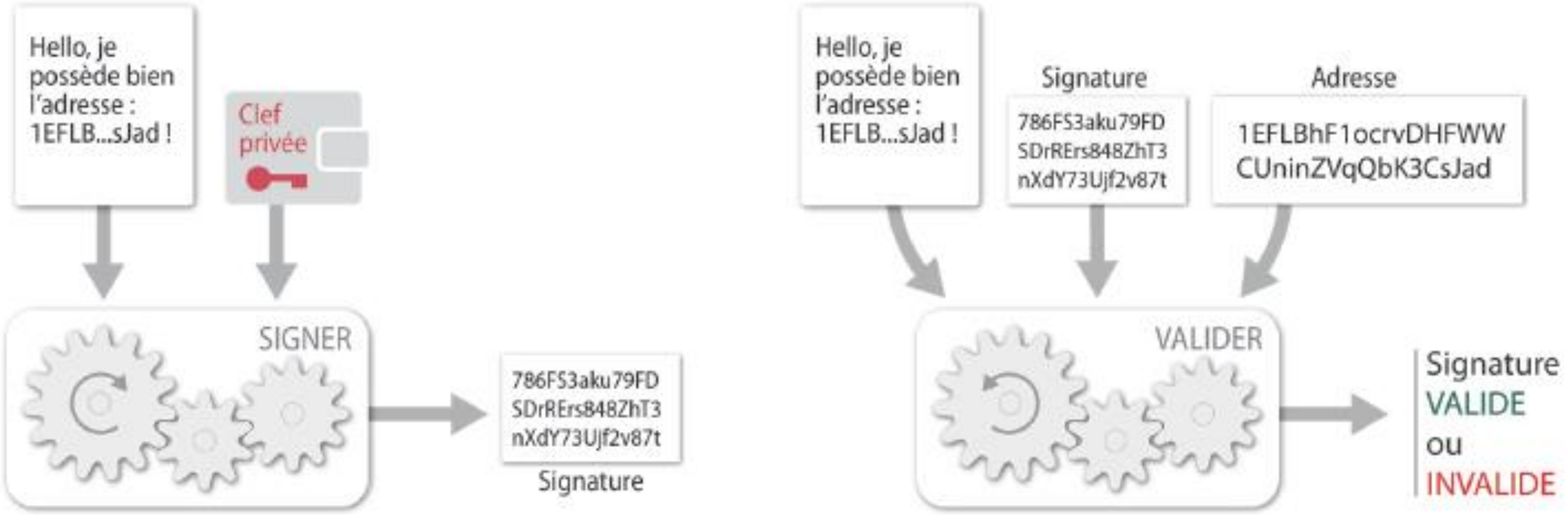
Une **signature numérique** permet de garantir deux **propriétés** :

1) L'**intégrité** du **contenu** signé
2) L'**authentification** du **signataire**



ASTUCE DE LA SIGNATURE BITCOIN

Dans Bitcoin, la **signature numérique** est utilisée pour prouver que l'on est **propriétaire** d'une **adresse de compte** (celle qui contient nos Coins).
Pour cela, il faut prouver que l'on détient la **clé privée** qui a signé la transaction.
Or, l'adresse du compte émetteur est le **haché de la clé publique**.



LOCK SCRIPTS

Dans Bitcoin, **les Coins sont dans la registre** (et non dans le portefeuille), **visibles de tous**, et sont protégés par des **verrous**.
Ces verrous sont de **scripts pré-déterminés**. Ils font parties des règles de Bitcoin.
Ce mécanisme de scripts est **précurseur** des smart contracts.

Le langage utilisé pour les scripts est basé sur **l'empilement d'opérations**.
Il est **minimaliste** et ne comporte pas la possibilité d'effectuer des boucles.
Il n'est pas **« turing-complet »**.

L'**interpréteur** suit les règles suivantes :

- Le script est lu de la gauche vers la droite
- Les constantes sont empilées lorsqu'elles sont rencontrées
- Les instructions (**OP**erators) peuvent empiler ou dépiler plusieurs éléments, les traiter et empiler le résultat
- Les opérateurs logiques évaluent une condition et empilent le résultat : TRUE = 1 et FALSE = 0

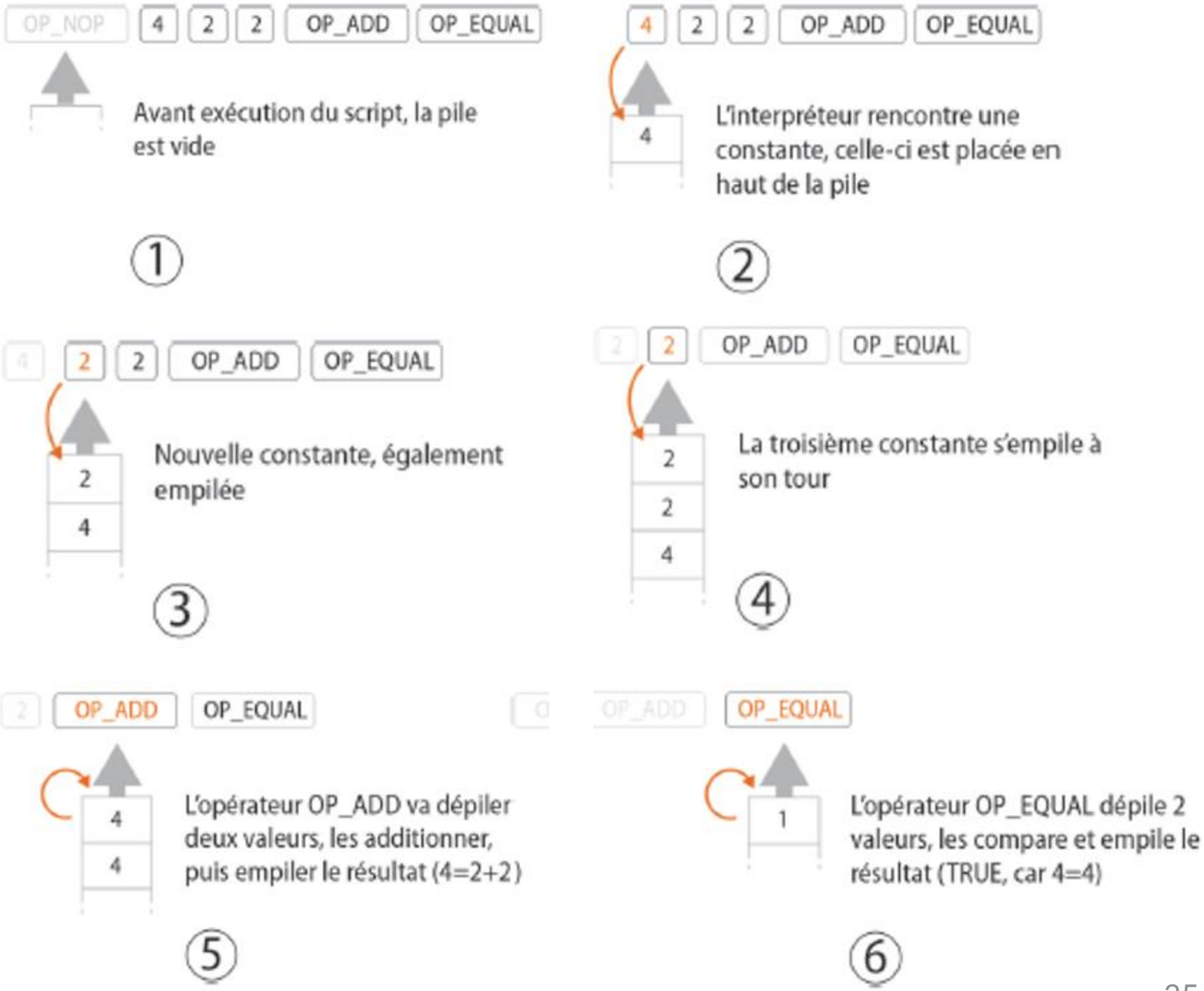
INTERPRÉTATION BAS-NIVEAU DANS LA PILE

Dans cet exemple :

On effectue une **addition** de 2+2 et on **compare** le résultat à la valeur attendue 4.

Deux opérateurs sont utilisés :

- **OP_ADD** pour l'addition
- **OP_EQUAL** pour la comparaison



LOCK SCRIPTS STANDARD

Scripts	*Description*
P2PKH	Pay-to-Public-Key-Hash ou paiement contre le hash d'une clef publique (adresse). Ces scripts sont aujourd'hui utilisés dans la majorité des transactions. Loc script : OP_DUP OP_HASH160 <Address> OP_EQUALVERIFY OP_CHECKSIG Unlock script : <signature> <Public Key>
P2PK	Pay-to-Public-Key ou Paiement contre une clef publique Version historique et plus simple du script P2PKH, mais qui nécessite davantage de place. En désuétude. Loc script : <Pub Key> OP_CHECKSIG Unlock script : <signature>
Multi-Signature	Permet de définir une condition telle que M signatures doivent être présentées sur N *Exemple :* Une sortie est protégée par 5 clefs publiques mais ne pourra être dépensée que sur présentation de 2 des 5 signatures. Loc script : M <public Key 1> <public Key 2> . . . <public Key N> N OPO_CHECKMULTISIG Unlock script : 0 <signature1> . . . <signature M>
P2SH	Pay-to-Script-Hash ou paiement contre le hash d'un …script de rachat. Le script de verrouillage n'est plus enregistré dans la sortie, mais devra être envoyé lors du déverrouillage. Seule l'empreinte du script est fournie lors de la construction de la transaction, laquelle pourra être encodée dans une adresse de type P2SH [12]. Loc script : OP_HASH160 <script Hash> OP_EQUAL Unlock script : <script arguments> <script>
OP_RETURN	L'opérateur OP-RETURN permet d'intégrer des données dans une transaction et *in fine* dans la blockchain. Une sortie comportant une instruction OP_RETURN ne peut être dépensée. La taille des données est actuellement limitée à 40 octets. Cet opérateur permet d'utiliser la blockchain à des fins de notariat électronique. Lock script : OP_RETURN <data> Unlock script : aucun

SCRIPTS P2PKH 1/6

Le script de verrouillage **Pay-to-Public-Key-Hash** est le plus utilisé pour le paiement.
Son **déverrouillage** requiert une **clé publique** et la **signature** effectuée avec la clé privée de l'adresse de compte correspondant.

On regarde la **transaction** avec laquelle Alice paie son kWatt d'énergie

txid : b36e7dedc141675e09e2b3b6ad05a016cb2a90f540713548ca102070de74c2de
Visible via : https://blockexplorer.com/api/tx/b36e7ded(...)de74c2de

On voit que la sortie vers l'adresse de Bob comporte le script de verrouillage suivant

scriptPubkey :
OP_DUP OP_HASH160 39ee7f7d4c80307d27e7657cbfef0d48d0eb84e5 OP_EQUALVERIFY OP_CHECKSIG

avec l'adresse de Bob en hexadécimal

39ee7f7d4c80307d27e7657cbfef0d48d0eb84e5 **(hexa)**
16HKE4qxjfNVpWCpe6DL2rSZVTkpZU27UL **(WIF, Base58check)**

Le script de déverrouillage requiert

scriptSig :
<signature de Bob> <clef publique de Bob>

SCRIPTS P2PKH 2/6

Dans la gestion des verrous de Bitcoin, deux scripts sont utilisés :

- Les scripts de **verrouillage : scriptPubKey**
- Les scripts de **déverrouillage : scriptSig**

Lors de la **vérification**, l'interpréteur commence par exécuter le script de **déverrouillage** (scriptSig), puis il exécute le script de **verrouillage** (scriptPubKey).

A l'issue de l'exécution, le résultat est un **booléen** : TRUE ou FALSE

Il faut que le résultat soit **TRUE** pour que **la transaction soit incluse dans le bloc**.

SCRIPTS P2PKH 3/6

SCRIPTS P2PKH 4/6

SCRIPTS P2PKH 5/6

<signature de Bob> <clef publique de Bob> OP_DUP OP_HASH160 39ee7f...84e5 OP_EQUALVERIFY OP_CHECKSIG

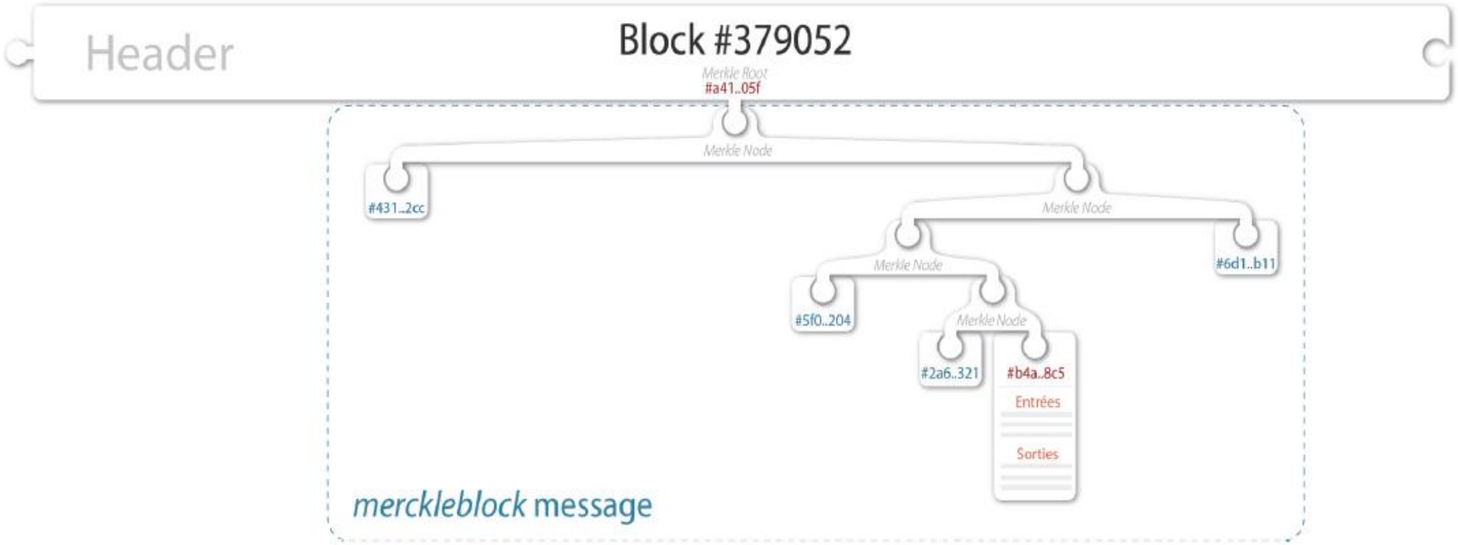
L'adresse de Bob est empilée
Il s'agit de l'adresse brute, sans encodage WIF.
(ie. sans prefix ni entête)

<signature de Bob> <clef publique de Bob> OP_DUP OP_HASH160 39ee7f...84e5 OP_EQUALVERIFY OP_CHECKSIG

Les deux valeurs en haut de la pile sont comparées
En cas de différence, le script est abandonné,
Si les valeurs sont identiques, elles sont retirées de la pile
Cela revient à vérifier que la chef publique envoyée correspond bien à l'adresse de la sortie.

SCRIPTS P2PKH 6/6

OP_CHECKSIG va dépiler la clef publique et la signature, puis vérifier que l'une correspond à l'autre.
Le résultat est empilé (TRUE ou FALSE)



Intégration de la transaction au bloc en construction

SCRIPTS OP-RETURN

Le script de **OP_RETURN** a été ajouté dans Bitcoin pour les applications de « preuve d'existence ».
Il permet d'**enregistrer** jusqu'à 40 octets de **données**.
Il présente la particularité qu'il n'y a pas de sortie indexée dans la base des UTXO (unspend Transaction Output)

Usage classique

```
OP_RETURN  <data length>  <data>
```

Exemple

```
OP_RETURN  28  444f4350524f4f46 55f86e0bf0c27068b068da7bfb9240e7784c7d8e12ba12153aabe3b36cf4136a
```

avec

28 :	longueur (en hexa), soit 40 octets
444f4350524f4f46 :	libellé de l'enregistrement sur 8 octets
55f86e0b…6cf4136a :	haché d'un document sur 32 octets

Application

https://proofofexistence.com/

UNLOCK SCRIPTS : scriptSig

Le mécanisme de **déverrouillage** est également un script comportant une ou plusieurs signatures.

Le rôle de ces signatures est :

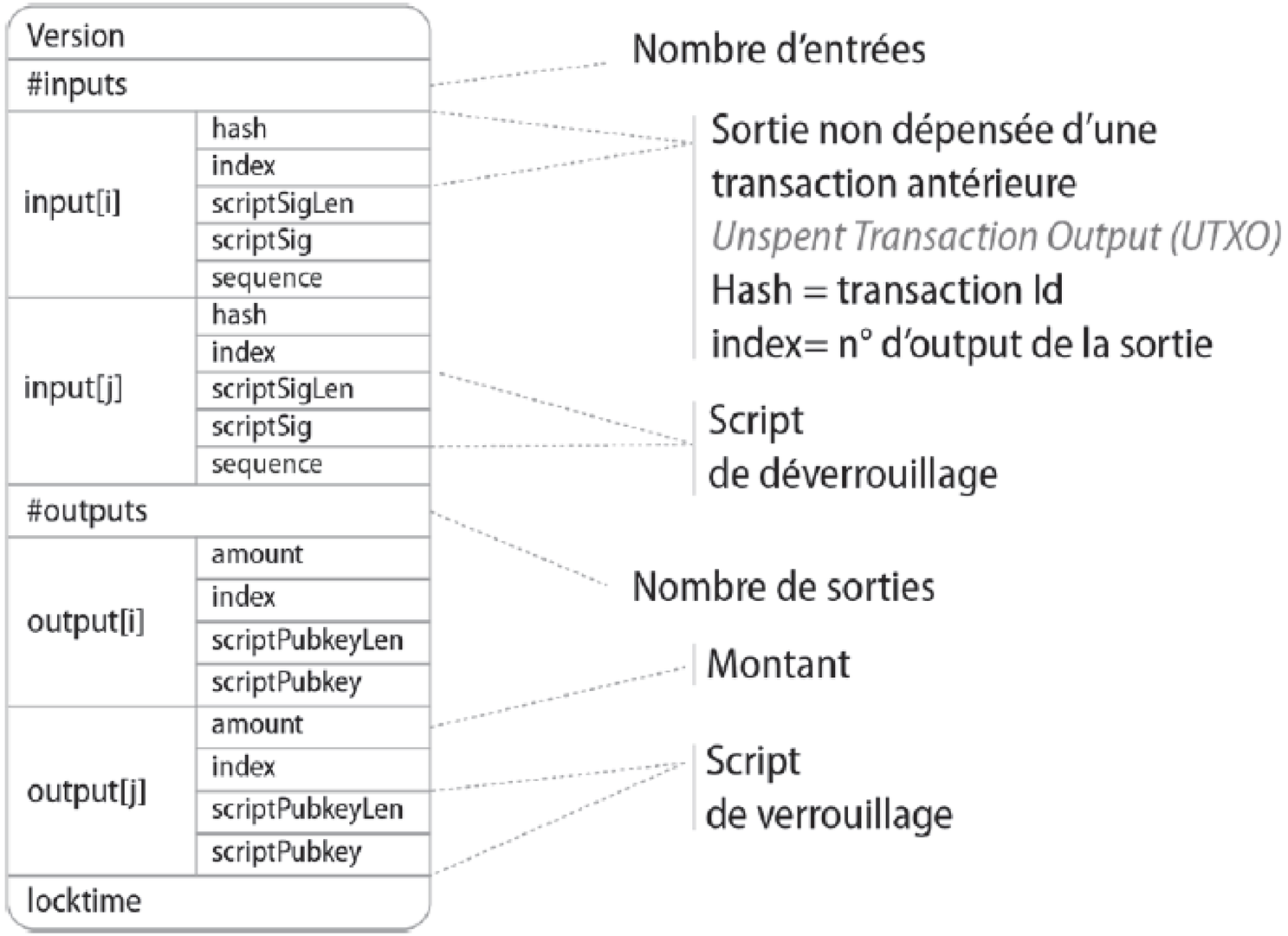
- d'apporter la **preuve de possession** d'une sortie non dépensée (UTXO)
- de **protéger le contenu** de la transaction contre d'éventuelles falsifications

Une fois la transaction construite, elle est diffusée dans le réseau où elle est vérifiée puis éventuellement incluse dans un bloc. Son **intégrité** est **garantie** par la signature numérique **depuis l'émission** de la transaction **jusqu'à l'enregistrement** dans un bloc.

Le script de signature permet de définir la **malléabilité** de la transaction et le type de modifications qu'il est possible de lui apporter

UNLOCK SCRIPTS : scriptSig

Structure d'une transaction Bitcoin



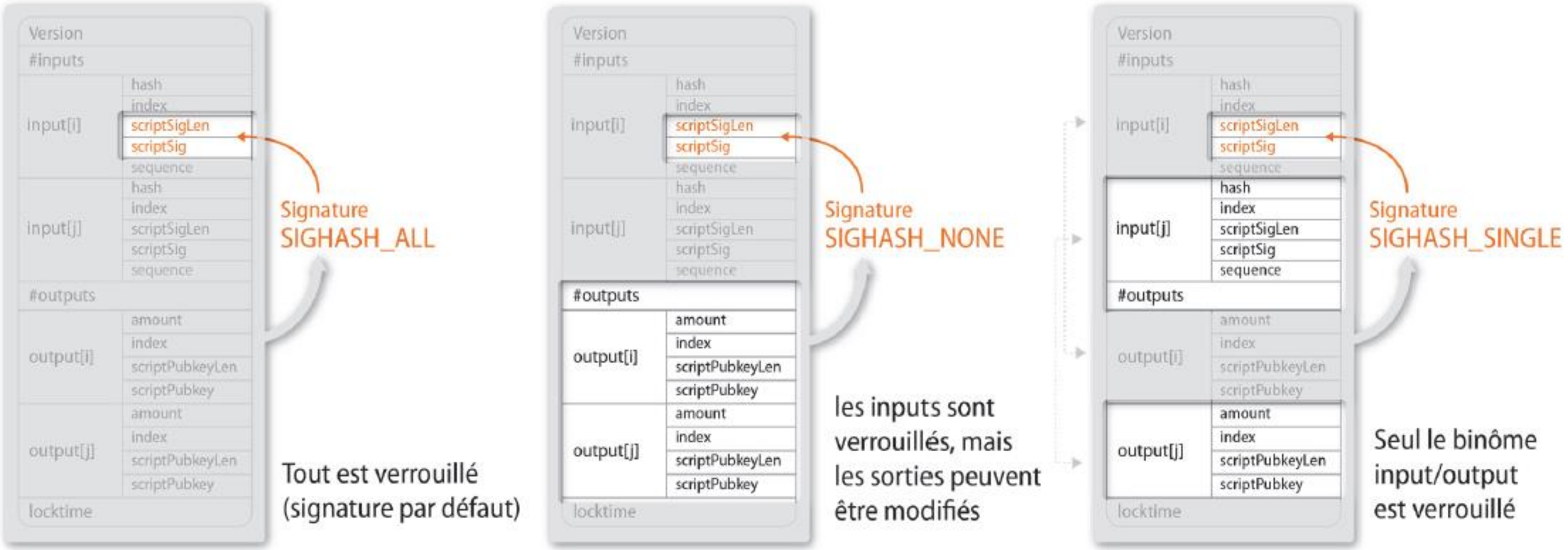
UNLOCK SCRIPTS : scriptSig

Le flag **SIGHASH** indique quelle partie de la transaction est signée avec l'algorithme ECDSA.
Ce mécanisme apporte de la flexibilité dans les usages, c'est la propriété de **malléabilité**.
Au total, six combinaisons peuvent être employées.

flag SIGHASH	Usage
SIGHASH_ALL	Flag utilisé par défaut. Il permet de signer toutes les entrées et sorties utilisées pour construire la transaction. Une fois signée, la transaction ne peut plus être modifiée.
SIGHASH_NONE	Flag permettant de signer toutes les entrées et aucune sortie. Les sorties peuvent donc être modifiées sans invalider la signature.
SIGHASH_SINGLE	Flag permettant de signer l'entrée et la sortie qui partagent le même index.
SIGHASH_ALL | ANYONECANPAY	Les sorties ne peuvent pas être modifiées sans invalider la signature, mais tout le monde peut contribuer en entrée.
SIGHASH_NONE | ANYONECANPAY	Une seule entrée est signée, et aucune sortie. Ce type de signature peut être utilisée pour attribuer des fonds à un destinataire.
SIGHASH_SINGLE | ANYONECANPAY	Les entrée et sortie de même index sont signées et ne peuvent pas être modifiées, mais de nouveaux binômes peuvent être ajoutés.

UNLOCK SCRIPTS : scriptSig

Le numéro de la transaction TXID et le LockTime sont **toujours** signés
Le script de signature scriptSig n'est **jamais** signé



UNLOCK SCRIPTS : scriptSig

Et, avec le flag **ANYONECANPAY** (chacun peut payer)

UNLOCK SCRIPTS : scriptSig

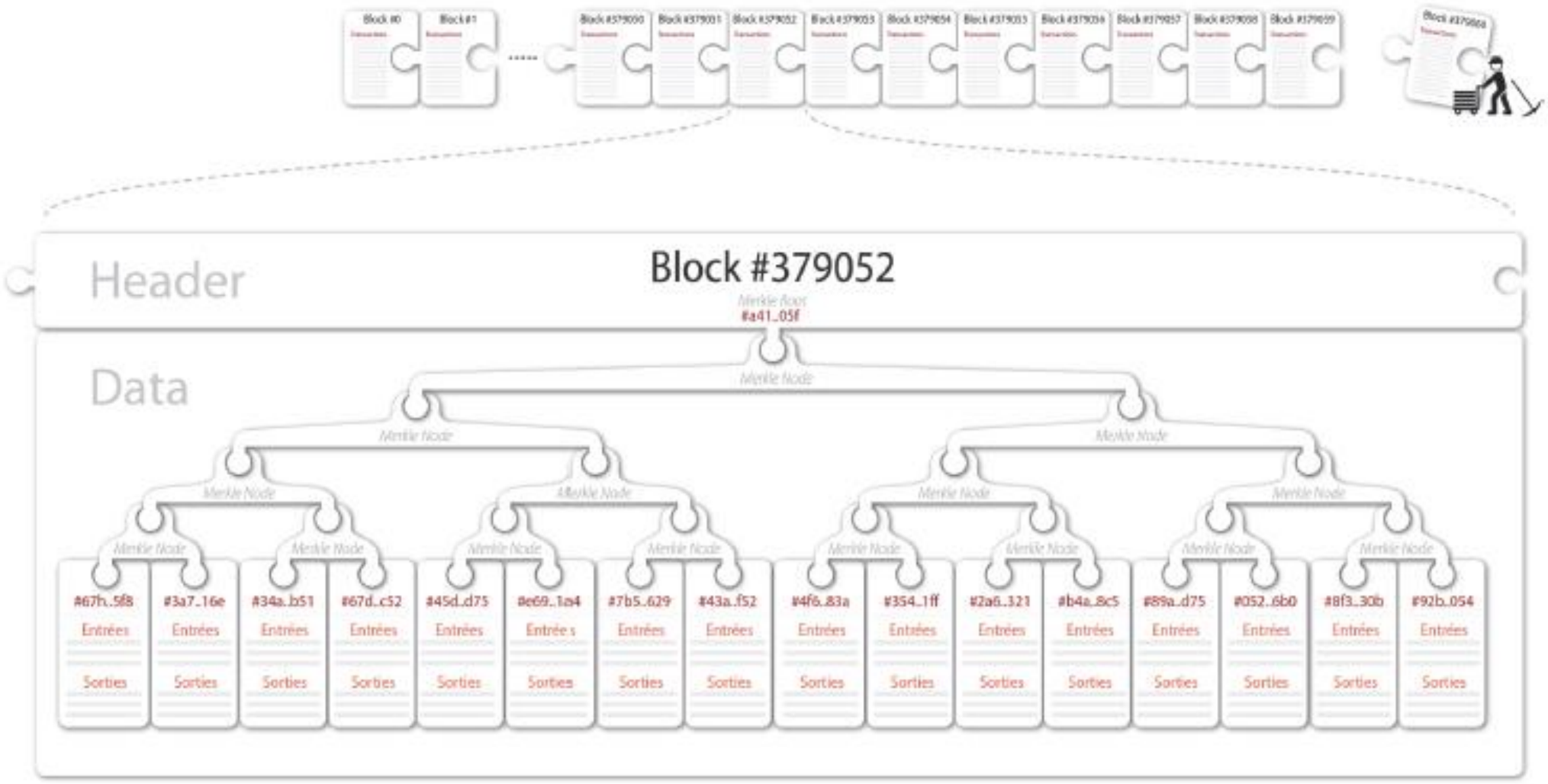
Les **possibilités** offertes par l'usage de ces scripts sont **nombreuses**.

Par exemple, l'usage des flags SIGHASH_ALL | ANYONECANPAY permet de mettre en œuvre une **collecte** au profit d'une personne, d'une cause.
Tout le monde peut participer en donnant le montant qu'il souhaite, et le destinataire est verrouillé.

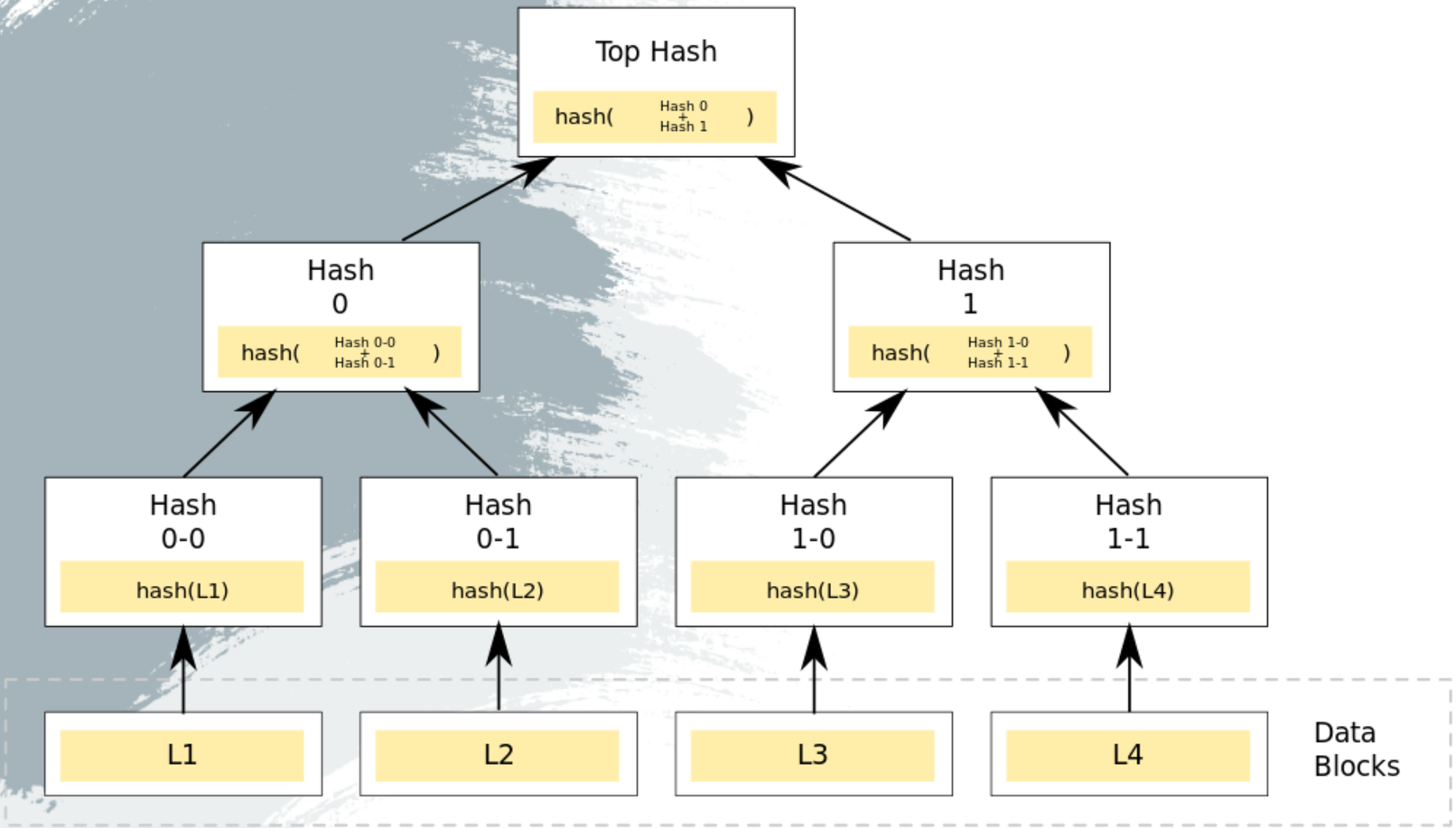
Les transactions portant le flag SIGHASH_NONE permet à quiconque d'ajouter ses sorties souhaitées à la transaction pour réclamer les fonds en entrée.

Les **scripts** dans Bitcoin sont **précurseurs** des **smart contracts**.
Ceux-ci étendent l'usage des scripts avec un langage de plus haut niveau, « **turing-complet** », c'est-à-dire permettant de coder tout type d'opérations.

CONSTRUCTION D’UN BLOC

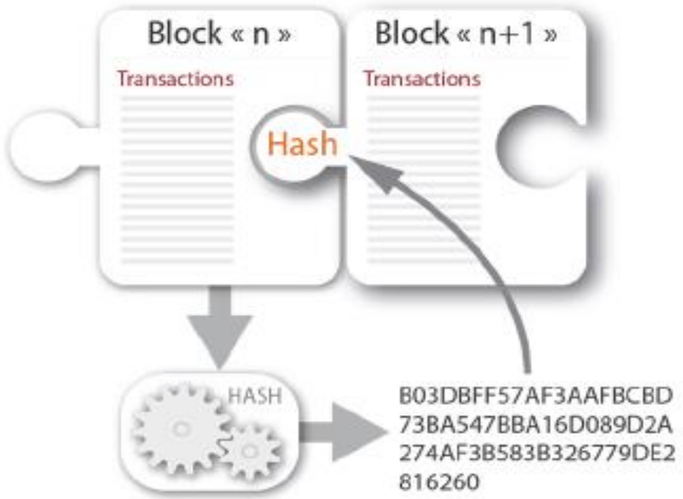


ARBRE DE MERKLE

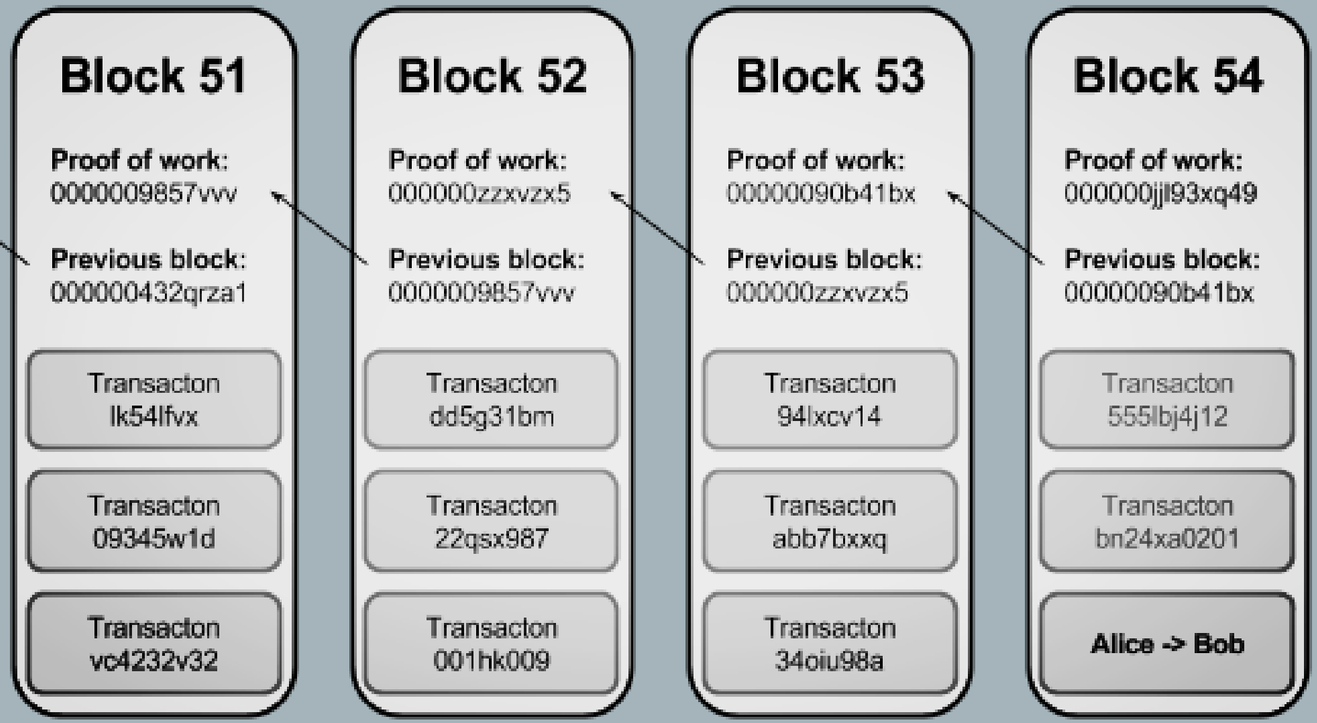


https://fr.wikipedia.org/wiki/Arbre_de_Merkle

CHAINAGE TEMPOREL

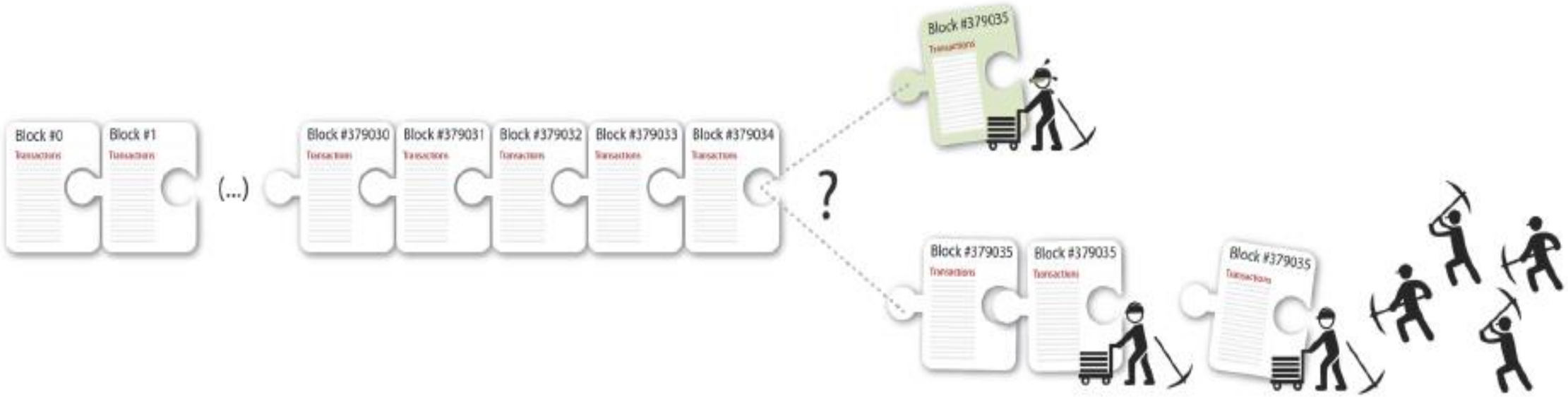


L'**empreinte** d'un bloc est intégrée dans l'entête du **bloc suivant**



LE MINAGE

Comment déterminer le **bloc** qui est **accroché** à la blockchain ?
…et qui désigne le **« vainqueur »,** celui qui remporte l'**incitation**



La **notion du temps**, et plus précisément du **temps écoulé** entre deux blocs, est une propriété de **sécurité** pour la blockchain.
C'est ce qui empêche un **attaquant** de **reconstruire** la blockchain depuis le **bloc genèse**.

