IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Blockchain: Principes de fonctionnement & Applications à l'énergie

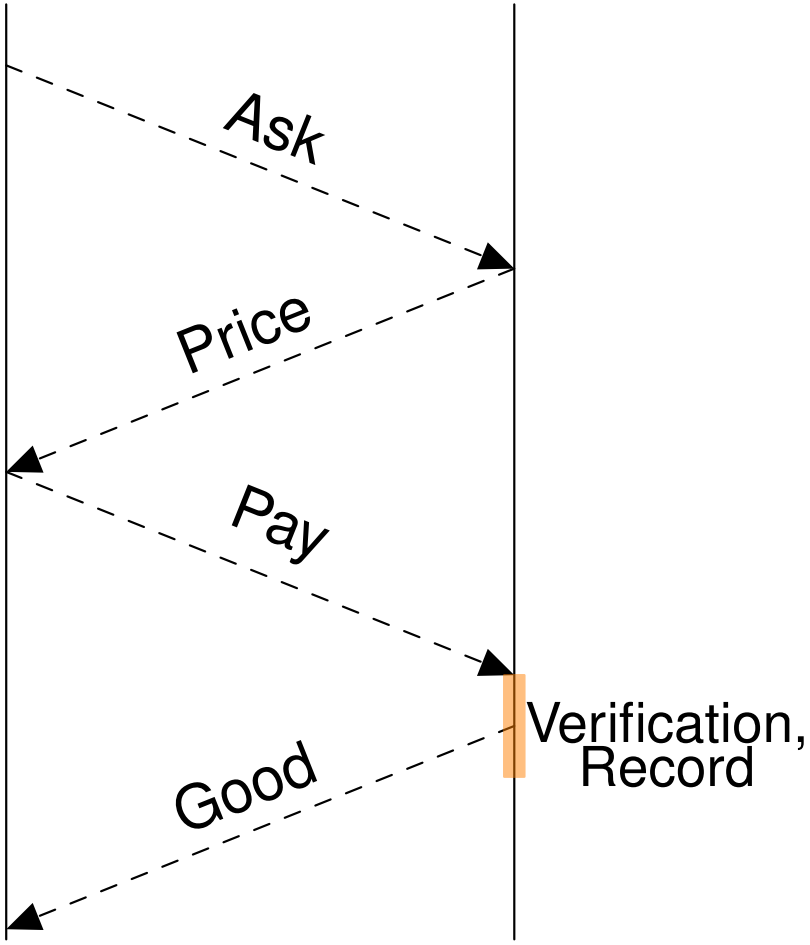
Romaric Ludinard
IMT Atlantique
22 juin 2018

SOMMAIRE

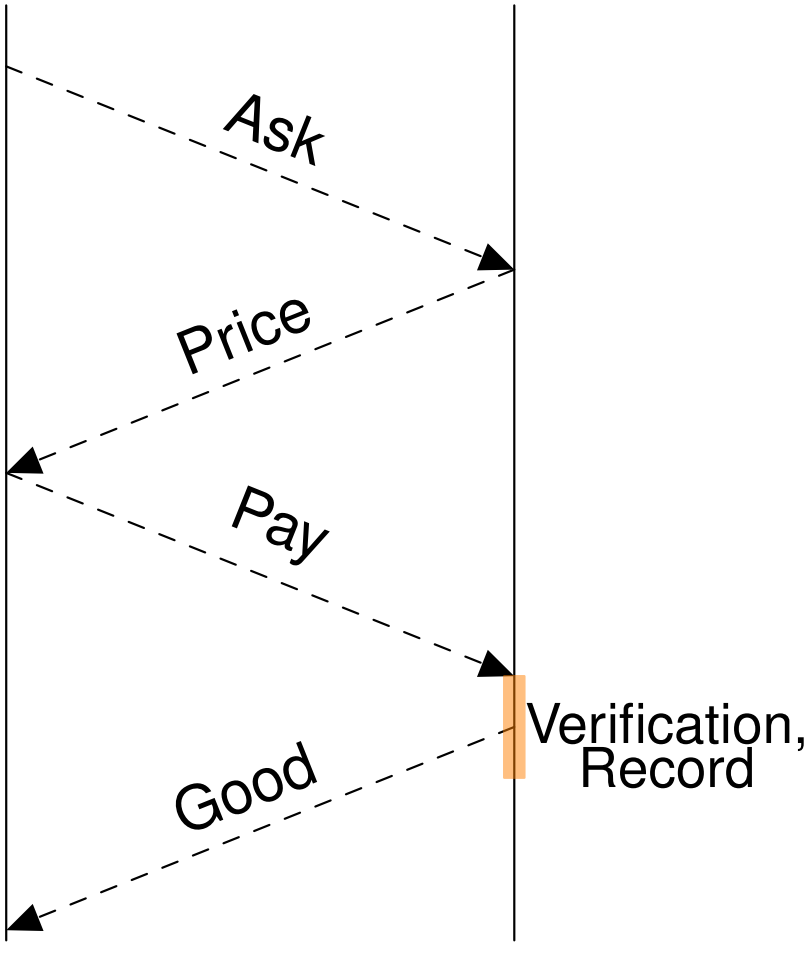
Propriétés d'une devise

- Sont générées par le système
- Sont détenues par une entité du système
- Ne peuvent être dépensées que par leur propriétaire
- Peuvent être utilisées à tout moment
- Ne peuvent être employées que pour une transaction

Utilisation de devises physiques



Utilisation de devises physiques

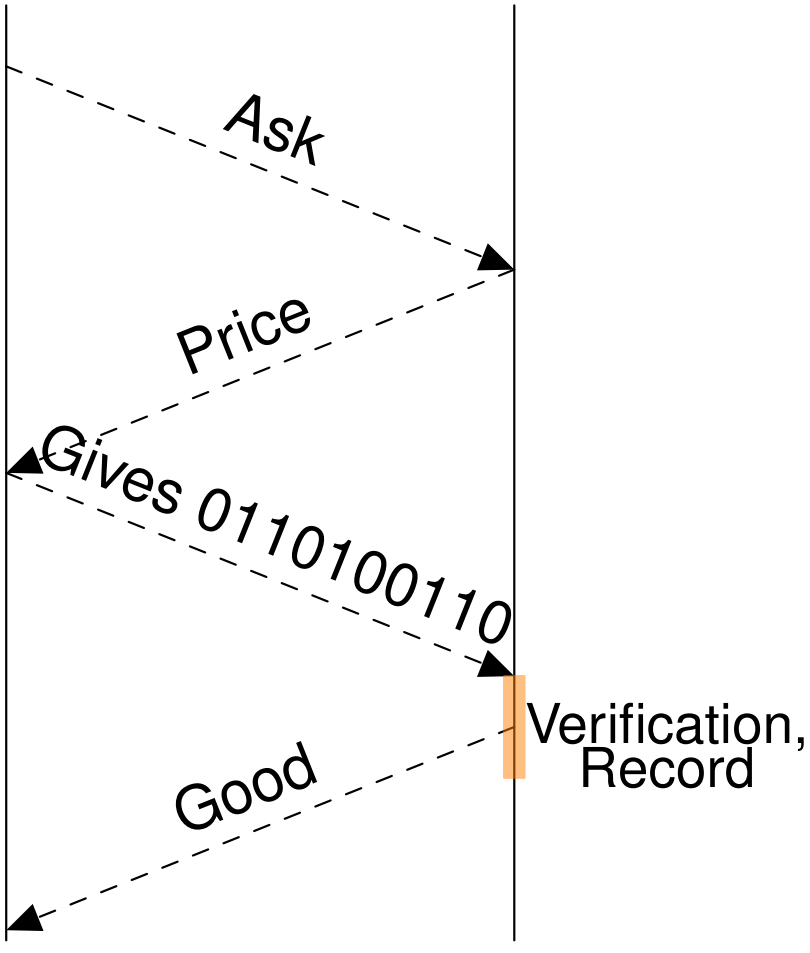


- Savoir partagé

- Test de validité

- Token physique
- Unicité de la transaction
- Enregistrement de la transaction

Utilisations de devises virtuelles



- Échange de transactions
- Vérification de la transaction
- Enregistrement de la transaction

Questions

- Qu'est ce qu'une transaction ?
- Comment vérifier si une transation est valide ?
- Comment enregistrer une transaction dans le système ?

Propriétés d'une devise

- Sont générées par le système
- Sont détenues par une entité du système
- Ne peuvent être dépensées que par leur propriétaire
- Peuvent être utilisées à tout moment
- Ne peuvent être employées que pour une transaction

Problème

Implémentation d'un système pour suivre les échanges de devises.

SOMMAIRE

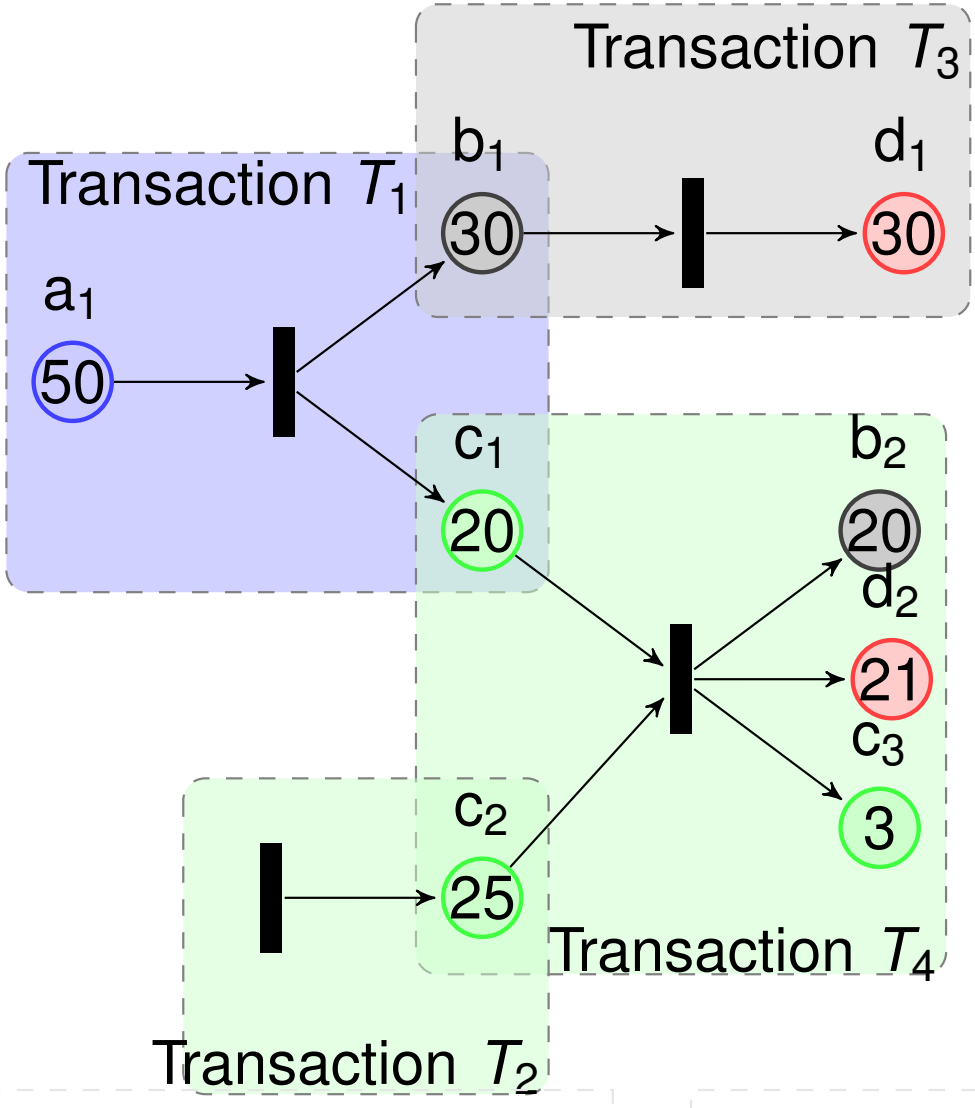
- Système pair-à-pair
- Transactions : transfert de valeur entre deux utilisateurs
- Trois types d'acteurs : utilisateur, pair, mineur
- Historique des transactions émises dans le système
- Registre répliqué sur chaque pair

