Preventing the 51%-Attack: a Stochastic Analysis of Two Phase Proof of Work in Bitcoin

Gaspare Armato

Introduzione

- Perchè la necessità di una seconda *proof-of-work*?
- In che modo si modella il protocollo Bitcoin attraverso *metodi formali*?

Introduzione Bitcoin

- Cosa sono i Bitcoin?
- Cos'è un *public ledger*?
- Cos'è una *transazione*?
- Cos'è una *transazione scrypt*?
- Che *proof-of-work* utilizza il Bitcoin?

Difficoltà e Pooling

- La *difficoltà* è un valore globale che viene ricalcolato ogni 2016 blocchi e che mira ad un average block release time di 10 minuti. È definita come:

$$\frac{target\_1}{current\_target}$$

- Perchè nascono le mining pool?

51% Attack

- I *pool operators* possono eseguire particolari attacchi sulla rete sfruttando la notevole quantità di potenza computazionale posseduta.
- L'obbiettivo di questo attacco è la generazione di una *longest chain* che in un *mid/long term* si rivelerà vincente.

51% Attack

- Consideriamo due blockchain cha hanno almeno un predecessore in comune e che abbiano tali lunghezze:

$$n, m \in \mathbb{N} | n > m$$

- Entrambe le blockchain possono generare una blockchain di lunghezza:

$$k \in \mathbb{N} | p = d^{k-l}$$

Two phase proof-of-work

- Per prevenire una situazione simile a quella di GHash qualche anno fa, è stata proposta una soluzione che:
- Preserva la blockchain esistente.
- Preserva i saldi associati agli indirizzi bitcoin esistenti.
- Preserva i soldi investiti dai miner in attrezzature per il mining.

Two phase proof-of-work

- La seconda proof-of-work firma l'header di una transazione con la chiave privata, un numero a 256bit.
- Sia, da qui in poi, X la prima proof-of-work ed Y la seconda proof-of-work.

Two phase proof-of-work

- La chiave per un sistema sicuro in cui i grandi pool non possono superare il 50% della potenza computazionale totale ed i piccoli pool possono crescere, si ottiene variando la difficoltà dei puzzle crittografici.
- Sia

$$X = dcurrent, Y = \infty$$

allora un valore più alto di Y significa meno difficoltà.

Two phase proof-of-work

- Diminuendo Y otteniamo puzzle crittografici più difficili.
- Per cui, come si previene la crescita dei grandi pool attraverso il two phase proof-of-work in modo da evitare il 51% attack?

Two phase proof-of-work: Obbiettivo

- L'analisi seguente risponderà esattamente alla domanda:

<< Quali valori di X ed Y permettono ai piccoli pool di esistere e contemporaneamente non stimolano la crescita dei grandi pool? >>

Two phase proof-of-work: Ipotesi

- Verrà dunque dimostrata che la seguente ipotesi è valida:

<< Le azioni appartenenti al mercato di un pool descrescono in modo lineare non appena un pool è incapace di dedicare abbastanza potenza computazionale alle Y-challenge. >>

Two phase proof-of-work: Approccio

- Ogni transizione è probabilistica, il modello matematico più adeguato è quello di Markov.
- Cos'è una catena di Markov discreta?
- Proprietà di Markov:

$$\mathbb{P}(X_n = x_n \mid X_{n-1} = x_{n-1}, \ldots, X_0 = x_0) = \mathbb{P}(X_n = x_n \mid X_{n-1} = x_{n-1}).$$

Two phase proof-of-work: Approccio

- Cos'è una catena di Markov continua?
- Attraverso *PRISM* un software di model checking formale costruiremo una CDMC.
- Esempio: a: [0..2] init 0
- Costruisce una CDMC con tre stati per ogni possibile valore dell'intero a e con stato iniziale uguale a zero.

Two phase proof-of-work: Approccio

- Ad una transizione si possono aggiungere label, guardie, rates e modificatori:
- [label] guard → rate:modifier;
- Per esempio:
- [y_found] x = 0 → 0.2:(x'=3) esprime in linguaggio naturale: se x è uguale a zero allora si transita con rate di una volta ogni 5 secondi e dopo si modifica l'ambiente della variabile x aggiornando il suo valore con 3.

Two phase proof-of-work: Approccio

- La semantica di una label permette a PRISM di transire se e solo se tutti gli altri moduli possono transire usando la stessa label.
- Ogni modulo descrive un insieme di transizioni diverse ed è aggiunto allo spazio degli stati possibili attraverso la composizione parallela.

Two phase proof-of-work: Approccio

- Considerando tre moduli distinti con 2,5 e 7 stati rispettivamente allora il numero totale degli stati sarà 70.
- PRISM riduce lo spazio degli stati attraverso l'uso delle label.