LA PROOF-OF-WORK

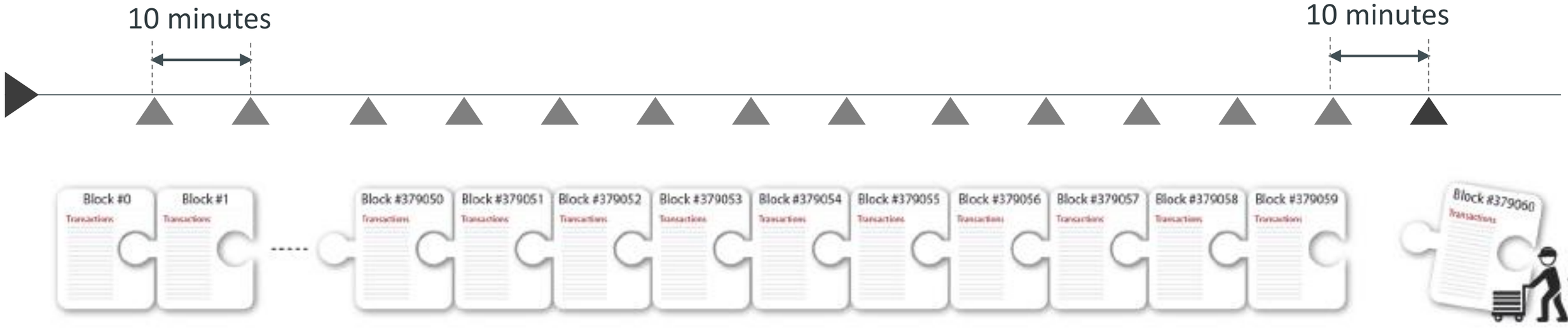
La preuve de travail consiste en un **challenge cryptographique** réalisé par « **brut force** » avec la fonction de hashage SHA256, dont la **difficulté** augmente régulièrement.
Il consiste à mettre en **concurrence** tous les mineurs pour remporter l'**incitation**, c'est-à-dire une **récompense** en crypto-monnaie.
Cette concurrence entre les mineurs pour remporter l'incitation a pour objectif de maintenir l'**équilibre** du système distribué, aussi bien du point de vue de la **sécurité** que de l'**économie**.

Données brutes (Bloc)	data = « ceci est mon bloc »
Difficulté	difficulty = 000000000ffffffffffffffffffffff
Challenge cryptographique	**trouver un nonce tel que :** SHA256(SHA256(data | nonce) < difficulty

LA PROOF-OF-WORK

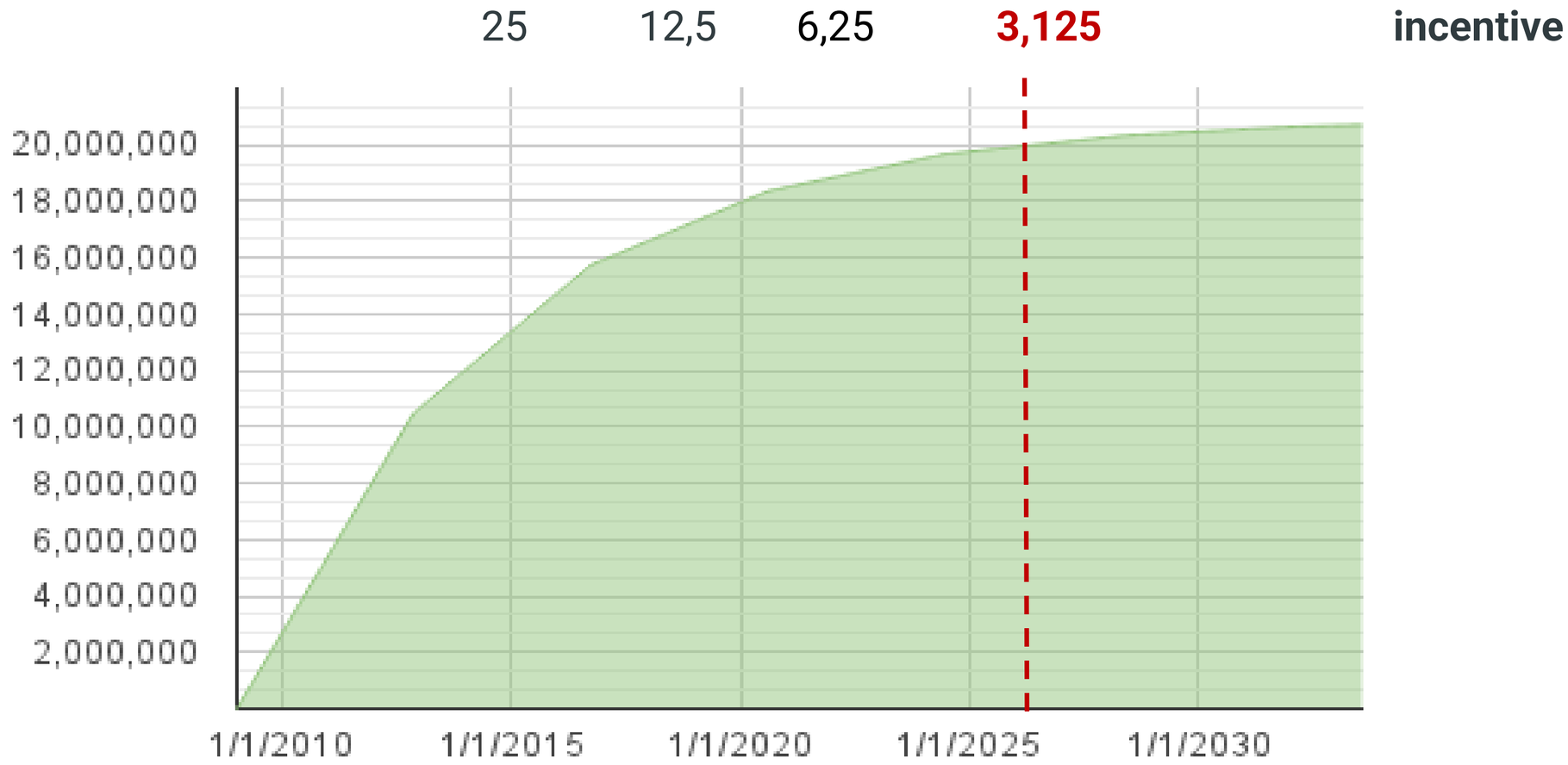
difficulty = **000000**ffffffffffffffffffffffffff

difficulty = **000000000**ffffffffffffffffffff



L’INCITATION DES MINEURS

Courbe d’émission dégressive du Bitcoin au cours du temps



DISTRIBUTION DU HASHRATE

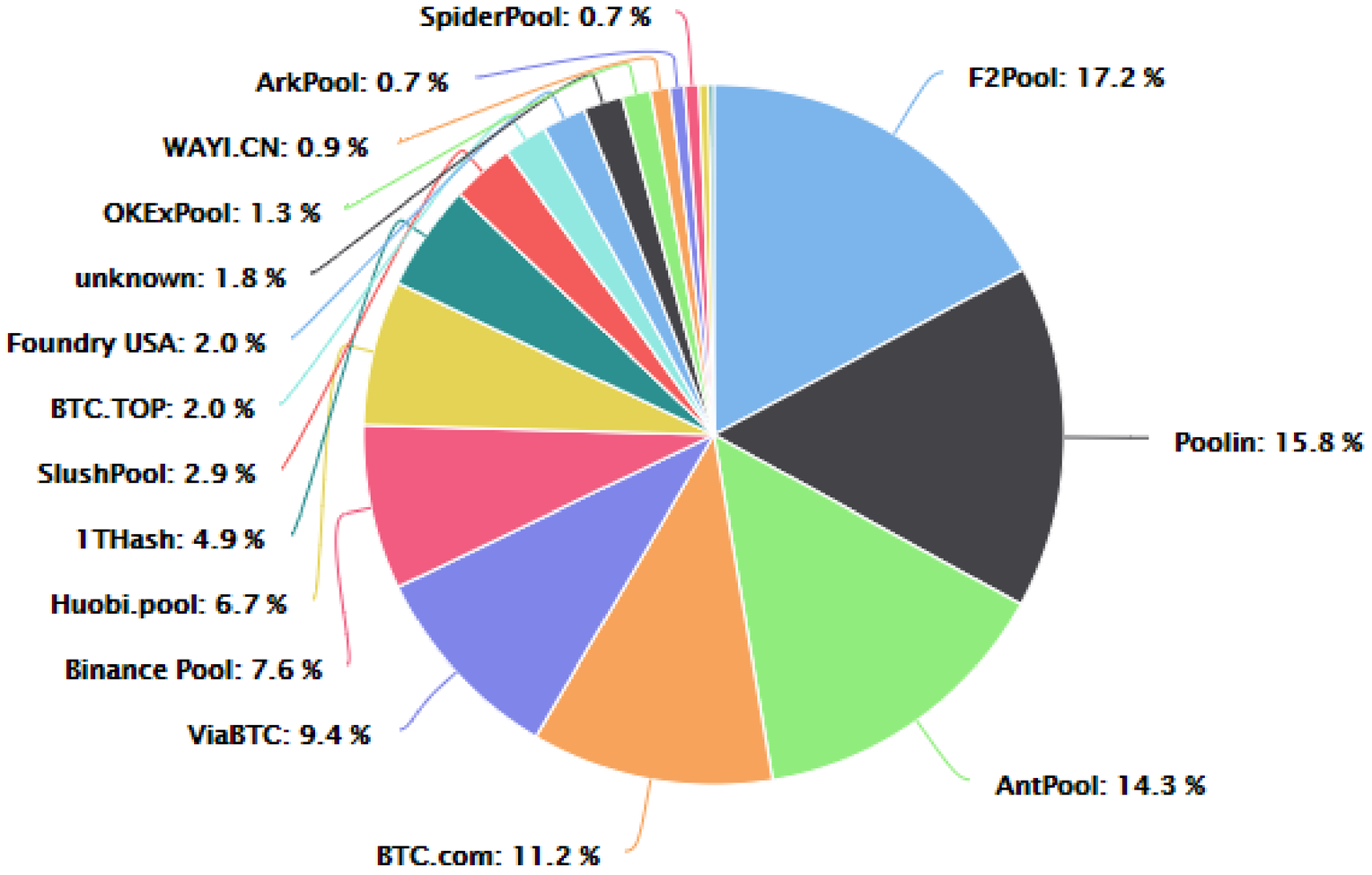
Répartition de l'**incentive** entre fermes de minage

Risque d'**attaque à 51%** …

Concurrence versus **Equilibre** entre fermes de minage

Forme de **Centralisation**

Gouvernance



MINAGE DES BLOCS

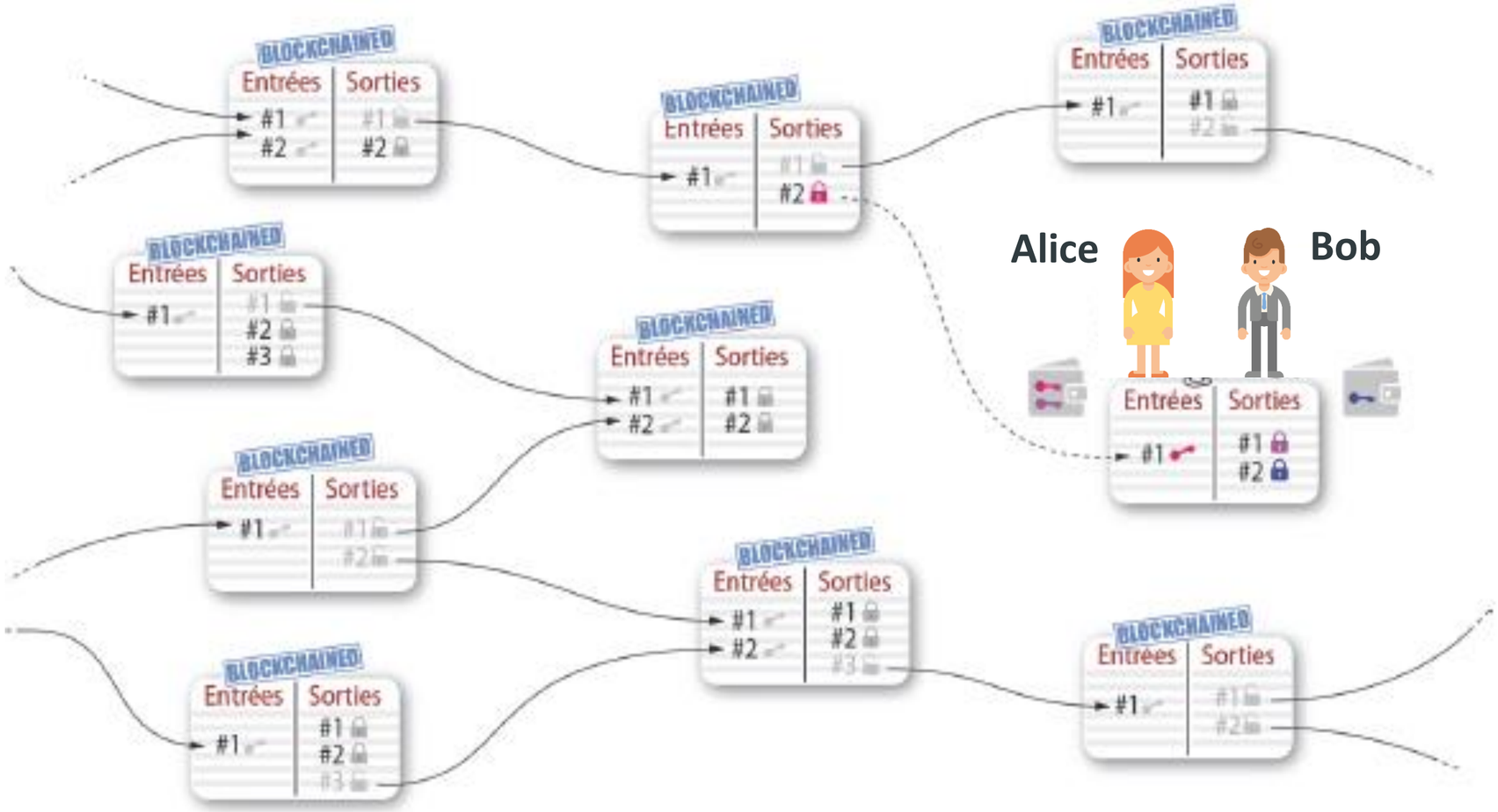
Latest Blocks

Height	Relayé par	Taille(B)	Récompense	Heure	Hash du bloc
678,804	BTC.com	1,496,669	6.55249813 BTC	6 minutes ago	0000000000000000004c0566e0a9a29e34aac45e39fdf01b8b3939d46b47d28
678,803	AntPool	1,356,324	6.82208651 BTC	12 minutes ago	0000000000000000003c86035f7f1a1e74f07797983286e5feecac4ba374c97
678,802	Poolin	1,565,180	6.34888861 BTC	25 minutes ago	0000000000000000000050dd9f98b6ef6224f91a64e6f33741ac90cf961918c5d
678,801	ViaBTC	1,504,546	6.81917259 BTC	27 minutes ago	0000000000000000000090040abf7290f3757a40e15e053dc21f1c0e833b20eaa
678,800	Poolin	1,392,908	6.35075800 BTC	39 minutes ago	0000000000000000000bd91483a66c464f3d6aa191e703a6c7dd24b322e7f554
678,799	ViaBTC	1,342,936	6.53170461 BTC	40 minutes ago	0000000000000000000b398b1b58d4870fd8dd4cb392201d14d677e254c5fd1a
678,798	F2Pool	1,327,063	6.49587307 BTC	44 minutes ago	0000000000000000000bc7558980a4fe31c22afa8dd6e0f9fa87e953992f7974
678,797	Poolin	1,307,861	6.69096686 BTC	47 minutes ago	0000000000000000000c1f9361e5abea761a1bdad34f03c28cb07ae32d6bf0e8
678,796	ViaBTC	1,376,096	7.02156036 BTC	56 minutes ago	0000000000000000000031fa0ded3294ba7639939c77a41ee1828da4e2ad6e5d4
678,795	F2Pool	1,466,490	6.90122940 BTC	1 heure 11 minutes ago	0000000000000000000041790a052fb6d2d96012002b5f1eba788b795f64ccf9a

https://btc.com/

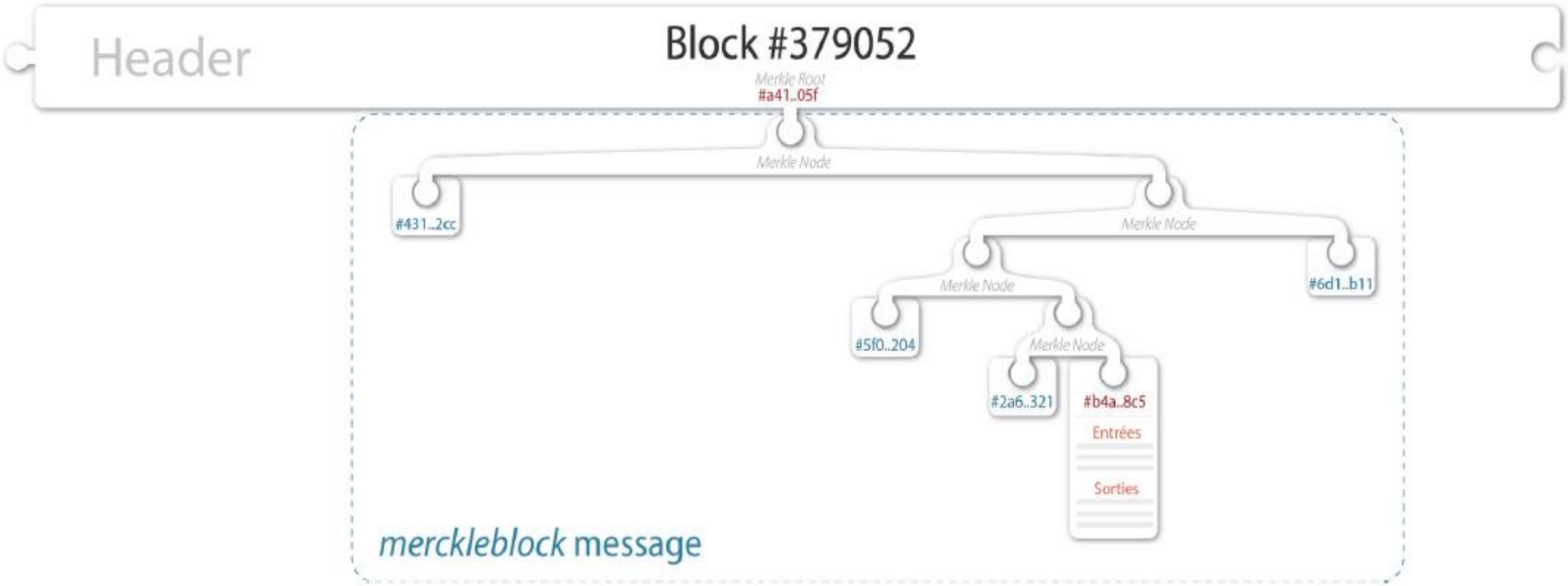
DAG DES TRANSACTIONS

(Directed Acyclic Graph)



SIMPLE PAYMENT VERIFICATION

Le **protocole SPV** (Simple Payment Verification) permet de vérifier l'**intégrité d'un bloc** ainsi que l'**appartenance d'une transaction** au bloc.
La consultation est **locale** pour les nœuds (validateurs) disposant d'une **copie complète du registre (ledger)**.



https://en.bitcoinwiki.org/wiki/Simplified_Payment_Verification

CONSULTATION DU LEDGER

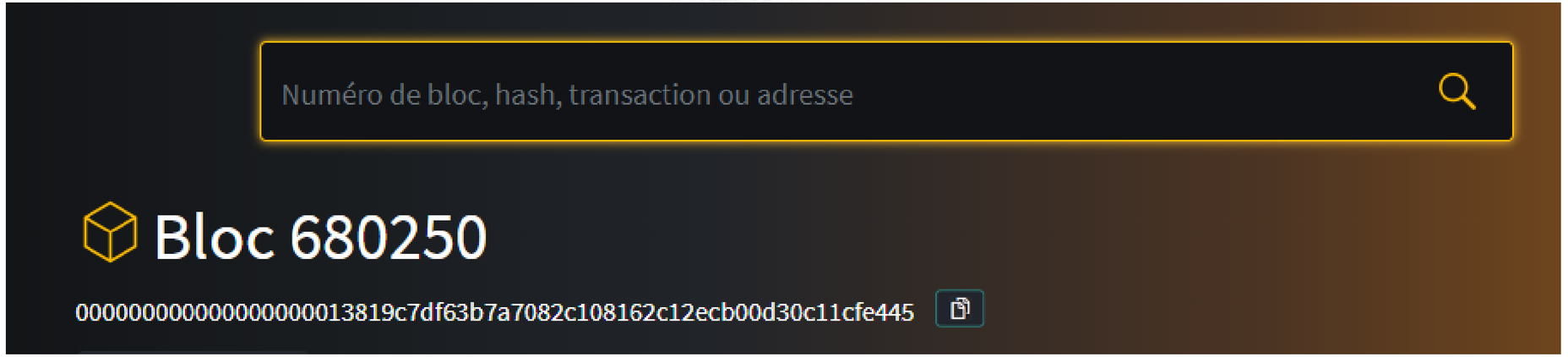
La blockchain Bitcoin étant publique, **tout le monde peut consulter son contenu**.
Il existe des API facilitant la consultation, comme par exemple :

https://blockstream.info/

https://www.blockchain.com/

https://live.blockcypher.com/btc/

https://btc.com/

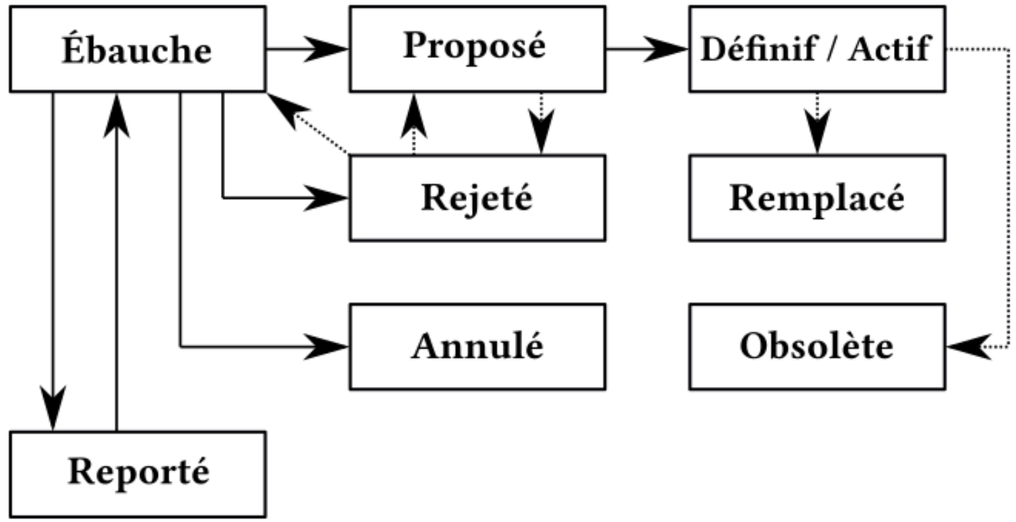


STANDARDISATION

Un **BIP**, acronyme de l'anglais ***Bitcoin Improvement Proposal***, est une proposition d'amélioration de Bitcoin, qui peut concerner les règles de consensus, la communication pair-à-pair, l'interface logicielle, les applications ou tout autre chose relative à Bitcoin.

Il s'agit généralement d'un court **document technique** décrivant la modification à effectuer, qui est ensuite utilisé comme standard.

Tout le monde peut proposer un BIP. Une fois qu'un BIP est jugé acceptable, un **numéro** lui est assigné et il est intégré au dépôt sous la forme d'une **ébauche**. Il peut par la suite changer de statut au cours du temps, l'objectif étant qu'il devienne définitif ou actif.



QUELQUES BIPs

Bitcoin **évolue**…

12	Consensus (soft fork)	OP_EVAL	Gavin Andresen	Standard	Withdrawn
13	Applications	Address Format for pay-to-script-hash	Gavin Andresen	Standard	Final
32	Applications	Hierarchical Deterministic Wallets	Pieter Wuille	Informational	Final
39	Applications	Mnemonic code for generating deterministic keys	Marek Palatinus, Pavol Rusnak, Aaron Voisine, Sean Bowe	Standard	Proposed
340		Schnorr Signatures for secp256k1	Pieter Wuille, Jonas Nick, Tim Ruffing	Standard	Draft

https://github.com/bitcoin/bips

QUELQUES ÉLÉMENTS D’ÉCONOMIES

COURS DU BITCOIN

Historique des prix BTC

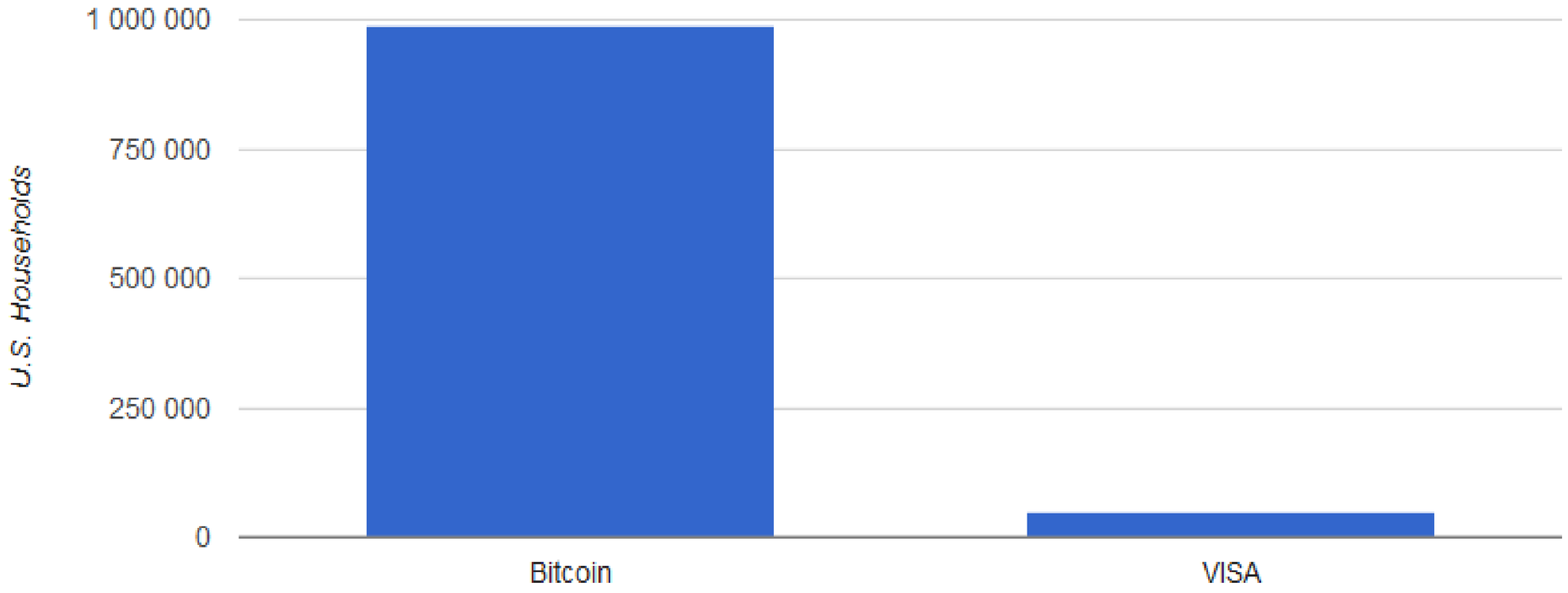
Bitcoin Prix aujourd'hui

Prix de Bitcoin	**€50,363.32**
Évolution du prix 24h	€230.32 ▲ 0.46%
Étendue intrajour des prix	€49,163.57 / €51,118.66
Volume d'échange 24h	€40,183,588,427.37 ▼ 12.47%
Volume / Market Cap	0.04274
Dominance du marché	54,27%
Rang sur le marché	#1

https://coinmarketcap.com/fr/currencies/bitcoin/

DÉBIT DES TRANSACTIONS

Estimation comparée de la consommation d'énergie du système Bitcoin avec le système VISA



Au moins 169 x 10^5 kWh
pour 100 000 transactions

169 kWh
pour 100 000 transactions

COÛT ÉNERGÉTIQUE

Avec une consommation annualisé d'environ 34 TeraWatt/h en 2018, le Bitcoin consomme autant d'électricité que…

- L'intégralité de la Suisse
- Ce qui correspond à 13 % de la consommation de la France,
- Et à 0.25 % de la consommation totale d'électricité du monde,
- plus d'électricité que 159 États dans le monde.

En 2025, une étude réalisée par le Cambridge Centre for Alternative Finance (CCAF), a estimé que le bitcoin consommait 138 TWh par an, soit :

- 0,54 % de la consommation électrique mondiale,
- la consommation de la Pologne.

Un **indice** pour suivre le **coût énergétique** du Bitcoin en temps réel :

https://cbeci.org/

RÉGULATION

Le Bitcoin, et plus généralement la crypto-monnaie, peut être considéré soit comme une **marchandise**, soit comme une **monnaie** dont la **quantité** est finie et dont la **valeur** fluctue en fonction de l'offre et de la demande.

Les Bitcoins peuvent être vendus ou achetés sur des places de marchés. On distingue :

- Les **places de commerce** où on peut acheter directement des Bitcoins (ex. *Paymium*) https://www.paymium.com/
- Les **places d'échange** qui mettent en relation les acheteurs et les vendeurs (ex. *localbitcoin*) https://localbitcoins.com/fr/

L'instance de régulation des marchés boursiers américains a estimé en 2015 que le commerce de crypto-monnaies relève de la législation du **commerce des marchandises**.

A contrario, la cours de justice européenne a estimé en 2015 que les échanges de devises contre des bitcoins sont **exonérés de TVA** au même titre que des échanges de devises traditionnelles.

RÉGULATION

La Commission européenne a adopté en 2023 le règlement **MiCA** (Markets in Crypto-Assets)

- MiCA est entré en vigueur en **juin 2024**
- MiCA vise à **encadrer** les émissions et les services sur crypto-actifs
- Chaque État membre met en place sa propre politique fiscale

Le règlement MiCA **ne s'applique pas** aux crypto-actifs qui sont uniques et non fongibles avec d'autres crypto-actifs (**NFT**)

QUELQUES RÉFÉRENCES

1. Satoshi Nakamoto, « Bitcoin: A peer-topeer electronic cash system », 2009
2. A. M. Antonospoulos, « Mastering Bitcoin », O'Reilly, https://bitcoin.fr/Bitcoin-explique-par-son-inventeur/
3. Susanne Köhler and Massimo Pizzol, « Life Cycle Assessment of Bitcoin Mining", Environmental Science & Technology, volume 53, number 23, pages 13598-13606, year 2019, doi 10.1021/acs.est.9b05687, https://pubs.acs.org/doi/pdf/10.1021/acs.est.9b05687
4. « The Security Evaluation of Time Stamping Schemes: The Present Situation and Studies" Masashi UNE Discussion Paper No. 2001-E-18
5. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=D44E454AD23AA67165916258C623FF91?doi=10.1.1.23.7486&rep=rep1&type=pdf
6. https://en.bitcoin.it/wiki/Main_Page
7. https://bitcoin.org/fr/
8. https://github.com/minium/Bitcoin-Spec

SECTION 3

ETHEREUM

ETHEREUM

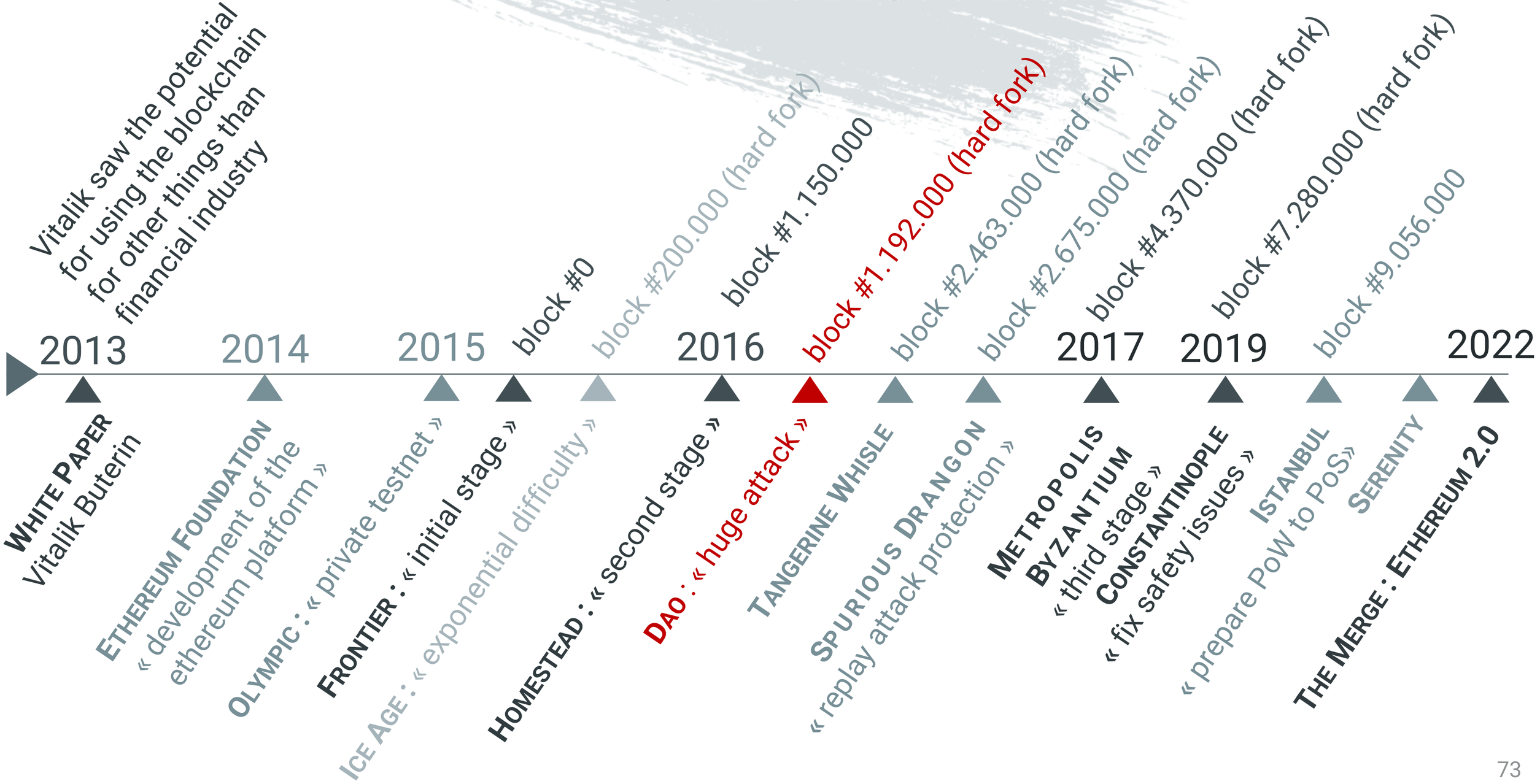
"Ethereum is an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology".