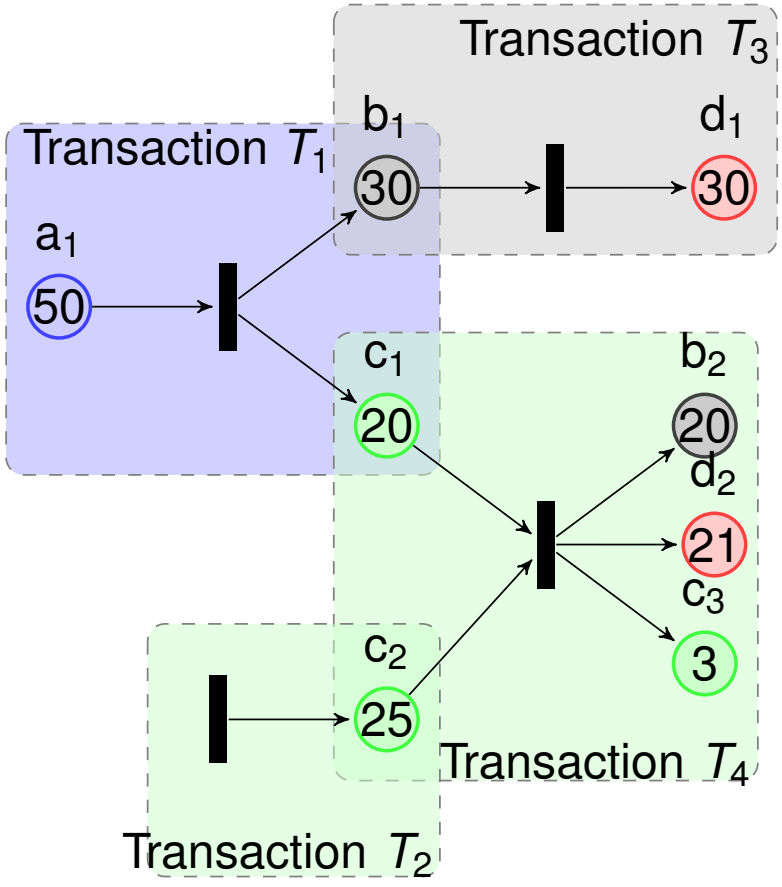
- Système pair-à-pair
- Transactions : transfert de valeur entre deux utilisateurs
- Trois types d'acteurs : utilisateur, pair, mineur
- Historique des transactions émises dans le système
- Registre répliqué sur chaque pair

Problème

En l'absence de tiers de confiance, comment

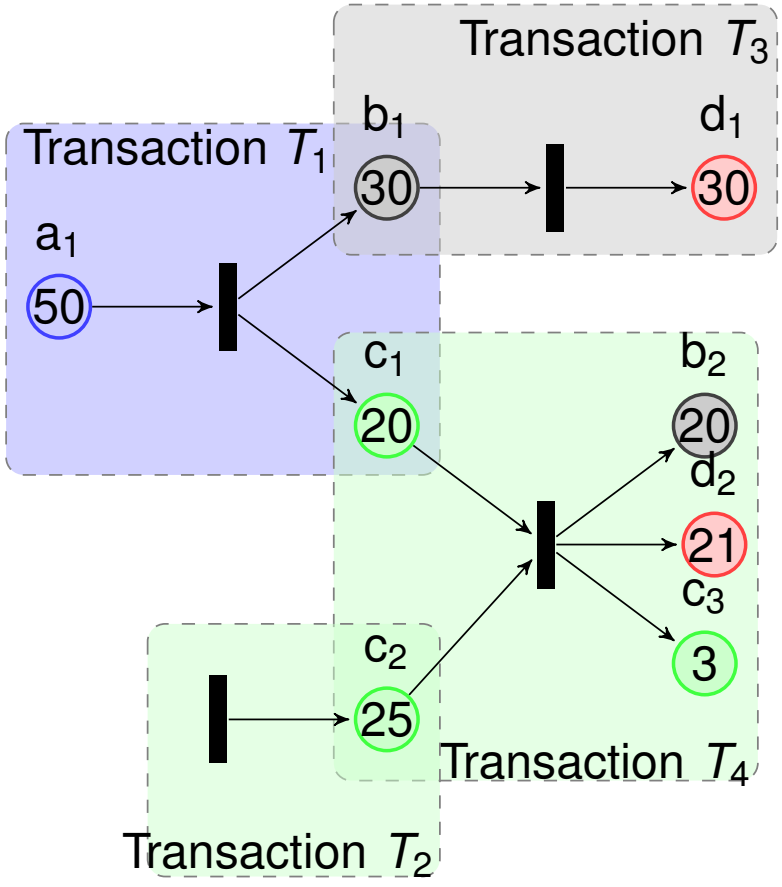
- vérifier la validité d'une transaction ?
- vérifier l'unicité d'une transaction ?
- enregistrer les transactions émises ?


Transaction T_3
b_1
d_1
30
30
Transaction T_1
a_1
50
c_1
20
b_2
20
d_2
21
c_3
3
c_2
25
Transaction T_4
Transaction T_2



Transaction T_4 dépense

- sortie 2 de T_1
- sortie 1 de T_2



Transaction T_4 dépense

- sortie 2 de $\mathfrak{h}(T_1)$
- sortie 1 de $\mathfrak{h}(T_2)$

Comment prouver que je suis légitime à dépenser ces sorties ?

Différentes étapes :

- Définir un bénéficiaire
- Éviter la double dépense
- Prouver que le bénéficiaire est légitime

Différentes étapes :

- Définir un bénéficiaire

⇒ Mise en place d'un challenge

- Éviter la double dépense

⇒ Enregistrement dans un historique

- Prouver que le bénéficiaire est légitime

⇒ Réponse au challenge

Transactions bien formées

Une transaction T est dite bien formée si

i) toutes les entrées appartiennent à l'utilisateur émettant T

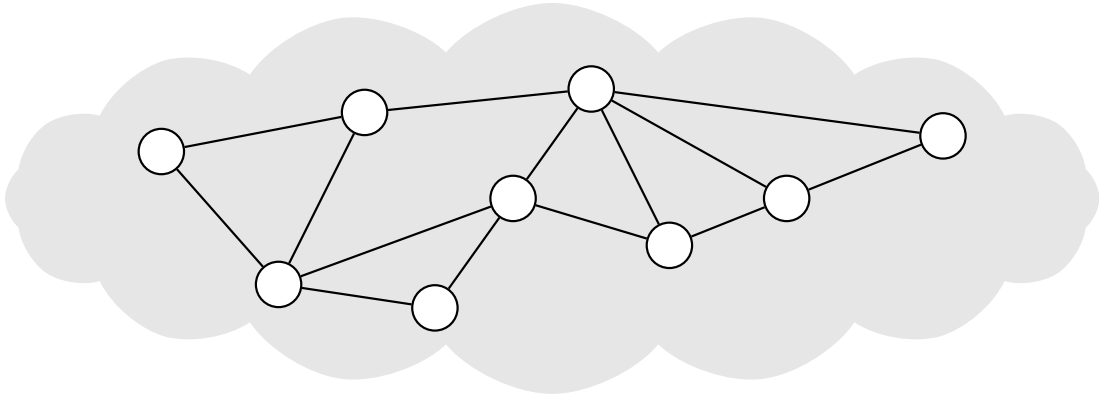
ii) le montant total en entrée est supérieur au montant de sortie.

iii) elle ne crée pas de double dépense.

