

Protocoles Économiques et *Smart-contracts*

18 octobre 2020

Plan du cours

1. Aperçu général
2. Protocoles économiques
3. Cryptographie et Privacy
4. Algorithmes de consensus
5. Systèmes distribués en milieu adversarial

Code commun à tous les participants de la chaîne

Ensemble des règles de la chaîne

- Algorithme de consensus
- Représentation des données (+ API)
- Validité et applications des opérations/blocs
- *Smart-contracts*
- ...

Représentation des données

Le protocole doit définir le format de l'état de la chaîne et des structures de données (e.g. opérations, blocs, ..)

Exemple : un *état* de la chaîne

```
/accounts:
```

```
  {id => balance}
```

```
/constants:
```

```
  rewards-per-block
```

```
  min-time-between-blocks
```

```
  max-op-per-block
```

```
  ...
```

Représentation des données

Le protocole doit définir le format de l'état de la chaîne et des structures de données (e.g. opérations, blocs, ..)

Exemple : un *état* de la chaîne

```
/accounts:
```

```
  {id => balance}
```

```
/constants:
```

```
  rewards-per-block
```

```
  min-time-between-blocks
```

```
  max-op-per-block
```

```
  ...
```

Pourquoi ?

Cohérence des données

La validité d'un bloc et des opérations dépendent de la cohérence des données.

Comment s'en assurer ?

\mathcal{B}

Pred. : # 380f004f Level : 345
Opérations : Alice $\xrightarrow{10\text{€}}$ Bob Bob $\xrightarrow{5\text{€}}$ Bob Bob $\xrightarrow{2\text{€}}$ Charles

\mathcal{S}

Alice	1230€
Bob	5432€
Charles	543€

$\mathcal{B}(\mathcal{S})$

Alice	1220€
Bob	5440€
Charles	545€

Cohérence des données

La validité d'un bloc et des opérations dépendent de la cohérence des données.

- On *hash* l'état résultant ¹
- Ce hash est inclus dans le bloc

\mathcal{B}

Pred. :	# 380f004f
Level :	345
State _{\mathcal{H}} :	$\mathcal{H}(\mathcal{B}(\mathcal{S}))$
Opérations : ...	

$\mathcal{B}(\mathcal{S})$

Alice	1220€
Bob	5440€
Charles	545€

1. n.b. racine d'un arbre de Merkle

Validité des opérations

$$X \xrightarrow{\langle \text{montant} \rangle} Y + \text{signature}$$

Propriétés que le protocole doit vérifier :

- X existe dans l'état
- X possède au moins $\langle \text{montant} \rangle$ dans son compte
- La signature est bien celle de X

Validité des opérations

$$X \xrightarrow{\langle \text{montant} \rangle} Y + \text{signature}$$

Propriétés que le protocole doit vérifier :

- X existe dans l'état
- X possède au moins $\langle \text{montant} \rangle$ dans son compte
- La signature est bien celle de X

Que se passe-t-il si Y n'existe pas ?

Validité des opérations

$$X \xrightarrow{\langle \text{montant} \rangle} Y + \text{signature}$$

Propriétés que le protocole doit vérifier :

- X existe dans l'état
- X possède au moins $\langle \text{montant} \rangle$ dans son compte
- La signature est bien celle de X

Que se passe-t-il si Y n'existe pas ?

Que se passe-t-il si Y ré-injecte à nouveau cette opération ?

On associe à chaque opération un *bloc d'ancrage*

- On dispose des opérations considérées comme « trop vieilles » ;
- Nécessite de stocker les (hash des) blocs récents dans l'état de la chaîne.

Mécanismes d'*anti-replay* (2)

On associe un compteur à chaque compte

- À chaque débit, le compteur du compte est incrémenté
- Les opérations doivent déclarer pour quel compteur elles supposent être valides
- Pour être valide, l'opération doit avoir le même compteur que celui du compte
- Un entier par compte à stocker dans l'état

Alice(25) $\xrightarrow{10\text{€}}$ Bob

Alice(24) $\xrightarrow{10\text{€}}$ Bob

S

Alice	#25	1230€
Bob	#32	5432€
Charles	#10	543€

Validité des blocs

\mathcal{B}

Pred. : # 380f004f
Level : 345
State $_{\mathcal{H}}$: # 9932c2ad
Opérations : op ₁ ...op _n

Propriétés à vérifier :

- Prédécesseur est cohérent
- Le niveau est cohérent
- Les opérations sont valides
- L'état résultant est cohérent

⇒ On doit stocker le bloc prédécesseur dans l'état

En général, on distribue une récompense au créateur du bloc

Comment peut-on implémenter cela ?