Ethereum's official documentation

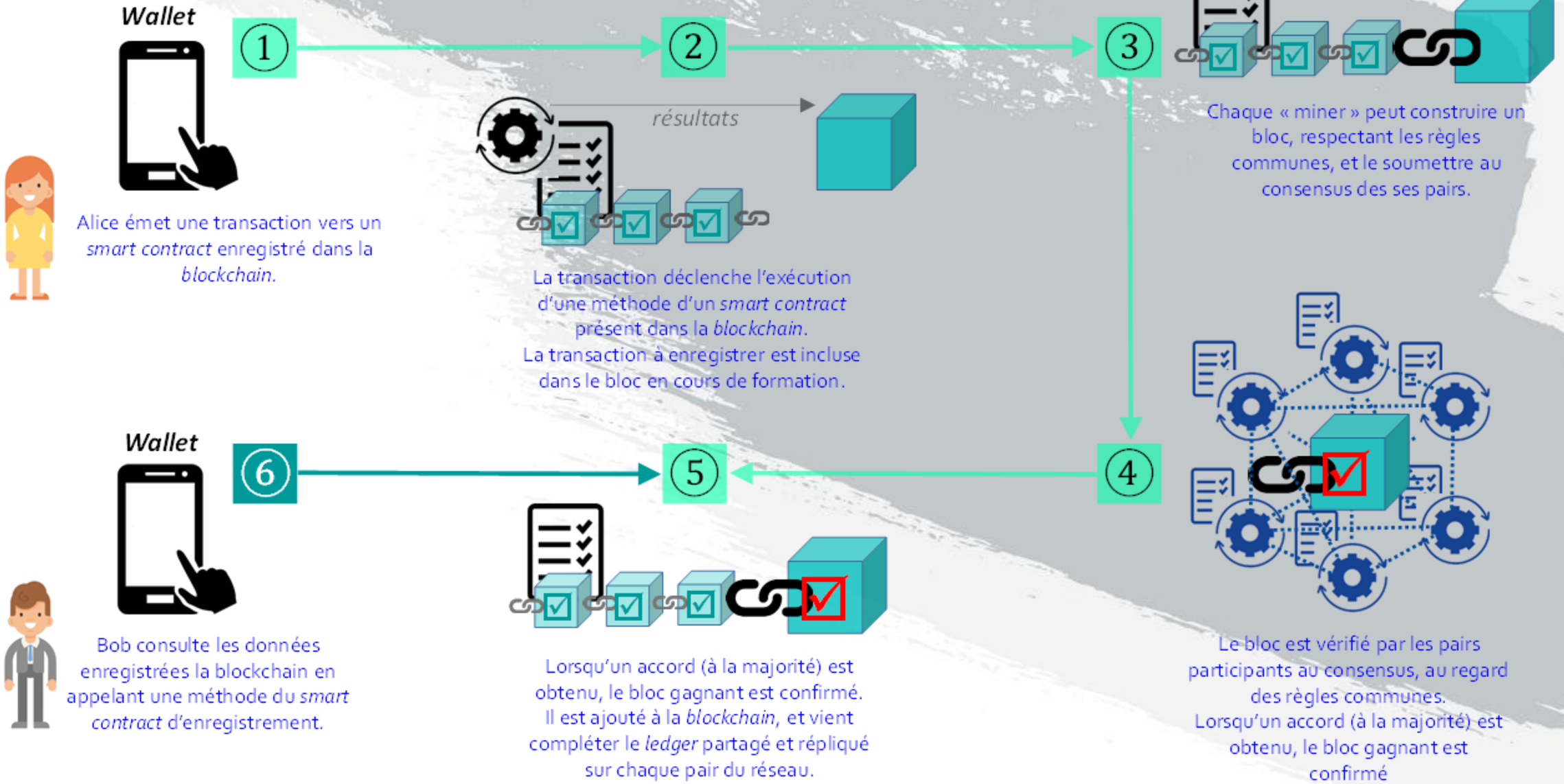
« the world computer »

Vitalik Buterin

ÉTAPES DU DÉVELOPPEMENT



COMMENT FONCTIONNE ETHEREUM ?



UTXO OU BALANCE DE COMPTE

- une transaction a comme origine les outputs (UTXO) de transactions précédentes et va générer à son tour de nouveaux outputs (UTXO).
- Le wallet garde la trace de toutes les UTXO associées aux adresses qu'il possède. Pour établir le solde du compte, il va parcourir toutes les UTXO associées au compte
- Usage : pour échanger de l'argent
- Blockchain : Bitcoin, Monero, Neo, Grin

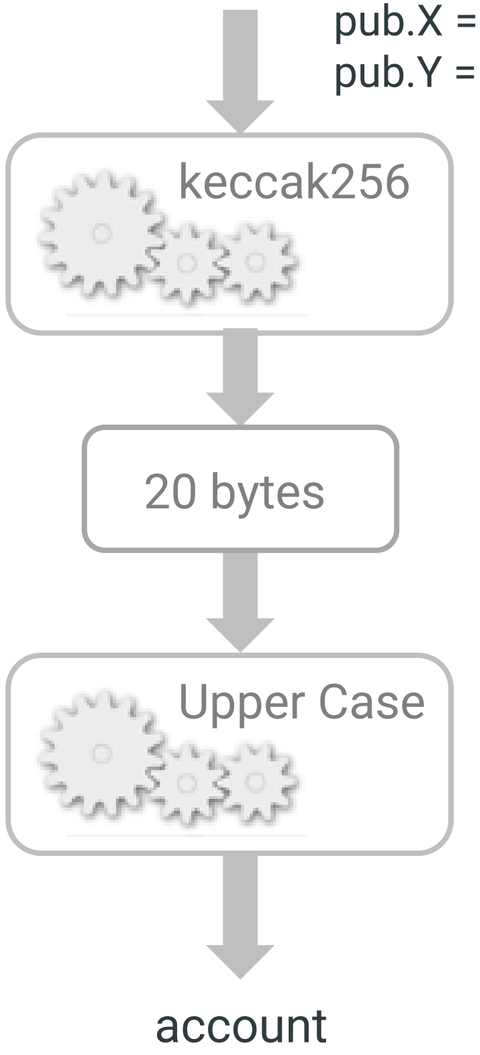
- Le compte est "stateful", l'état correspond au résultat des opérations entrantes
- Le solde du compte est estimé de façon instantanée
- Chaque transaction donne lieu à un récépissé
- Usage : exécution de smart contracts
- Blockchain : Ethereum, EOS, Tron

Les blockchains qui utilisent massivement le **sharding** utilisent généralement les **deux mécanismes** en parallèle. La **synchronisation** entre shards s'effectue suivant le mécanisme des **UTXO** et l'exécution de **smart contracts** suit le mécanisme des **Balance de compte**

ADRESSE DE COMPTE ETHEREUM

clé publique

pub.X = 904D1127AD59B93C45F02A2F80E8FCA122746BC77AAA1416A6B0FE0C5D0982FD
pub.Y = 9AF40ADB96C0C92E59242E59EFBFDEF999FE120536C4B1F32F30BD8A1CF1ABEA



① Calcul du haché

② Sélection des 20 octets LSB

416CC7CC63FC93830750A3666A3C4BFD83B3F832

③ Calcul et Application du masque (checksum)

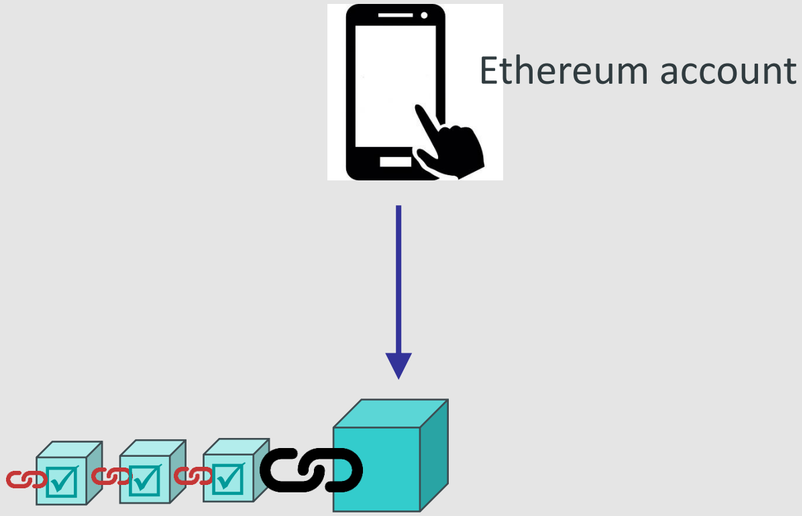
Keccak256(account address) → mask
For each quartet acc[i] in account : if mask[i] >=8 and account[i] is a letter → take upper case letter

④ Upper Case Account Address

account = 416CC7Cc63Fc93830750a3666a3C4bfD83b3F832

DEUX TYPES DE TRANSACTIONS

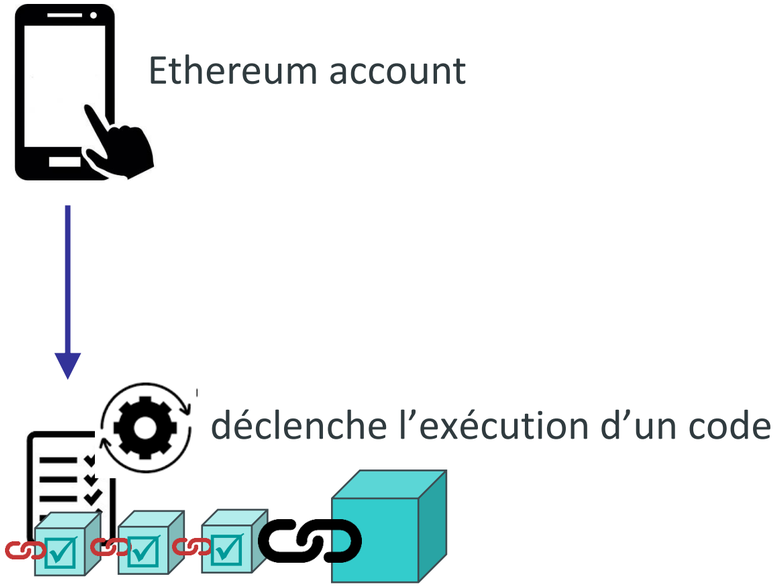
Transaction de compte à compte



change la balance de account source et account destination

```
tx = {
    'from': to_checksum_address(source),
    'to': to_checksum_address(destinataire),
    'value': value,
    'gasPrice': self.conn.gasPrice,
    'gas': 2 * self.conn.estimateGas(),
    'nonce': self.conn.eth_getTransactionCount(source),
    'chainId': 1
    }
```

Transaction de compte à smart contract



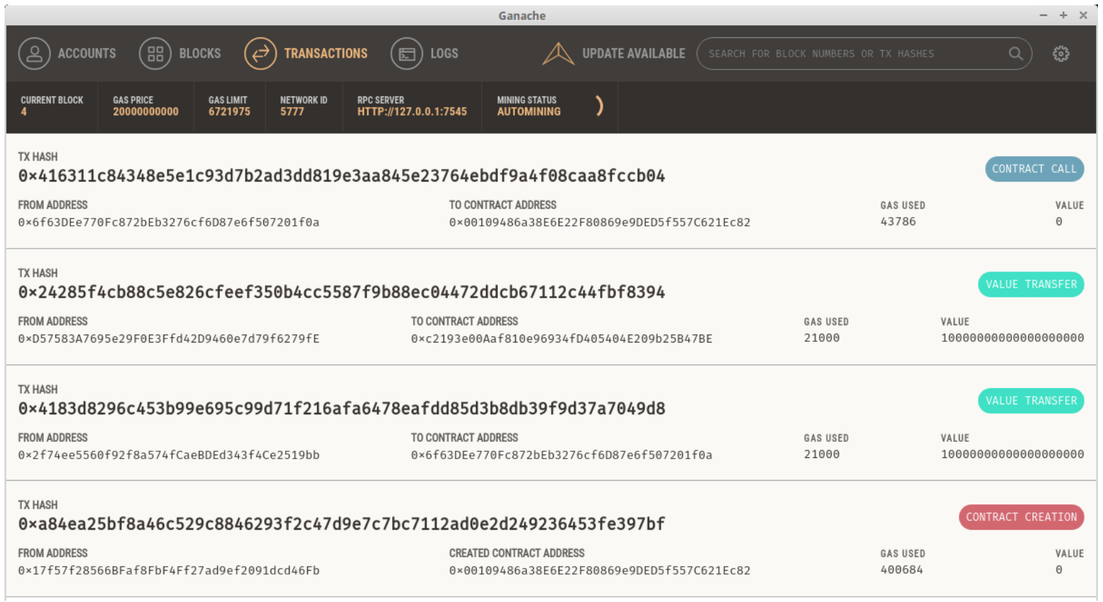

```
tx = {
    'from': to_checksum_address(source),
    ‘to': to_checksum_address(contract_address),
    'gas': self.conn.eth_estimateGas(source), ‘
    'gasPrice': self.conn.eth_gasPrice(),
    'nonce': self.conn.eth_getTransactionCount(source),
    'data': method | arguments
    'chainId': 1,
   }
```

EXEMPLE AVEC GANACHE

Etape ① : déploiement du smart contract

Etape ② : Approvisionnement du compte utilisateur

Etape ③ : Envoi d'une transaction depuis le compte utilisateur vers le smart contract



MNEMONIC

Bitcoin Improvement Proposal (BIP) 39

Une solution pour le **backup & recovery** de la clé privée

1) **génération** d'une **suite de mots** aléaoirement piochés dans un **dictionnaire** : **mnemonic**

   12 mots : 128 bits d'entropie → 24 mots : 256 bits d'entropie

   https://github.com/bitcoin/bips/blob/master/bip-0039/bip-0039-wordlists.md

   indoor dish desk flag debris potato excuse depart ticket judge file exit

2) **déduction** de la **seed** (512 bits) à partir du **mnemonic**

   3bd0bda567d4ea90f01e92d1921aacc5046128fd0e9bee96d070e1d606cb79225e
   e3e488bf6c898a857b5f980070d4d4ce9adf07d73458a271846ef3a8415320

3) **dérivation** de la **clé privée** sur la famille de courbe elliptique **secp256k1** à partir de la **seed**

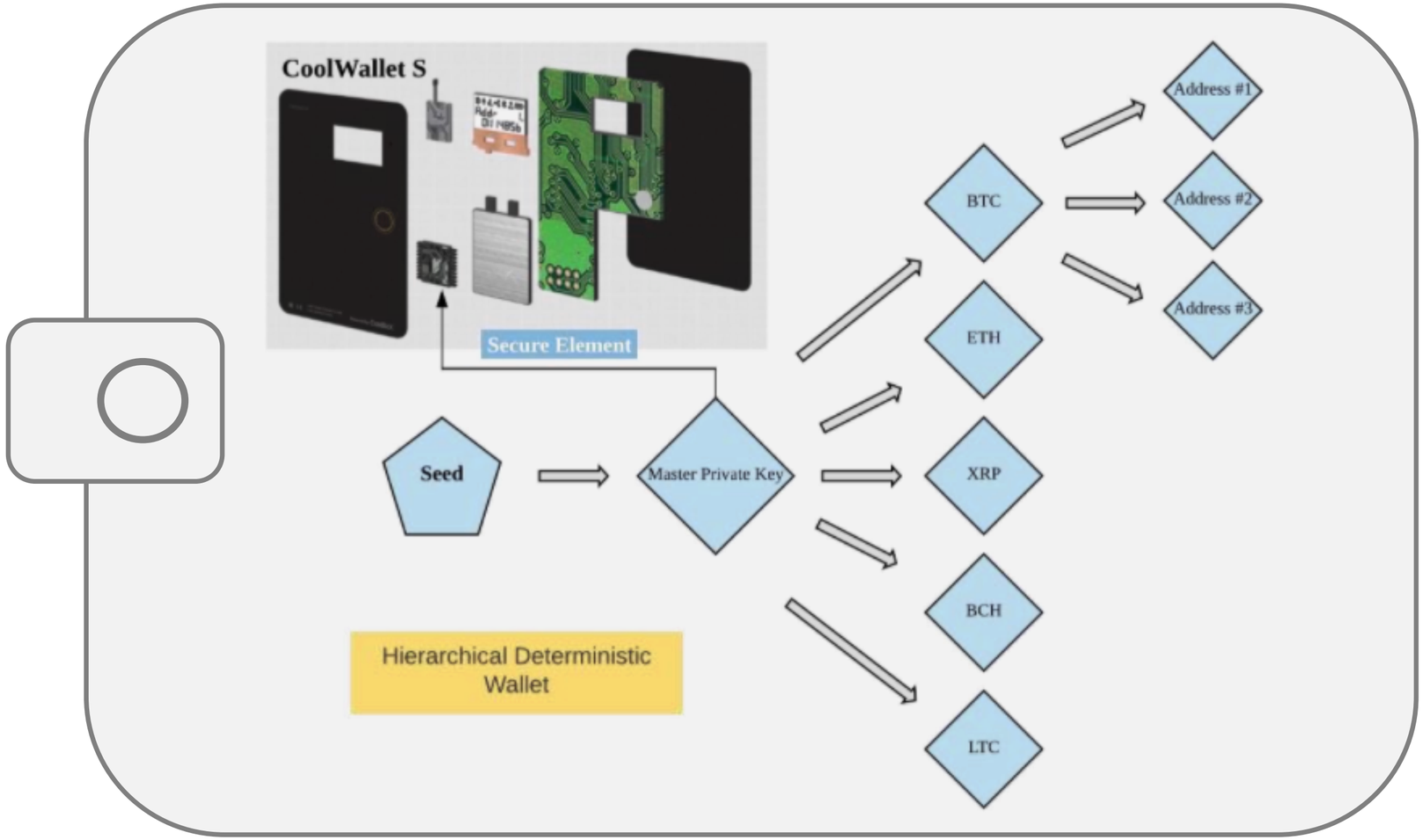
   xprv9s21ZrQH143K3wP239Ccn722nReih7LW9W4uiAciQss8a8Co2cUrmcEFCuKQ
   2a5ZGuGhiHvo51D4pj5zd3CRsNgs6r7yrVHE1EFvjUUGSH8

LES DIFFÉRENTS WALLETS

Les **Coins** sont présents dans le **LEDGER**.
Le **WALLET** héberge les **clés secrètes** permettant d'utiliser les **Coins**.

Portefeuille	Papier	Logiciel	Matériel
Chaud		**Portefeuille Web** - MyCrypto - Opera **Portefeuille navigateur** - Metamask	
Froid		**Portefeuille de bureau** - wallet.dat - MyEtherWallet **Portefeuille mobile** - Trust Wallet - Metamask	**Dispositif avec TPM** - Nano Ledger - Trezor

DETERMINISTIC HIERARCHICAL WALLET

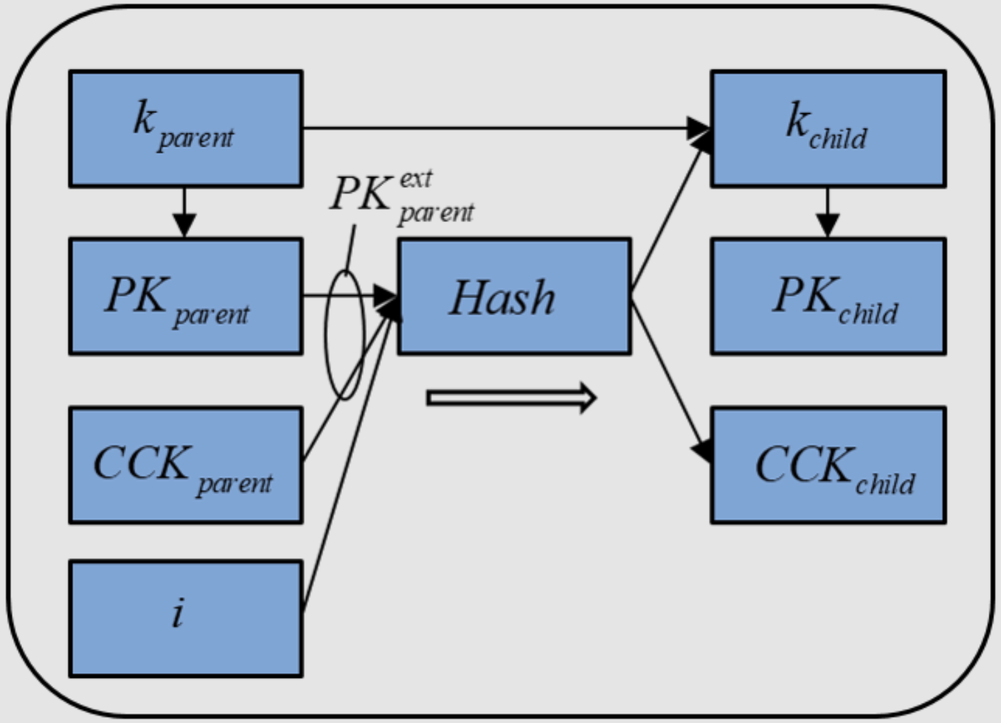


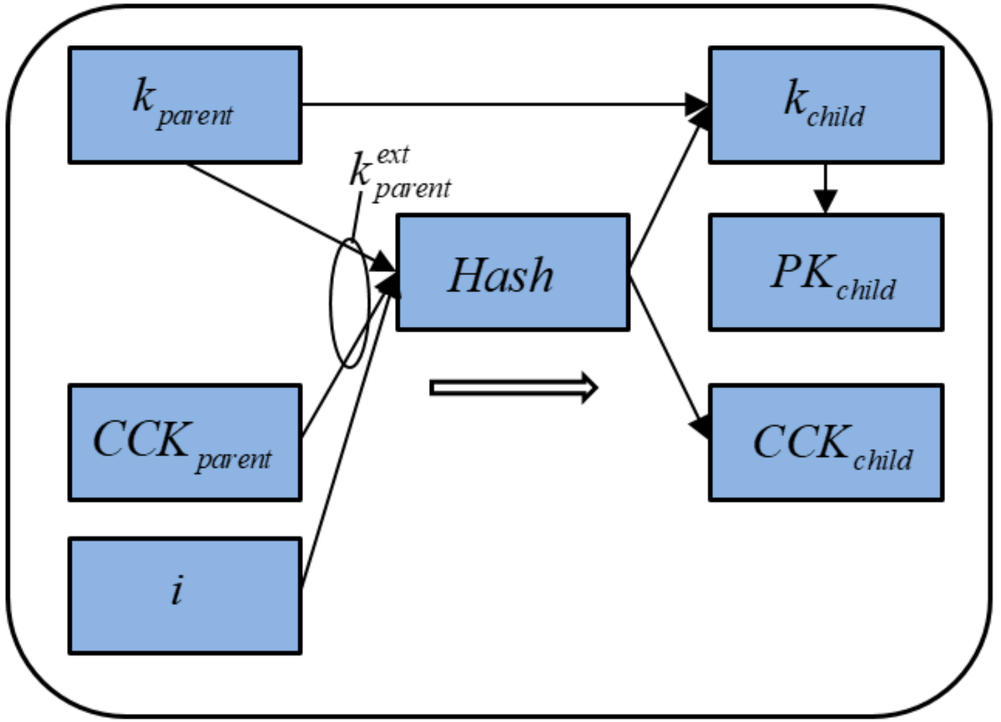
https://en.bitcoin.it/wiki/BIP_0032

https://en.bitcoin.it/wiki/BIP_0044

```
m / purpose' / coin_type' / account' / change / address_index
```

TECHNIQUES DE DÉRIVATION DE CLÉS





ADRESSE DE COMPTES

mnemonic : indoor dish desk flag debris potato excuse depart ticket judge file exit

Path Toggle	Address Toggle	Public Key Toggle	Private Key Toggle
m/44'/60'/0'/0/0	0x3f1Eae7D46d88F08fc2F8ed27FCb2AB183EB2d0E	0x0226cc24348fbe0c2912fbb0aa4408e089bb0ae488a88ac46bb13290629a737646	0xb6b15c8cb491557369f3c7d2c287b053eb229daa9c22138887752191c9520659
m/44'/60'/0'/0/1	0xe2148eE53c0755215Df69b2616E552154EdC584f	0x0380a059c1159ce87f4a6136fc34192553602d4c0d1c533044ee513778831faf49	0xcb5790da63720727af975f42c79f69918580209889225fa7128c92402a6d3a65
m/44'/60'/0'/0/2	0x6A568afe0f82d34759347bb36F14A6bB171d2CBe	0x03018426b811a3397e407ea21fade1bb72a9dd695deed32c59213d3992ba603feb	0x182fecf15bdf909556a0f617a63e05ab22f1493d25a9f1e27c228266c772a890
m/44'/60'/0'/0/3	0x863C904166E801527125D8672442D736194A3362	0x03279e993c7e49044e6553612a5f15018cec91c6245b562cae395214de8a00dfdb	0xecdf21cb41c65afb51f91df408b7656e2c8739a5877f2814add0afd780cc210e
m/44'/60'/0'/0/4	0x3E6134aAD4C4d422FF2A4391Dc315c4DDf98D1a5	0x023888991dcfaae7ce00fe9d89b4e7c1dfbefc222ffc019691810b3ee9d1986824	0x90f899754eb42949567d3576224bf533a20857bf0a60318507b75fcb3edc6f5f
m/44'/60'/0'/0/5	0x5E1497dD1f08C87b2d8FE23e9AAB6c1De833D927	0x032ff983dd3a95196c35384fc8c8e2026d5ee1592784881488f281559d3a63c8fc	0xdc04c5399f82306ec4b4d654a342f40e2e0620fe39950d967e1e574b32d4dd36
m/44'/60'/0'/0/6	0x46225F4cee2b4A1d506C7f894bb3dAeB21BF1596	0x02db02841583e911c074e40d05373b292ee8407ae9875d8baaa4f86b30c0a7a5ae	0xb0c3d5fa3891e7029918fdf0ed5448e0d6b7642c4ee2c8fa921bc703b4bc7c9f
m/44'/60'/0'/0/7	0x19ED240ddd4DDEeDdF2B77aA279F258eFC52f9b7	0x03bc4a1dfd67d08340c63a253c8e6612576ababf39864b11eac7f271ee65bce9d6	0x19b611a70d1cbed3eb0678f3fc1fa78141d3a7c7b3a18242043d30c35e768b9d
m/44'/60'/0'/0/8	0xaDef93B37f314ae8c8d711C76dB2C17587738DB8	0x022b5cadc08ff75bb7f2cdbfa5d23c16ce2065244353db0d5df70ff2932085b296	0x4b8c12d91a78348c9fb0b0bb505e41ba08efbb14a955dde43a2fabe0e3c2793e
m/44'/60'/0'/0/9	0xcEeD7eFb59a5De0B885Af9b5D416bc5E9725C8e0	0x02c637abbf1989fac52663b71eaba23f07a311249a611088a769825b14702dfaa6	0x0677fe91341830074O1d356e1ec970114afde6363895f493b4f8aec3bfe20d86
m/44'/60'/0'/0/10	0xb911217f08f23e49F02151354c91c3f213C040CE	0x03e819c84c66d9bd18dc5ca03a5e499920a7a4c90aafa02f704d307cd69bd253d4	0xccae2b523a08f622a9c14e9008efccfdb57a0f344398de564620da0cf915299a
m/44'/60'/0'/0/11	0x0831A8b22377779D19d014810754ceD6fC6b7dA0	0x03d95dcfd190a7e5d52bd39684ebda7b3df43caad0efaca1f026b24d5c6dae8a47	0x852a94fbf20024dc1e45c210a8a769d128e095745e52e31bf0e26883450179ac
m/44'/60'/0'/0/12	0x1F623279865ea3036825d22BDE370B50E0f4D8B0	0x03729b6ea6a46bc17b86d1f344f91633bd8761fe1149dceef80c92c94b14cb8de6	0x927e6f1e82fa71f8da1436deeed6533154ff9f2790fb5a0695c05f7ccf2330ea

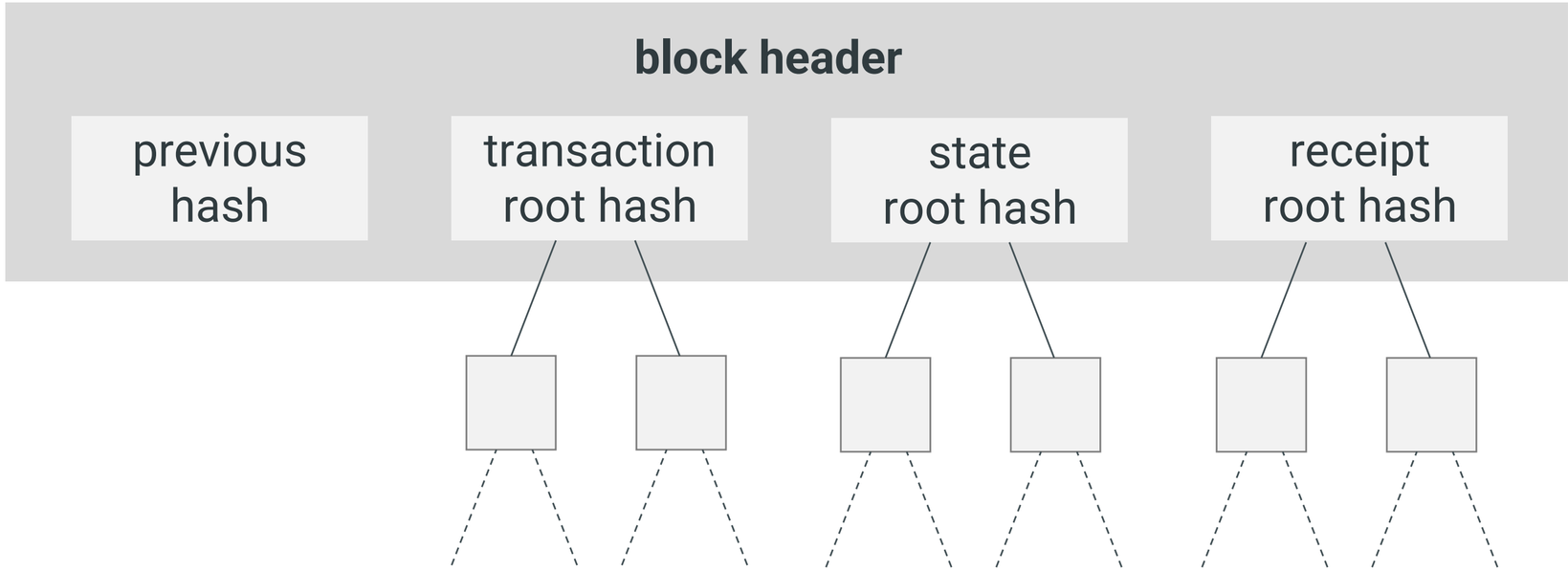
https://iancoleman.io/bip39/#english

STRUCTURE DES DONNÉES

Un bloc ethereum contient trois arbres de données distincts :

- Un arbre de **transactions**
- Un arbre d'états (**state**)
- Un arbre de récépissé (**receipt**)

Chaque arbre est un **Patricia Merkle Tree**



PATRICIA MERKLE TREE

La structure de données **Patricia Merkle Tree** utilisée par Ethereum s'appuie sur **4 notions**

① **Arbre de Merkle**

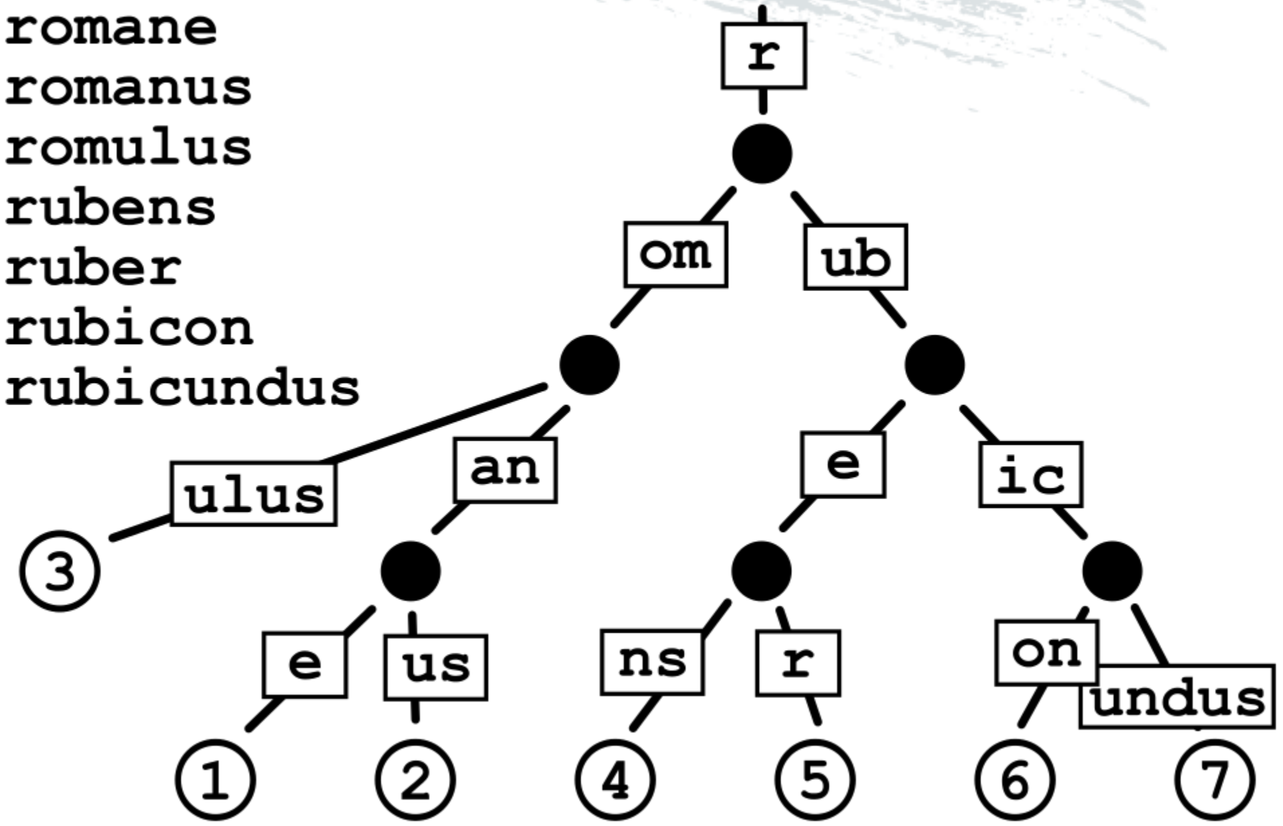
② **Radix Tree**

③ **Hash Table**

④ **Encodage RLP**

RADIX TREE

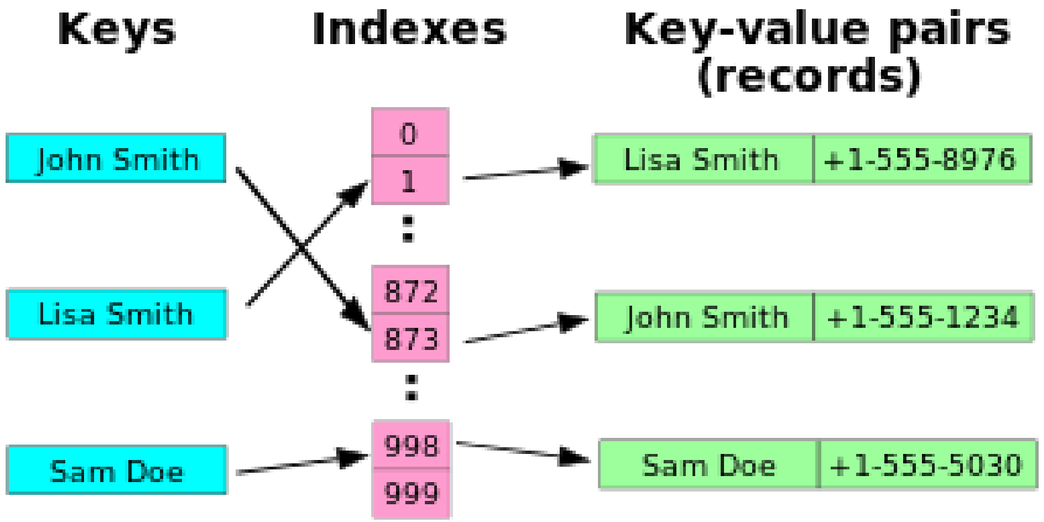
1 romane
2 romanus
3 romulus
4 rubens
5 ruber
6 rubicon
7 rubicundus



https://en.wikipedia.org/wiki/Radix_tree#cite_note-10

HASH TABLE

Une table de hachage est une structure de données qui permet une association **[clé, valeur]**



- n’est pas ordonné
- accès à chaque valeur du tableau par le haché de la clé qui sert d’index
- Présente l’avantage de retrouver très facilement des données
- Le risque de collision est réduit en utilisant une fonction de hachage adéquate

https://en.wikipedia.org/wiki/Hash_table

ENCODAGE RLP

Recursive Length Prefix

RLP encode des éléments : tableaux de bytes, structures (transaction par exemple)

Un élément ou une liste d'éléménts peut être :

- “dog”
- []
- [“dog”]
- [[], “dog”, [“cat”], “ ”]

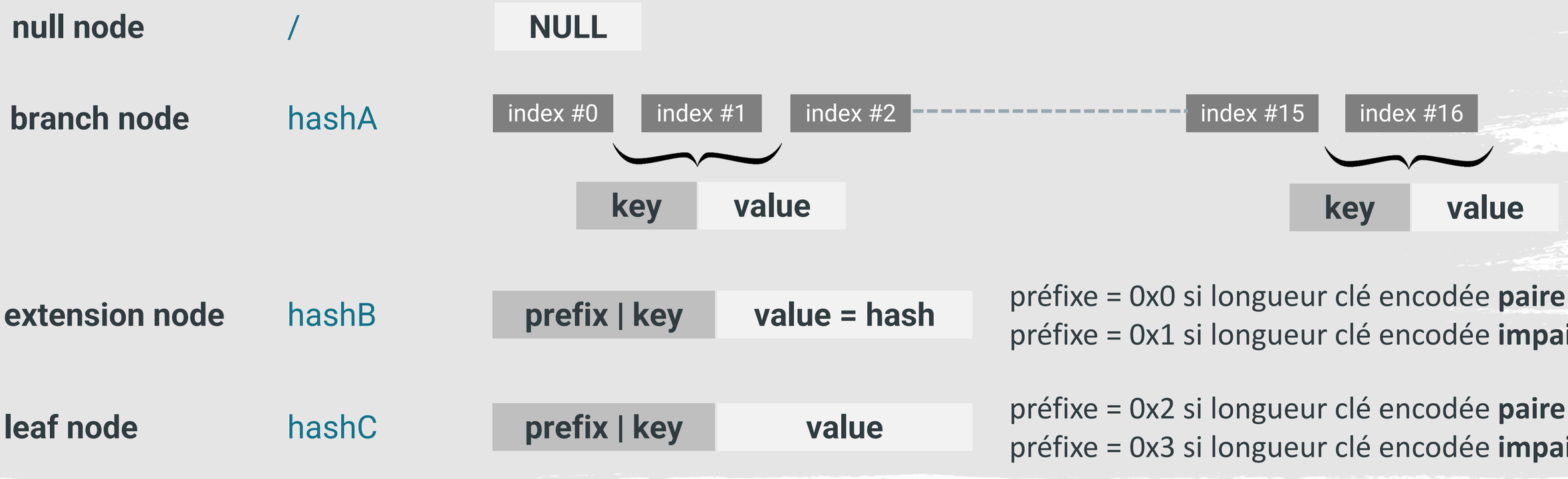
“hello world” = [**0x8b**, 0x68, 0x65, 0x6c, 0x6c, 0x6f, 0x20, 0x77, 0x6f, 0x72, 0x6c, 0x64]

Parce que “hello world” est une chaine de 11 caractères, soit 0xb en hexadécimal, concaténé avec le préfixe 0x80, cela donne 0x8b suivi du codage des caractères

https://eth.wiki/en/fundamentals/rlp

PATRICIA MERKLE TREE

La **structure de données** Patricia Merkle Tree inclut une **authentification cryptographique** qui peut être utilisée pour stocker des associations **[clé, valeur]**



https://eth.wiki/en/fundamentals/patricia-tree

PRÉFIXE

Un seul caractère hexadécimal (nombre codé sur4 bits) est appelé un **nibble.**

Les nœuds **branche** comportent **16 enfants** (index 0 à 15 codant un nibble en hexa).
L'**index 16** est réservé pour stocker **sa propre [clé, valeur]**.