Utilisation des transactions

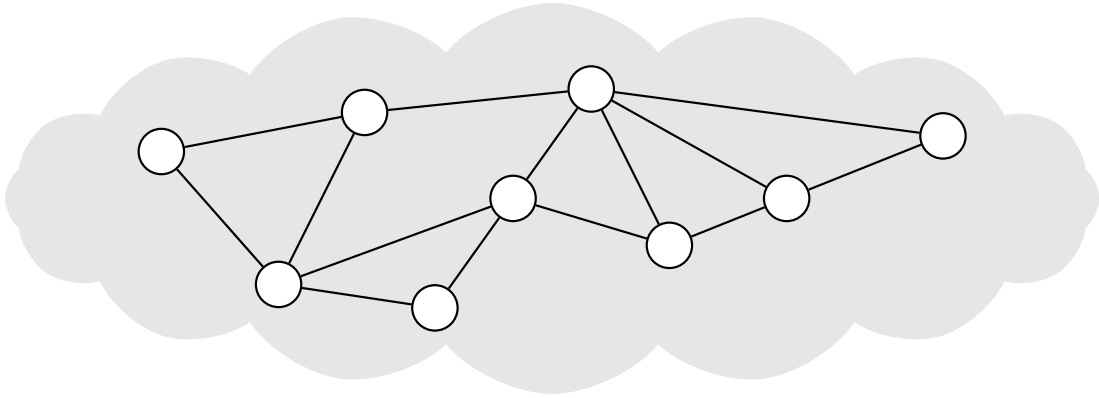
Seller

Client

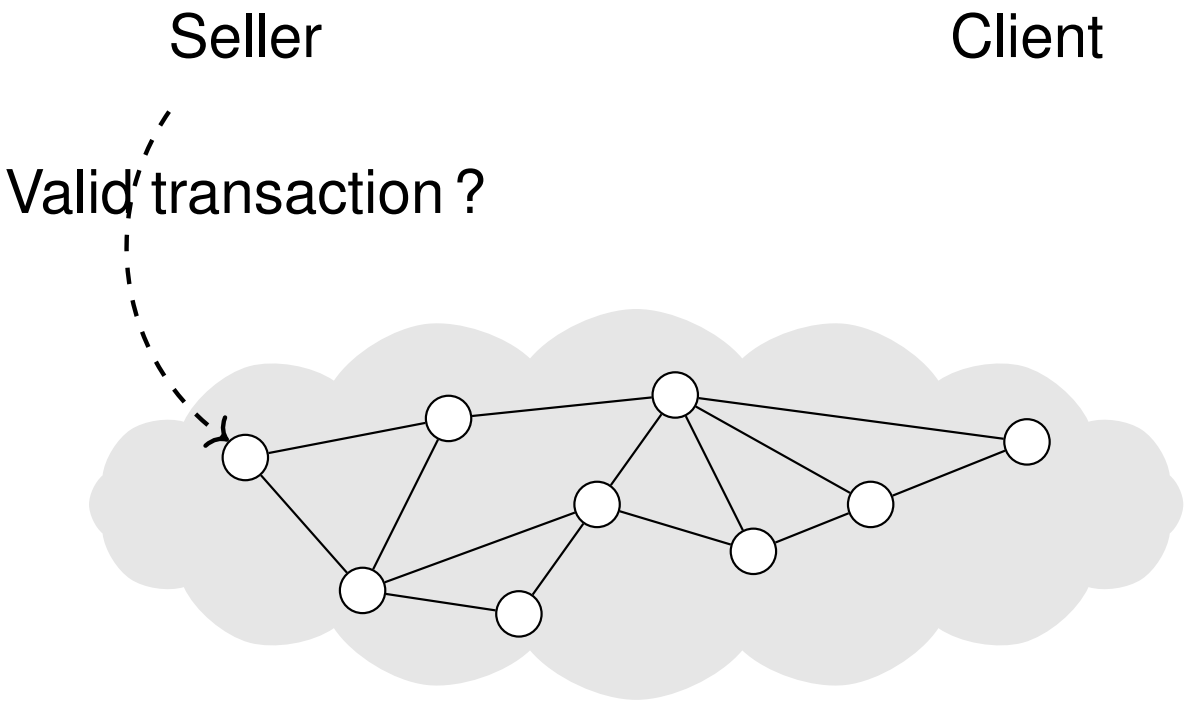


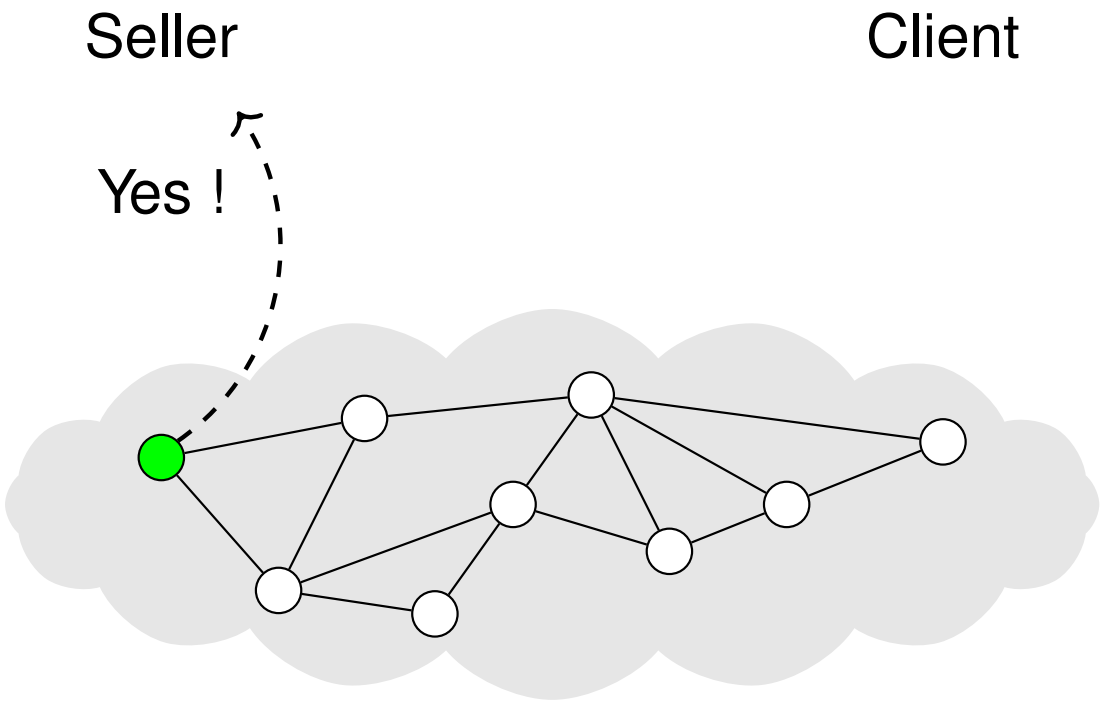
Signed transaction

Seller <- - - - - - - - - - - - - - - - - - Client

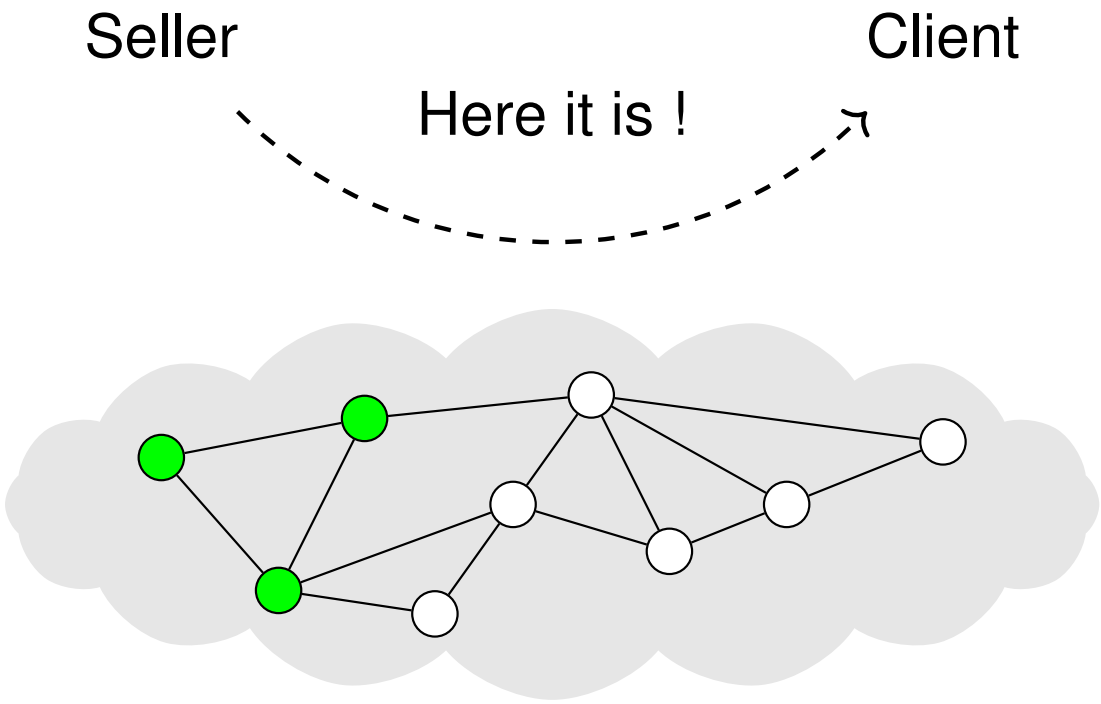


Utilisation des transactions




Seller
Client
Yes !

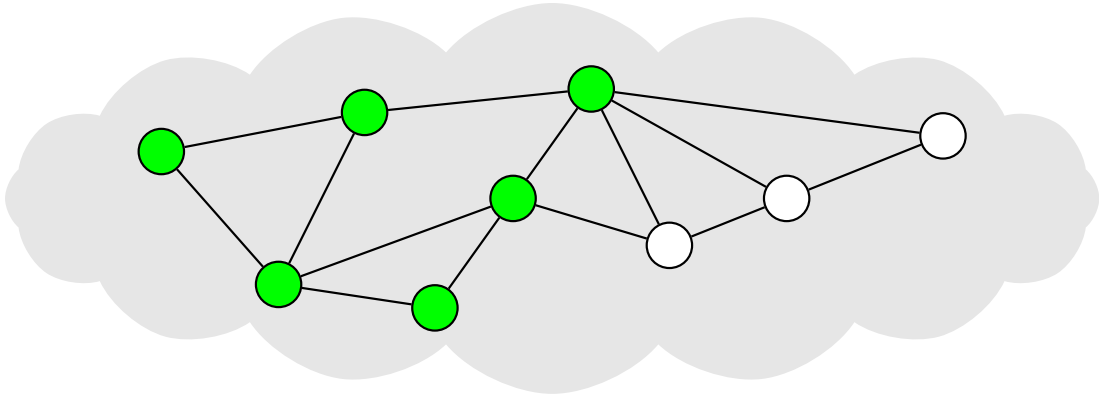
Utilisation des transactions



Utilisation des transactions

Seller

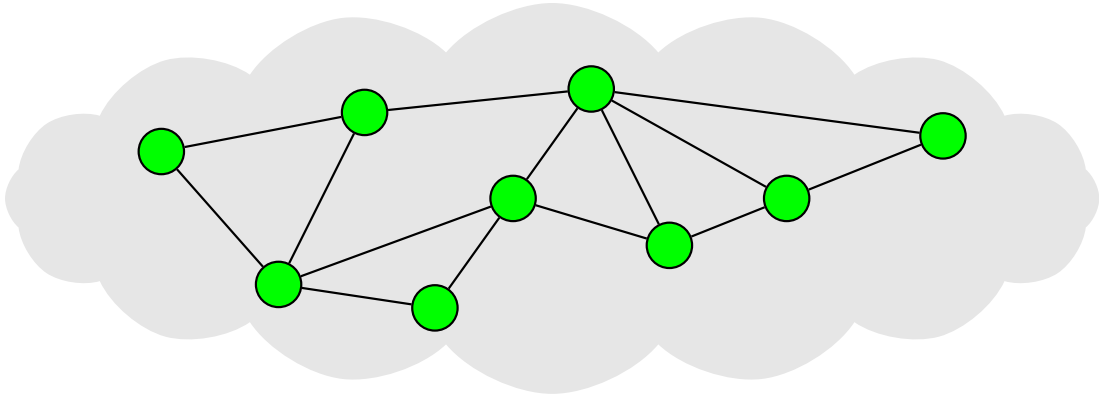
Client



Utilisation des transactions

Seller

Client



SOMMAIRE

Blockchain : implémentation de la vue locale

- Donnée localement valide éligible à l'inclusion dans un bloc
- Preuve d'intégrité : Arbre de Merkle

- Donnée localement valide éligible à l'inclusion dans un bloc
- Preuve d'intégrité : Arbre de Merkle
- Résilience aux attacks Sybil

- Donnée localement valide éligible à l'inclusion dans un bloc
- Preuve d'intégrité : Arbre de Merkle
- Résilience aux attacks Sybil

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

Unsolved Sudoku

- Donnée localement valide éligible à l'inclusion dans un bloc
- Preuve d'intégrité : Arbre de Merkle
- Résilience aux attacks Sybil

