Protocoles Économiques et *Smart-contracts*

22 octobre 2019

Plan du cours

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

Protocole Économique

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*
- ...

Rappel : algorithmes de consensus

Proof-of-Work

Proof-of-Stake

Rappel : algorithmes de consensus

Proof-of-Work

- Calcul d'un hash de bloc en modifiant une nonce
- Premier arrivé, premier servi

Proof-of-Stake

- Le validateur est choisi aléatoirement
- Les chances sont pondérées par le *stake*

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 $\xrightarrow{10€}$ Bob	
Bob $\xrightarrow{5€}$ Bob	
Bob $\xrightarrow{2€}$ 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 (racine de l'arbre de Merkle)
- Ce hash est inclus au bloc

$\mathcal{B}$

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

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

Alice	1220€
Bob	5440€
Charles	545€

Validité des opérations

$$X \xrightarrow{\texttt{<montant>}} Y \quad (+ \text{ signature})$$

Propriétés que le protocole doit vérifier :

- X existe dans l'état
- X possède au moins `<montant>` dans son compte
- La signature est bien celle de X

Validité des opérations

$$X \xrightarrow{\texttt{<montant>}} Y \quad (+ \text{ signature})$$

Propriétés que le protocole doit vérifier :

- X existe dans l'état
- X possède au moins `<montant>` dans son compte
- La signature est bien celle de X

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

Validité des opérations

$$X \xrightarrow{\texttt{<montant>}} Y \quad (+ \text{ signature})$$

Propriétés que le protocole doit vérifier :

- X existe dans l'état
- X possède au moins `<montant>` 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 ?

Mécanismes d'*anti-replay*

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

- On dispose des opérations considérées comme « trop vieilles »
- Nécessite de stocker tous les blocs (hashs) dans l'état
  $\rightarrow$ Cela peut être coûteux

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

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

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

$\mathcal{S}$

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

Validité des blocs

$\mathcal{B}$

Pred. :	# 380f004f
Level :	345
State :	# 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

$\Rightarrow$ On doit stocker le bloc prédécesseur dans l'état

Application des blocs

En général, on distribue une récompense au créateur du bloc

Comment peut-on implémenter cela ?

Application des blocs

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 ?

Protocole Bitcoin

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 protocoles

Vitesse de création des blocs

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

Déterminer les constantes du protocoles

Vitesse de création des blocs

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

Déterminer les constantes du protocoles

Vitesse de création des blocs

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

Déterminer les constantes du protocoles (2)

Taille d'une opération

$$
\begin{aligned}
& 2 \text{ adresses } (2 \times 32 \text{ bytes}) + \text{signature } (32 \text{ bytes}) \\
& \text{compteur } (8 \text{ bytes}) + \text{montant } (8 \text{ bytes}) \\
= \quad & 112 \text{ bytes}
\end{aligned}
$$

Nombre d'opérations autorisées dans un bloc

- 10 opérations autorisées (1Kb) $\Rightarrow$ Effet ?

Déterminer les constantes du protocoles (2)

Taille d'une opération

2 adresses (2×32 bytes) + signature (32 bytes)
compteur (8 bytes) + montant (8 bytes)
$=$ 112 bytes

Nombre d'opérations autorisées dans un bloc

- 10 opérations autorisées (1Kb) $\Rightarrow$ Effet ?
- 10.000 opérations autorisées (1,1Mb) $\Rightarrow$ Effet ?

Déterminer les constantes du protocoles (2.2)

Il faut prendre en compte :

- Le temps de validation d'une opération et d'un bloc
- La propagation et l'aggrégation dans le réseau
- Les effets sur le consensus :
  - Exemple : les blocs vides sont plus rapides à créer que des blocs pleins
- 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 protocoles (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, . . .

Protocole économique – Conclusion

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

Définition

Un *Smart-contract* est un compte spécial possédant :

- Une addresse (identifiant) et un montant
- Un code à exécuter
- Un espace de stockage

Le code et le stockage sont visibles de tous.