Un **préfixe hexadécimal** (HP) est utilisé pour encoder les clés.

HP code également si la clé est de longueur **paire** ou **impaire** :

- Un nibble correspond à 4 bits soit un quartet en hexadécimal
- La mémoire gère des octets, soit 8 bits en hexadécimal
- Selon que la longueur de la clé est paire ou impaire, on optimise

MISE EN APPLICATION 1/4

[clé, valeur]

```
<64 6f> : 'verb'
<64 6f 67> : 'puppy'
<64 6f 67 65> : 'coin'
<68 6f 72 73 65> : 'stallion'
```

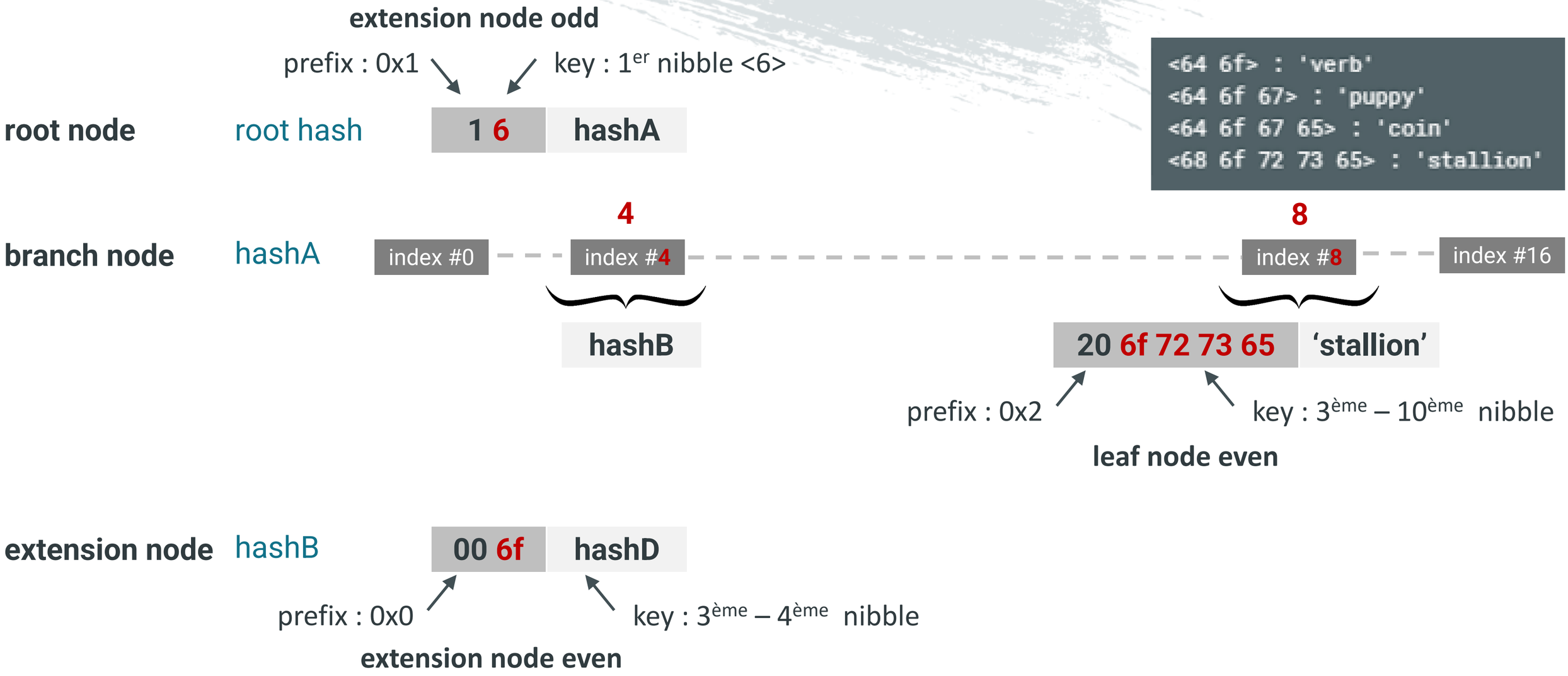
Patricia Merkle Tree

```
rootHash: [ <16>, hashA ]
hashA:    [ <>, <>, <>, <>, hashB, <>, <>, <>, [ <20 6f 72 73 65>, 'stallion' ], <>, <>, <>, <>, <>, <>, <>, <> ]
hashB:    [ <00 6f>, hashD ]
hashD:    [ <>, <>, <>, <>, <>, <>, hashE, <>, <>, <>, <>, <>, <>, <>, <>, <>, 'verb' ]
hashE:    [ <17>, [ <>, <>, <>, <>, <>, <>, [ <35>, 'coin' ], <>, <>, <>, <>, <>, <>, <>, <>, <>, 'puppy' ] ]
```

Pour s’exercer :

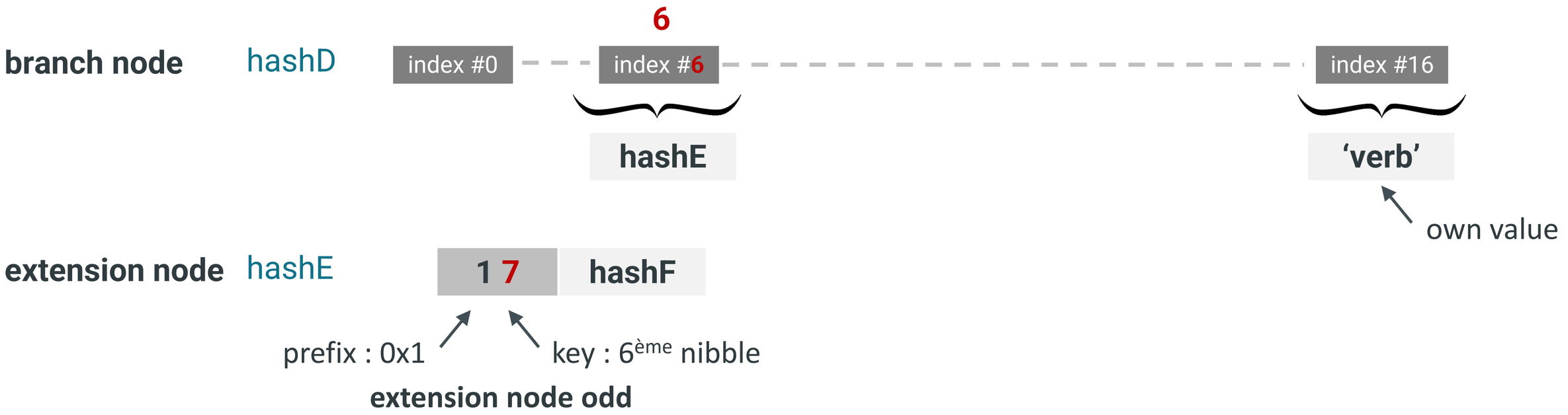
git clone git@github.com:ebuchman/understanding_ethereum_trie

MISE EN APPLICATION 2/4



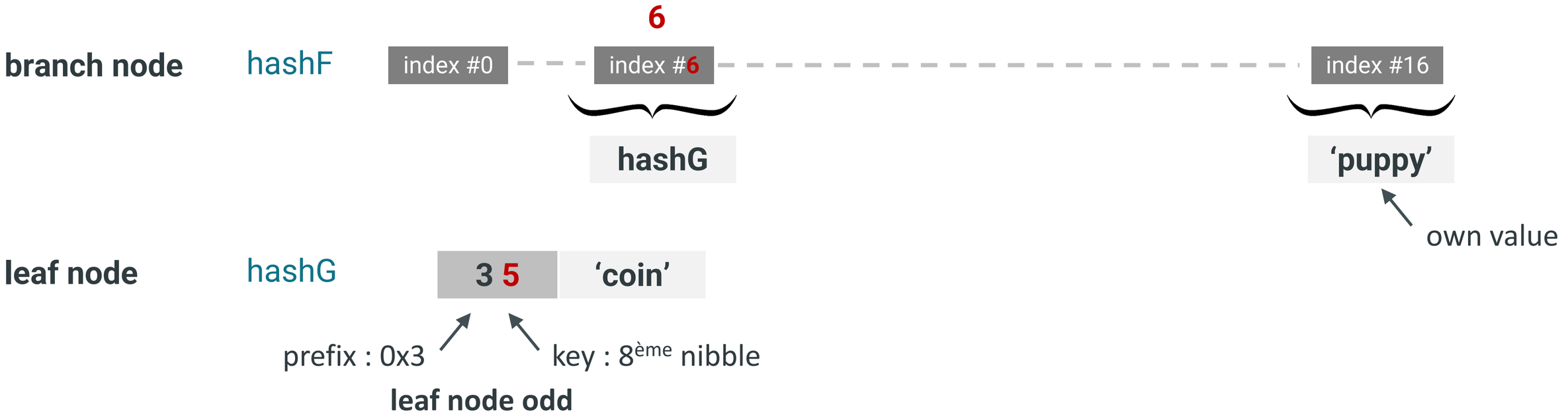
MISE EN APPLICATION 3/4

```
<64 6f> : 'verb'
<64 6f 67> : 'puppy'
<64 6f 67 65> : 'coin'
<68 6f 72 73 65> : 'stallion'
```



MISE EN APPLICATION 4/4

```
<64 6f> : 'verb'
<64 6f 67> : 'puppy'
<64 6f 67 65> : 'coin'
<68 6f 72 73 65> : 'stallion'
```

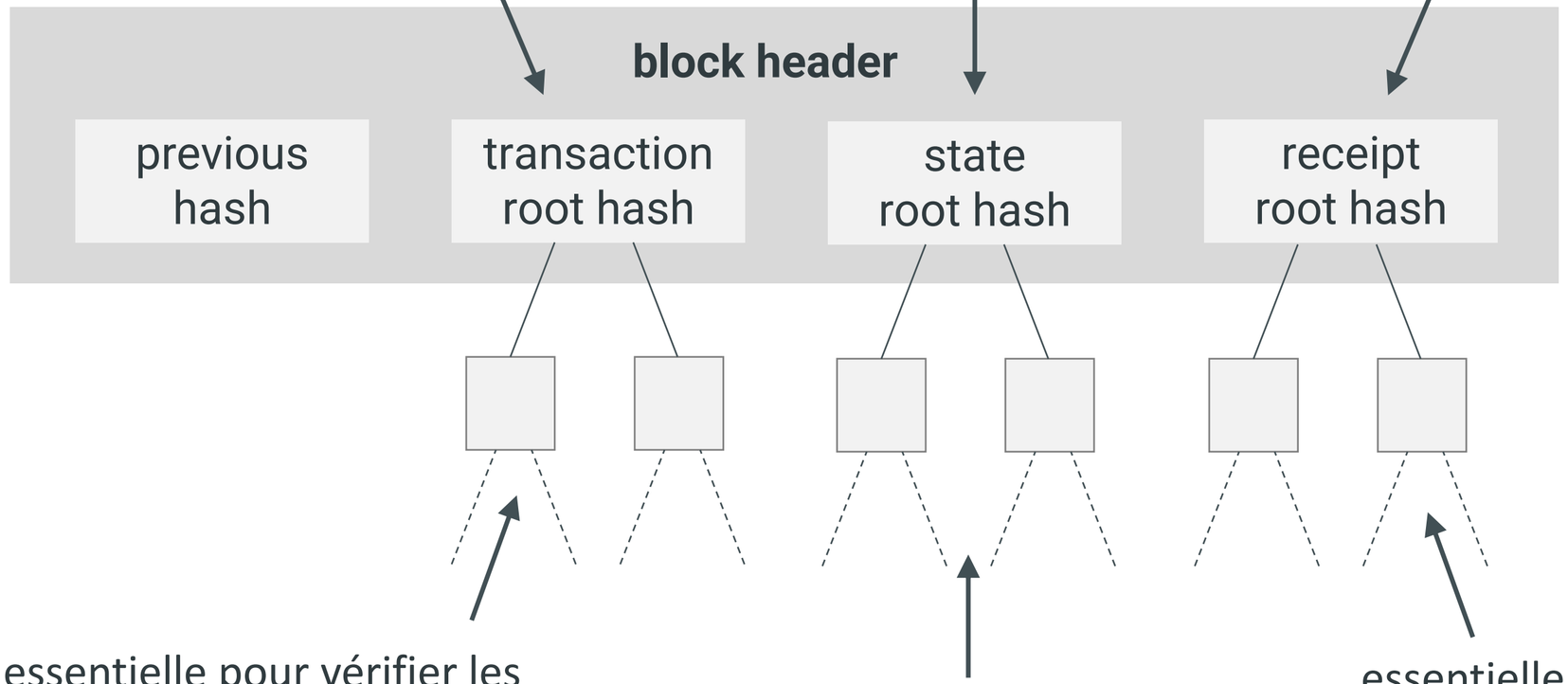


A QUOI SERT CETTE STRUCTURE ?

liste des transactions, dont la clé est le numéro de la transaction dans le bloc

variable d'état (dont solde du compte), dont la clé est l'adresse de compte ethereum associé

Récépissé des transactions, dont la clé est l'adresse du compte ethereum émetteur

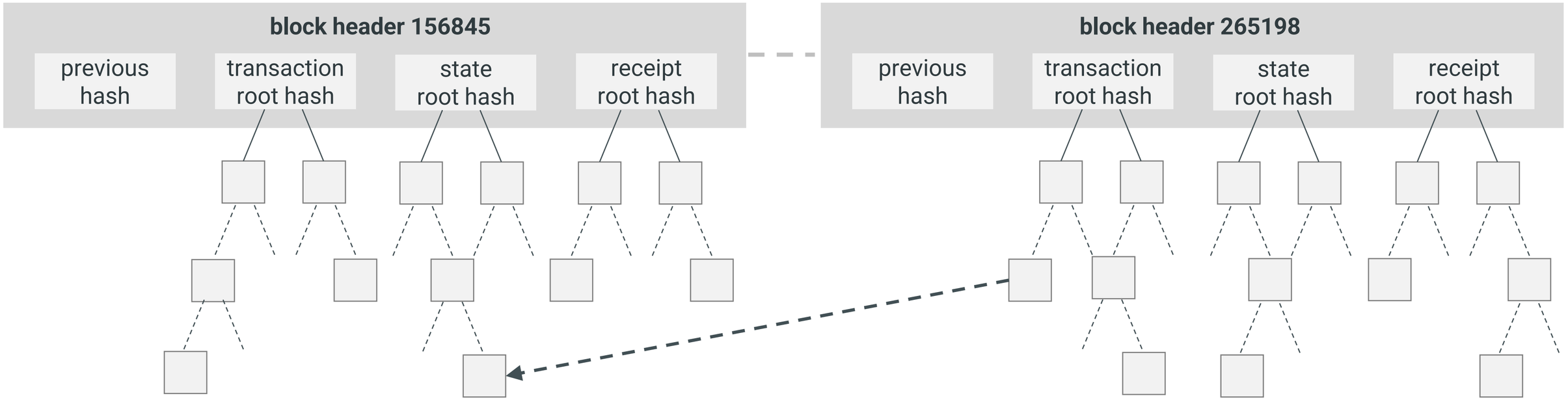


essentielle pour vérifier les nouvelles transactions

essentielle pour vérifier l'intégrité du ledger (les états) depuis le bloc genèse

essentielle pour effectuer l'audit de l'historique d'un compte et pour éviter le « *double spending* »

VÉRIFICATION ENTRE BLOCS



Pour vérifier une transaction entrante, on consulte les états dans les blocs précédents :

- Balance du compte émetteur
- Compteur du compte émetteur (nonce)
- Smart contracts destinataire (dans la chaine authentifié par son adresse)
- …

UN PUISSANT PROCÉDÉ DE CONSULTATION

Cette structure est le support d'un **protocole** de **client léger très avancé**.

Les clients légers peuvent émettre des **requêtes vérifiables** et obtenir très vite la réponse (sans besoin de puissance de calcul) :

- Quel est le solde actuel de mon compte ?
- A quelle transaction correspond ce récépissé ?
- Dans quel bloc est incluse cette transaction ?
- Quelles sont toutes les instances d'un événement de type X (par exemple les versements sur un contrat de crowdfunding) émises par cette adresse au cours des 30 derniers jours ?
- Ce compte existe-t-il ?
- …

ORACLE

Le terme Oracle vient de la mythologie grecque :
« Réponse d'une divinité au fidèle qui la consultait »

Qu’est-ce qu’un Oracle ?

- source externe à la blockchain qui fournit des données à un smart contract.
- outils permettant de lier la blockchain au monde réel/physique.
- Pas d’authentification des données provenant de l’extérieur par la blockchain

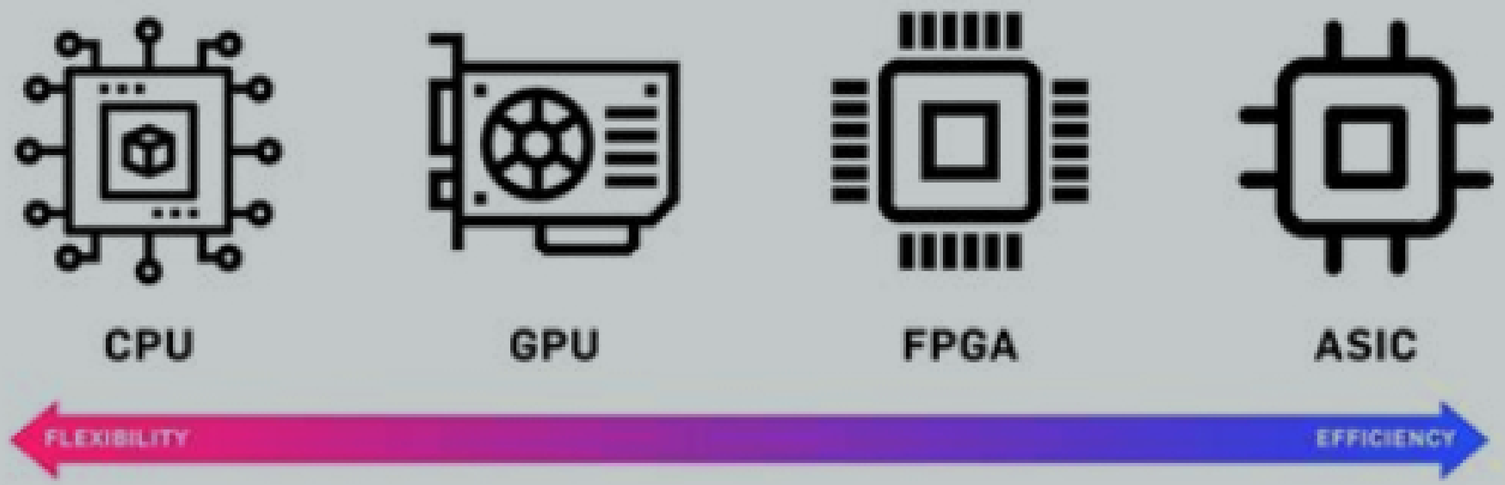
Pourquoi utiliser un Oracle ?

- Les machines virtuelle ethereum ne peuvent pas gérer les nombres aléatoires
- Seul le champ ‘data’ des transactions peut contenir des données, de petite taille et non confidentielles

Comment l’implémenter ?

- Collecte de la donnée sur une source externe
- Inclusion dans une transaction signée vers un smart contract
- Mise à disposition de la donnée dans un bloc dans l’arbre d’états

PoW SUR GPU

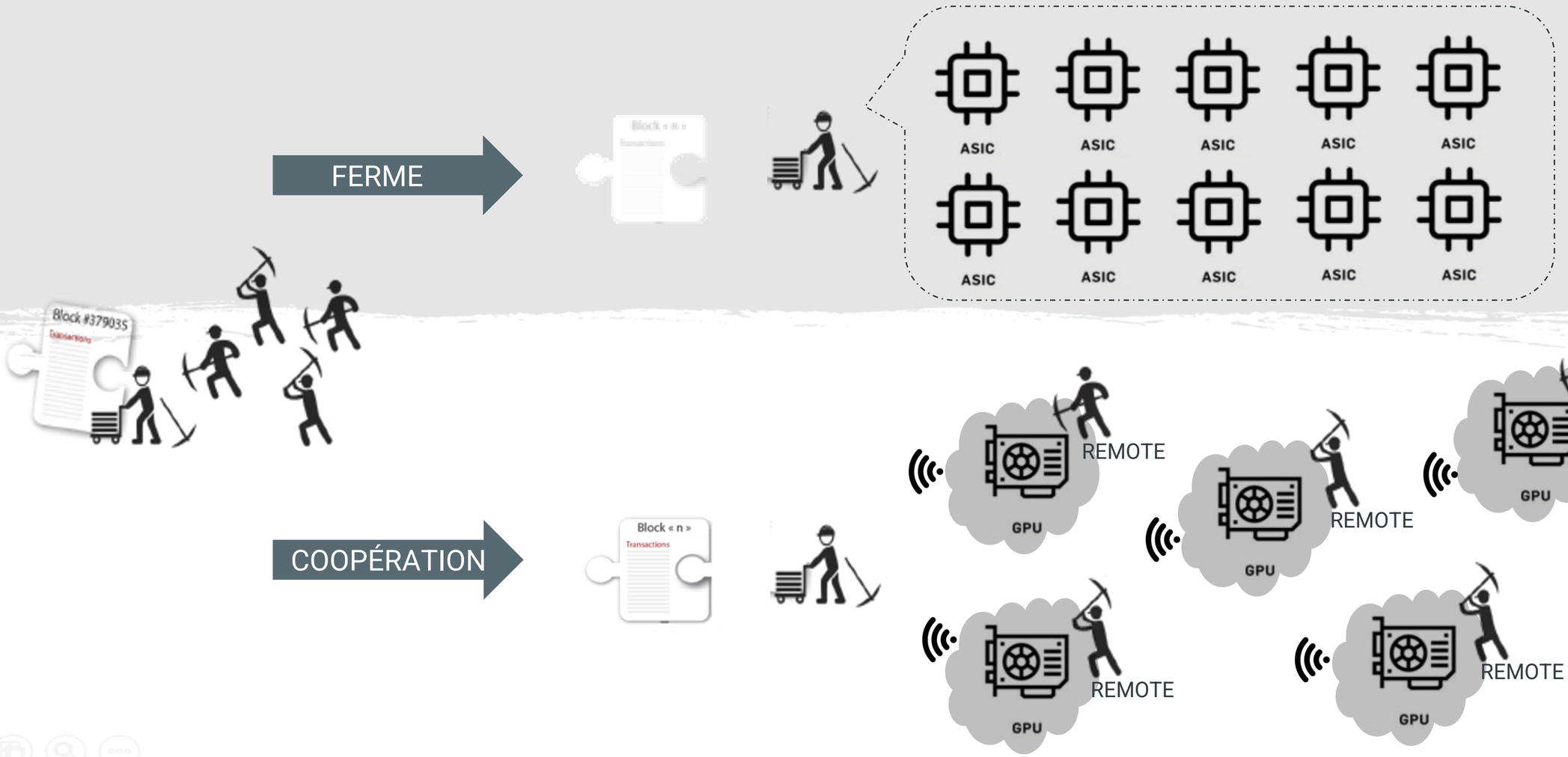


Ethereum introduit le **minage sur GPU** dans le but d'éviter le regroupement en fermes de mining sur ASIC.

L'algorithme **Etash** propose une implémentation de la PoW dont le calcul est plus efficace sur GPU que sur ASIC : il requiert le **chargement de données contextuelles** (provenant du bloc précédent) **en mémoire vive**.

MINING : FERME VERSUS COOPÉRATION

La **PoW** avec **Ethash** favorise le **mining sur GPU** et évite le regroupement en fermes de mining basées sur des batteries d'ASICs.
Le **regroupement en ferme** est quand même possible mais sous forme de **coopération.**



FERME DE MINING

https://www.bitmain.com/

ETHASH 1/2

- **Ethash** est une évolution de l'algorithme Dagger-Hashimoto.
- La difficulté s'ajuste dynamiquement de sorte qu'en moyenne, un bloc est produit par le réseau toutes les 12 secondes.
- Ethash est basé sur la construction d'un **DAG** (Directed Acyclic Graph) chargé dynamiquement en mémoire vive tous les 30 000 blocs, soit environ 100 heures appelé un **epoch**.
  La taille actuelle du DAG d'Ethereum est d'environ 4 Go. Il continuera de croître au fur et à mesure que la difficulté augmentera.
  Le nonce est tiré aléatoirement parmi les données du DAG.
- Le terme **memory hard** fait référence à une situation dans laquelle le temps nécessaire à la réalisation d'un problème de calcul donné est principalement déterminé par la quantité de mémoire requise pour contenir les données. Le temps consacré à lire les données dans le DAG est beaucoup plus long que le temps consacré au calcul.

  https://github.com/chfast/ethash

ETHASH 2/2

Cette technique permet de rendre le calcul de la PoW plus efficace sur GPU qu'avec un ASIC.

Ethash peut être implémenté sur une des cartes GPU suivantes :

- MSI GTX 1050Ti Gaming X 4G
- MSI RX 580 Gaming 8G
- MSI RX 560 AERO ITX OC 4G
- Sapphire RX 560 Pulse OC 4G
- Sapphire RX 550 Pulse 4G
- Sapphire RX 470 Mining 4G

https://github.com/ethereum-mining/ethminer

La récompense pour le mineur gagnant est de 3 éthers, en plus du prix du gas consommé par l'exécution de toutes les transactions qui sont contenues dans le bloc.
La récompense pour les mineurs de blocs valides mais non retenus (uncle block) est de 7/8 de la récompense statique du bloc, soit 2,625 éthers.

DISTRIBUTION VERSUS GOUVERNANCE

La **PoW** avec **Ethash** favorise le minage sur GPU.

L'objectif est d'**éviter** le regroupement en fermes de mining basées sur des batteries d'**ASICs** fonctionnant en parallèle.

Le regroupement en ferme est quand même possible mais sous forme de **coopération**.

- Vitalik & Co réfléchissait à un consensus par PoS, « Casper », avant l'attaque « The DAO » qui a conduit à un hard fork pour restituer la somme volée et invalider le smart contract bogué.
- Avant de réaliser le « hard fork », Vitalik & Co ont effectué un sondage auprès des fermes de mining pour savoir s'ils adopteraient le « fork ».
- Le sondage a conduit à 80% des fermes favorables à l'adoption de la nouvelle branche.
- Le « hard fork » a été réalisé conduisant à deux branches ETC (secondaire) et ETH (principale).
- Retour d'expérience : si le protocole « Casper », nettement plus distribué (et équitable) avait été déployé, le « hard fork » aurait été impossible.
- Vitalik & Co choisissent de prendre du recul sur ce sujet dans le but de conserver la gouvernance de la blockchain… et pour cela, ils font évoluer « Casper ».

CASPER : PROOF-OF-STAKE

Casper CBC
Correct-by-Construction

- recherche menée par **Vlad Zamfir**, chercheur à la Fondation Ethereum
- Modèle **prouvé formellement** (correct par construction)
- initialement axée sur les protocoles PoS pour les **blockchains permissionless**
- évolution vers un champ d'étude plus large, comprenant une famille de modèles PoS
- pourrait remplacer ou compléter le Casper FFG à l'avenir

Casper FFG
Friendly Finality Gadget

- FFG privilégie une **transition en plusieurs étapes** pour le déploiement de la PoS
- FFG n’est pas prouvé formellement
- recherche menées par **Vitalik Buterin**, cofondateur d'Ethereum
- la proposition initiale consistait en un système hybride PoW/PoS (beacon chain)
- depuis The Merge (2022), Casper FFG repose sur un modèle de **PoS pur**

CASPER FFG

- la vérification et la validation des nouveaux blocs de transactions sont effectuées par des **validateurs** de blocs, qui sont **sélectionnés proportionnellement à leur mise**.
- Pour devenir un validateur de bloc dans la première phase de Serenity, les validateurs doivent déposer **une mise de 32 éthers** (ETH) au minimum sur un smart contract enregistré sur la blockchain Ethereum (1.0).
- Par exemple, un validateur ayant déposé 64 ETH aura le double du poids de vote d'un validateur ayant déposé 32 ETH.
- Tous les utilisateurs capable d'immobiliser au moins 32 ETH peuvent **contribuer** au protocole de **consensus**, ce qui favorise la décentralisation.
- Avant The Merge, les récompenses de bloc sont constituées uniquement de frais de transaction. Désormais, les **récompenses** sont un **pourcentage** de la mise.
- Avec cette structure, l'efficacité et la sécurité ne sont pas encore prouvées.
  Casper FFG n'est pas encore totalement résistant aux attaques à 51%. Une spécification formelle est encore nécessaire pour définir une règle de fork nécessaire pour répondre aux attaques.

SERENITY ROADMAP

1er décembre 2020 à midi

Phase 0 : Beacon Chain

- La chaîne Beacon est une nouvelle chaîne parallèle à Ethereum
- Elle introduit la PoS avec Casper et une nouvelle monnaie ETH2
- Pour être mineur, il faut investir 32 ETH sur cette chaine (non récupérable)

Phase 1 : Shard Chains

- Le sharding consiste à étendre la blockchain au-dessus de la chaine Beacon
- 100 shards seront initialement déployés
- Les validateurs valideront les transactions de leurs shards

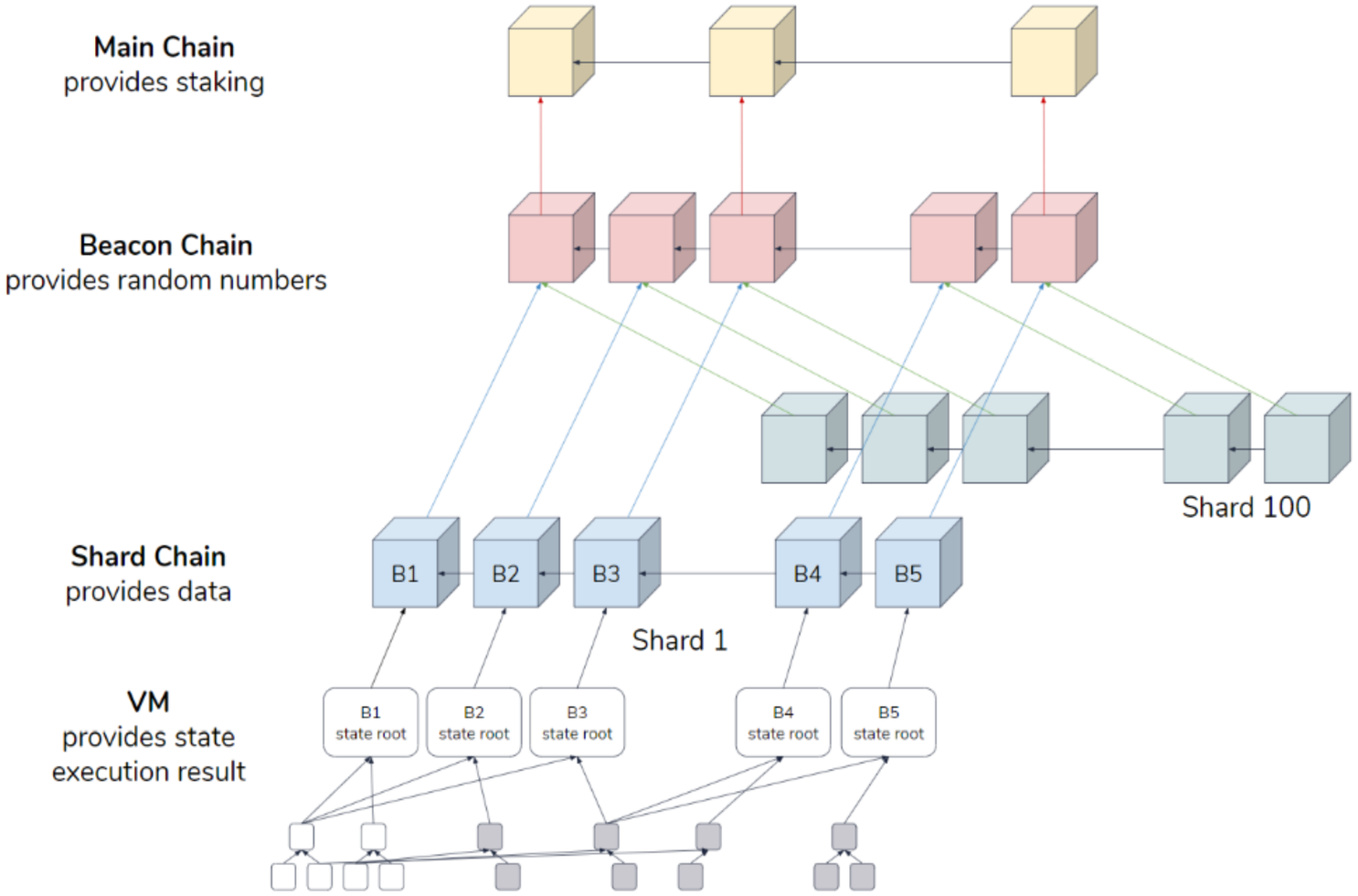
Phase 2: eWASM

- Nouvelle machine virtuelle Ethereum appellee eWASM (Ethereum WebAssembly)
- Support de différents langages de smart contract

Phase 3: Continued Improvement

- Cross-shard transactions
- Lightweight clients
- …

ETHEREUM 1.0 (AVANT LE 14/09/2022)



- PoW sur GPU
  GOUVERNANCE
- PoS avec engagement de 32 ethers
  DISTRIBUTION
- Sharding
  AUGMENTATION DU DÉBIT DES TRANSACTIONS

BEACON CHAIN

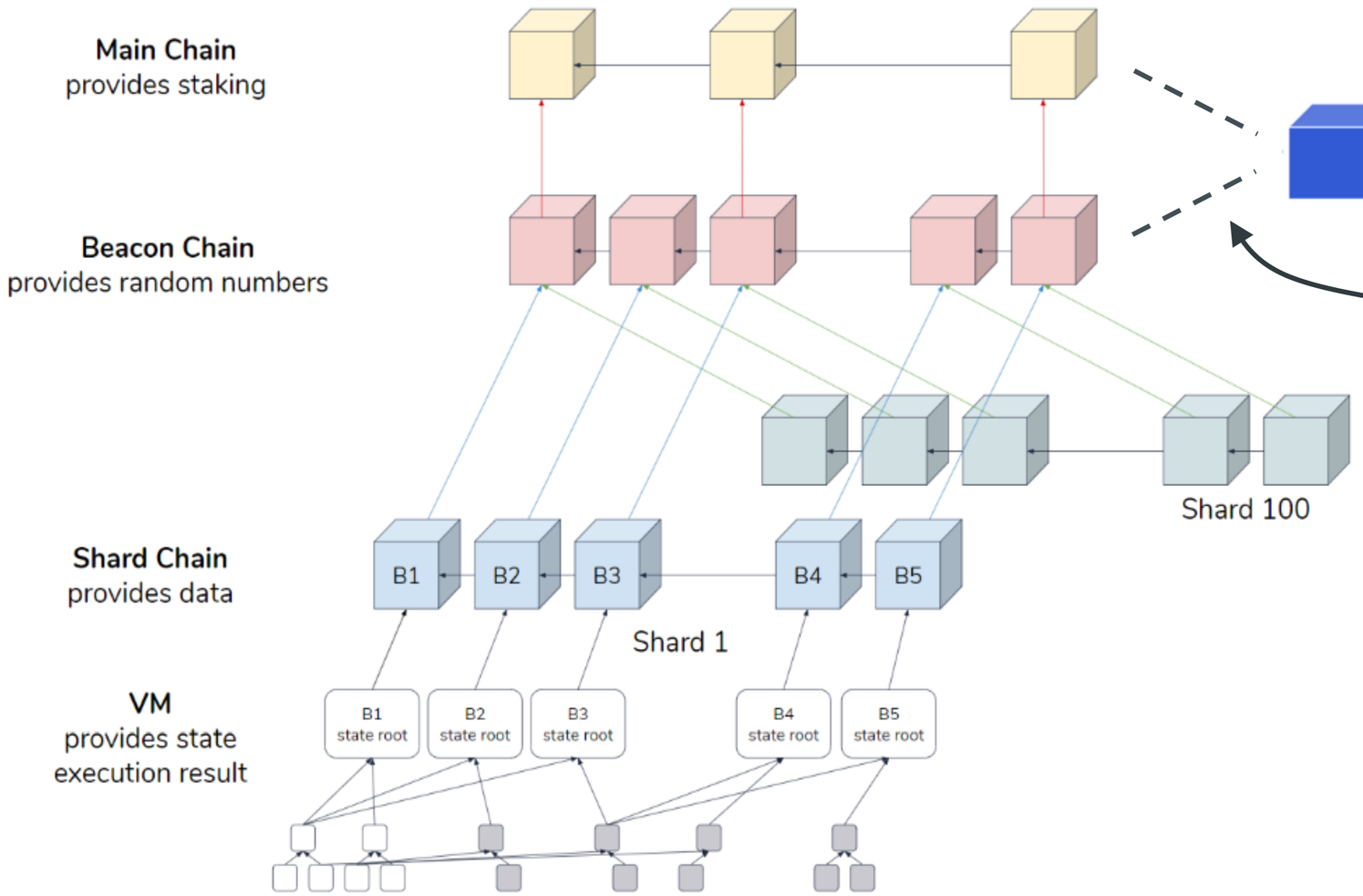
La **Beacon chain** contrôlera et coordonnera le **réseau étendu de shards.**
Mais elle ne gèrera pas les comptes ethereum, ni les contrats intelligents.