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

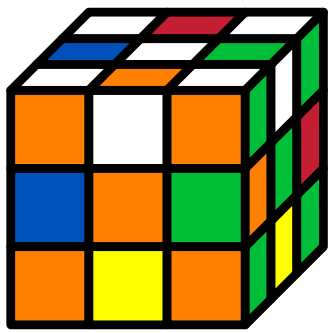
Unsolved Sudoku

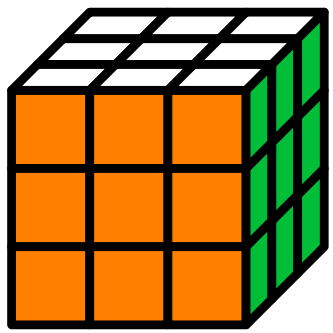
4	2	*6*	5	*7*	1	*3*	9	*8*
8	*5*	*7*	2	*9*	3	*1*	*4*	6
1	3	*9*	*4*	6	*8*	*2*	7	*5*
9	*7*	1	*3*	*8*	*5*	6	*2*	*4*
5	4	*3*	*7*	*2*	*6*	*8*	1	9
6	*8*	2	*1*	*4*	*9*	7	*5*	*3*
7	9	*4*	*6*	3	*2*	*5*	8	*1*
2	*6*	*5*	8	*1*	4	*9*	*3*	7
3	1	*8*	9	*5*	7	*4*	6	*2*

Solved Sudoku

Blockchain : implémentation de la vue locale

- Donnée localement valide éligible à l'inclusion dans un bloc
- Preuve d'intégrité : Arbre de Merkle
- Résilience aux attacks Sybil

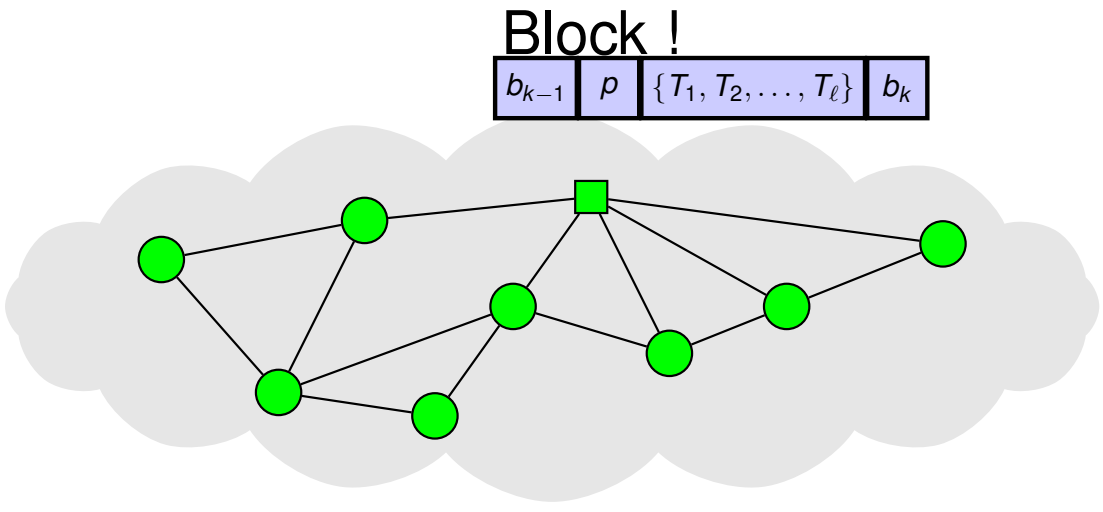




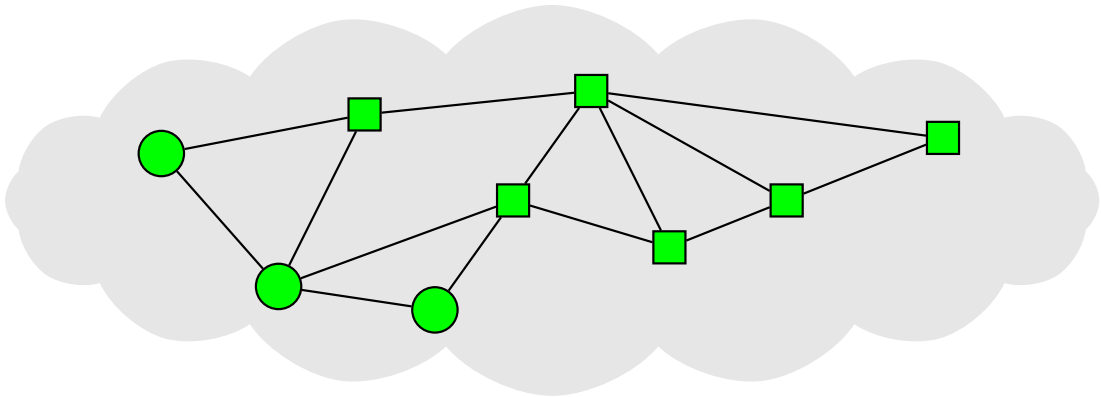
here

- Créer des bitcoins = créer des blocs
- Difficile d'un point de vue calculatoire. . .
- . . . mais loin d'être impossible
- Résultat peu couteux à vérifier
- But : trouver une preuve de travail valide pour la blockchain courante
- Délai moyen inter-bloc 10 minutes