En général, on distribue une récompense au créateur du bloc

Comment peut-on implémenter cela ?

1. On ajoute aux blocs l'identifiant du créateur
2. La fonction d'application de bloc crédite la récompense
3. L'état doit également être mis-à-jour

Rappel : Création monétaire \Rightarrow Inflation

Protocole Bitcoin

Rappel : le protocole Bitcoin veut un bloc toutes les 10min

Comment faire ?

Rappel : le protocole Bitcoin veut un bloc toutes les 10min

Comment faire ?

1. On ajoute à l'état la difficulté actuelle
2. On ajoute à l'état une moyenne des temps
3. On ajoute aux blocs un temps de création et un *nonce*
4. La fonction d'application doit :
 - Calculer le delta de temps entre les deux blocs
 - Vérifier que ce delta est cohérent
 - Vérifier le hash vis-à-vis de la difficulté et du temps
 - Mettre à jour la moyenne et, la difficulté au besoin

Déterminer les constantes du protocole

Vitesse de création des blocs

- Création d'un bloc toutes les heures \Rightarrow Effet ?

Déterminer les constantes du protocole

Vitesse de création des blocs

- Création d'un bloc toutes les heures \Rightarrow Effet ?
 \Rightarrow Lenteur d'inclusion des transactions
- Création d'un bloc toutes les secondes \Rightarrow Effet ?

Déterminer les constantes du protocole

Vitesse de création des blocs

- Création d'un bloc toutes les heures \Rightarrow Effet ?
 \Rightarrow Lenteur d'inclusion des transactions
- Création d'un bloc toutes les secondes \Rightarrow Effet ?
 \Rightarrow Latence réseau, synchronisation difficile, ...

Déterminer les constantes du protocole (2)

Taille d'une opération

2 adresses (2×32 octets) + signature (32 octets)

compteur (8 octets) + montant (8 octets)

= 112 octets

Nombre d'opérations autorisées dans un bloc

- 10 opérations autorisées ($\sim 1\text{Ko}$) \Rightarrow Effet ?

Déterminer les constantes du protocole (2)

Taille d'une opération

2 adresses (2×32 octets) + signature (32 octets)

compteur (8 octets) + montant (8 octets)

= 112 octets

Nombre d'opérations autorisées dans un bloc

- 10 opérations autorisées ($\sim 1\text{Ko}$) \Rightarrow Effet ?
- 10.000 opérations autorisées ($\sim 1,1\text{Mo}$) \Rightarrow Effet ?

Déterminer les constantes du protocole (2.2)

Il faut prendre en compte :

- Le temps de validation d'une opération et d'un bloc
- La propagation et l'agrégation dans le réseau
- Les effets sur le consensus :
 - Exemple : les blocs vides sont plus rapides à créer que des blocs remplis
- Le *throughput* = $\frac{\text{Nb. opérations par bloc}}{\text{Vitesse de création d'un bloc}}$
(e.g. Visa = 1.736 op/s)
- ...

Déterminer les constantes du protocole (3)

Conclusion – Déterminer les constantes est très difficile :

- Incitation économique vs. limitations techniques
- Variables économiques du “vrai” monde
- Considérer les états possibles du réseau (surchargé, attaqué, . . .)
- Choix technologiques, architectures matérielles, . . .

Implémentation des règles

- Tout le monde choisit d'exécuter le même code ;
- Le code implémente la logique de la chaîne ;
- Le code est loi.

Criticité du protocole

- Bug de création d'argent \Rightarrow dévalorisation monétaire ;
- Une exception non-rattrapée \Rightarrow arrêt de la chaîne, ...

Smart-contracts

Un *Smart-Contract* est un compte régi par un script possédant :

- Une adresse (identifiant) et un montant ;
- Du **code** à exécuter ;
- Un **espace de stockage**.

- Le code et le stockage sont également visibles de tous.
- Le protocole définit :
 - Le langage du smart-contract : syntaxe & sémantique ;
 - Son modèle d'exécution.

Modèle d'exécution (1/2)

Création

- Tout le monde peut créer un smart-contract¹ ;
- Le créateur (manager) doit fournir le code du smart-contract à créer et fournir un stockage initial.

1. En fonction des contraintes établies par le protocole

Exécution

- Un smart-contract est exécuté via une transaction ;
- Seul le stockage varie entre chaque appel ;
- Selon son code, un smart-contract peut :
 - Émettre des transactions ;
 - Appeler un autre smart-contract ;
 - Créer de nouveaux smart-contracts.
- Son exécution est généralement atomique.