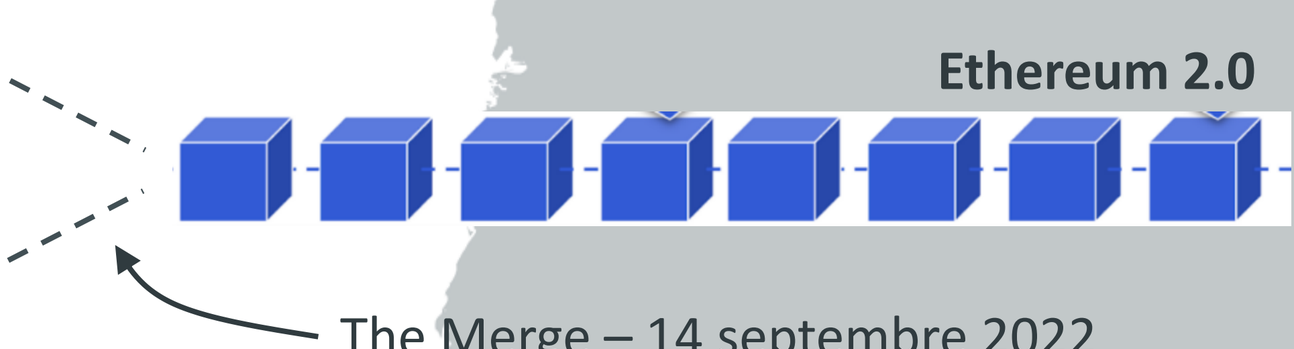
Dans un premier temps, la **Beacon chain** sera séparée de la blockchain Ethereum **Mainnet**.
A terme, ils seront **connectés**.
L'objectif est de **fusionner** le Mainnet avec le système de preuve d'enjeu de la chaîne Beacon.

Après la fusion de Mainnet et de Beacon Chain, l'étape suivante introduira des **shards chains** pilotées par la Beacon chain et le protocole de Proof-of-Stake Casper.
Les **shards** augmenteront la **capacité du réseau** et amélioreront la **vitesse des transactions** en étendant le réseau à **64 blockchains**.

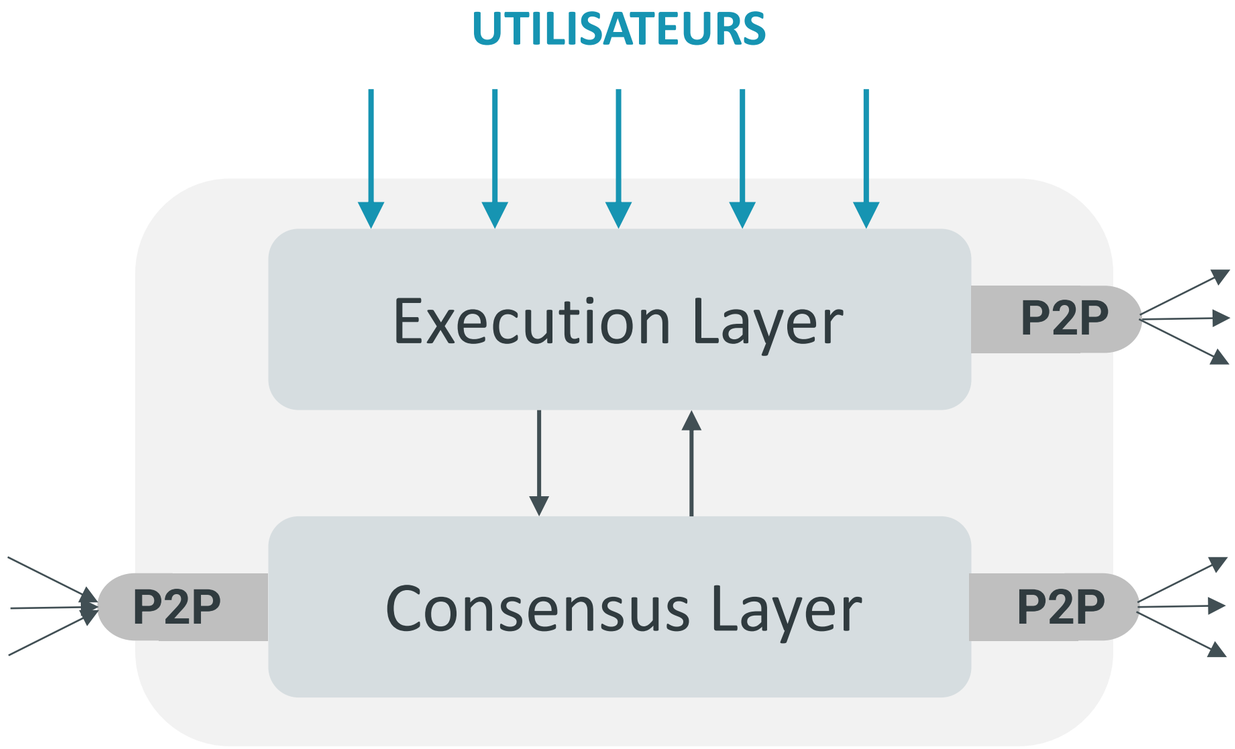
https://beaconscan.com/

ETHEREUM 2.0 (APRÈS LE 14/09/2022) : THE MERGE





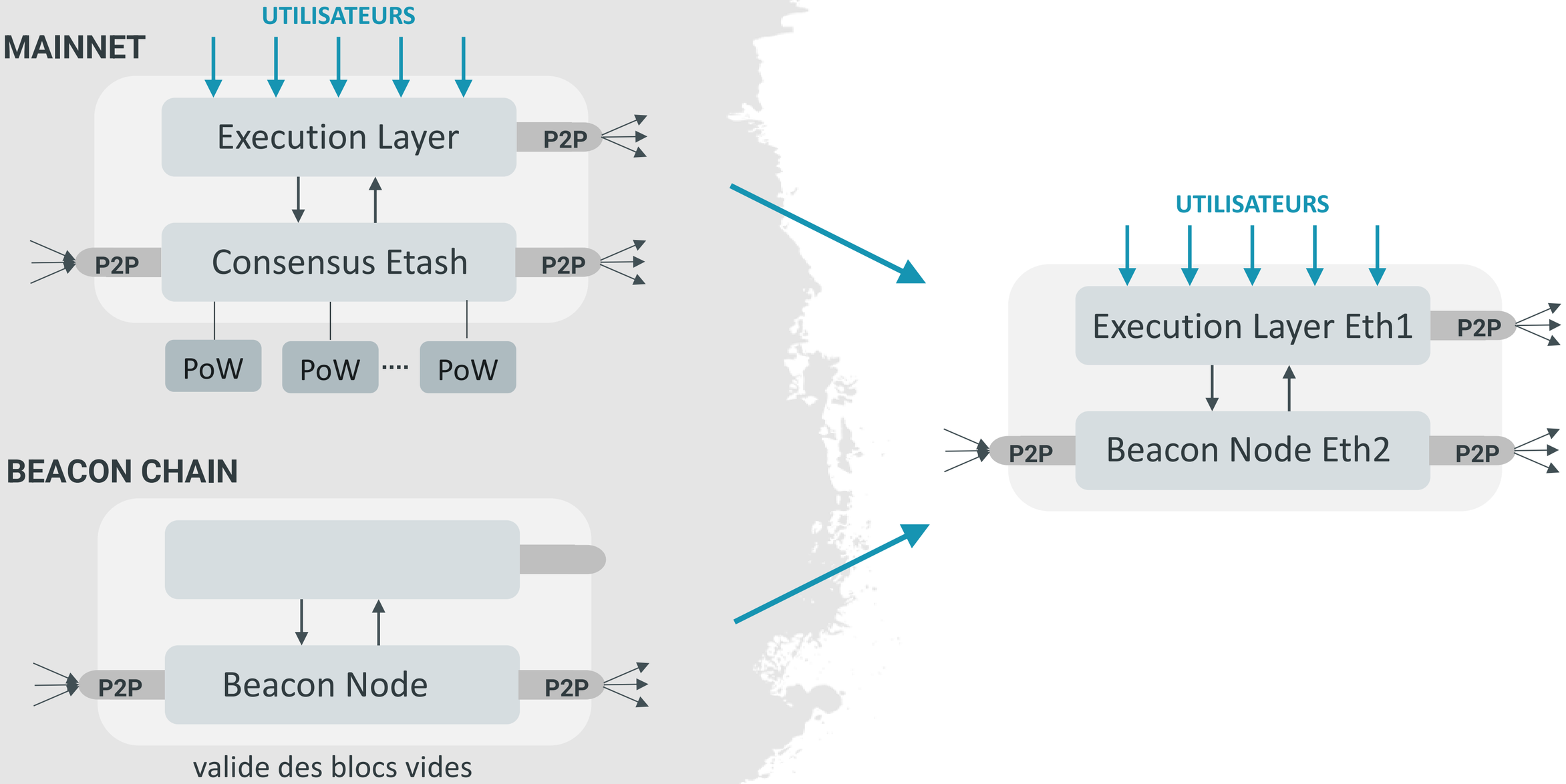
NŒUD VALIDATEUR



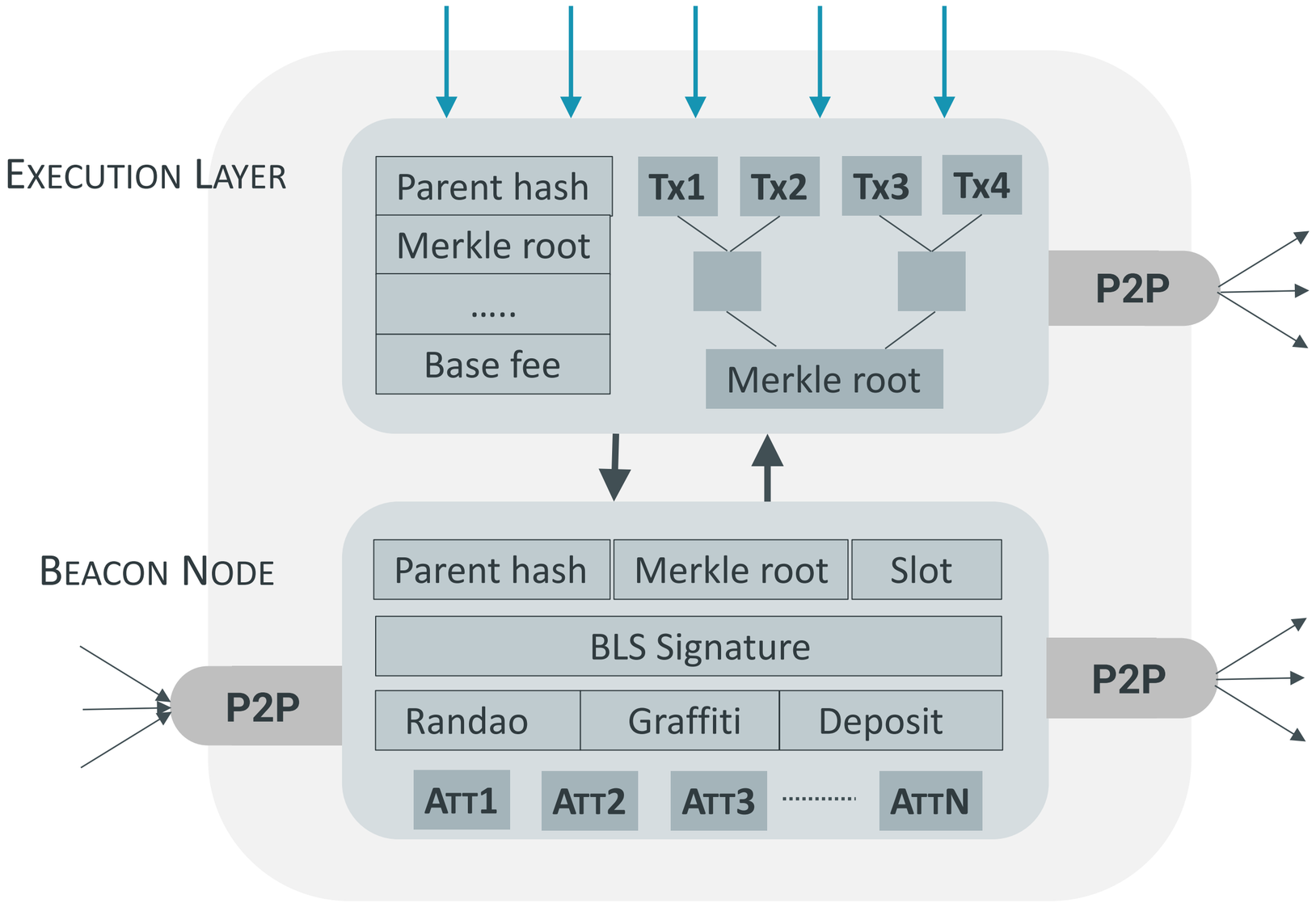
- Construction des nouveaux blocs
- Maintenance du MemPool
- Exécution des smart contracts
- Synchronisation
- Propagation P2P (Gossip) des Tx

- Investissement de 32 ETH
- Traque des blocs HEAD dans le réseau P2P
- Propagation P2P (Gossip) des blocs
- Attestation des blocs
- Réception des récompenses (Rewards)

ETHEREUM 1.0 → ETHEREUM 2.0

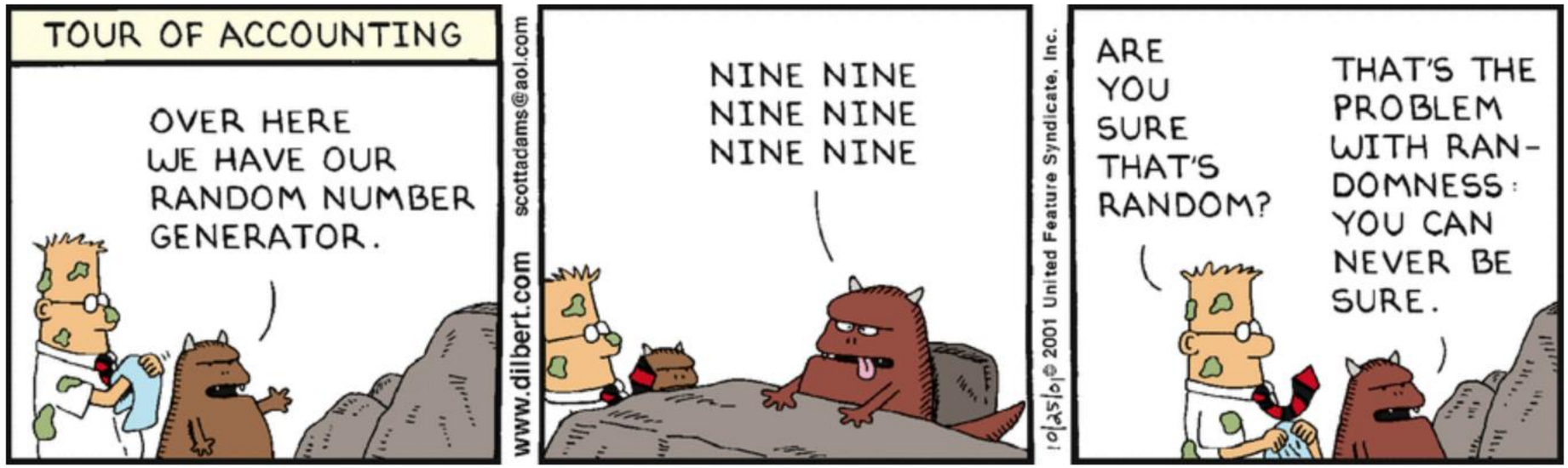


NOUVEAU NŒUD VALIDATEUR



CE QUI CHANGE AVEC ETHEREUM 2.0 …

Le opcode DIFFICULTY est remplacé par PREVRANDAO



En l’absence de PoW, la DIFFICULTY est fixée à ZERO. Mais comment choisit-on le prochain bloc ?
Un epoch de 32 validateurs est construit afin d’éviter que le validateur du bloc gagnant soit prédictible, ce qui ouvrirait la surface d’attaques autour de ce validateur.

CE QUI CHANGE AVEC ETHEREUM 2.0 …

La signature ECDSA est remplacée par la signature BLS (pour les blocs uniquement)

La signature numérique de Boneh-Lynn-Shacham (BLS) cryptographic signatures est plus efficace que la signature ECDSA pour aggréger les attestations des différents validateurs

Finalisation : les blocs sont validés avec un consensus au 2/3 des validateurs

Pour être accepté, le nouveau bloc doit être choisi par plus de 2/3 des validateurs.
Une fois que le bloc est confirmé, il est marqué d'un nouveau tag « finalized », ce qui le rend immuable.

L'intervalle entre les blocs est exactement de 12 secondes après The Merge

UNE COMMUNAUTÉ ACTIVE

Un projet **open-source** avec une **communauté** de développeurs **active**

développement

- **Réseau** : Ganache, testnet, mainnet
- **Clonage** : à partir du code source py-ethereum, go-ethereum (geth) avec la PoA « Clique » intégrée
- **Interface** : web3.py, web3.js…
- **Développement smart contracts et API** : brownie

ressources

S'impose de facto comme un **standard** :

- **ERC** (Ethereum Request for Comment) - https://eips.ethereum.org/
- **EIP** (Ethereum Improvment Proposal) - https://eips.ethereum.org/erc

explorateur

- etherchain - https://www.etherchain.org/
- ethplorer - https://ethplorer.io/fr/
- ethblockexplorer - https://ethblockexplorer.org/
- blockchain - https://blockchair.com/ethereum
- … et en **ligne de commande** avec brownie par exemple

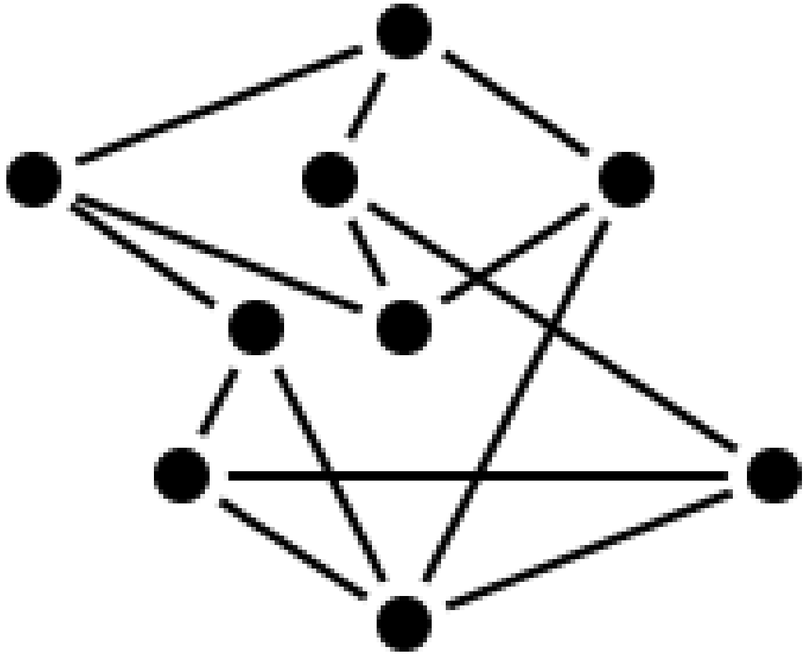
QUELQUES RÉFÉRENCES

1. « white paper », https://ethereum.org/en/whitepaper/
2. « yellow paper », Gavin Wood, « ethereum: a secure decentralized generalized ledger », pertersburg version 41c1837, 14 february 2021, https://ethereum.github.io/yellowpaper/paper.pdf
3. « mauve paper », https://www.ethereum-france.com/%EF%BB%BFethereum-2-0-mauve-paper-traduction-francaise/
4. Andreas M. Antonopoulos, « Mastering Ethereum », 2018
5. Algorithms and Data Structures Research & Reference Material: PATRICIA, by Lloyd Allison, Monash University
6. Patricia Tree, NIST Dictionary of Algorithms and Data Structures
7. Christine Hennebert and Florian Barrois, "Is the blockchain a relevant technology for the industry 4.0?," 2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), Paris, France, 2020, pp. 212-216, doi: 10.1109/BRAINS49436.2020.9223290

SECTION 4

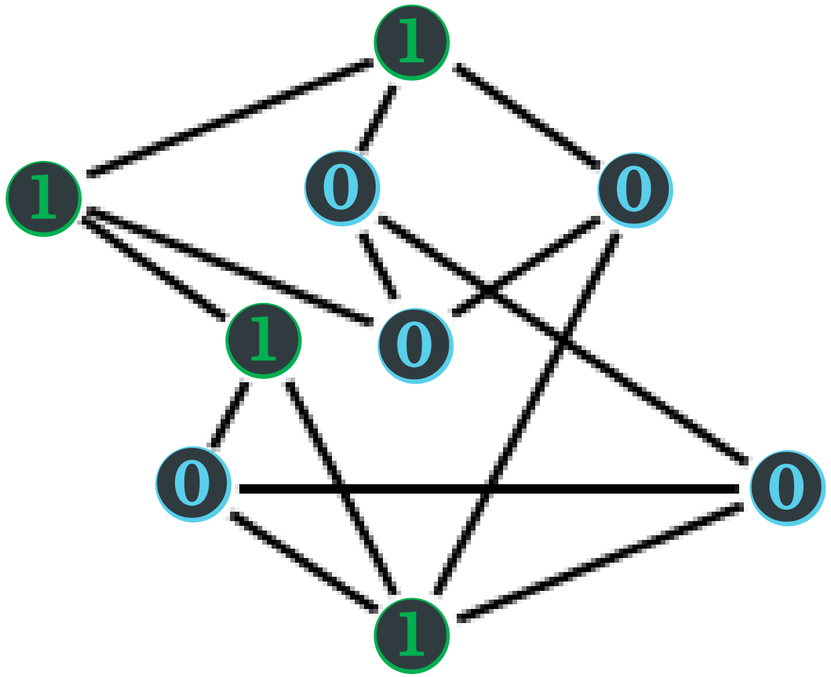
PROTOCOLES DE CONSENSUS

PROPRIÉTÉS D'UN SYSTÈME DISTRIBUÉ



- **Concurrence** des **dispositifs**
- Pas d'**horloge** globale
- **Défaillance possible** d'un dispositif

BON FONCTIONNEMENT ?



Jeu : Supposons que chacun des nœuds tire aléatoirement soit le chiffre 0, soit le chiffre 1.

Règle commune : le résultat validé correspond au chiffre qui obtient la majorité

Correctness : Comment détermine-t-on que le système se comporte correctement ?

SAFETY & LIVENESS

S’accorder sur l’état d’un registre et le maintenir au cours du temps fait appel à deux notions :
La « ***safety*** » assure que le réseau reste un bon état
La « ***liveness*** » assure que le réseau est toujours disponible

Propriété	***Safety***	***Liveness***
Définition	Aucun évènement n’arrivera	Un évènement doit arriver

De façon triviale,
- la « ***safety*** » est assurée lorsqu’aucune nouvelle transaction n’est acceptée dans le registre,
- la « ***liveness*** » pourrait consister à accepter toutes les transactions nouvellement soumises.

De la tension entre ces deux notions émergent de nombreux mécanismes de consensus.

CORRECTNESS

Le système se comporte correctement lorsque les règles communes sont respectées

Un algorithme de consensus garantit le bon fonctionnement d'un système distribué (***Correctness***) en assurant les **fonctionnalités** de :

Validity : toute valeur décidée doit être proposée par au moins l'un des dispositifs
Agreement : tous les dispositifs non défectueux doivent s'accorder sur la même valeur
Termination : tous les nœuds non défectueux propose un résultat dans le temps imparti

	Validity	***Agreement***	***Termination***
Safety	☑	☑	
Liveness			☑

CAP THÉORÈME

En 1985, Fisher, Lynch (MIT) et Patterson ont montré qu'aucun algorithme déterministe de consensus ne peut satisfaire simultanément les trois propriétés de Cohérence (***Consistency***), Disponibilité (***Availability***) et Tolérance au Partitionnement (***Partition Tolerance***) dans un réseau distribué asynchrone.

- **Cohérence** : tous les nœuds du système voient exactement les mêmes données au même moment
- **Disponibilité** : garantie que le système est opérationnel
  toutes les requêtes reçoivent une réponse dans le temps imparti
- **Tolérance au partitionnement** : le système continue à fonctionner correctement,
  y compris en présence de partitionnement du réseau

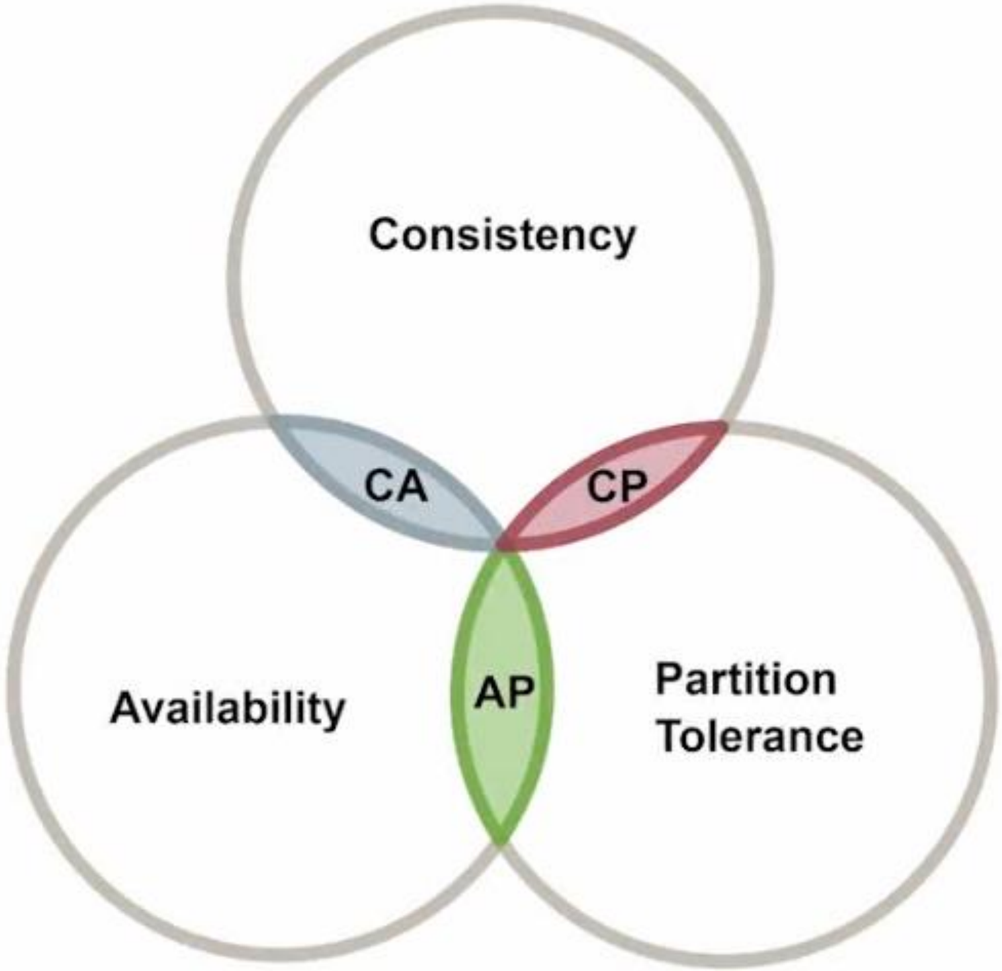
Tout système de calcul distribué ne peut garantir à un instant donné que **deux** de ces contraintes.

CAP THÉORÈME

Availability & Consistency
Il n’y a pas de nœuds au comportement byzantin (défaillant)

Availability & Partition Tolerance
La cohérence est assurée avec un certain retard

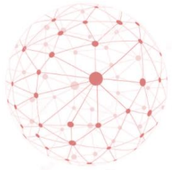
Consistency & Partition Tolerance
On a besoin de synchronisme pour que le système soit fonctionnel



MÉCANISME DE LOTERIE VERSUS ÉLECTION

Construction du bloc

Calcul de la preuve (respect des règles communes)



Propagation dans le réseau

Validation du bloc

Confirmation du bloc

Mécanisme de loterie	Mise en concurrence des mineurs Besoin d'une incitation à participer Besoin d'un mécanisme pour décourager la tricherie → Crypto-monnaie	Fork possible	Besoin d'un mécanisme de résolution des forks
Mécanisme d'élection	Le leader est déterminé à l'avance Le choix du leader est effectué par un algorithme Pas besoin d'incitation à participer → Pas besoin de Crypto-monnaie	Pas de fork	Retardée si le leader est évincé

NAKAMOTO CONSENSUS

Availability & Partition Tolerance

Mécanisme de loterie

Il y a des règles et procédures communes

Le bloc du gagnant est ajouté à la blockchain

Les pairs votent implicitement en mettant à jour leur copie locale de la blockchain

Le bloc est confirmé après un certain temps (profondeur d'enfouissement)

Comment prévenir la triche ?

Proof-of-Work : PoW

Proof-of-Stake : PoS

Proof-of-Elapsed-Time : PoET

PROOF-OF-STAKE : PoS

Le mécanisme de **PoS** a été introduit par **Peercoin** en **2011**.

- Avec la **PoW**, un validateur est considéré digne de confiance s'il engage une quantité considérable d'énergie pour générer le nouveau bloc.

- Avec la **PoS**, la confiance repose sur la richesse des validateurs. En pratique, la confiance accordée à un validateur est proportionnelle à sa mise, c'est-à-dire la quantité de *coins*, qu'il engage pour générer un bloc, au risque de perdre sa mise.

Plutôt que de s'appuyer sur des récompenses pour la sécurité, la **Proof-of-Stake** s'appuie sur des **pénalités**.
Si un participant place sa mise sur un bloc malhonnête, il est pénalisé et perd le montant qu'il a mis en jeu.
Ainsi, **un acte malveillant** est **pénalisé** beaucoup **plus lourdement** que le gain obtenu en agissant honnêtement.

SÉCURITÉ VERSUS MOTIVATION

PoW

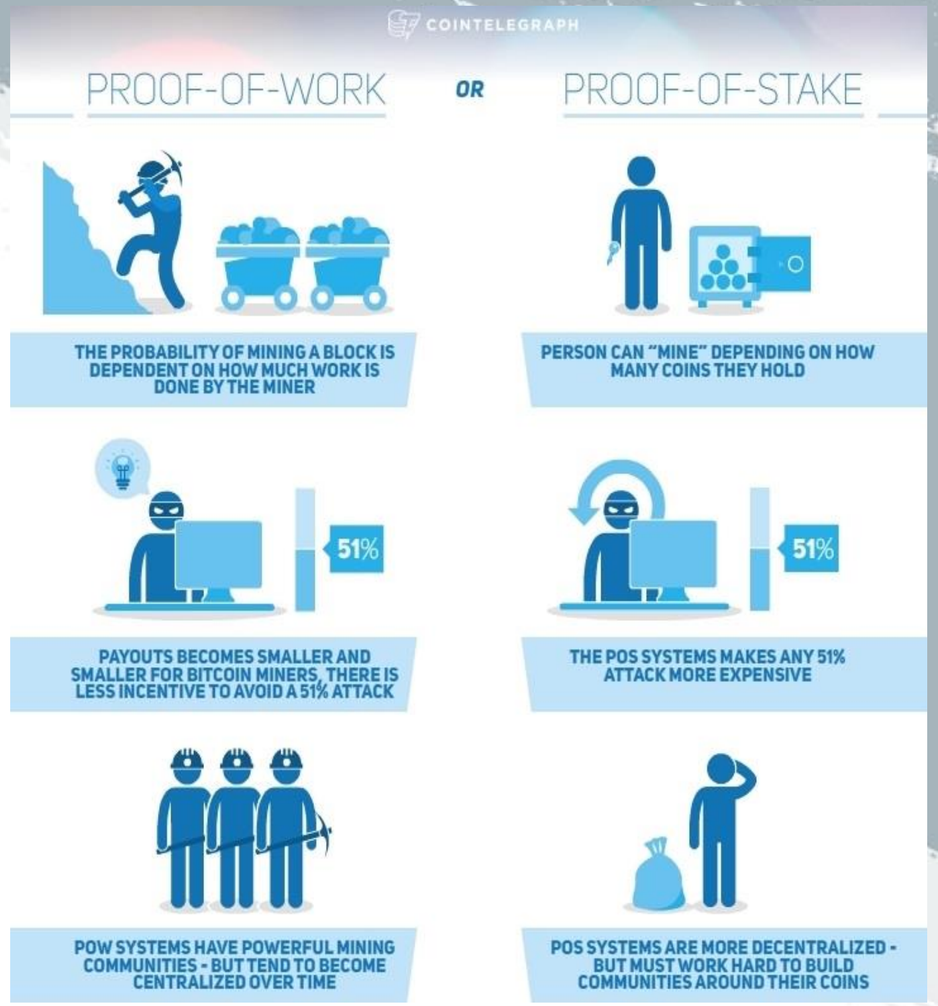
- pas d'avantage pour le défenseur
- le coût de l'attaque et le coût de la défense sont dans un rapport de 1:1
- La quantité de ressources dépensée est la même que l'acteur se comporte honnêtement ou malhonnêtement
- les contraintes sont inflexibles
- il n'y a rien en place pour prévenir ou décourager les acteurs malveillants. Le PoW le permet

PoS

- On peut spécifier les propriétés que l'on veut maintenir à l'avantage du défenseur
- le cout de l'attaque doit être supérieur au coût de la défense :
  - introduit des pénalités
  - punit les comportements malhonnêtes davantage que la PoW
  - la sécurité vient de l'immobilisation du capital pour un temps suffisamment long
- Décourage les comportements malhonnêtes avec des répercutions explicites

DILEMME

PoW - Problème des fermes de mining



PoS - Problèmes « Nothing at Stake »
« Long-Range Attack » ...