Modèle d'exécution

Création

- Tout le monde peut créer un smart-contract
- On doit fournir un code et un stockage initial

Exécution

- On appelle un smart-contract via une transaction
- Seul le stockage varie entre chaque appel
- Un smart-contract peut, de lui-même :
  - Faire des transactions
  - Appeler un autre smart-contract
  - Créer de nouveaux smart-contracts
- L'exécution est atomique

Exemple

```
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  }
```

Un smart-contract P

Opération de création :

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

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

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

Exemple – Création

$\mathcal{B}$

Pred. :	# 380f424f
Level :	4201
State :	# 2952a3de
Opérations : op	

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

Alice	1220€		
Bob	5440€		
Charles	545€		
Alice_{SC}	0€	0	P

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

Exemple – Appels

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

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

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

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

Alice	1220€
Bob	5440€
Charles	545€
Alice_{SC}	2€ 27 P

Tout le monde va exécuter ($2\times$) le smart-contract pour valider et mettre à jour l'état

Coût des Smart-contracts

Que se passe-t-il si :

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

Coût des Smart-contracts

Que se passe-t-il si :

- On appelle un smart-contract qui est 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 ?

Coût des Smart-contracts

Que se passe-t-il si :

- On appelle un smart-contract qui est 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

Notion de *Gas*

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

Le montant payé pour le gas est reversé au validateur du bloc

Stockage des Smart-contracts

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

Le montant payé pour le stockage est « brûlé »

Exemples de « Dapps »

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 » (2)

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.

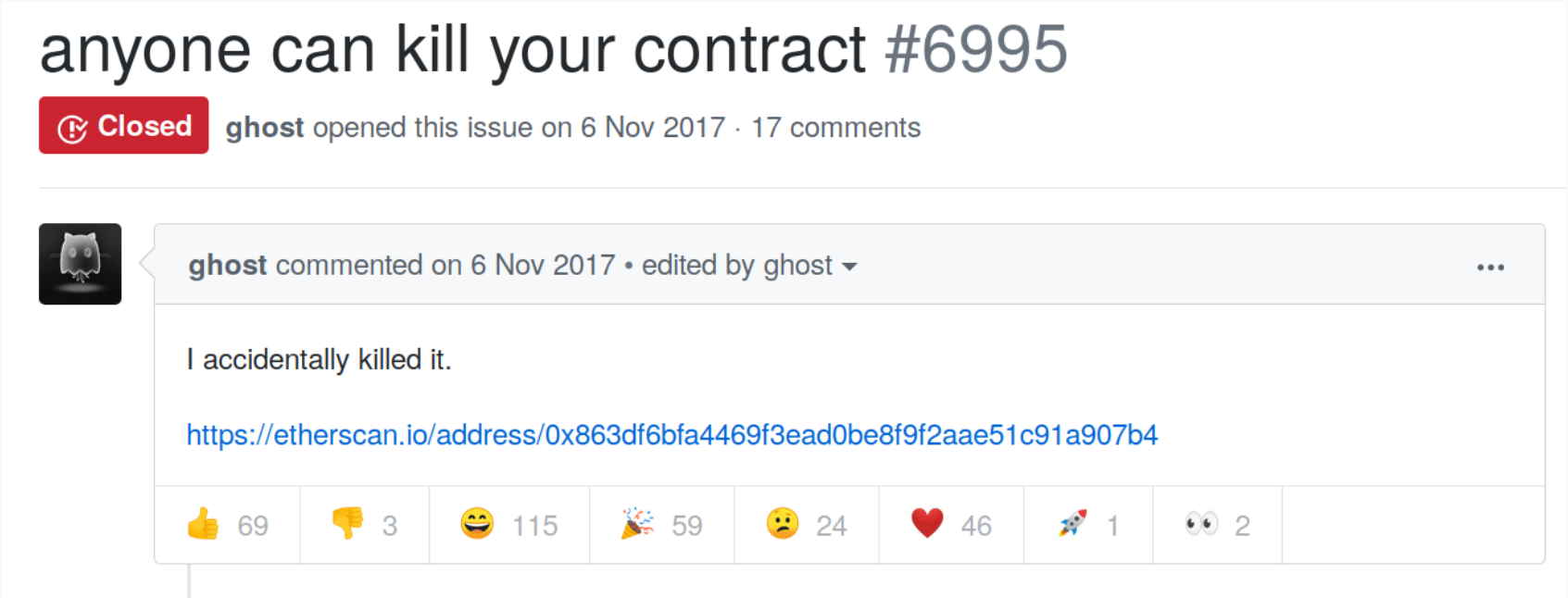
Exemples de « Dapps » (3)

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



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 des failles
  $\Rightarrow$ Il est important de s'assurer de leur correction