Exemple

Code

```
1  /* add the parameter
2     to the storage
3     for each call */
4  int storage;
5  code(int parameter){
6     storage += parameter ;
7     return ;
8  }
```

Opération de création :

- Un *manager*
- Un montant initial
- Un stockage initial
- Le code à exécuter

Exemple :

$op_crea = \langle \text{manager} : \text{Alice}, \text{balance} : 0, \text{storage} : 0, \text{code} : P \rangle$

\Rightarrow Propagation de l'opération sur la chaîne (après signature)

Exemple – Création

\mathcal{B}

Pred. :	# 380f424f
Level :	4201
State \mathcal{H} :	# 2952a3de
Opérations : op_crea	

$\mathcal{B}(S)$

Alice	1220€	
Bob	5440€	
Charles	545€	
Alice $_{SC}$	0€	Storage : 0

- Tout le monde va stocker le nouveau smart-contract
- Il sera présent dans la chaîne pour toujours

Exemple – Appels

$$op_1 = \text{Alice} \xrightarrow{\langle 0\text{€}, param:10 \rangle} \text{Alice}_{SC}$$

$$op_2 = \text{Bob} \xrightarrow{\langle 2\text{€}, param:17 \rangle} \text{Alice}_{SC}$$

Après inclusion des opérations dans un bloc \mathcal{B} :

$\mathcal{B}(\mathcal{S})$

Alice	1220€
Bob	5440€
Charles	545€
Alice _{SC}	2€ Storage : 27

Tous les participants doivent **exécuter les appels au smart-contract** pour valider et mettre à jour leur état.

Que se passe-t-il si :

- On appelle un smart-contract très long à exécuter ou qui ne termine pas ?

Que se passe-t-il si :

- On appelle un smart-contract très long à exécuter ou qui ne termine pas ?
- Le code du smart-contract est très gros ou stocke (ou va stocker) énormément de données ?

Que se passe-t-il si :

- On appelle un smart-contract très long à exécuter ou qui ne termine pas ?
- Le code du smart-contract est très gros ou stocke (ou va stocker) énormément de données ?

On doit faire payer

Gas

- Les personnes appelant des S-C doivent fournir du *gas* **en payant**
- Chaque exécution de smart-contract a une limite de *gas*
- Chaque instruction exécutée engendre un coût en *gas*
- Si l'exécution dépasse le *gas* fourni par l'utilisateur, l'appel au S-C échoue

En général, le montant payé pour le *gas* est reversé au validateur du bloc.

On paye l'espace de stockage utilisé :

- À la création, on paye en proportion de :
 - La taille du code injectée
 - Le stockage initial à créer
- À l'appel, on paye le stockage qui va être généré

En général, le montant payé pour le stockage est « brûlé ».

Vote électronique (version naïve)

- Le contrat contient les adresses des participants autorisés à voter ;
- Lorsqu'un participant vote, le contrat marque dans son stockage que le participant à voter ;
- Après une certaine date, le vote est terminé et le contrat n'accepte plus d'appel. L'issue du vote et son historique restent consultables.

Exemples de « Dapps » (2)

Multi-signature M-N

- Un compte commun géré par N comptes ;
- Un transfert ne peut s'effectuer que si M parmi N signatures sont récoltées.

Exemples de « Dapps » (3)

Escrow (séquestre)

- Permet de sécuriser des échanges
- L'acheteur et le vendeur passent par le contrat :
 - Chacun donne $2\times$ le montant de la vente au contrat
 - Si les deux partis confirment le bon déroulement alors le transfert est effectué et le surplus est reversé sinon l'argent est maintenu dans le contrat tant qu'aucun accord n'est trouvé.

Exemples de « Dapps » (4)


Crypto-kitties... <https://www.cryptokitties.co/>

Langages de Smart-contract

- Bitcoin Script – Bitcoin
- Solidity, Vyper, EVM – Ethereum
- SmartPy, Ligo, Michelson – Tezos
- ...

Criticité des Smart-contracts

anyone can kill your contract #6995

 Closed ghost opened this issue on 6 Nov 2017 · 17 comments



ghost commented on 6 Nov 2017 · edited by ghost

I accidentally killed it.

<https://etherscan.io/address/0x863df6bfa4469f3ead0be8f9f2aae51c91a907b4>



69



3



115



59



24



46



1



2

Coût : 300 millions de dollars en tokens Ethereum

Conclusion

- Les smart-contracts permettent d'automatiser des actions sur la chaîne
- Chacun peut observer le code et les interactions des smart-contracts
- Les smart-contracts ont un coût d'utilisation et de potentielles failles
⇒ Il est important de s'assurer de leur correction