PoW
Plus on est puissant,
Plus on s'enrichit

PoS
Plus on est riche,
Plus on s'enrichit

ATTAQUE À 51%

PoW

- Requiert pour l’attaquant de disposer de plus de 51% de la puissance de calcul du réseau
- Ne décourage pas explicitement cette attaque
- Perte de l’incitation lors d’un tour, mais possibilité de participer à nouveau au tour suivant
- En pratique, perte de valeur du Coin nuisible au business des mineurs

PoS

- Requiert pour l’attaquant de disposer de plus de 51% des Coins du réseau
- Requière mettre beaucoup de Coins en jeu, au risque de les perdre
- Dissuasion d'attaquer un réseau dans lequel on détient une part majoritaire

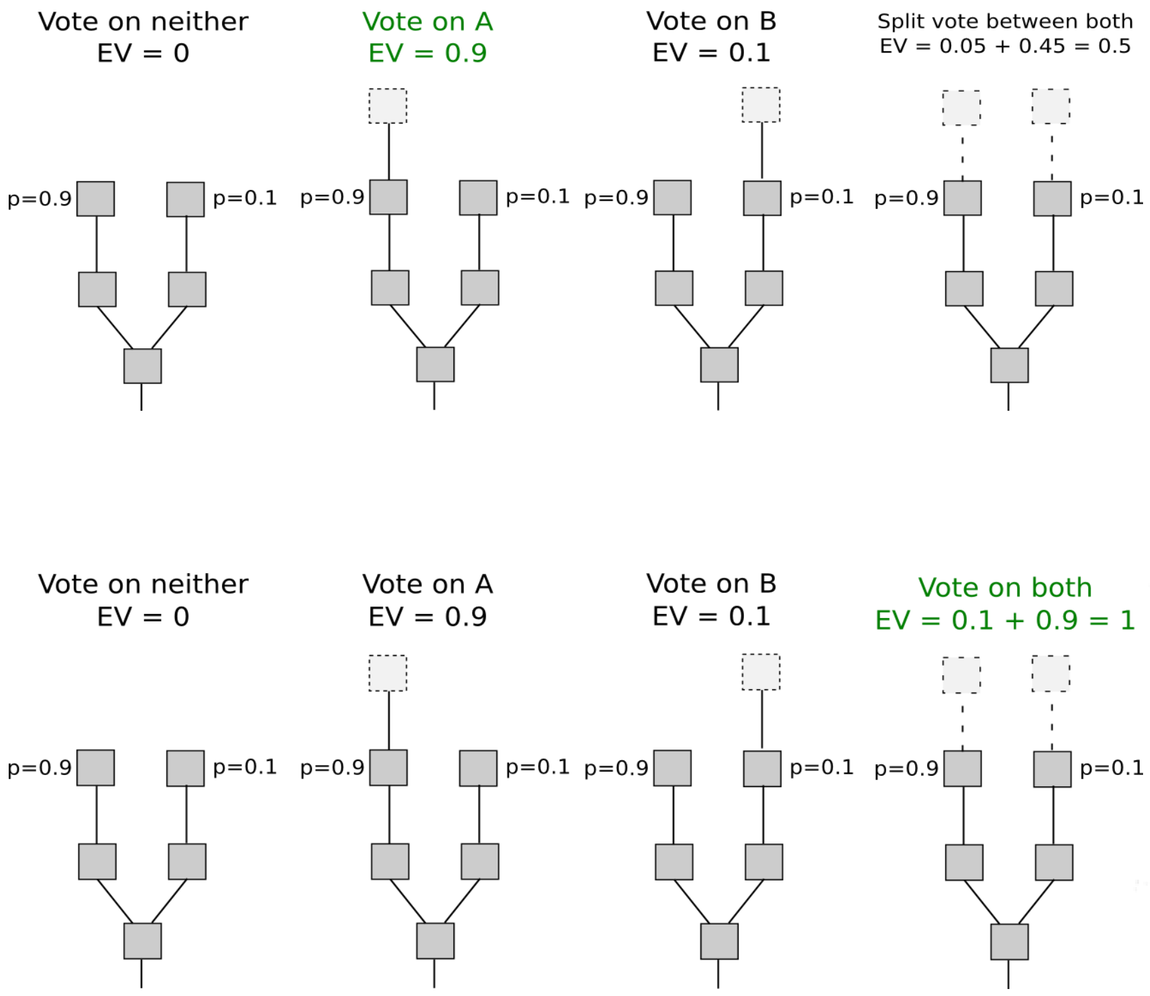
VULNÉRABILITÉS DE LA PoS

Le problème du « ***nothing-at-stake*** » :

les validateurs ont intérêt à créer plusieurs branches, en générant plusieurs blocs lors du même tour afin de maximiser leur chance que ce soit leur bloc qui soit validé et ajouté à la chaine, et ainsi remporter l'incitation.

L'attaque « ***long-range*** » consiste à créer une branche à partir d'un bloc de l'historique de la chaine, puis à forger des faux blocs à un rythme plus élevé que sur la chaine légitime jusqu'à ce que la fausse branche devienne plus longue que la chaine légitime. La chaine de blocs étant la chaine la plus longue, l'attaquant pend alors le contrôle, sa branche devenant la chaine principale.

NOTHING-AT-STAKE



Choix de la branche à miner

PoW

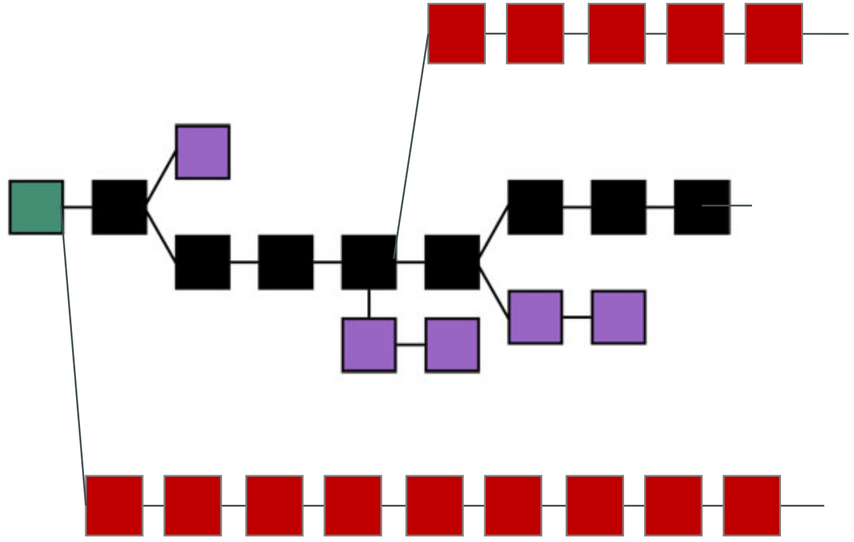
- un mineur doit faire un choix parmi toutes les branches possibles.
- Les options sont mutuellement exclusives et en choisir deux simultanément revient à gaspiller ses forces.
- La stratégie payante est celle de « miner » sur la branche où on a le plus de chance de gagner, ce qui revient à choisir la branche la plus longue.

PoS

- Dans un PoS, il faut trouver un moyen pour éviter qu'un utilisateur « riche » répartisse son solde sur plusieurs comptes pour se donner plus de chance.
- Une solution, appelée « Slasher », consiste à pénaliser les validateurs qui génèrent plusieurs blocs lors du même tour, en confisquant leur mise.

LONG-RANGE ATTACK

Un acteur qui détient suffisamment de Coins peut construire une branche à partir d'un bloc qu'il a signé (ou vendre ses clés privés à un attaquant)

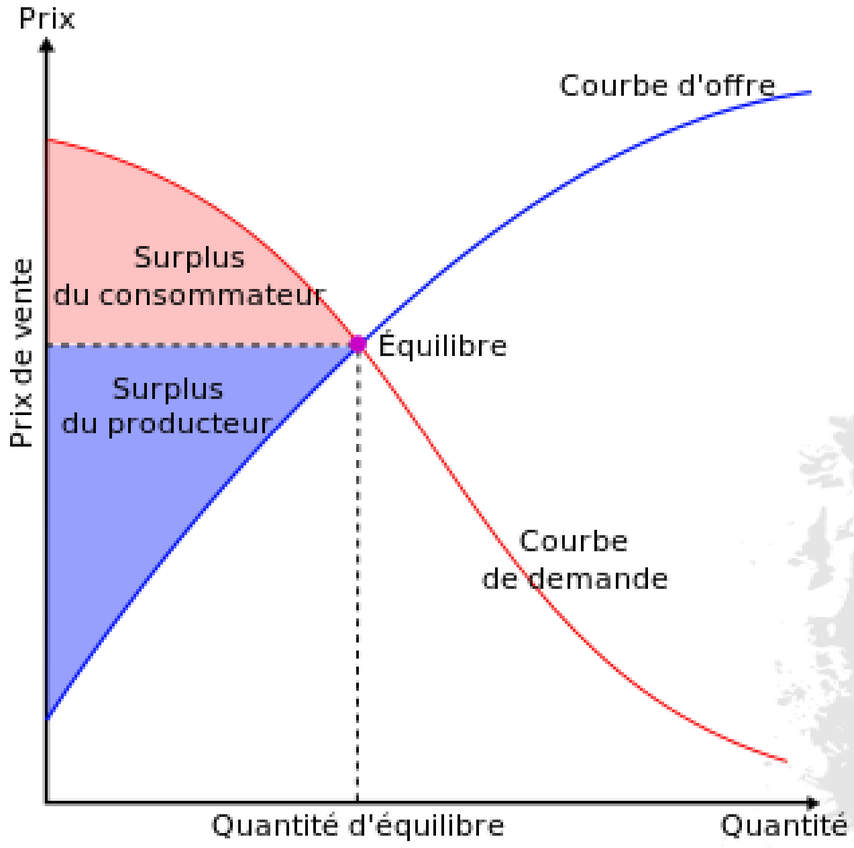


Un attaquant peut construire une branche à partir du Genesis block s'il détient au moins un Coin à l'origine

Quelle est la branche légitime ?
La plus longue ?

→ weak subjectivity

PROBLÈME DE LIQUIDITÉ



La **PoS** introduit un problème de **liquidité** :

- Puisque les validateurs doivent **immobiliser** leurs fonds afin d'avoir un intérêt dans le réseau, le **montant réel** des **fonds disponibles** pour effectuer des transactions est beaucoup **plus faible**.
- Cela **réduit** la **liquidité** de la crypto-monnaie elle-même, diminue le montant des fonds disponibles et **augmente le prix et la demande**.
- Cela **encourage** aussi les validateurs à **conserver** leurs fonds et à les **vendre** lorsque les prix augmentent, plutôt que de participer aux transactions, c'est-à-dire à spéculer.

PROOF-OF-ELAPSED-TIME : PoET

Mécanisme de loterie introduit par Intel qui exploite l'**enclave de sécurité SGX** des processeur **core i7.**
PoET est destinée aux blockchain **permissioned** et est intégrée dans Hyperledger Sawtooth.
SGX est un **TEE** (Trusted Execution Environment) qui permet l'**exécution d'un code signé** numériquement dans une zone de confiance.

Le protocole PoET se déroule en plusieurs étapes :

1. Le dispositif disposant d'une enclave SGX se procure le code embarqué de PoET signé par Intel
2. Le dispositif le charge dans l'enclave et vérifie que ce code est légitime
3. Le dispositif envoie vers ses pairs une attestation « join » signée avec sa propre clé pour demander à rejoindre le réseau de validateur
4. A chaque tour (nouveau bloc), chaque dispositif tire un nombre aléatoire.
5. Celui qui a tiré le nombre le plus petit diffuse son certificat, et s'il est accepté comme leader, il propose ensuite son bloc au consensus

PoET : AVANTAGES ET INCONVÉNIENTS

Avantages

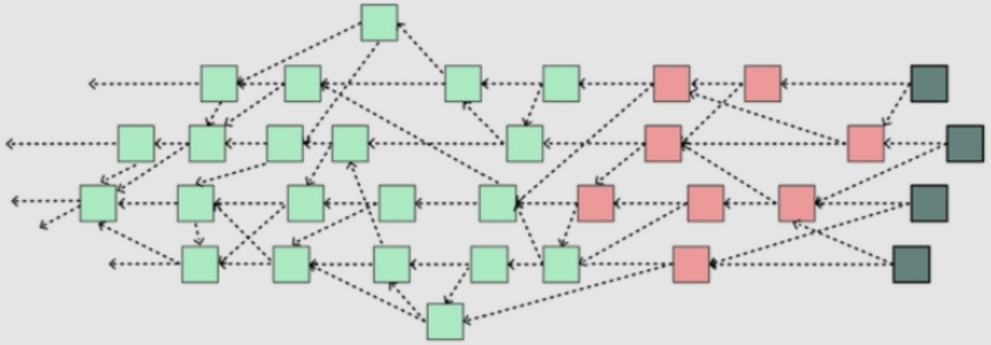
- PoET propose une amélioration **écologique** à la PoW
- Introduit une solution élégante au problème de **sélection aléatoire d'un leader**
- Introduit de la flexibilité dans la gestion de l'**incentive**

Inconvénient

- SGX est vulnérable aux attaques **Spectre** et **Meltdown**
- Requiert de disposer d'un processeur Intel avec SGX intégré et d'**activer la licence**
- Intel se positionne comme **tiers de confiance** et toute la confiance repose sur son matériel

IOTA

The Tangle protocol



- IOTA est un **DLT *permissionless***
- Le protocole **Tangle** est introduit avec un DAG
- dédié à l'Internet des Objets (**IoT**)
- pour des **transactions (M2M)** *machine-to-machine*

https://github.com/IOTAledger

Pour **émettre** une transaction, un dispositif doit avoir **validé au préalable** deux transactions.
Les transactions ne sont **pas ordonnées.**
On vérifie la balance des comptes pour savoir si le **Ledger** est **consistent.**
IOTA utilise un **fragment** de la **PoW** avec une petite difficulté.

IOTA

Scalability

- Plus le nombre de dispositifs est élevé, plus les transactions sont validées rapidement
- Le passage à l'échelle est ainsi favorisé, et souhaitable

Micro-transactions

- Il n'y a pas de frais de transaction
- La multiplicité de transactions est ainsi encouragée, avec une bande passante « dite » élevée

Quantum resistance

- Utilise le schéma « Winternitz One-Time Signature » à la place de ECDSA https://eprint.iacr.org/2011/191.pdf
- Basé sur des opérations ternaires (et non binaires) comme contre-mesure à une attaque visant à retrouver la clé privée https://en.wikipedia.org/wiki/Ternary_operation
- Quid de l'implémentation en embarqué ???

MAM

- Les dispositifs peuvent échanger des données chiffrées

PROBLÈME DES GÉNÉRAUX BYZANTINS

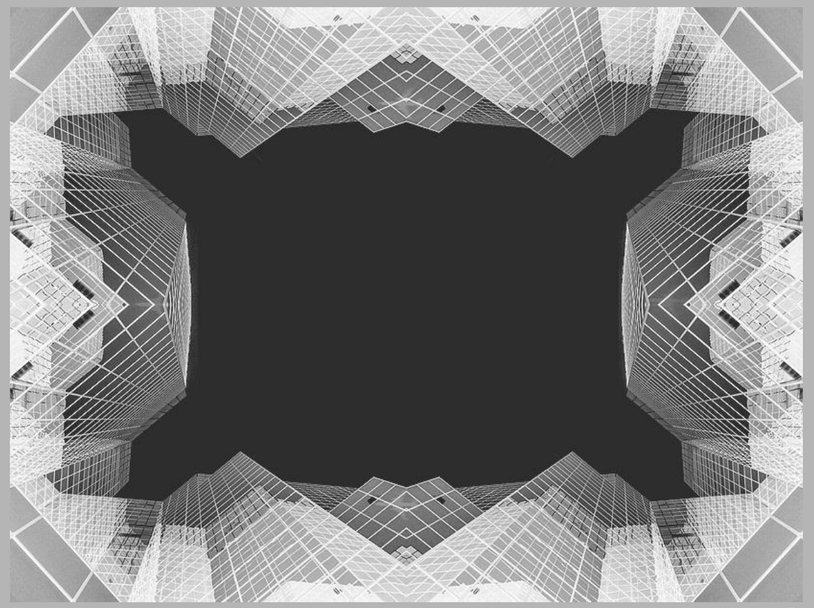
Ce problème traité en profondeur en **1982** par Lamport, Shostak et Pease,illustre la question de la **confiance dans un système décentralisé**.

Il peut se **résumer** ainsi :

« Imaginons qu'un **groupe de généraux**, chacun commandant une partie de l'armée Byzantine, entoure une cité ennemie. Les généraux ne peuvent **communiquer que par messages**, et pour conquérir la cité, ils doivent **s'accorder sur un plan de bataille**.

Mais, au moins **un des généraux** peut être un **traître**, qui en falsifiant un message peut saboter les plans. Le problème est donc de **trouver un algorithme** pour s'assurer que les généraux se mettent **d'accord** sur **un plan de bataille qui ne soit pas saboté**. Et en corollaire cela consiste à déterminer **combien de traîtres peuvent être infiltrés** dans l'armée sans que celle-ci cesse de **fonctionner comme une seule force** ? »

BYZANTINE FAULT TOLERANCE



repli **Général #C** reçoit :
2 attaques

attaque

repli

Général #A

reçoit :
1 attaque
1 repli

attaque

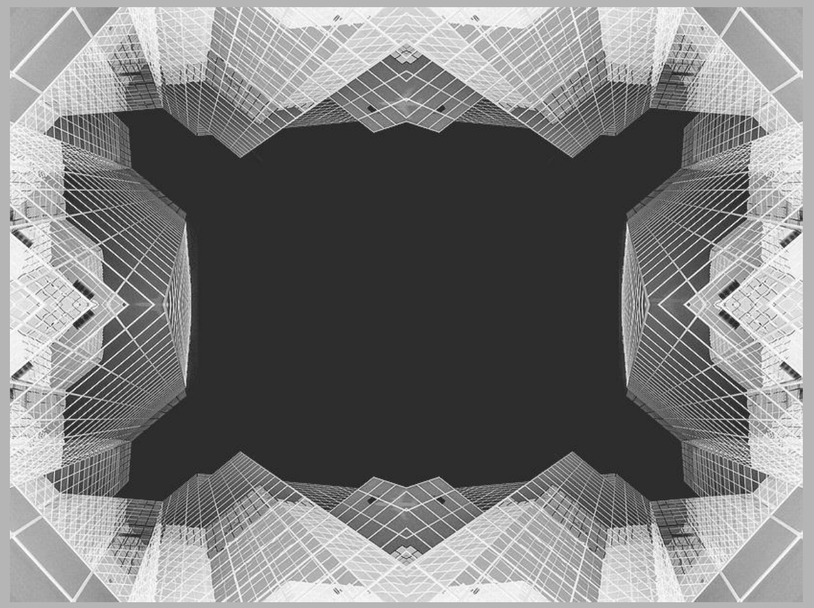
attaque

attaque **Général #B**

reçoit :
1 attaque
1 repli

N=3 Généraux

BYZANTINE FAULT TOLERANCE



attaque

repli

repli

attaque

attaque

attaque

attaque

Général #A

reçoit :
2 attaques
1 repli

Général #C

reçoit :
3 attaques

Général #B

reçoit :
2 attaques
1 repli

Général #D

reçoit :
2 attaques
1 repli

N=4 Généraux

BYZANTINE FAULT TOLERANCE

Pour assurer la **confiance** dans un réseau **distribué**, les protocoles de la famille des **BFT** requièrent qu'au moins **2N + 1 nœuds** du réseau soient **honnêtes** en présence de N nœuds byzantins.

- **Tendermint**
- **PoA** :
  - introduit par **Ethereum** pour la construction de blockchain **permissioned** deux algorithmes :
    - **Aura** : utilisé par Parity
    - **Clique** : utilisé par go-ethereum (Geth)
  - distribution équitable du rôle de création du nouveau bloc parmi un ensemble de nœuds dits « ***authority*** »
  - le **temps** est divisé en « *steps* », à chaque « *steps* » un nouveau **leader** est déterminé (élu)
  - suppose que le réseau est **partiellement synchrone** et que l'horloge de toutes les autorités est synchronisée sur une **horloge de référence** (UNIX time t)
- ...

TENDERMINT

Consistency & Partition Tolerance

Protocole **BFT** qui favorise la cohérence sur la disponibilité.

Les nœuds **validateurs** sont **authentifiés** et **autorisés**.
Un **leader**, choisi au hasard (*round-robin*) parmi les nœuds validateurs, **propose son bloc**.

L'ajout d'un bloc au registre se déroule en **trois phases,** entre chaque phase les nœuds s'attendent :

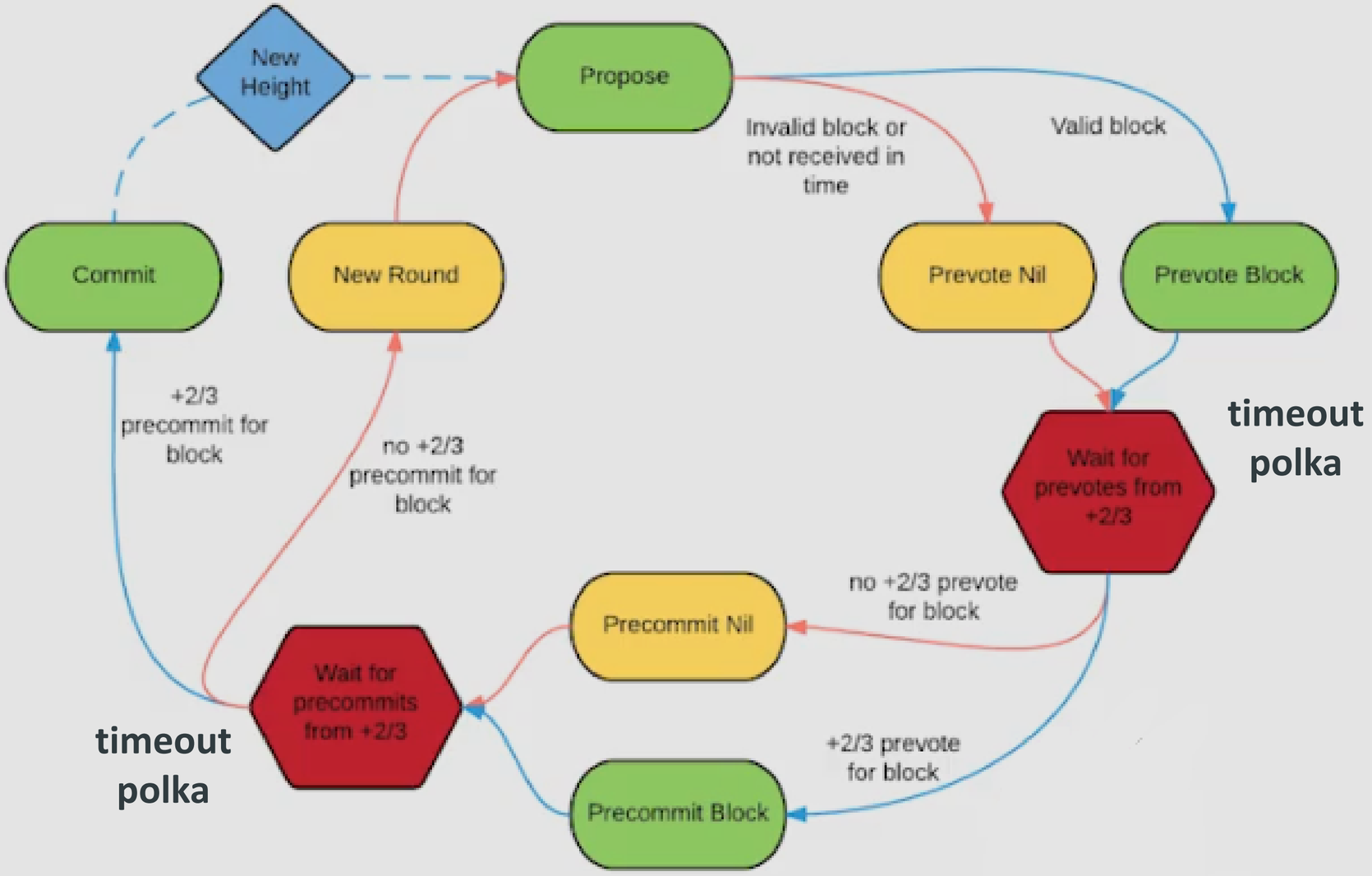
1. **Prevote** : Les pairs du réseau indique leur vote (VALID ou NIL)
   si un validateur ne répond pas, par défaut, son vote est NIL

2. **Precommit** : Si plus de 2/3 des validateurs ont voté VALID : chaque validateur ayant voté VALID détermine s'il « Precommit » le bloc
   sinon, le bloc n'est pas retenu

3. **Commit** : Si le bloc a recueilli plus de 2/3 de « Precommit », il est enregistré dans le registre
   sinon, on repart pour un tour et un nouveau leader est choisi pour proposer son bloc

TENDERMINT

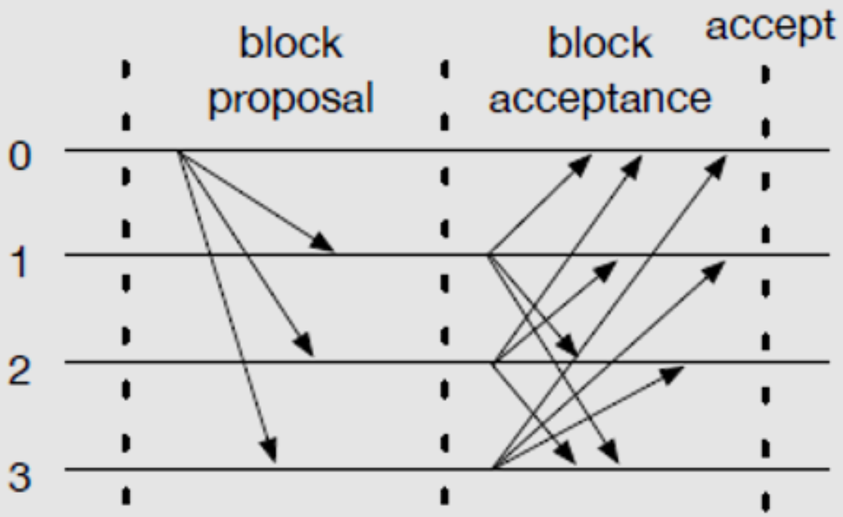
Consistency & Partition Tolerance



PROOF-OF-AUTHORITY : PoA

AURA

Consistency & Partition Tolerance



La **durée** d'un « *step* » est fixe : $step\_duration$

L'**index** d'un « *step* » est déterminé par : $s = \frac{t}{step_{duration}}$

Le ***leader*** du « *step* » s est : $l = s \ mod \ N$

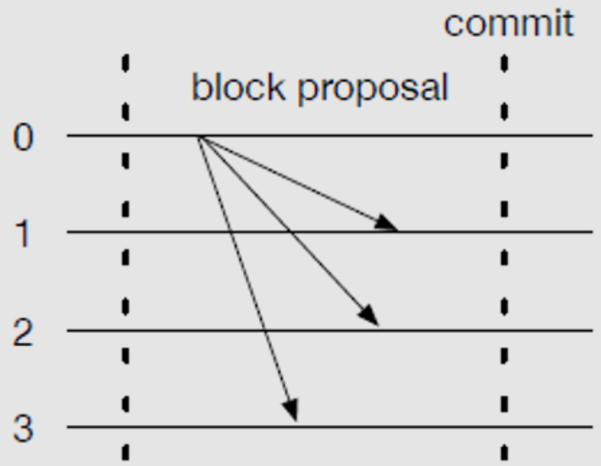
Le ***leader*** est considéré **malhonnête** lorsque :

- aucun bloc n'est proposé durant le « *step* »,
- plusieurs blocs sont proposés durant le « *step* »,
- différents blocs sont proposés à différentes autorités.

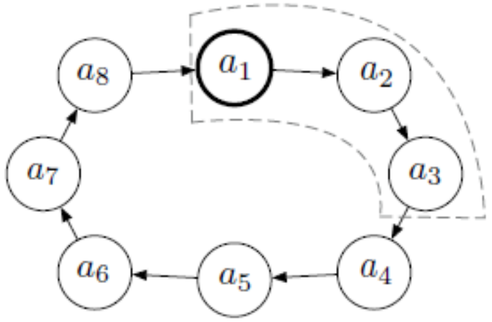
PROOF-OF-AUTHORITY : PoA

CLIQUE

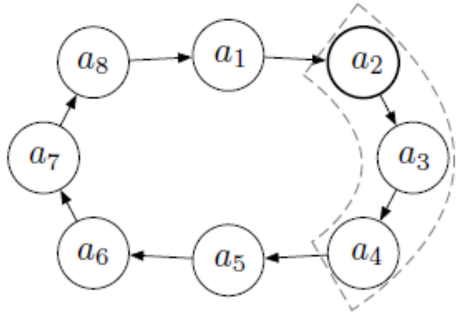
Consistency & Partition Tolerance



step i



step i+1



Le temps est divisé en « *epochs* » comprenant plusieurs « *steps* »

Un « *epoch* » est identifié par une empreinte des blocs précédents

A chaque « *step* », $N - \left(\frac{N}{2} + 1\right)$ *leaders* sont autorisés à proposer un bloc

Le *leader* principal est le 1er à proposer son bloc

Les autres *leaders* proposent leur bloc avec un délai

Lorsque des *forks* apparaissent, ils sont résolus avec l'algorithme GHOST

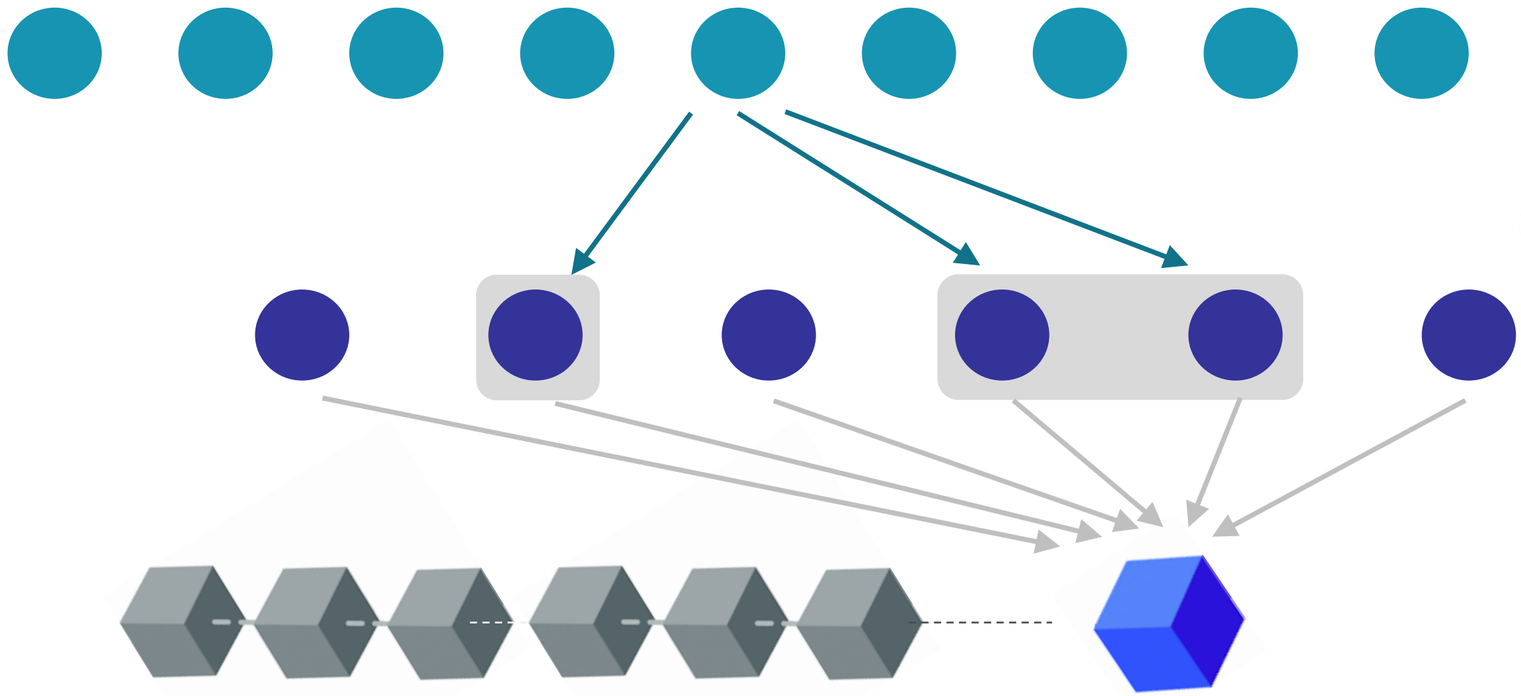
GHOST consiste à attribuer des scores aux blocs, le bloc de score le plus élevé est le gagnant

FEDERATED CONSENSUS

Mécanisme de loterie	Mécanisme d'élection
Nakamoto-based consensus	BFT-based consensus
Vote implicite via l'usage de ressources	Vote explicite pour un bloc
Tout le monde peut participer	Tous les participants sont connus
Blockchain permissionless	Blockchain permissioned
Résistance à la censure	Économe en ressources
Combine les deux mécanismes pour exploiter les avantages de chacun	
Fonctionnement analogue au système des grands électeurs aux Etats-Unis	

QUORUM SLICE

Un **quorum** est un ensemble de nœuds suffisant pour **parvenir** à un **consensus**



individual

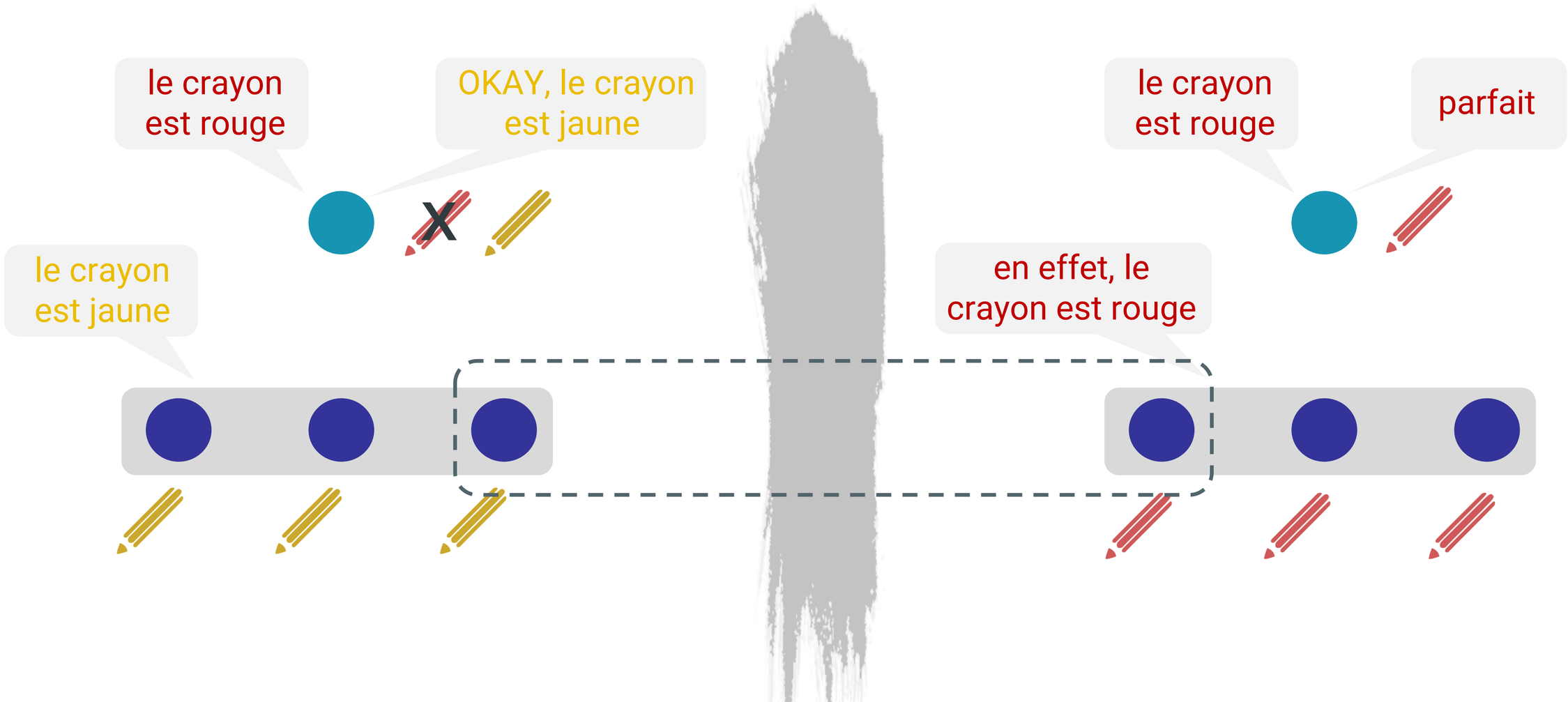
ÉLECTION
CHOIX DES REPRÉSENTANTS

quorum

CONSENSUS
ENTRE SLICES

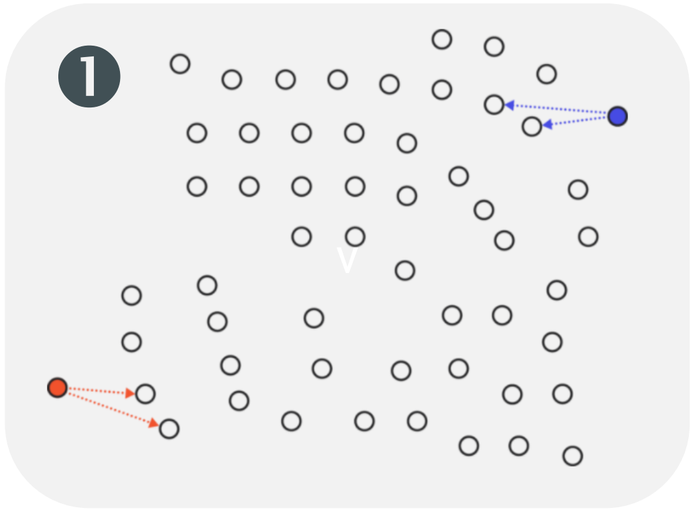
choix du bloc

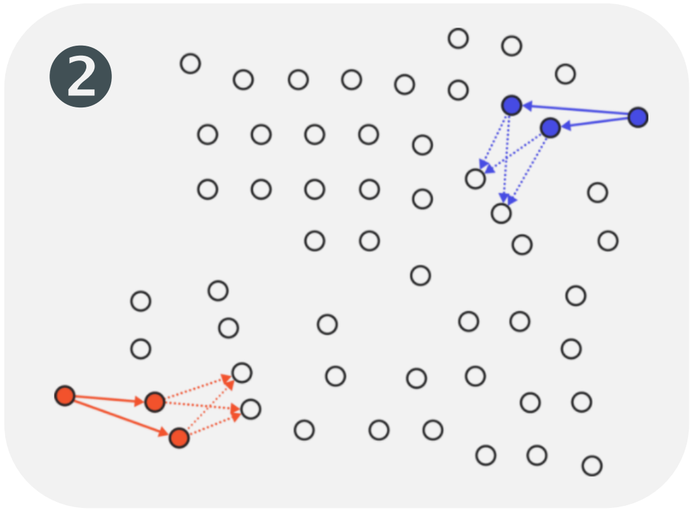
QUORUM INTERSECTION

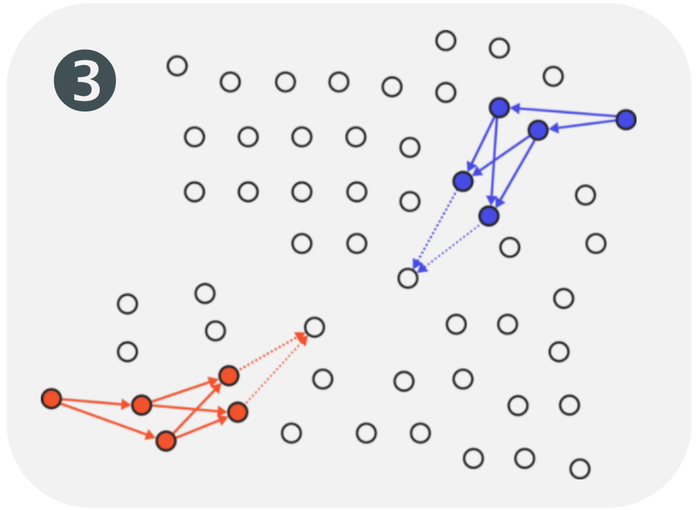


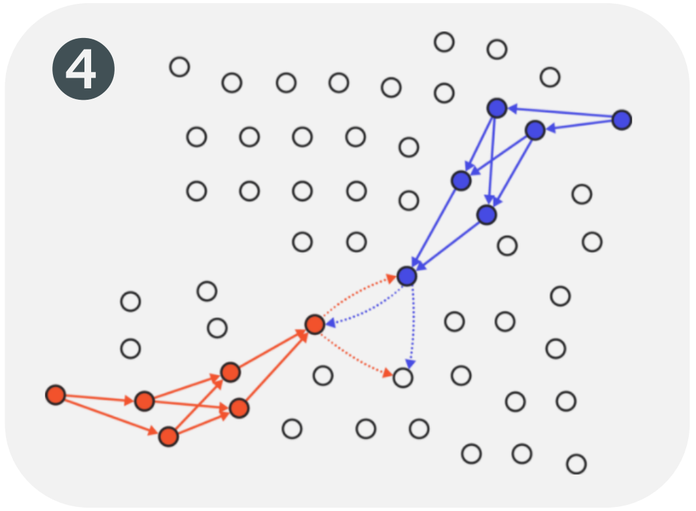
Le **bon fonctionnement** du protocole requière qu'il y ait toujours une **intersection** entre les slices

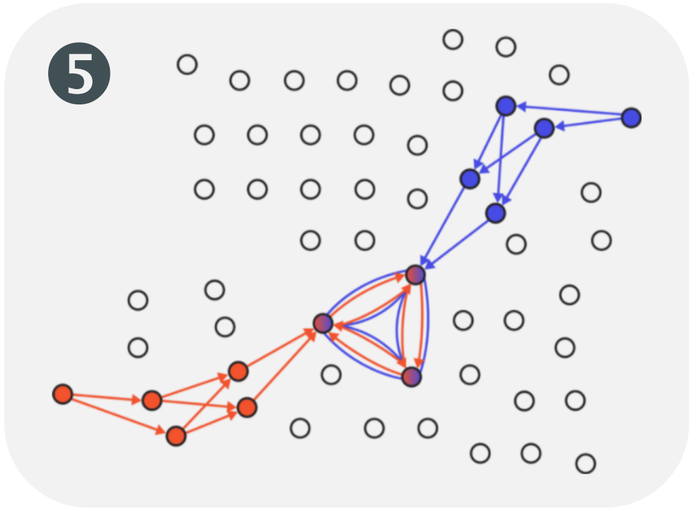
QUORUM INTERSECTION











Consensus sur
Rouge ou Bleu ?