Seller Client

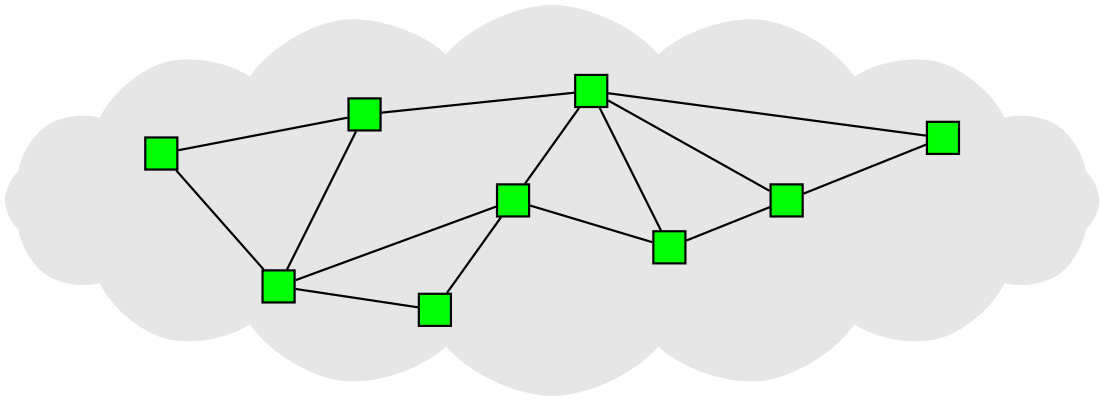


Seller Client



Seller Client

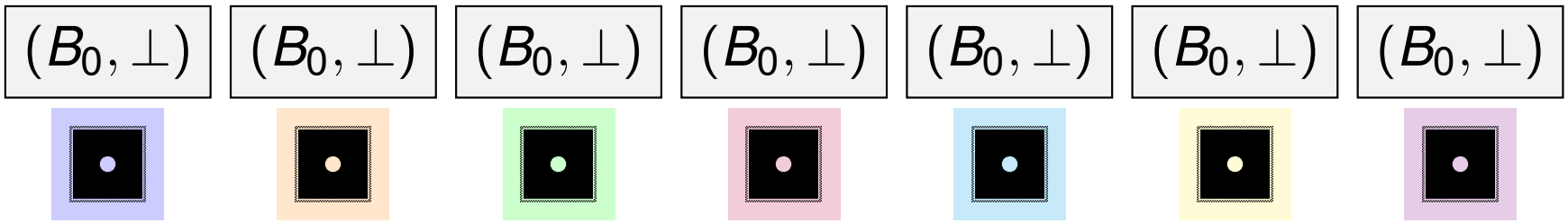
Transaction confirmed

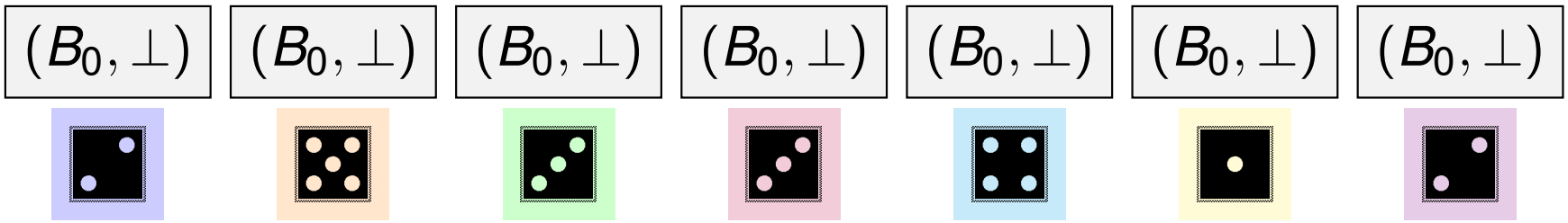


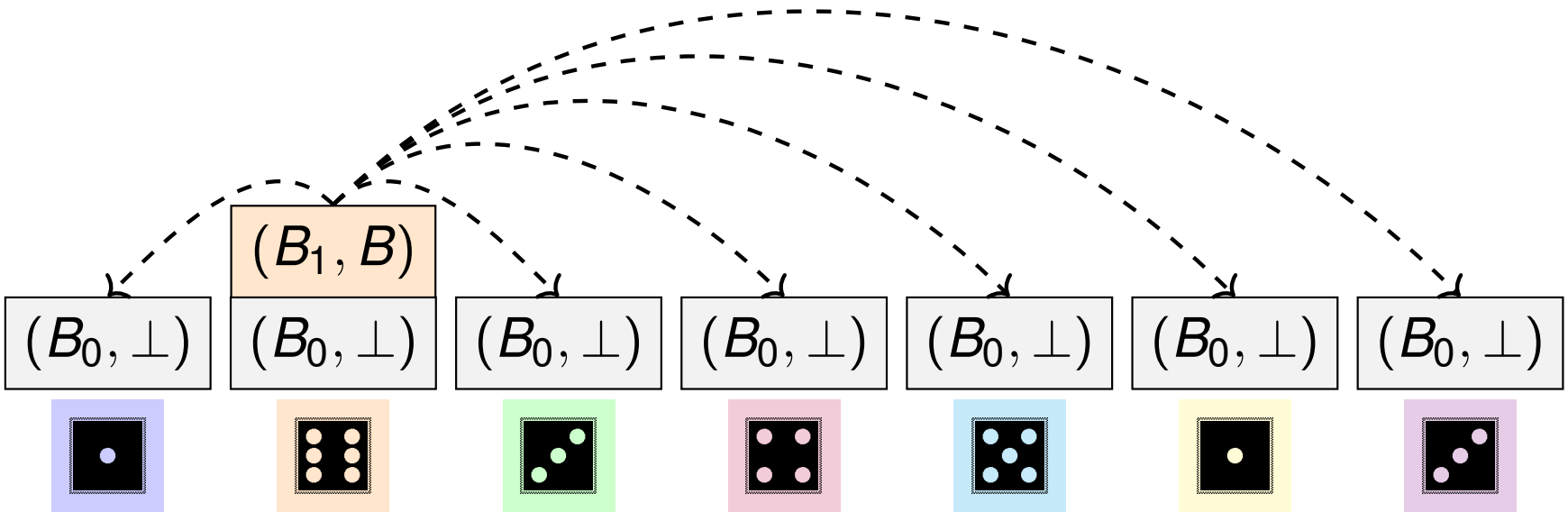
Confirmation locale de transaction

Étant donné un pair q du réseau Bitcoin, et une transaction localement valide T,

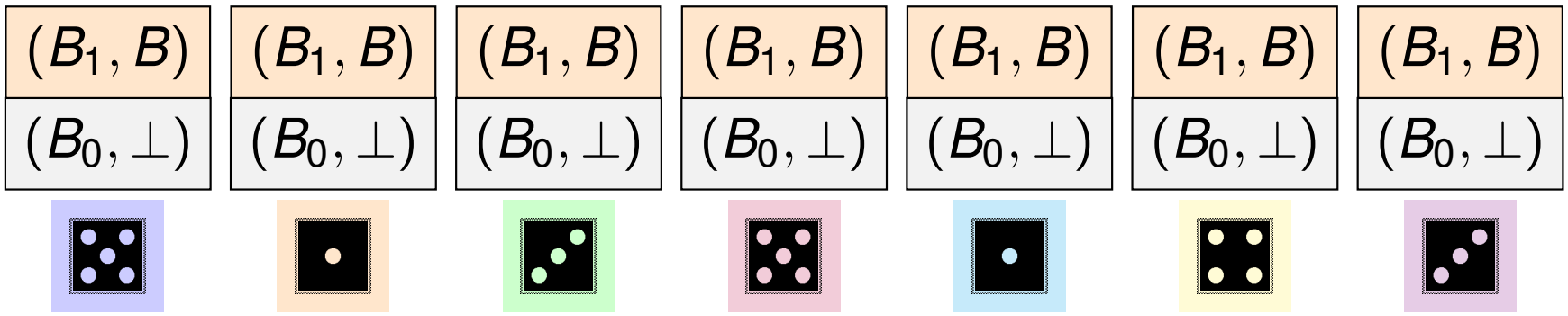
T est localement confirmée $\iff$ T appartient à l'historique local.

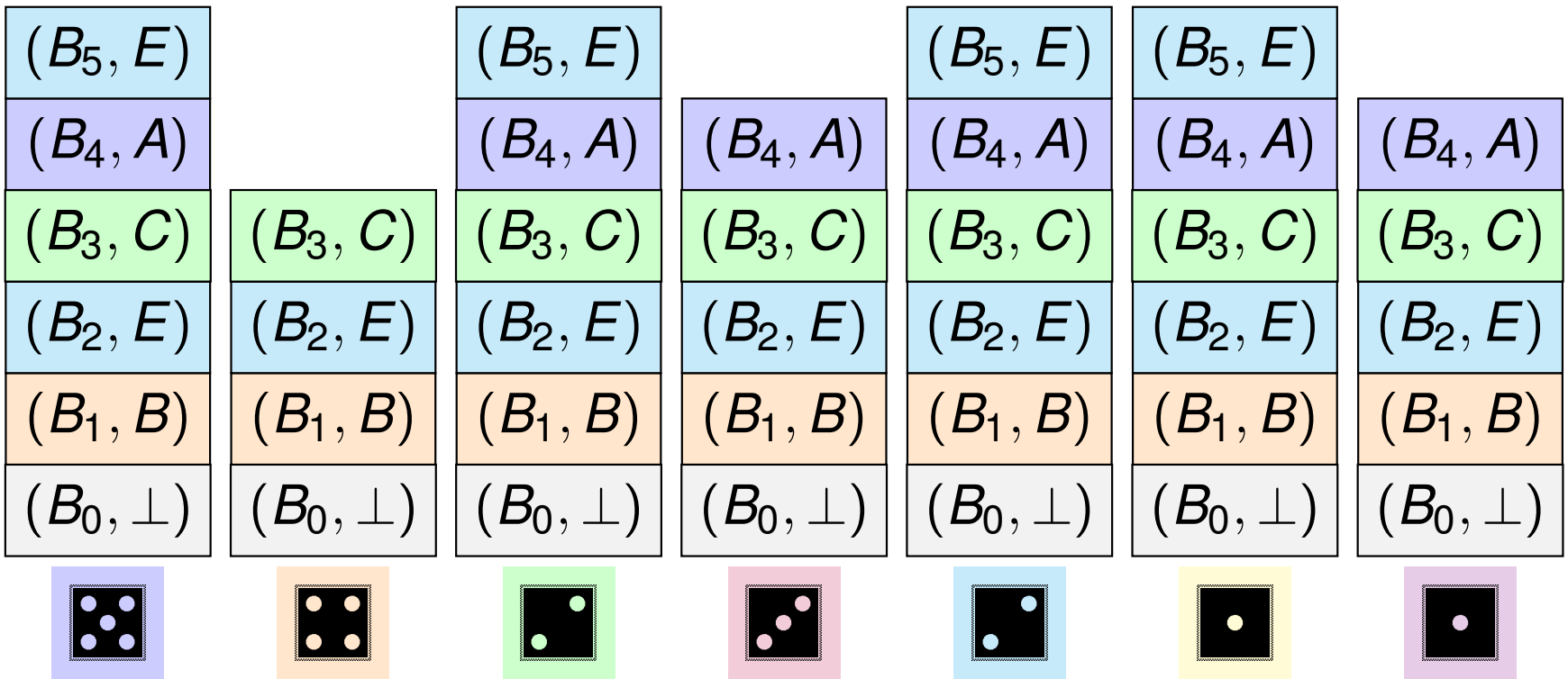

$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
B
C
D
E
F
G


$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
B
C
D
E
F
G


(B_1, B)
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
$(B_0, \perp)$
B
C
D
E
F
G

Block !


(B_1, B)
$(B_0, \perp)$
B
C
D
E
F
G


(B_5, E)
(B_4, A)
(B_3, C)
(B_2, E)
(B_1, B)
$(B_0, \perp)$
B
C
D
E
F
G

- Mise à jour de Bitcoin lente
- Instabilité de la chaîne
- Course à l'armement (CPU $\rightarrow$ GPU $\rightarrow$ ASIC)

- Mise à jour de Bitcoin lente

⇒ augmenter la fréquence des blocs

- Instabilité de la chaîne

⇒ changement de politique d'arbitrage

- Course à l'armement (CPU → GPU → ASIC)

⇒ changement du mécanisme de création de bloc

- Programme embarqué dans la blockchain
- Turing complet
- Exécution de pas de calculs à reception de transactions
- Execution vérifiable (i.e. pas d'aléatoire)

⇒ Limité à la blockchain

⇒ Stockage de l'état interne de chaque contrat

- Contrats ERC-20

SOMMAIRE

Limites actuelles

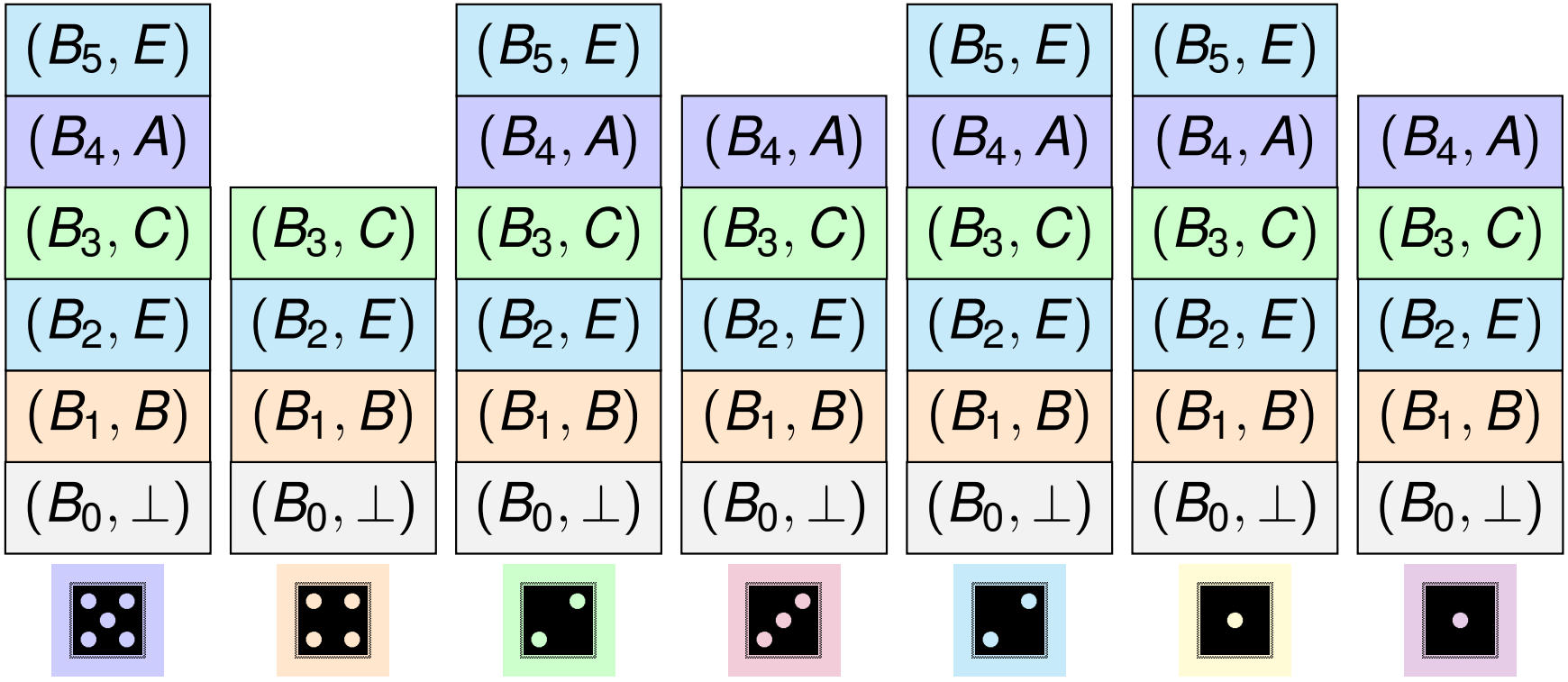
- Système ouvert
- Mécanisme de validation de bloc énergivoire (PoW)
- Création “monétaire”
- Énergie “périssable”
- Confidentialité de la consommation ?