Two phase proof-of-work: Approccio

```
module a
  x: [0..2] init 0;
  [label_a] x = 0 -> 0.2:(x'=1);
  [label_b] true -> (x'=2);
endmodule

module b
  y: [0..2] init 0;
  [label_b] y = 1 -> 0.2:(y'=2);
  [label_a] true -> (y'=1);
endmodule
```

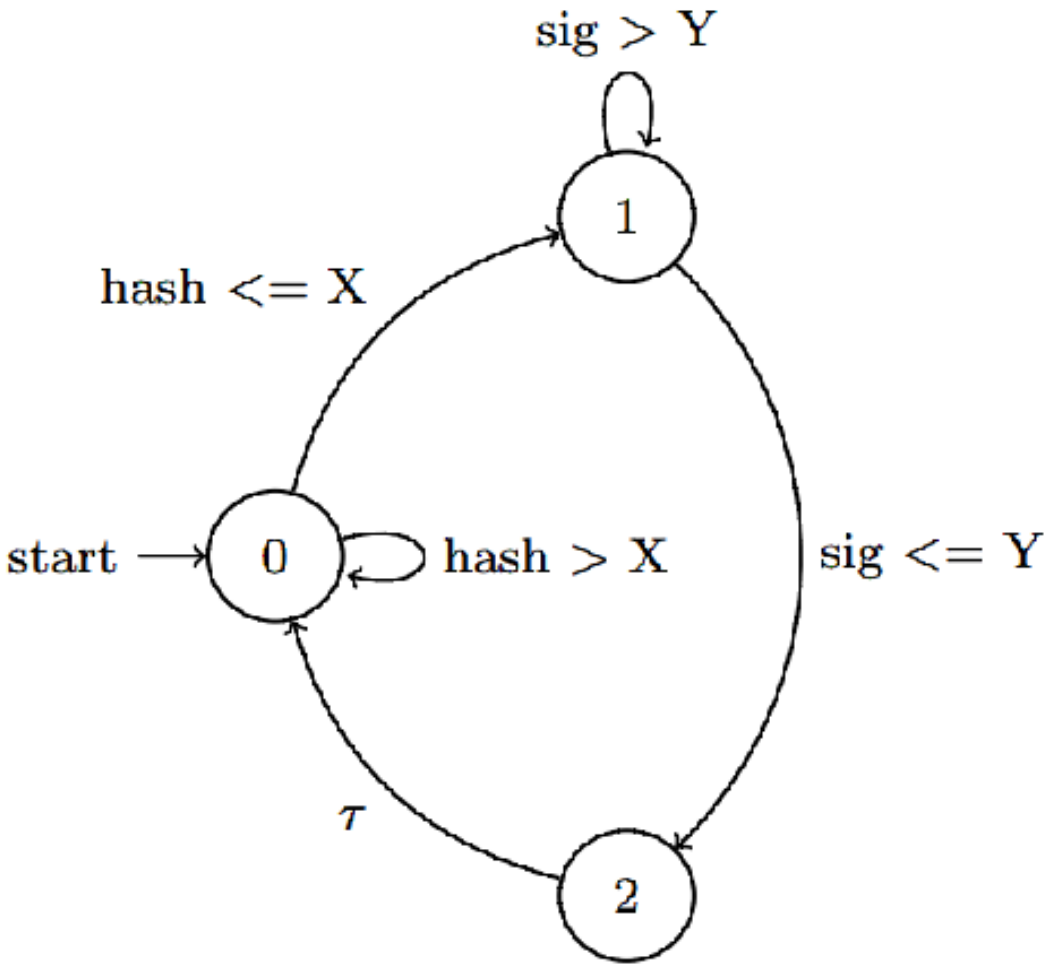
PRISM queries

- PRISM fornisce simulazioni a partire da modelli matematici ben formati e deriva conclusioni formali a partire da proprietà.
- Le due proprietà più importanti sono la S e la P.
- La proprietà S è usata per capire il *long term behavior* di una probabilità.
- P è usata per derivare la probabilità che un evento e occorra.

Modello

- Per analizzare le proprietà discusse precedentemente abbiamo bisogno di un accurato modello formale, il quale una volta definito permetta l'esecuzione di simulazioni.

Modello



Modello

- La figura precedente era una macchina a stati relativa ad un miner di Bitcoin.
- Un miner genera un hash combinando dati e nonce. Se non trova alcun match ci riprova cambiando nonce. Se trova il nonce si muove nello stato 1 ed esegue una Y-challenge.
- Dopo aver risolto la Y-challenge il miner viene ricompensato dalla rete.

Modello

- Un miner si troverà quindi nello stato 0 se non ha risolto alcun hash, nello stato 1 se ha risolto l'hash X e nello stato 2 se ha risolto la Y-challenge. Una volta raggiunto lo stato 2 il miner viene ricompensato.
- Ogni transizione ha dei rate collegati.

Modello

- La difficoltà target:

$$dtarget \in [target_1, \infty] | dtarget = \frac{0xffff * 2^{208}}{D}$$

- Dove D è la difficoltà corrente.
- Un target è un numero $\in [1, 2^{256} - 1]$ che indica il massimo valore valido assumibile da un hash SHA-256. Più basso è più difficile è.