Et la **surface d'attaque**…

https://medium.com/interstellar/understanding-the-stellar-consensus-protocol-423409aad32e

IMPLÉMENTATIONS DU QUORUM

Contrôle décentralisé : il n’y a pas d’autorité centrale
Latence faible : le consensus est rapidement atteint
Confiance flexible : chaque nœud choisit en quels autres nœuds il donne sa confiance

- **Ripple** : protocole utilisé pour les transferts entre banques
- **Stellar** : version open-source et publique de Ripple

SYNTHÈSE

	PoW Satoshi consensus Etash ProgPoW	**PoS** Percoin Casper (Ethereum)	**PoET**	**BFT** PoA Clique & Aura PBFT TenderMint	**Federated BFT** Ripple Stellar	**IOTA**
Blockchain type	Permissionless	Both	Permissioned	Permissioned	Permissioned	Both
Transaction Finality	Probabilistic	Probabilistic	Probabilistic	Immediate	Immediate	Probabilistic
Transaction Rate	Low	High	Medium	High	High	Low
Token needed	Yes	Yes	No	No	No	Yes
Scalability of peer network	High	High	High	Low	High	High
Trust model	Untrusted	Untrusted	Untrusted	Semi-trusted	Semi-trusted	Untrusted
Adversary Tolerance	51%	Algorithm dependent	Unknown	33%	33%	Unknown
Open-source	Yes	Yes	Partially	Algorithm dependent	No	No

VULNÉRABILITÉS DU CONSENSUS 1/2

sujet

- Coalition de personnes morales ou physiques exploitant les nœuds participant au consensus pour s'accorder sur le même vote (choix du même bloc)
- Attaque Sybil : Regroupement de plusieurs identités (personnes morales ou physiques) derrière un même identifiant
- Usurpation de l'identifiant d'un dispositif validateur

nœud

- Regroupement en fermes visant à disposer de « tickets » de loterie pour un même identifiant ou d'une influence plus importante dans le choix du « leader »
- Possibilité pour un nœud de soumettre plusieurs blocs à un même vote
- Possibilité pour un participant de voter sur deux blocs différents lors du consensus, c'est-à-dire de maintenir deux copies différentes du registre
- Présence d'un malware dans un code distribué, infiltrant chaque nœud validateur
- Création d'une nouvelle branche suite à une mise à jour : Hard Fork versus Soft Fork

VULNÉRABILITÉS DU CONSENSUS 2/2

réseau de noeuds

- Désynchronisation des participants ne permettant pas le vote de chacun quant au choix du nouveau bloc
- Taux de nœuds byzantins : quand le pourcentage de nœuds byzantins tolérés par le protocole est faible, la confiance est plus rapidement mise en défaut en cas d'attaque

réseau distribué

- Déni de service : ralentissement de la propagation dans le réseau
- Corruption des messages échangés sur le réseau
- Attaque Eclipse : isolement d'un nœud du réseau, puis injection des fausses transactions lui faisant croire qu'elles sont majoritaires

QUELQUES RÉFÉRENCES

1. Michael J. Fisher, Nancy A. Lynch, M. S. Paterson, « Impossibility of Distributed Consensus with One Faulty Process », Journal of the ACM, 32(2):374–382, 1985
2. Seth Gilbert, Nancy A. Lynch, « Perspectives on the CAP Theorem », ACM Transactions on Computer Systems, 20(3):239–282, 2002
3. Roger WattenHoffer, « The science of blockchain », *Published by Createspace Independent Publishing Platform*, Jan 2016, ISBN : 978-1-5227-5183-0, *https://dl.acm.org/doi/book/10.5555/300270*
4. E. Deirmentzoglou, G. Papakyriakopoulos and C. Patsakis, "A Survey on Long-Range Attacks for Proof of Stake Protocols," in IEEE Access, vol. 7, pp. 28712-28725, 2019, DOI: 10.1109/ACCESS.2019.2901858
5. Leslie Lamport, Robert Shostak et Marshall Pease, « The Byzantine Generals Problem », ACM Transactions on Programming Languages and Systems, vol. 4, no 3, juillet 1982
6. Christine Hennebert, "A first Step towards a Protection Profile for the Security Evaluation of Consensus Mechanisms", 3rd IEEE International Symposium on Future Cyber Security Technologies (FCST 2020), 14-15 december 2020, Paris, France

SECTION 5

SMART CONTRACT & TOKEN

LANGAGES

Solidity

- Langage de haut-niveau orienté objet
- Supporte l'héritage : extension vers d'autres smart contracts
- Permet de construire des librairies de code appelées par différents smart contracts

Vyper

- Basé sur le langage Python
- Code compilé court
- moins de fonctionnalités que Solidity pour être plus sûr et plus facile à auditer

Michelson

- Utilisé dans la blockchain Tezos
- Langage plus formel basé sur le langage OCaml
- Ordonnancement des données dans la pile et le tas

… et aussi

- Yul : plus bas niveau optimisé pour les EVM
- Yul+ : introduit de nouvelles fonctionnalités à Yul
- Fe : très récent (2021), inspiré de Python et Rust

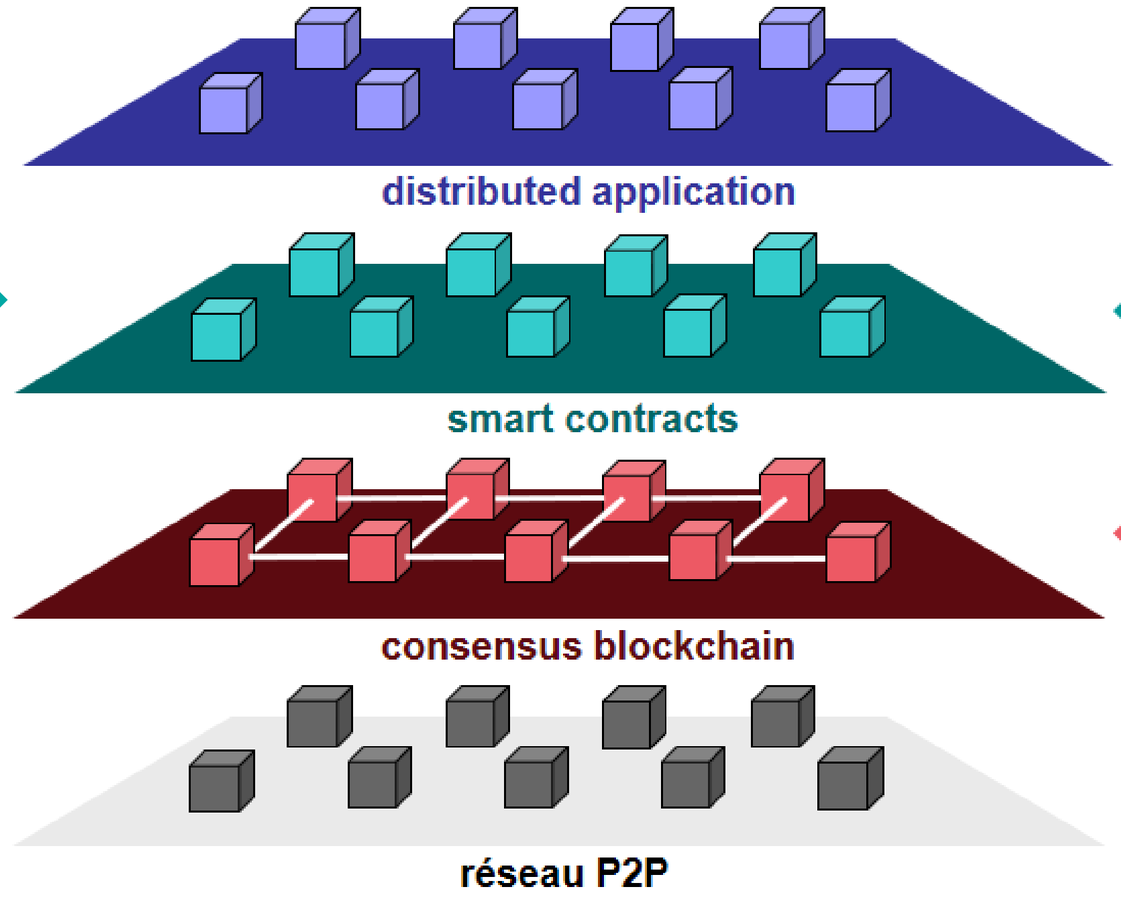
EXEMPLE DE CODE SOLIDITY

```
pragma solidity ^0.6.0;

import "interfaces/IERC20.sol";
import "math/SafeMath.sol";

contract service {

		IERC20 tokenContract;

		enum partitionStates {
				STATUS_ACTIVE,
				STATUS_FREEZED,
				STATUS_SOLD
		}

		mapping (address => uint256) public dataBase;

		constructor(address _tokenContractAddress) public {
				creator = msg.sender;
				tokenContract = IERC20(_tokenContractAddress);
		}

		function storeData( uint256 _data ) public{
				require(tokenContract.transferFrom(msg.sender, receiver, 1));
				dataBase[msg.sender] = _data;
		}
```

CURRENCY VERSUS TOKEN

Le code du smart contract est exécuté par chaque nœud validateur dans une machine virtuelle (EVM)

Le consensus porte sur la résultat de l'exécution



Token
Programmation d'un token avec un Smart Contract

Currency
Création de crypto-monnaie grâce à l'incitation

CURRENCY = CRYPTO MONNAIE

Propriétés

Fongible : une unité est équivalente à une autre
Divisible : chaque unité peut être divisée en petites unités de valeur
Acceptable : largement accepté comme moyen d'échange
Création : un mécanisme d'incentive permet de créer les unités
Quantité limitée : la quantité d'unités en circulation est limitée et/ou plafonnée
Portable : les unités peuvent être échangées contre une autre currency
Durable : les unités peuvent être utilisées sans être dégradées

Usages

Transfert d'argent
Réserve de valeurs
Unité de compte
… pour **déployer** des **smart contracts**

TOKEN

représentation numérique d'une valeur

Token Standard : **Ethereum Request for Comment (ERC)**

Token fongible : standard ERC-20, optimisé ERC-233, sécuritaire ERC-777…

- un token fongible est interchangeable avec un autre token du même type

Token non fongible : standard ERC-721

- un token non fongible présente des caractéristiques uniques, et ne peut pas être échangé avec un autre.
- les tokens non fongibles peuvent avoir des valeurs différentes les uns des autres.

Utility Token :

- peut représenter un droit d'accès
- permet d'accéder à un bien ou à un service fourni via une blockchain

Security Token : Standard **ERC-1400**

- hérite des caractéristiques des token fongible **ERC-20** (ou non fongible ERC-721)
- représente la propriété d'un bien ou d'une valeur
- est régulé quant à l'identité, la juridiction ou la valeur utilisée

TOKEN ERC20

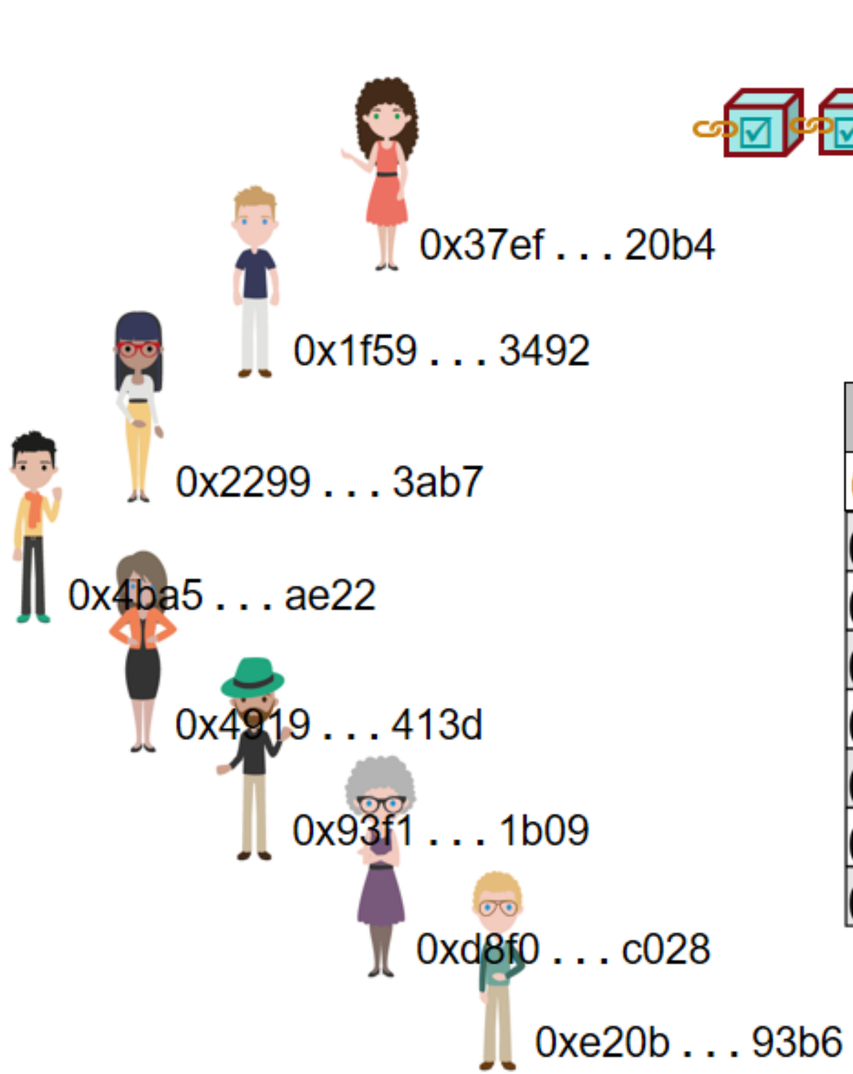
Ethereum Improvment Proposal (EIP) ≡ Implémentation des ERC en langage **Solidity**

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

- fournit le **prototype des fonctions standards** du smart contract de token ERC20 qui implémente un livre de compte accessible aux utilisateurs depuis leur adresse de compte

- les **implémentations de référence** :
  - openzeppelin : https://github.com/OpenZeppelin/openzeppelin-contracts/
  - consensys : https://github.com/ConsenSys/Tokens/

TRANSFERT DE COMPTE À COMPTE 1/2



smart contract ERC-20
« our-coin »
our-coin-address

holder account	balance
0x37ef . . . 20b4	100
0x1f59 . . . 3492	100
0x2299 . . . 3ab7	100
0x4ba5 . . . ae22	100
0x4919 . . . 413d	100
0x93f1 . . . 1b09	100
0xd8f0 . . . c028	100
0xe20b . . . 93b6	100
Total supply	**800**

TRANSFERT DE COMPTE À COMPTE 2/2

0x1f59 . . . 3492

transfert(dst = 0x93f1…1b09, amount = 10)

balanceOf(account = 0xd8f0…c028)

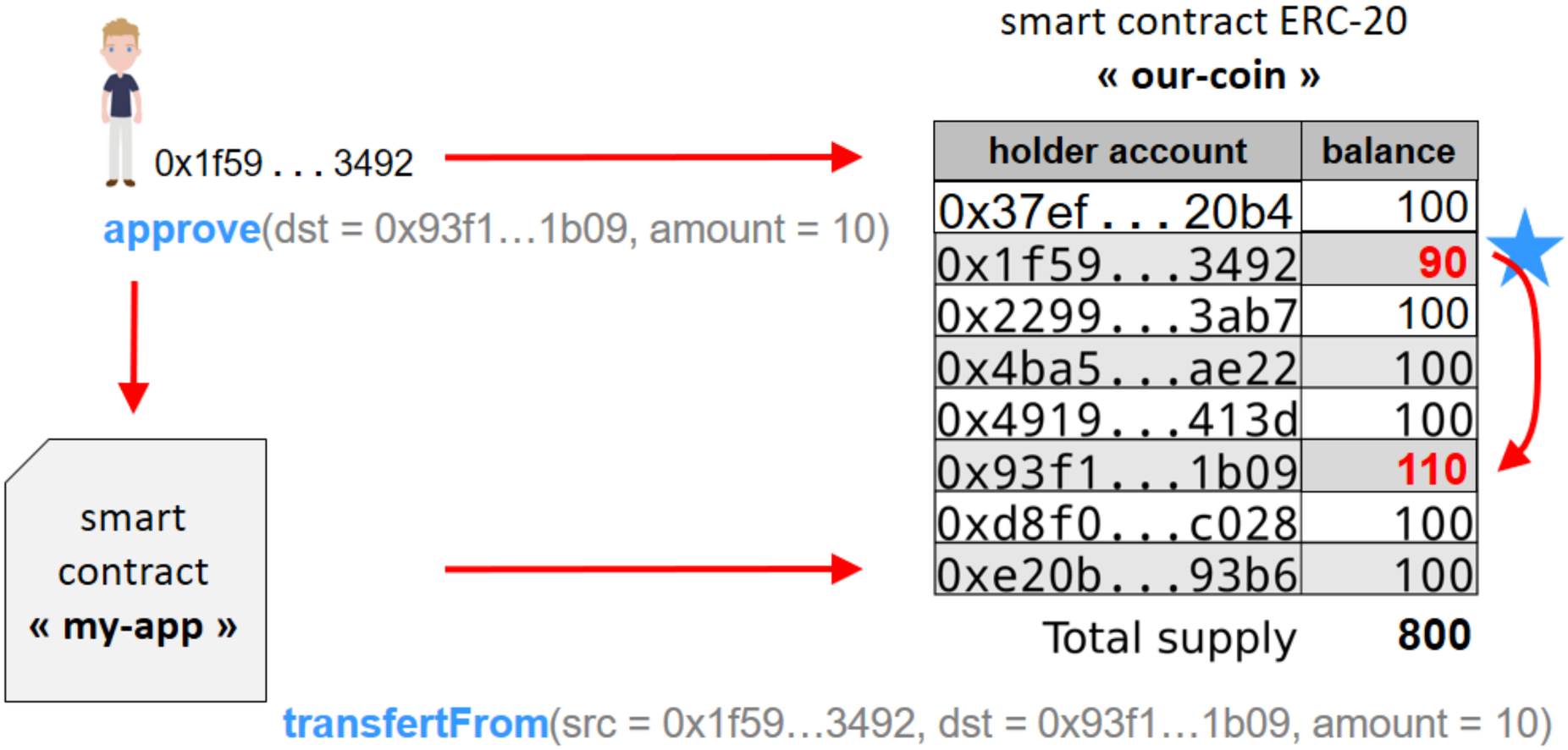
↵ **100**

smart contract ERC-20

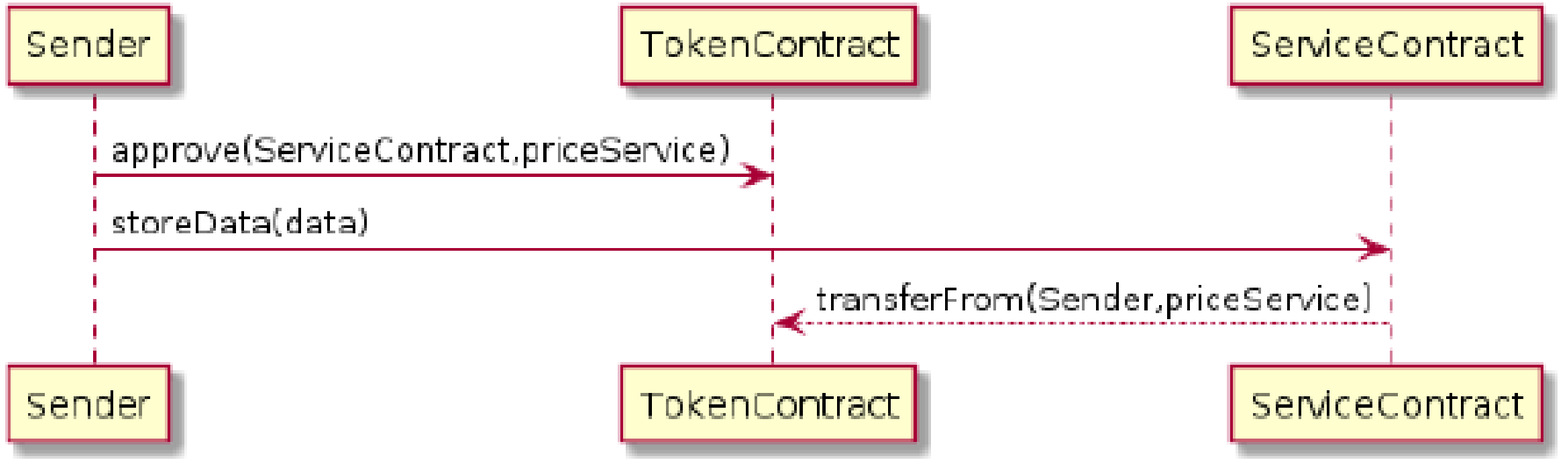
« our-coin »

holder account	balance
0x37ef . . . 20b4	100
0x1f59 . . . 3492	**90**
0x2299 . . . 3ab7	100
0x4ba5 . . . ae22	100
0x4919 . . . 413d	100
0x93f1 . . . 1b09	**110**
0xd8f0 . . . c028	**100**
0xe20b . . . 93b6	100
Total supply	**800**

TRANSFERT DE COMPTE À CONTRACT 1/2



TRANSFERT DE COMPTE À CONTRACT 2/2

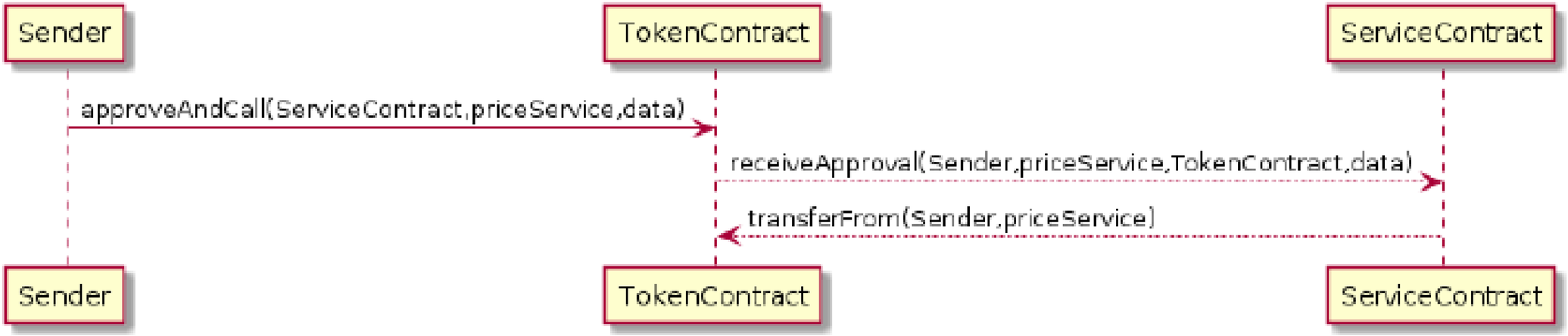


L'utilisateur envoie deux transactions, ce qui implique :

- De payer deux fois du gas
- D'ouvrir une faille quant à la possibilité d'être débiter sans que le service soit effectué, ou que le service soit effectué sans être débité
- D'où l'usage de *requirement* dans le code du smart contract de service

```
function storeData( uint256 _data ) public{
                        require(tokenContract.transferFrom(msg.sender, receiver, 1));
                        dataBase[msg.sender] = _data;
                }
```

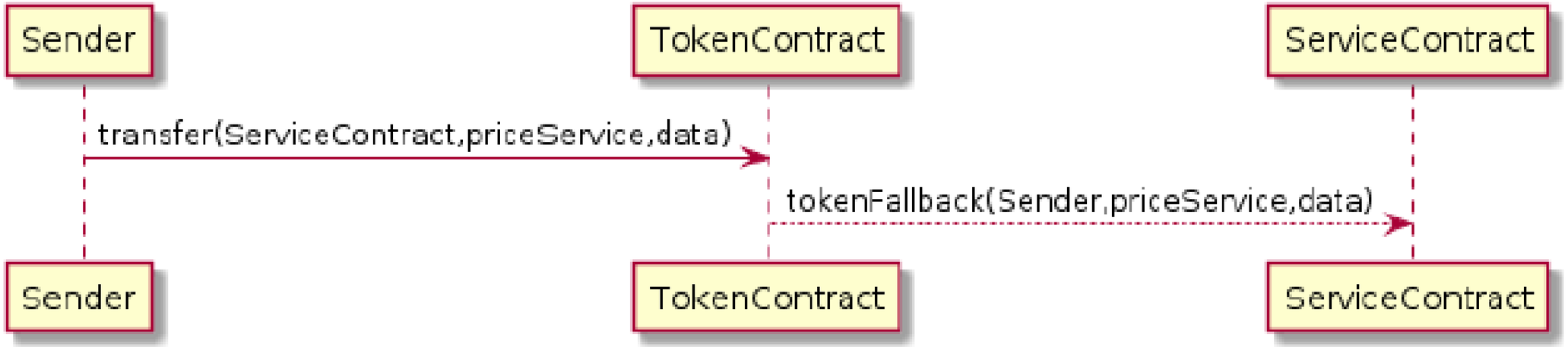
ERC20 AMÉLIORÉ



Une amélioration au smart contract standard ERC20 consiste à utiliser la fonction **approveAndCall** :

- Cette fonction n'est pas standard
- Elle est envoyée vers le smart contract de token qui appelle le service puis débite le compte
- Le code de la fonction **approveAndCall** imbrique plusieurs fonctions, ce qui le rend moins optimal que de faire appel aux deux transactions du cas précédent

TOKEN ERC223



Avec ERC223 l'utilisateur n'envoie qu'une seule transaction avec la fonction **transfer** qui est modifiée par rapport à l'implémentation de référence ERC20 :

- **transfer** fait appel à la fonction **tokenFallBack** pour lancer la transaction inter smart contract vers le smart contract de service
- Cela permet de dépenser moins de **gas** que dans les deux cas précédents
- Cela ouvre potentiellement une faille selon le soin apporté au code de la nouvelle fonction **transfer** : si le service échoue, le compte ne doit pas être débité, mais le **gas** est perdu

VULNÉRABILITÉS

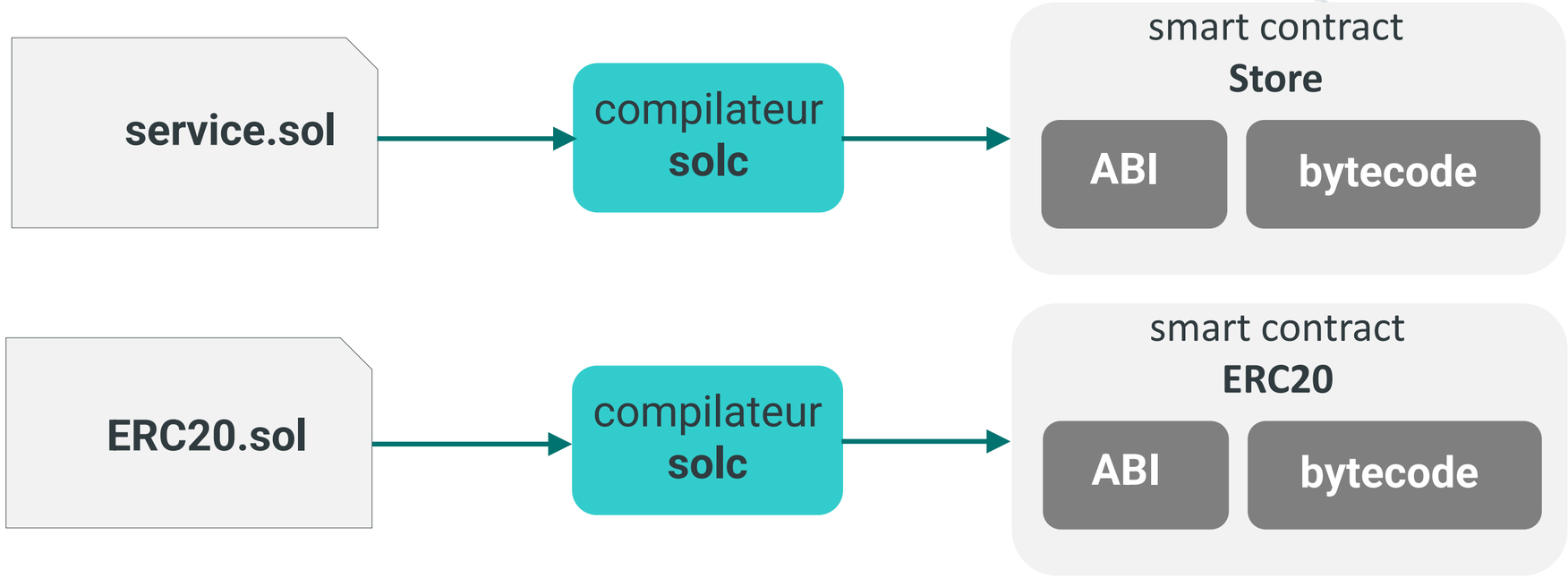
Smart Contract Weakness Classification and Test Cases

https://swcregistry.io/

Exemple

```
mapping (address => uint256) private _balances;

function _transfer(address sender, address recipient, uint256 amount) internal virtual {
        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");

        _beforeTokenTransfer(sender, recipient, amount);

        _balances[sender] = _balances[sender] - amount;
        _balances[recipient] = _balances[recipient] + amount;
        emit Transfer(sender, recipient, amount);
    }
```

COMPILATION 1/2



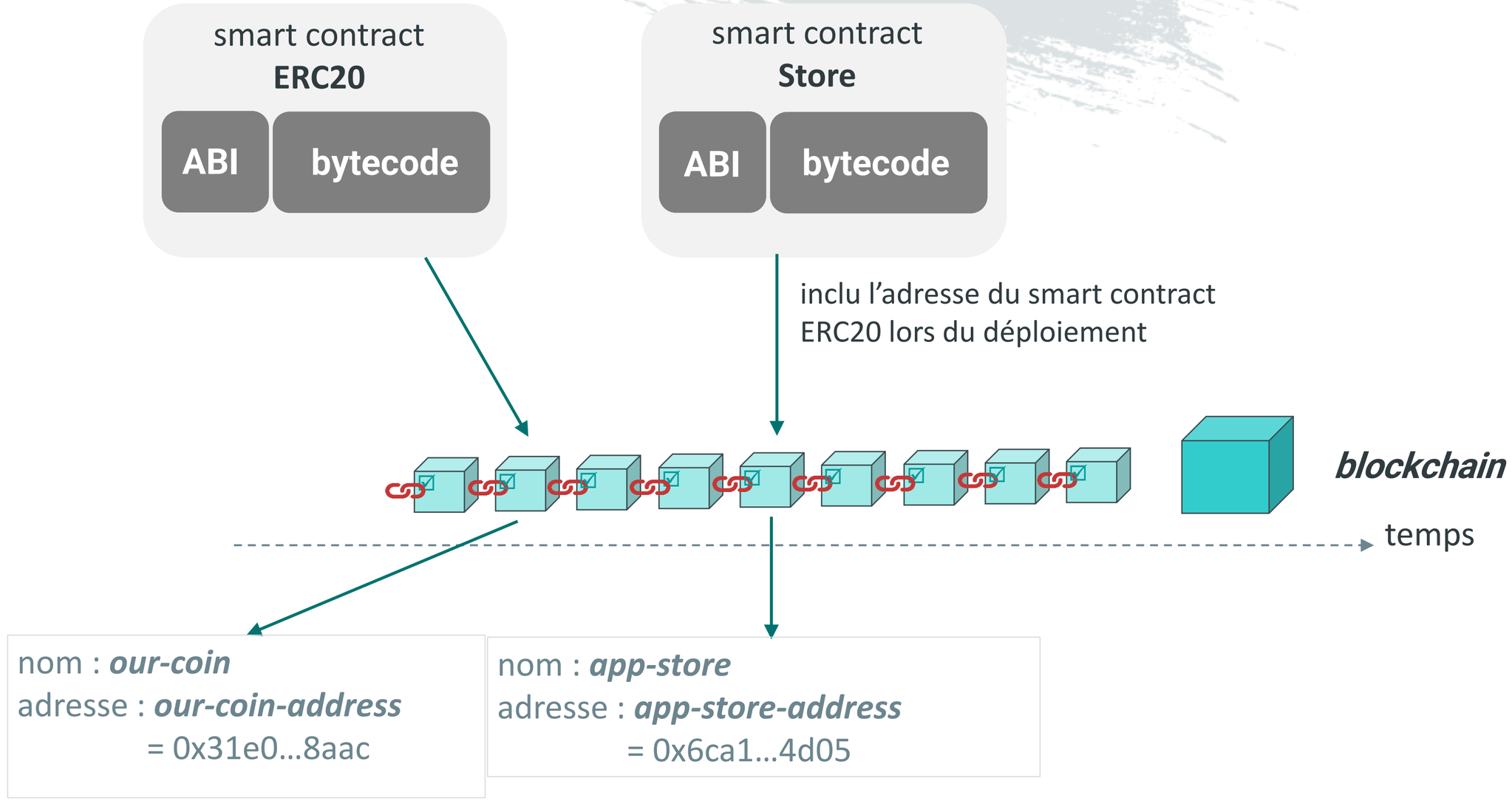
COMPILATION 2/2

```
BasicToken = pyeth.contracts.contract('BasicToken',
b'6080604052348015610010576000800fd5b506104178061002060003960000f3006080604052600436106100575760003357c0100000000
0000000000000000000000000000000000000000000900463ffffffff16806318160ddd1461005c57806370a082311461008757806
a9059cbb146100de575b600080fd5b348015610068576000800fd5b506100716101435656b6040518082815260200191505060405180910
90f35b348015610093576000800fd5b506100c860048036038101908080357ffffffffffffffffffffffffffffffffffffffff1690602001909291905050
061014d565b6040518082815260200191505060405180910390f35b348015610ea57600080fd5b506101296004803603810190808035
3ffffffffffffffffffffffffffffffffffffffff169060200190929190803590602001909291905050506101955656b6040518082151515158152602001
150506040518091039f35b60006001549050905656b60008060008373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff
ff16815260200190815260200160002054905091905056 5b60008073ffffffffffffffffffffffffffffffffffffffff168373ffffffffffffffffffffffffffffffffffffffff
fff16141515156101d257600080fd5b600803373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff1681526020019081
526020016000205482111515156102 1f57600080fd5b610270826000803373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff
ffffffffff16815260200190815260200160002054 6103b490919063ffffffff16565b6000803373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffff
ffffffffffffffffffffffff16815260200190815260200160002081905550610303826000808673ffffffffffffffffffffffffffffffffffffffff1673ffffffffffff
ffffffffffffffffffffffff16815260200190815260200160002054 6103cd90919063ffffffff16565b6000808573ffffffffffffffffffffffffffffffffffffffff
ff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002081905550827 3ffffffffffffffffffffffffffffffffffffffff163373ffffffffff
ffffffffffffffffffffffff167fddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef8460405180828152602001915
05060405180910390a3600190509291505056 5b60008282111515156103c257fe5b8183039050929150505656b6000808284019050838 1
101515156103e157fe5b8091505092915050560 0a165627a7a72305820cff2321f765033dd90f987a9d06d2bcd0cf0f6861d5c440616ec78
15c440aeb10029',
[{"constant":true,"inputs":[],"name":"totalSupply","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function
"},{"constant":true,"inputs":[{"name":"_owner","type":"address"}],"name":"balanceOf","outputs":[{"name":"balance","type":"uint256"}],"payable":fa
lse,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"_to","type":"address"},{"name":"_value","type":"uint256"}],"nam
e":"transfer","outputs":[{"name":"","type":"bool"}],"payable":false,"stateMutability":"nonpayable","type":"function"},{"anonymous":false,"inputs":[{
"indexed":true,"name":"from","type":"address"},{"indexed":true,"name":"to","type":"address"},{"indexed":false,"name":"value","type":"uint256"}],"n
ame":"Transfer","type":"event"}])
```