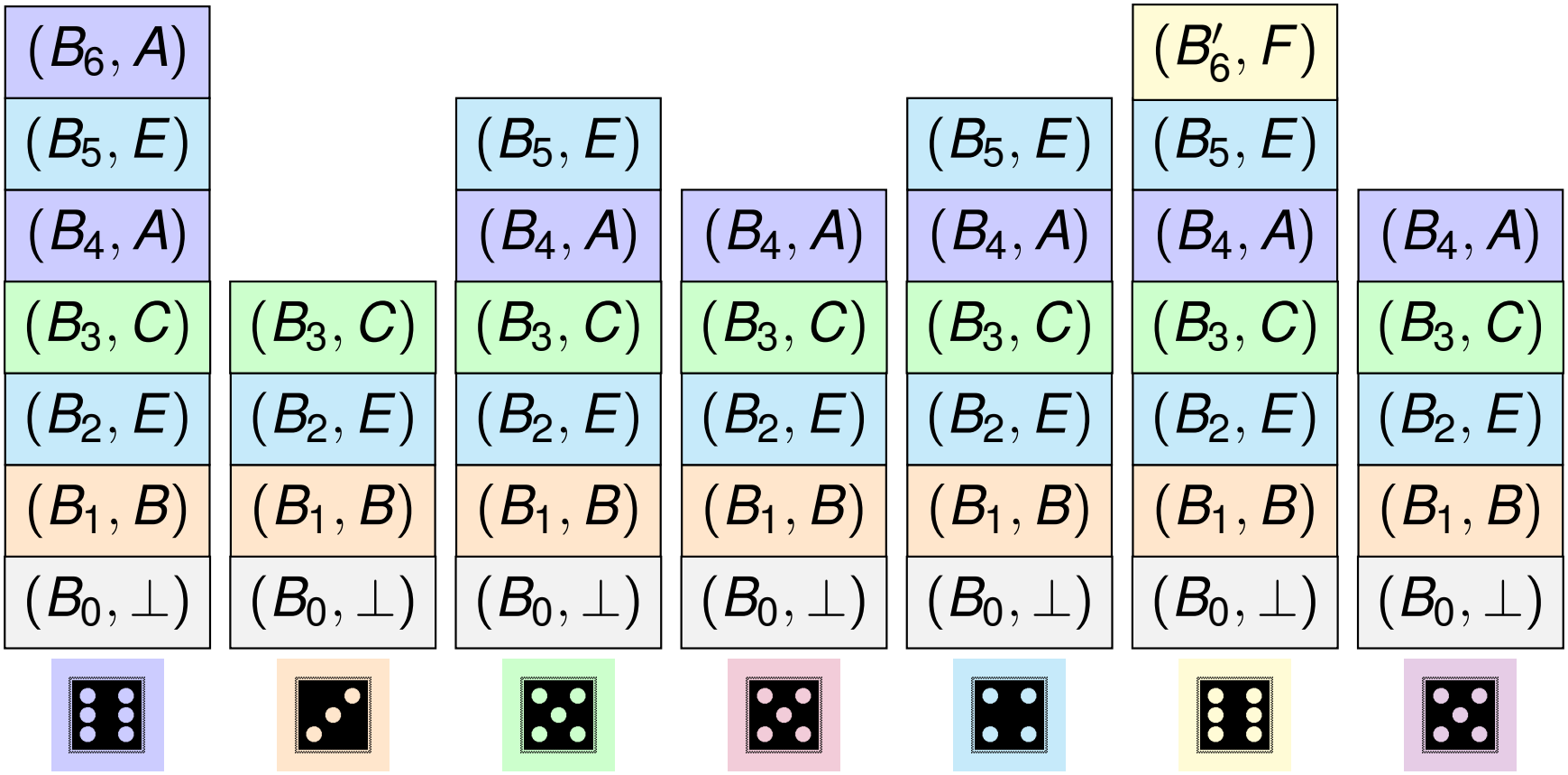
Limites actuelles

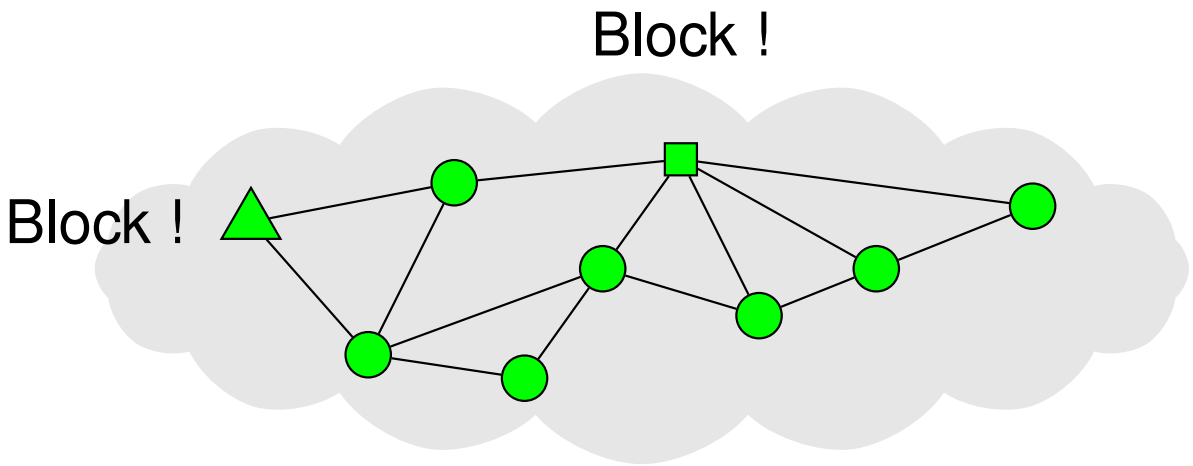
- Système ouvert
  $\Rightarrow$ alternatives ?
- Mécanisme de validation de bloc énergivoire (PoW)
  $\Rightarrow$ alternatives ?
- Création “monétaire”
  $\Rightarrow$ auditabilité ?
- Énergie “périssable”

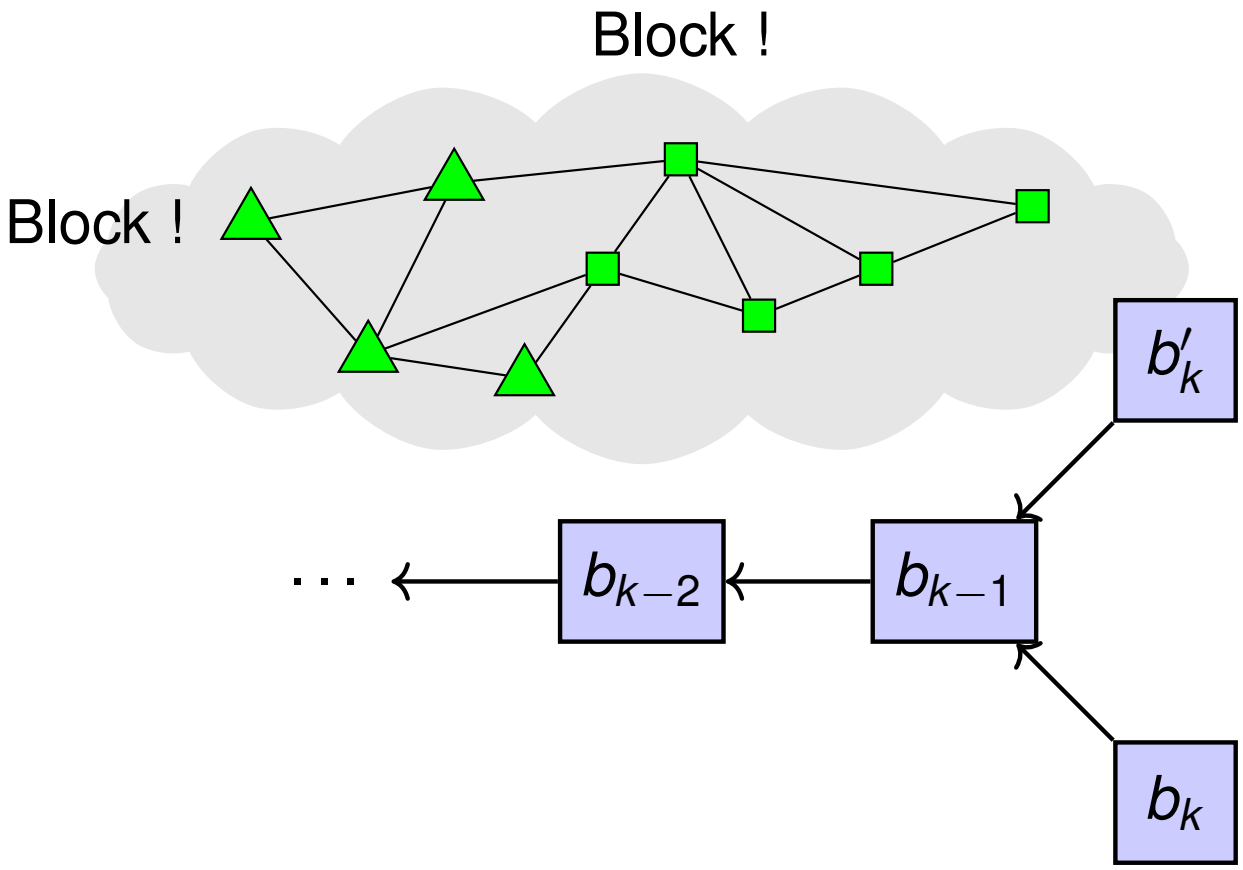
- Confidentialité de la consommation ?

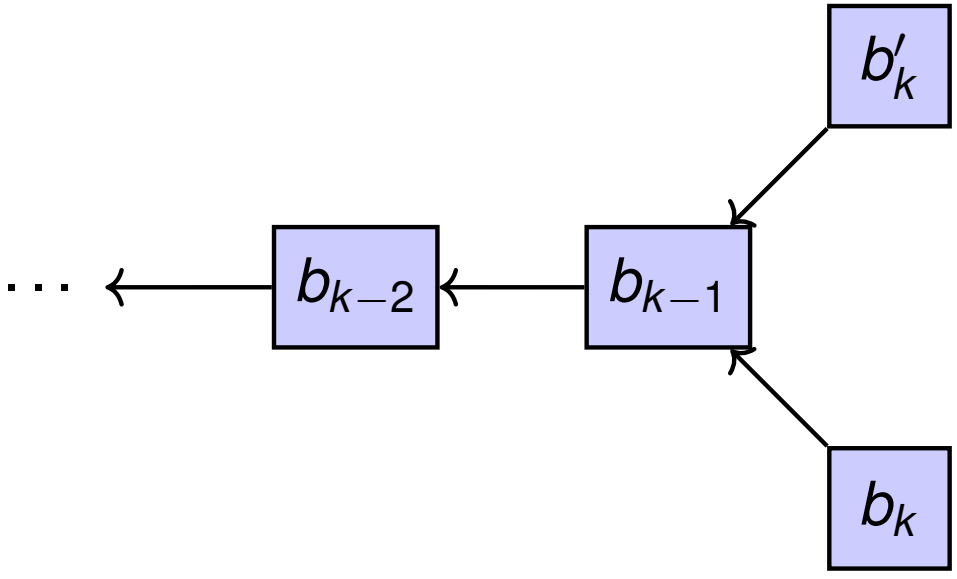
Questions : `romaric.ludinard@imt-atlantique.fr`