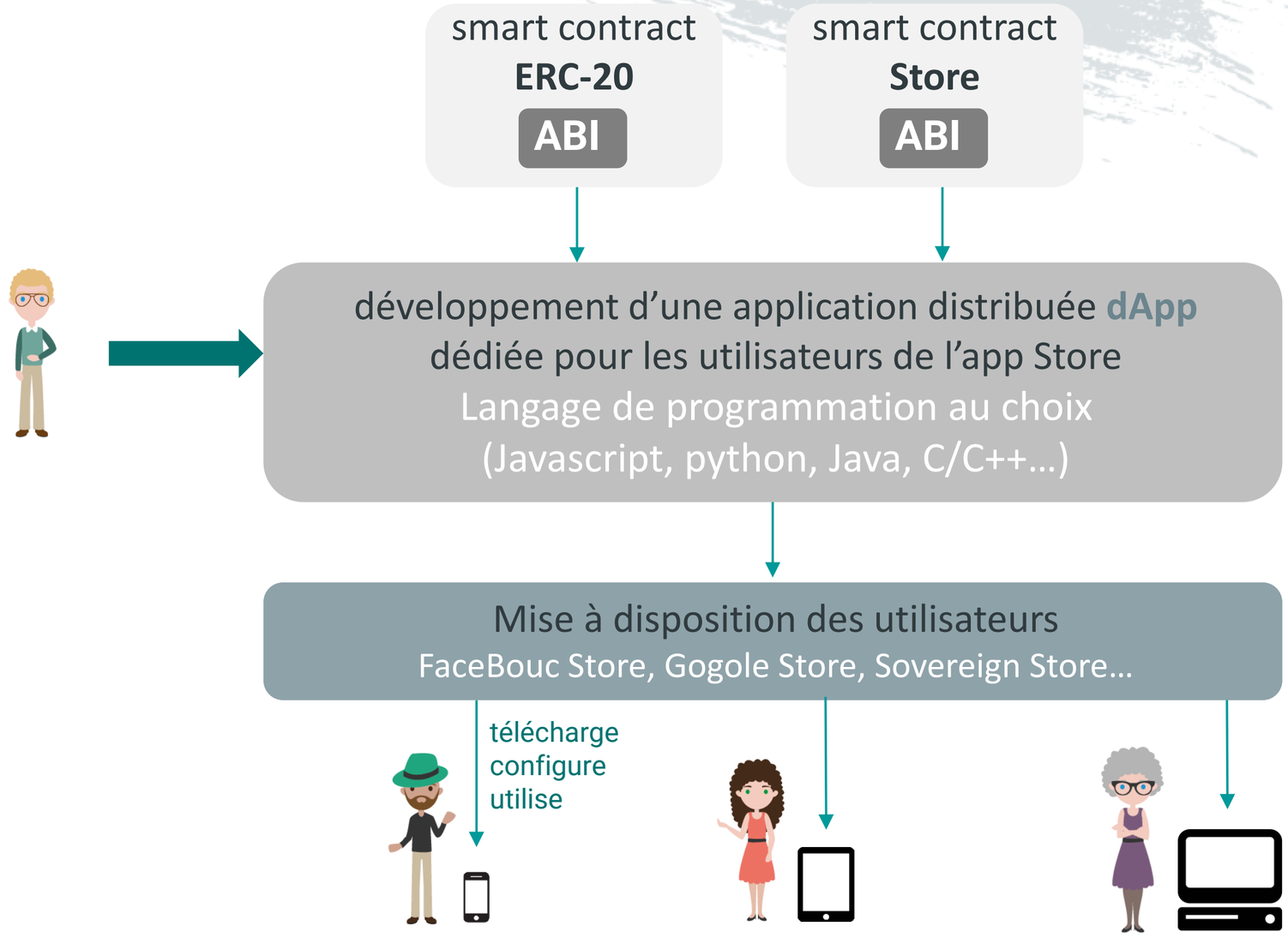
BasicToken.sol

```
pragma solidity ^0.4.18;

import "./ERC20Basic.sol";
import "../../math/SafeMath.sol";

contract BasicToken is ERC20Basic {

using SafeMath for uint256;

mapping(address => uint256) balances;
uint256 totalSupply_;

function totalSupply() public view returns (uint256) {
                return totalSupply_;
                }

function transfer(address _to, uint256 _value) public returns (bool) {
                require(_to != address(0));    require(_value <= balances[msg.sender]);
                balances[msg.sender] = balances[msg.sender].sub(_value);
                balances[_to] = balances[_to].add(_value);
                Transfer(msg.sender, _to, _value);
                return true;
                }

function balanceOf(address _owner) public view returns (uint256 balance) {
                return balances[_owner];
                }
}
```

DÉPLOIEMENT 1/2



DÉPLOIEMENT 2/2



QUELQUES RÉFÉRENCES

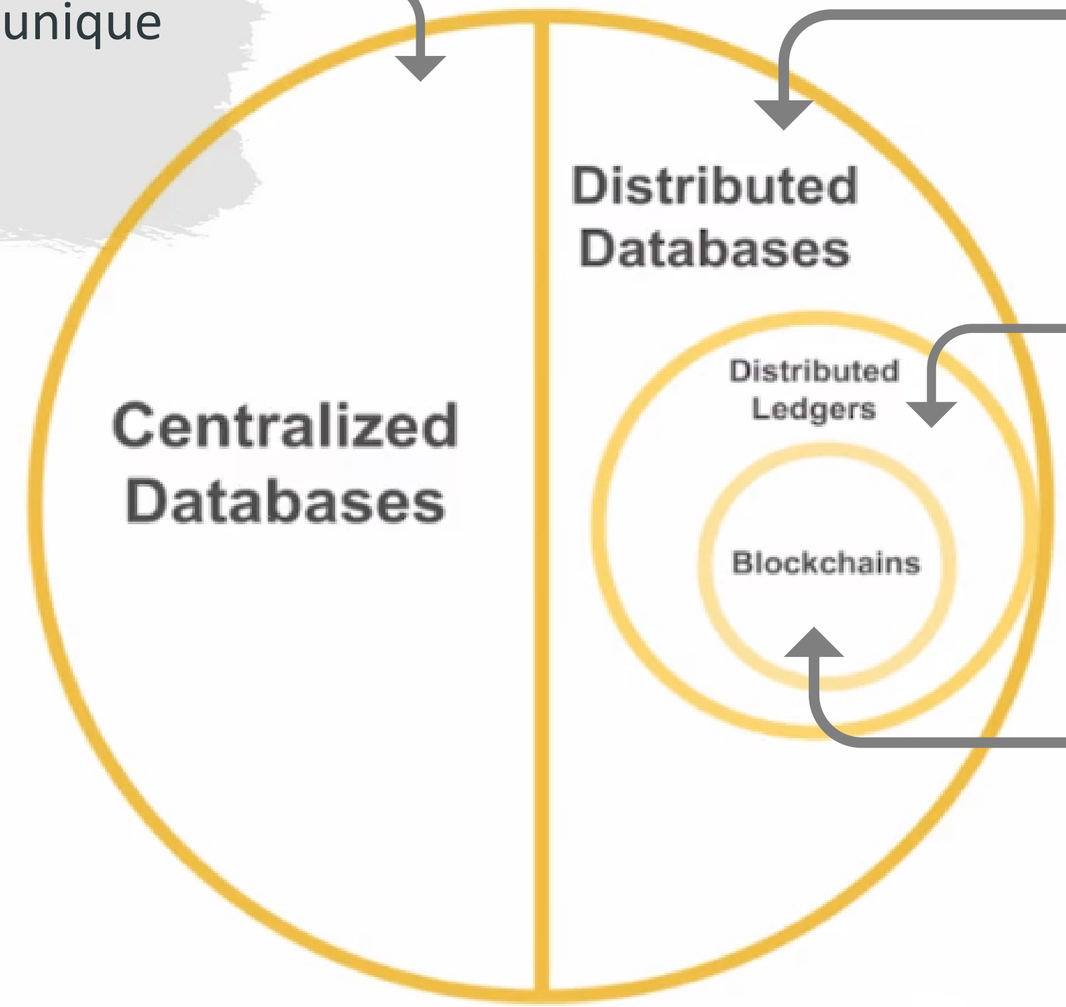
1. librairie openzeppelin, https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/token
2. librairie consensys, https://github.com/ConsenSys/Tokens/
3. Solidity readthedocs, https://docs.soliditylang.org/en/latest/
4. Solidity by example, https://docs.soliditylang.org/en/latest/solidity-by-example.html
5. Solidity github, https://github.com/ethereum/solidity/
6. Vyper readthedocs, https://vyper.readthedocs.io/en/stable/
7. Vyper by example, https://vyper.readthedocs.io/en/latest/vyper-by-example.html
8. Vyper github, https://github.com/vyperlang/vyper

SECTION 6

HYPERLEDGER & DLT

BLOCKCHAIN VERSUS DATABASE

Un tiers de confiance unique
Pas de consensus



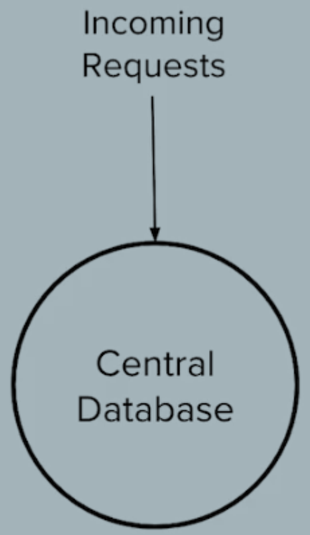
Un tiers de confiance unique
Besoin de cohérence entre les répliques de la database

Tiers de confiance multiples
Besoin d'un consensus

Acteurs multiples
Besoin d'un consensus
Structure de blocs

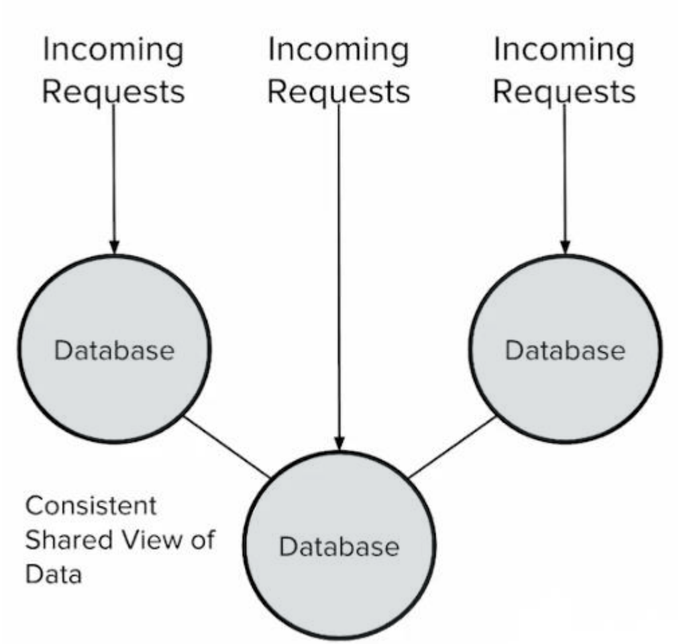
BLOCKCHAIN VERSUS DATABASE

centralized database



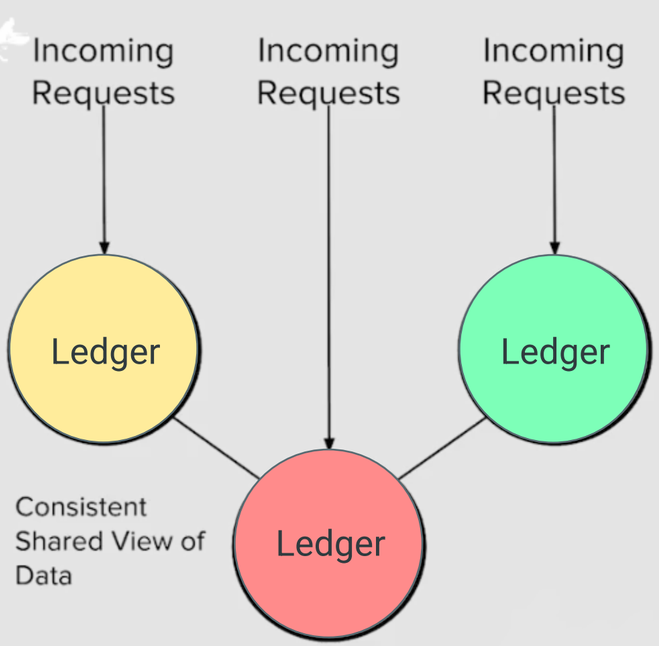
point de défaillance unique
simple à maintenir
débit de requête élevé

distributed database



confiance mutuelle
complexe à maintenir
tolérance aux fautes

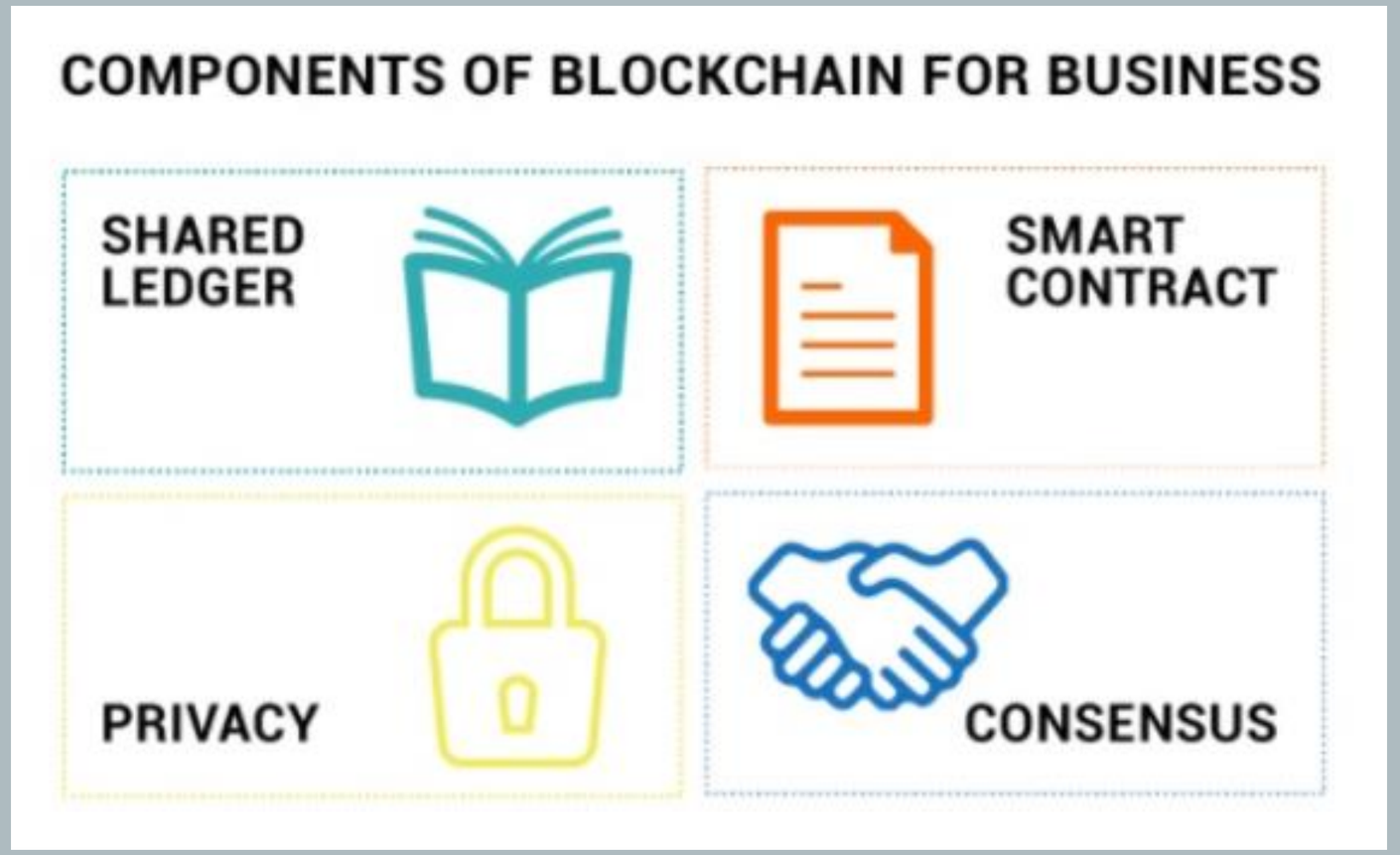
distributed ledger



pas de confiance mutuelle
tolérance aux fautes
protocole de consensus

PROPRIÉTÉS DE HYPERLEDGER

La notion de **Partage** est à regarder de près : **entre quels acteurs ?**

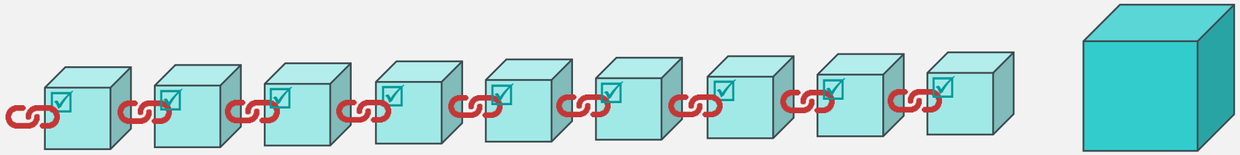


La notion de Privacy peut impliquer la **Confidentialité**

PRÉSENCE D'AGENTS

Dispositif utilisateur

Agent connecteur
gère les droits d'accès au ledger en écriture ET en consultation

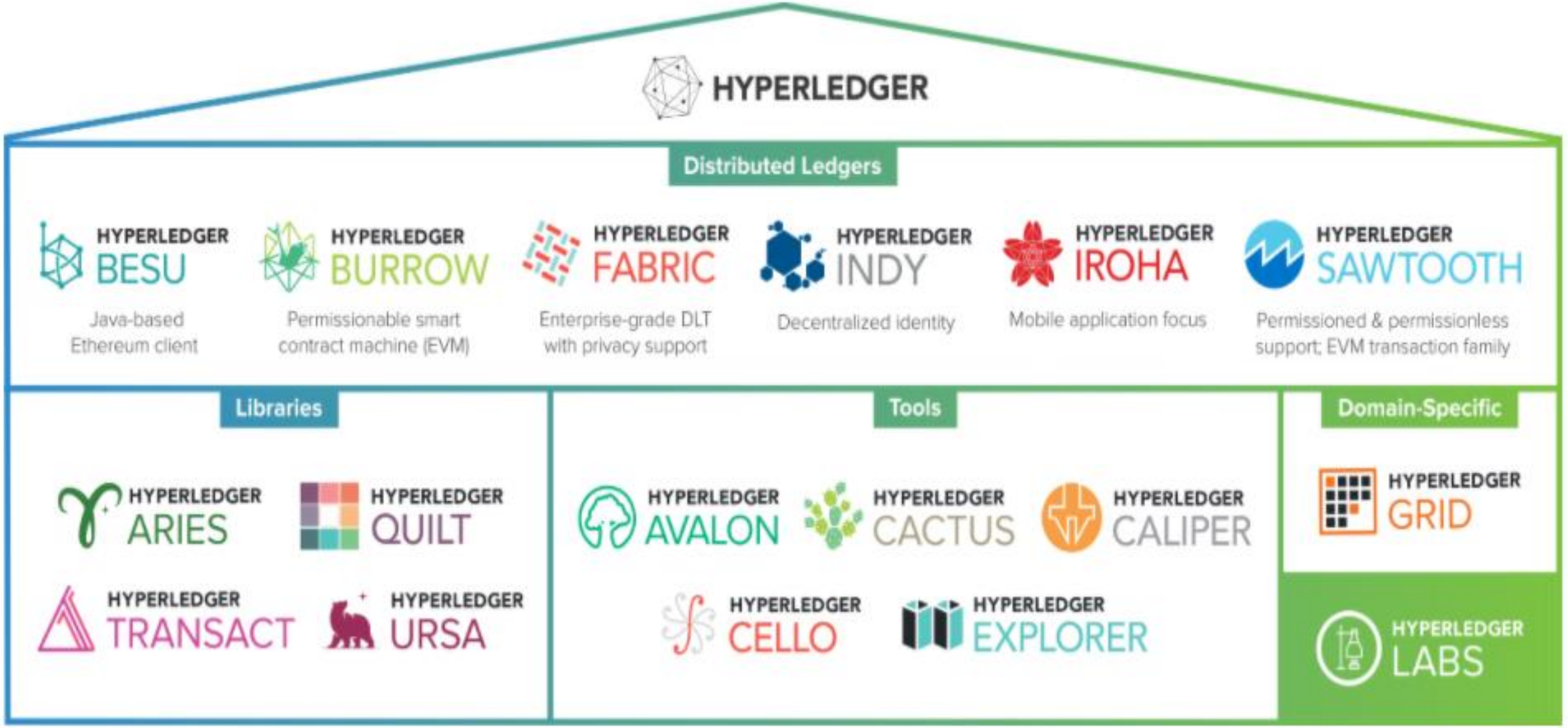


Nœud validateur
détient une copie complète du LEDGER

UN REGROUPEMENT DE PROJETS

- Hyperledger est un regroupement de projets open source
- Certains sont des systèmes de type blockchain, comme Fabric, Sawtooth et Iroha
- Les blockchain construites avec Hyperledger sont « permissioned », ce qui signifie que les acteurs qui rejoignent le réseau doivent être authentifiés et autorisés
- Le principal objectif d'Hyperledger est de fournir un ensemble d'outils pour les entreprises pour répondre à différents cas d'usage

ÉCOSYSTÈME AUTOUR DE HYPERLEDGER



SECTION 7

LES USAGES

A QUOI SERT UNE BLOCKCHAIN ?

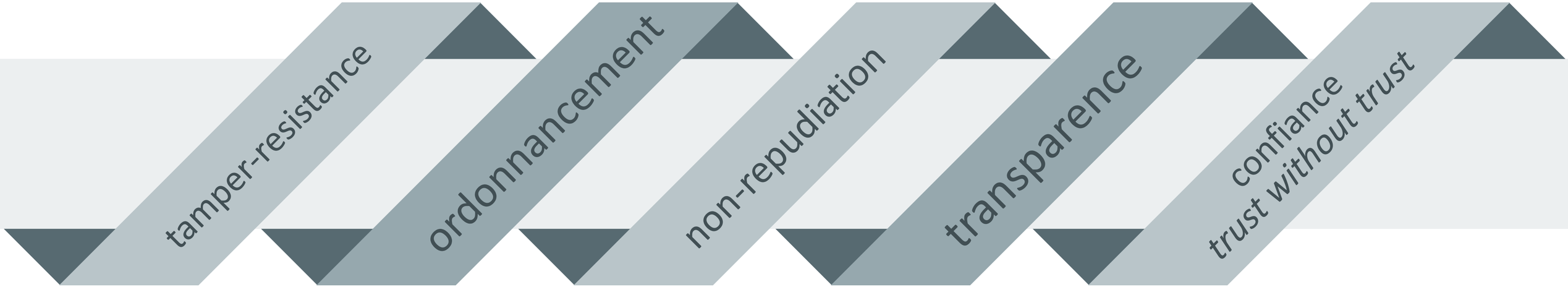
La blockchain est un outil :

- De **traçabilité**
- D'**ordonnancement**
- D'**horodatage**
- De garantie d'**intégrité**
- D'**authentification**
- D'**attestation**
- De **transactions** multi-parties
- D'enregistrement de **preuves**

La blockchain n'est pas un outil :

- De performance (débit et bande passante)
- De confidentialité
- De stockage de données volumineuses

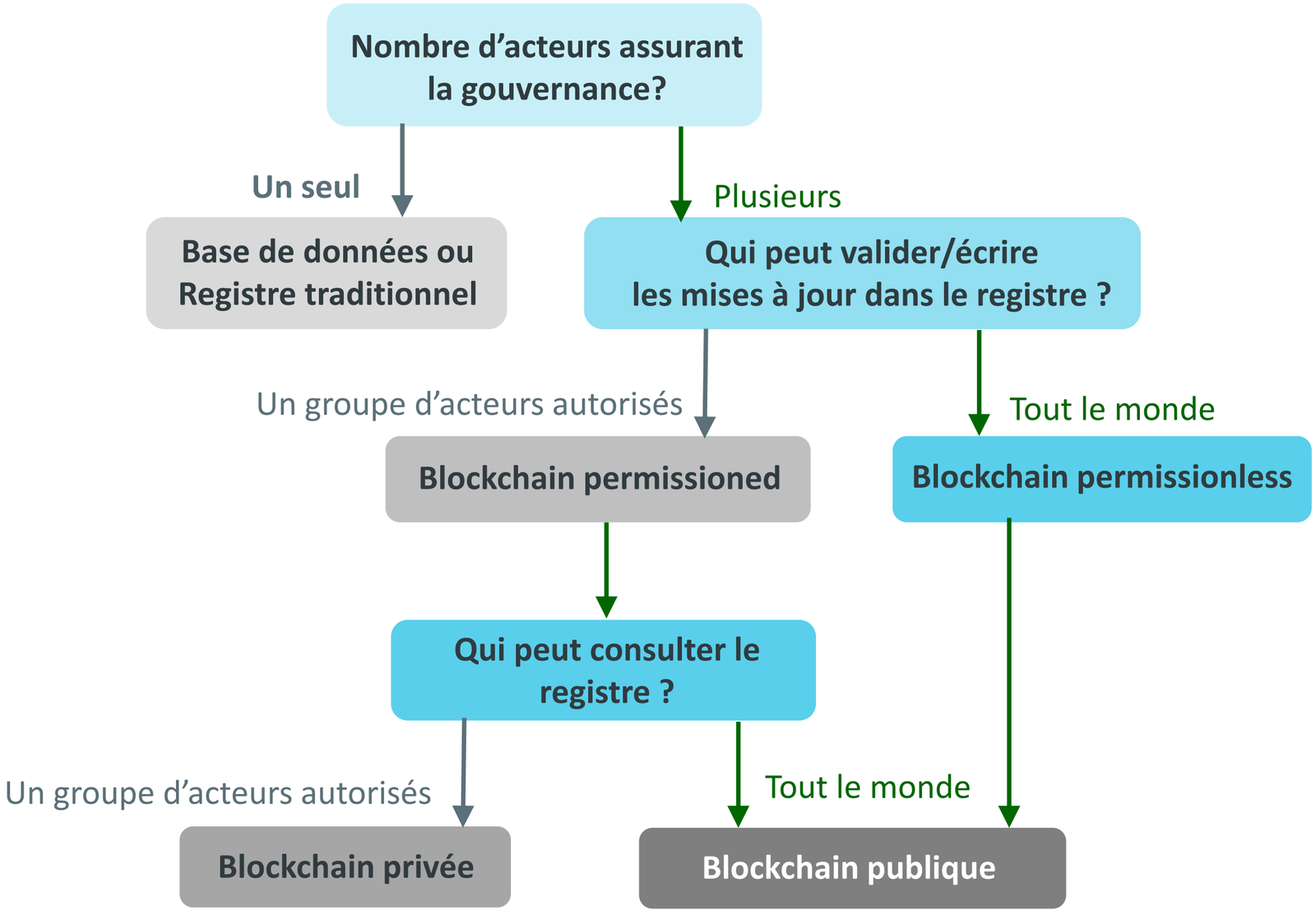
PROPRIÉTÉS DE LA BLOCKCHAIN



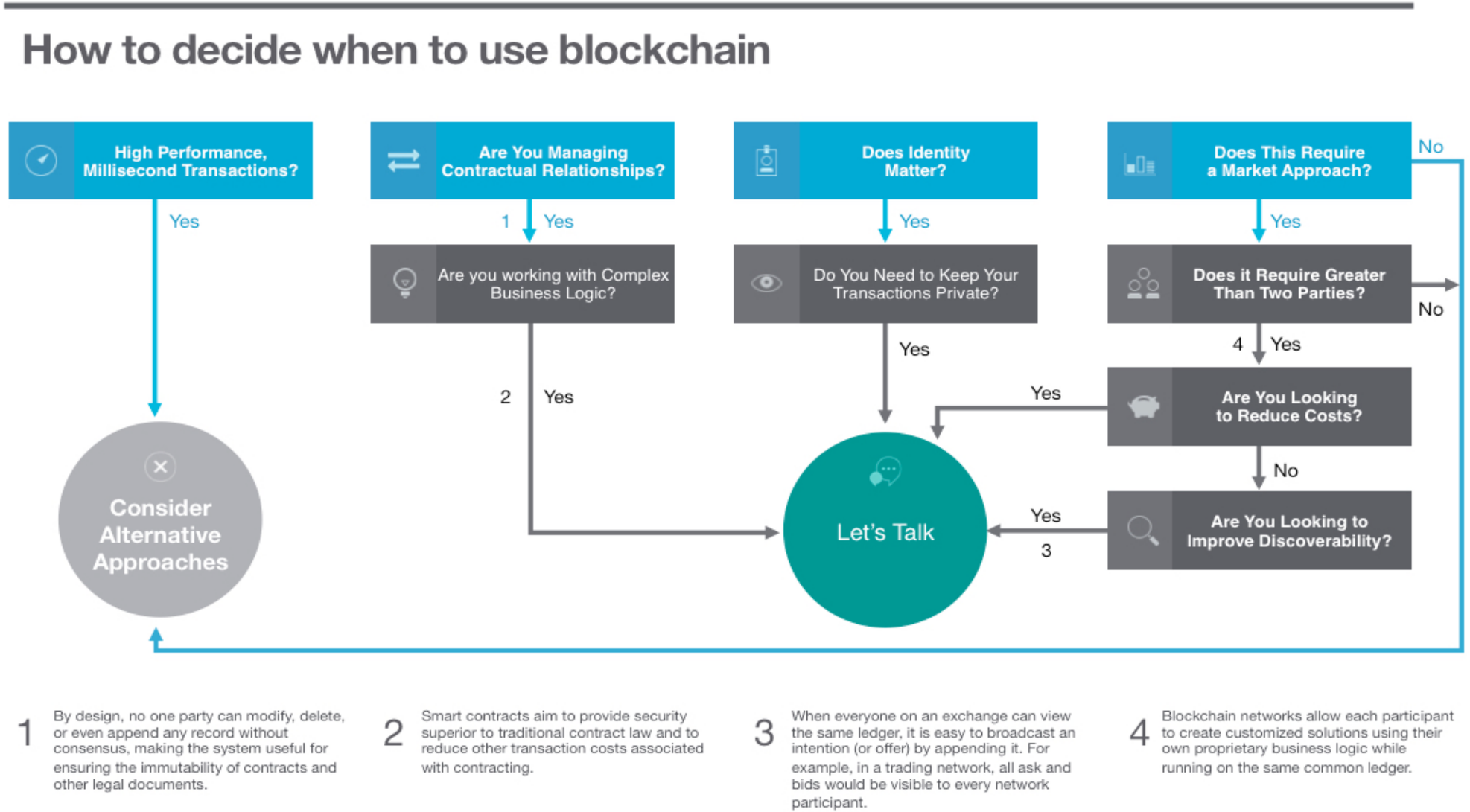
La blockchain permet :

- la **preuve de possession**
- L'**usage d'une valeur**

A-T-ON BESOIN D'UNE BLOCKCHAIN ?

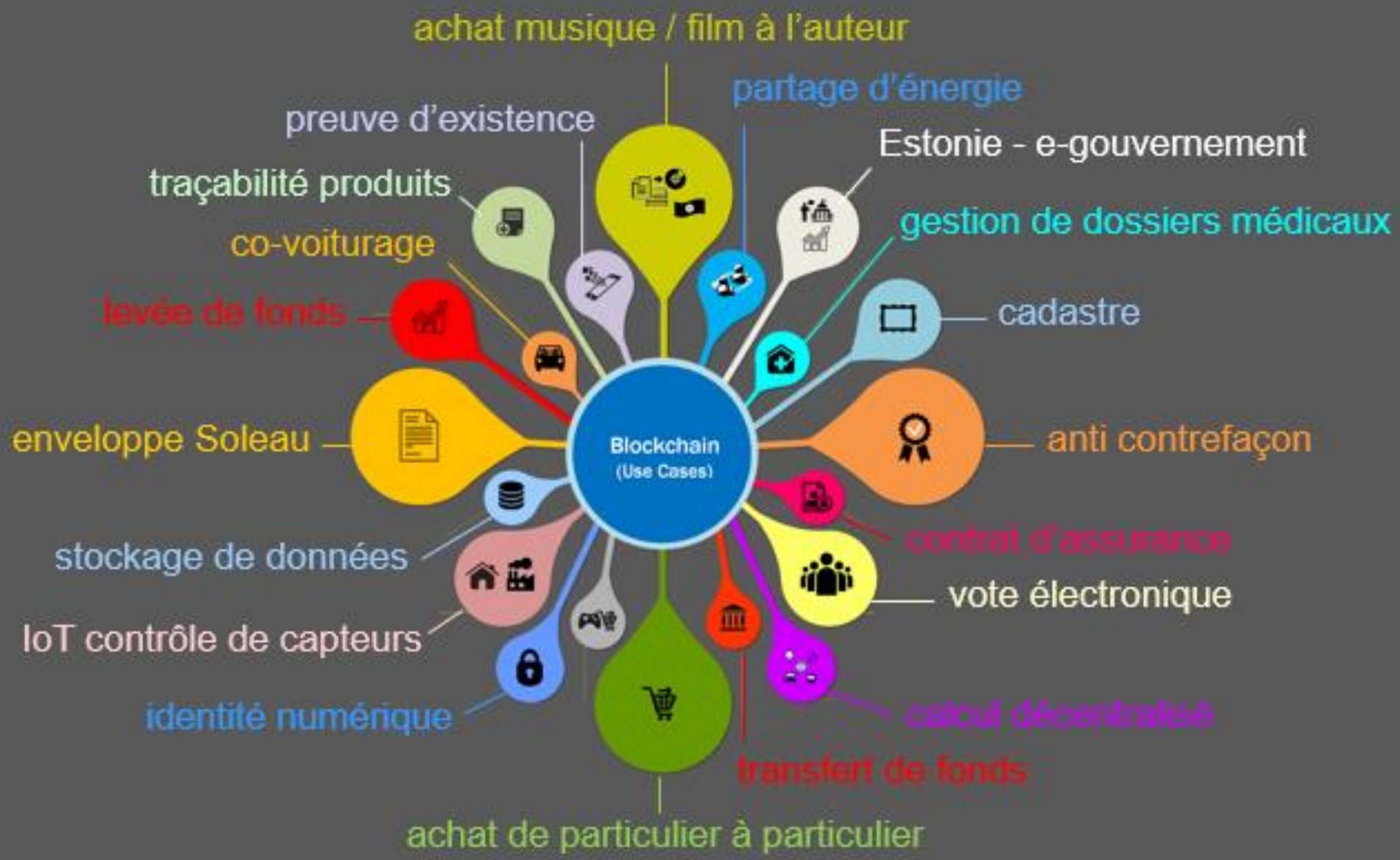


GRAPHE DE DÉCISION



source IBM

QUELQUES CAS D'USAGE



SECTION 8

EVALUATION

Sujet : Réfléchir au cas d'usage de l'authentification des diplômes

Indications :
- Le diplôme est une donnée personnelle, donc confidentielle
- Dans l'exercice, il s'agit d'un diplôme numérique et non numérisé

Questions en support à la réflexion :
- Réfléchir au rôle de la blockchain dans ce cas d'usage
- Quels sont les principaux acteurs ?
- Proposer un schéma d'architecture de système pouvant être mis en œuvre pour répondre à ce cas d'usage ? Quels éléments (dispositifs) sont requis et quelles sont les liens/communications entre eux ?
- Quel type de blockchain, permissioned versus permissionless, public versus privé, est préconisé ? Pourquoi ?
- Quel smart contract pourrait-on utiliser ? En quoi pourrait consister la dApp (decentralized application) ?
- Proposer un scénario d'usage ?

Rédiger un document avec une ébauche de solution technique susceptible de retenir l'attention d'un client.

christine.hennebert@cea.fr