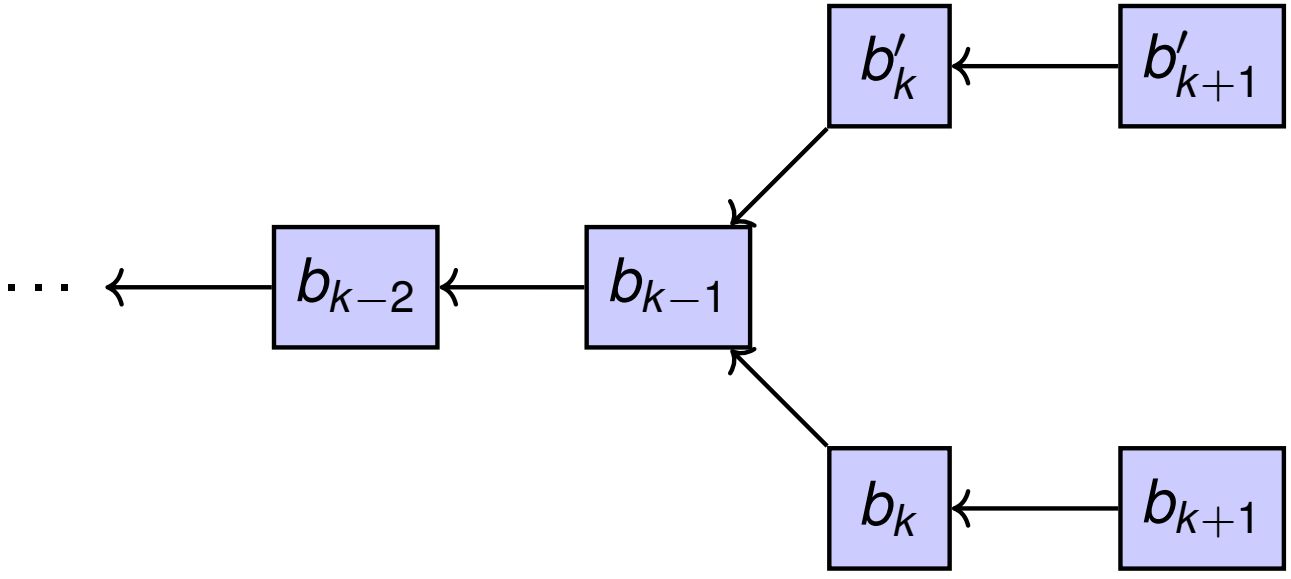

(B_5, E)
(B_4, A)
(B_3, C)
(B_2, E)
(B_1, B)
$(B_0, \perp)$
B
C
D
E
F
G

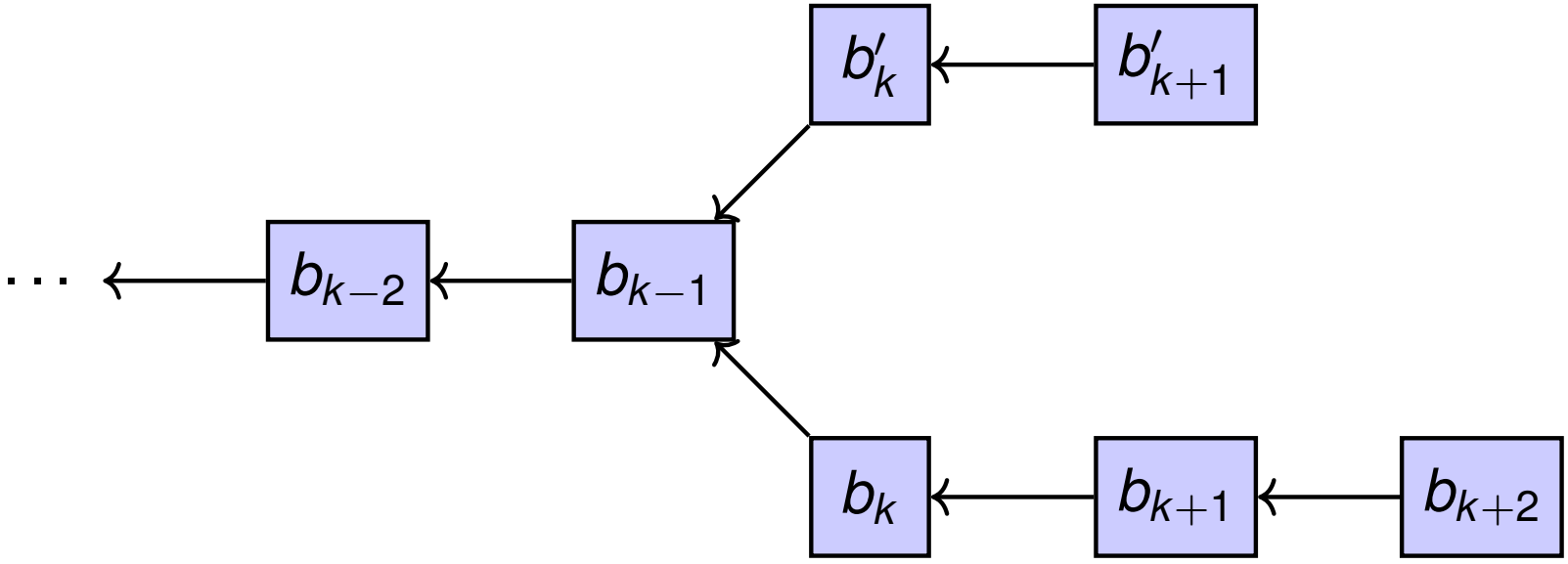

(B_6, A) (B_5, E) (B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B_5, E) (B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B_5, E) (B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B'_6, F) (B_5, E) (B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
B
C
D
E
F
Block !
G

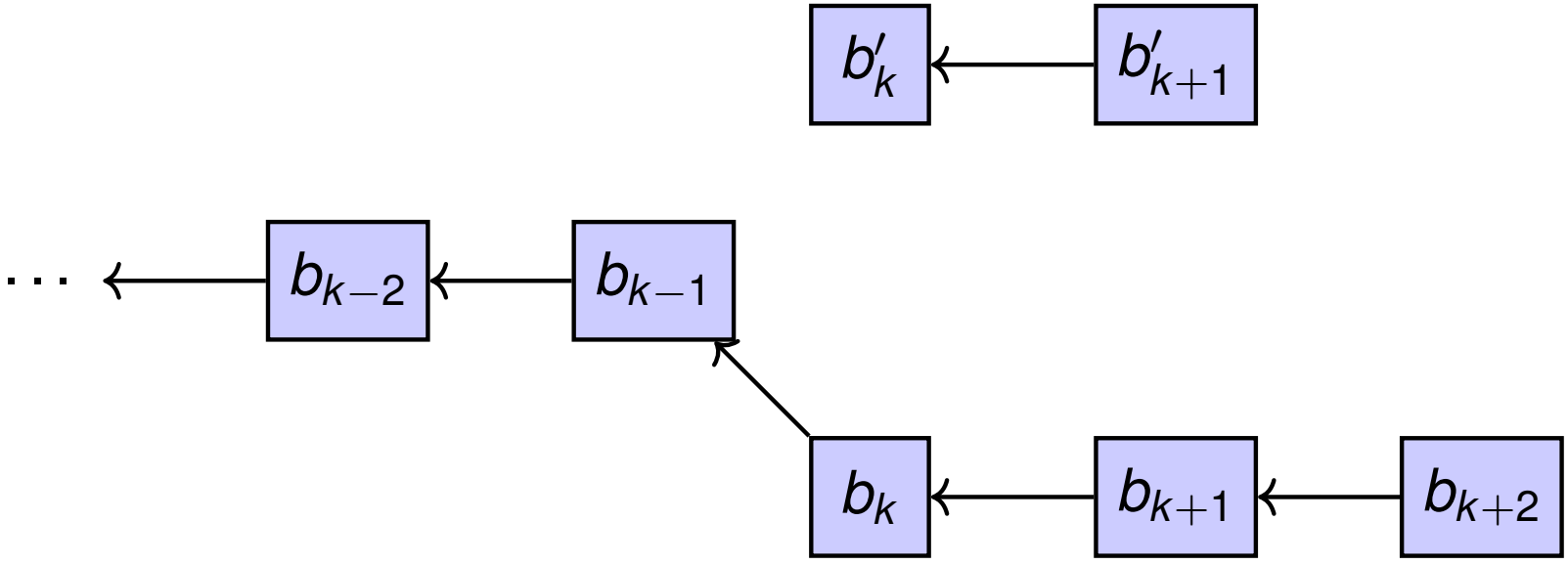

Block !
Block !

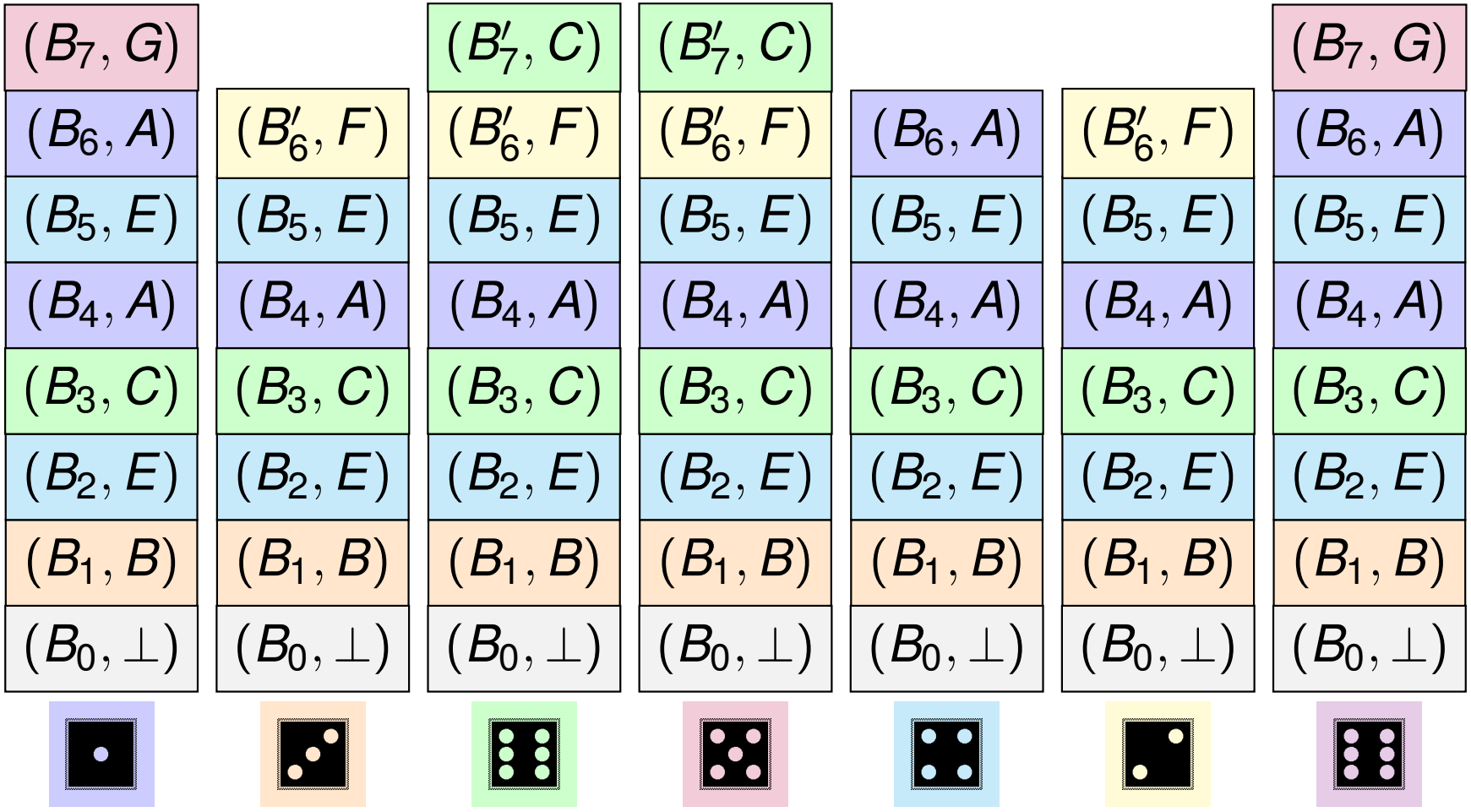

Block !
Block !
b'_k
b_{k-2}
b_{k-1}
b_k

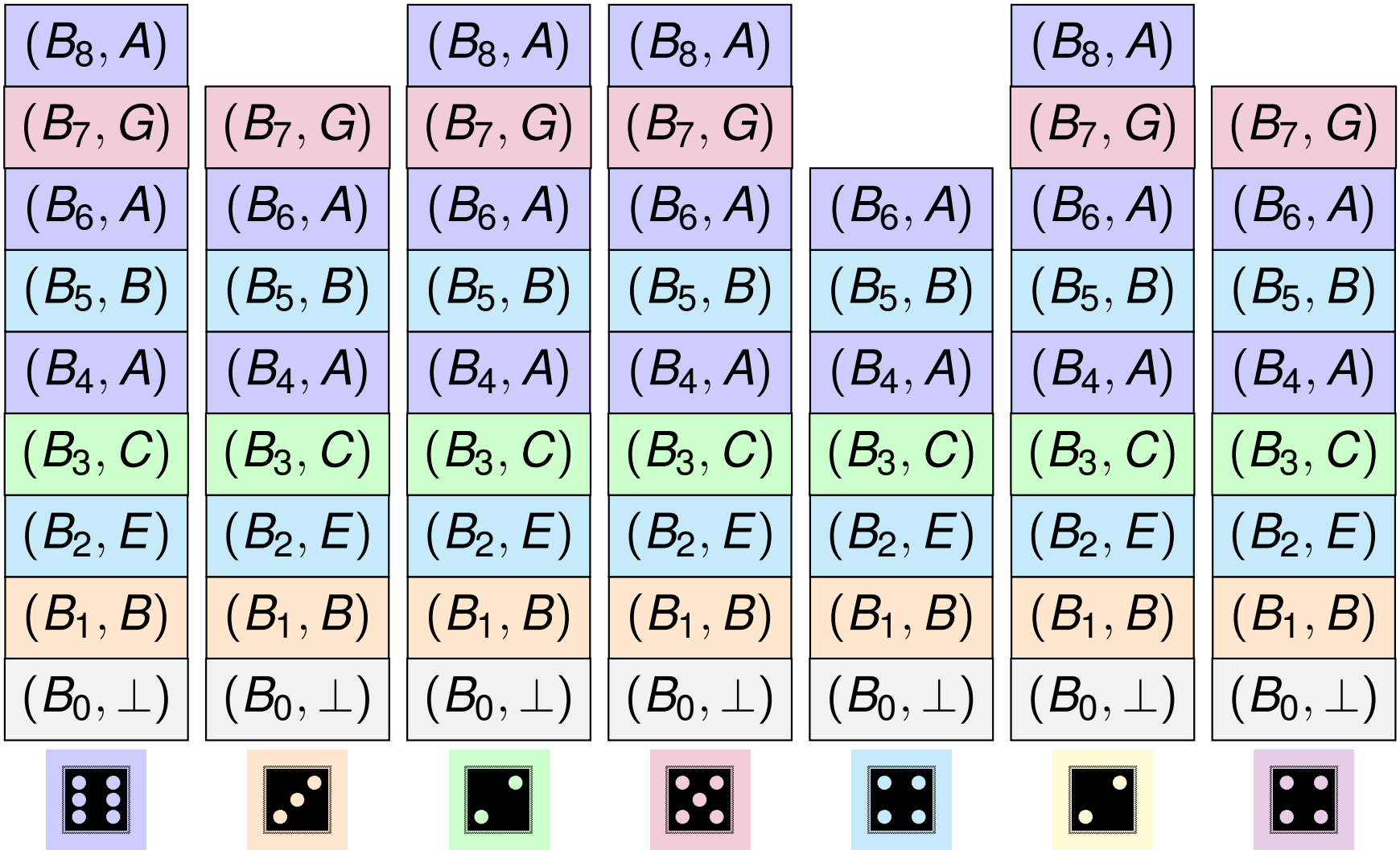

b'_k
b_{k-2}
b_{k-1}
b_k


b'_k
b'_{k+1}
b_{k-2}
b_{k-1}
b_k
b_{k+1}


b'_k
b'_{k+1}
$\cdots$
b_{k-2}
b_{k-1}
b_k
b_{k+1}
b_{k+2}


b'_k
b'_{k+1}
$\cdots$
b_{k-2}
b_{k-1}
b_k
b_{k+1}
b_{k+2}


(B_7, G) (B_6, A) (B_5, E) (B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B'_6, F) (B_5, E) (B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B'_7, C) (B'_6, F) (B_5, E) (B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B'_7, C) (B'_6, F) (B_5, E) (B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B_6, A) (B_5, E) (B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B'_6, F) (B_5, E) (B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
(B_7, G) (B_6, A) (B_5, E) (B_4, A) (B_3, C) (B_2, E) (B_1, B) $(B_0, \perp)$
B
C
Block !
D
E
F
G
Block !


(B_8, A)
(B_7, G)
(B_6, A)
(B_5, B)
(B_4, A)
(B_3, C)
(B_2, E)
(B_1, B)
$(B_0, \perp)$
B
C
D
E
F
G

string=HelloWorld !, nonce=0, difficulty=000

string=HelloWorld !, nonce=0, difficulty=000,

```
HelloWorld!0 : 3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb609106a
```

string=HelloWorld !, nonce=0, difficulty=000,

```
HelloWorld!0 : 3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb609106a
HelloWorld!1 : b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eefc22a27
```

string=HelloWorld !, nonce=0, difficulty=000,

```
HelloWorld!0 : 3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb609106a
HelloWorld!1 : b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eefc22a27
HelloWorld!2 : 5b6fd9c27fcb54ca23404d9428f081b7c9280ba6370e33a6a20b16f40ce76320
```

string=HelloWorld !, nonce=0, difficulty=000,

```
HelloWorld!0 : 3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb609106a
HelloWorld!1 : b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eefc22a27
HelloWorld!2 : 5b6fd9c27fcb54ca23404d9428f081b7c9280ba6370e33a6a20b16f40ce76320
HelloWorld!3 : 9c5d769416aa0ca894abf22bd17bd30fbb6959291423ae1903a9f86a1fe7ce78
....
```

string=HelloWorld !, nonce=0, difficulty=000,

```
HelloWorld!0 : 3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb609106a
HelloWorld!1 : b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eefc22a27
HelloWorld!2 : 5b6fd9c27fcb54ca23404d9428f081b7c9280ba6370e33a6a20b16f40ce76320
HelloWorld!3 : 9c5d769416aa0ca894abf22bd17bd30fbb6959291423ae1903a9f86a1fe7ce78
....
HelloWorld!94 : 7090a0e5d88cff635e42ea33fcd6091a058e9cdd58ab8cd5c21c1c70421e35c6
```

string=HelloWorld !, nonce=0, difficulty=000,

```
HelloWorld!0 : 3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb609106a
HelloWorld!1 : b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eefc22a27
HelloWorld!2 : 5b6fd9c27fcb54ca23404d9428f081b7c9280ba6370e33a6a20b16f40ce76320
HelloWorld!3 : 9c5d769416aa0ca894abf22bd17bd30fbb6959291423ae1903a9f86a1fe7ce78
....
HelloWorld!94 : 7090a0e5d88cff635e42ea33fcd6091a058e9cdd58ab8cd5c21c1c70421e35c6
HelloWorld!95 : b74f3b2cf1061895f880a99d1d0249a8cedf223d3ed061150548aa6212c88d43
```

string=HelloWorld !, nonce=0, difficulty=000,

```
HelloWorld!0 : 3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb609106a
HelloWorld!1 : b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eefc22a27
HelloWorld!2 : 5b6fd9c27fcb54ca23404d9428f081b7c9280ba6370e33a6a20b16f40ce76320
HelloWorld!3 : 9c5d769416aa0ca894abf22bd17bd30fbb6959291423ae1903a9f86a1fe7ce78
....
HelloWorld!94 : 7090a0e5d88cff635e42ea33fcd6091a058e9cdd58ab8cd5c21c1c70421e35c6
HelloWorld!95 : b74f3b2cf1061895f880a99d1d0249a8cedf223d3ed061150548aa6212c88d43
HelloWorld!96 : 447ca2fa886965af084808d22116edde4383cbaa16fd1fbcf3db61421b9990b9
```

string=HelloWorld !, nonce=0, difficulty=000,

```
HelloWorld!0 : 3f6fc92516327a1cc4d3dca5ab2b27aeedf2d459a77fa06fd3c6b19fb609106a
HelloWorld!1 : b5690c48c2d0a09481186aaa99e4e090901ff2ac4d572e6706dfd30eefc22a27
HelloWorld!2 : 5b6fd9c27fcb54ca23404d9428f081b7c9280ba6370e33a6a20b16f40ce76320
HelloWorld!3 : 9c5d769416aa0ca894abf22bd17bd30fbb6959291423ae1903a9f86a1fe7ce78
....
HelloWorld!94 : 7090a0e5d88cff635e42ea33fcd6091a058e9cdd58ab8cd5c21c1c70421e35c6
HelloWorld!95 : b74f3b2cf1061895f880a99d1d0249a8cedf223d3ed061150548aa6212c88d43
HelloWorld!96 : 447ca2fa886965af084808d22116edde4383cbaa16fd1fbcf3db61421b9990b9
HelloWorld!97 : 000ba61ca46d1d317684925a0ef070e30193ff5fa6124aff76f513d96f49349d
```

here