Modello

- Un target1 è invece un valore usato per calcolare la difficoltà. È definito per essere uguale a:
- 0x00000000 seguito da 56 volte F.
- Viene espresso in forma troncata attraverso il bitshift operator:

$$0xFFFF << 52$$

- Nelle macchine viene semplicemente memorizzato in un floating point.

Modello

- La cardinalità degli hash attesi da un pool sarà dunque:

$$E(H) = \frac{2^{256}}{\frac{\texttt{0xffff} \cdot 2^{208}}{D}} = \frac{2^{256} D}{\texttt{0xffff} \cdot 2^{208}}.$$

Esempio

- Supponendo di avere i seguenti valori:

Scope	Property	Value
Network	Hashrate	$290.562.134\ \mathrm{GHz\,s^{-1}}$
Network	Difficulty	35.985.640.265
Pool A	Hashrate	$0.25 \cdot 290.562.134\ \mathrm{GHz\,s^{-1}}$
Pool B	Hashrate	$0.18 \cdot 290.562.134\ \mathrm{GHz\,s^{-1}}$

Esempio

- Allora vale:

$$E(H) = \frac{2^{256} \cdot 35.985.640.265}{\texttt{0xffff} \cdot 2^{208}} = 1.55 \cdot 10^{20}$$

- Per cui l'expected time con cui un pool trova un blocco si ottiene dividendo l'expected number degli hash totali con l'hash rate del pool.

Esempio

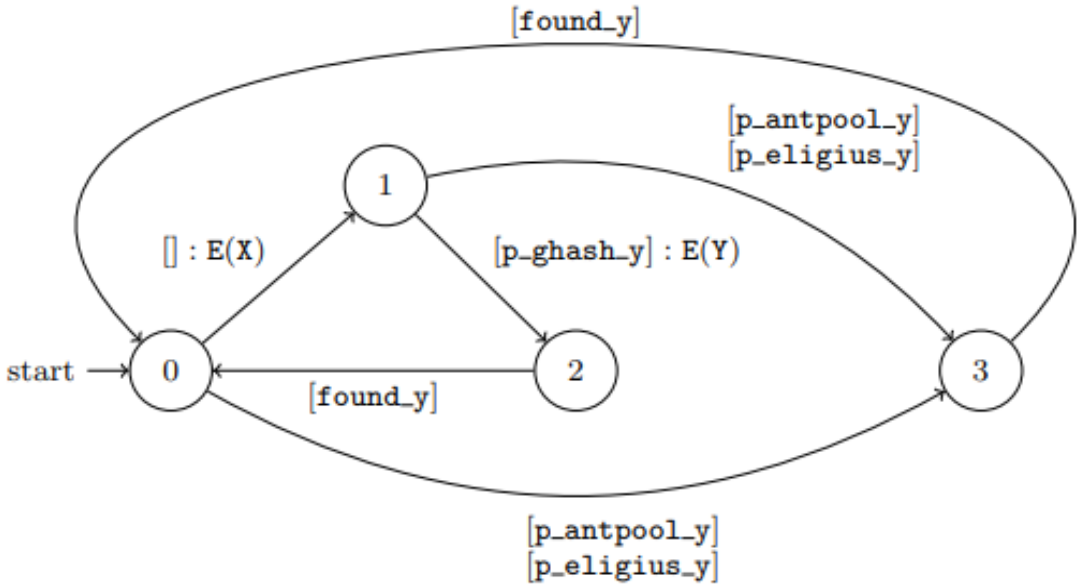
$$E(H_A) = \frac{1.55 \cdot 10^{20}}{0.25 \cdot 290.562.134 \cdot 10^9} = 2228\,\mathrm{s}$$

- Quindi il pool A trova un blocco ogni 37 minuti circa con un hashrate del 25% sul totale.

Composizione di pool

- Come si sincronizzano i pool?
- Se viene trovato un nuovo blocco tutti i miners dovrebbero lavorare sul più nuovo ed abbandonare il vecchio per diminuire la probabilità di block racing.
- Block racing = due miner che trovano lo stesso blocco in uno stesso tempo t.
- Nei modelli comuni questo viene ignorato poichè ha un'occorenza media di 1 volta in 10 giorni, nel modello qui presentato non si

Modello di sincronizzazione



- Il pool A trova la soluzione X con un certo rate a_x, ovvero il pool A transita dallo stato 0 allo stato 1 con un rate a_x.
- [] a_state = 0 → a_x:(a_state'=1);

Modello di sincronizzazione

- Il pool A si muove dallo stato 0 allo stato 1 se trova la soluzione X.
- Dallo stato 1 si può muovere nello stato 3 se qualcun'altro ha trovato la Y-solution oppure andare nello stato 2 se lui stesso ha trovato il blocco.
- Se un altro pool trova il blocco si muove comunque nello stato 3 e tutti gli altri pool si risincronizzano su found_y in modo da ripartire dallo stato iniziale.

Modello di sincronizzazione

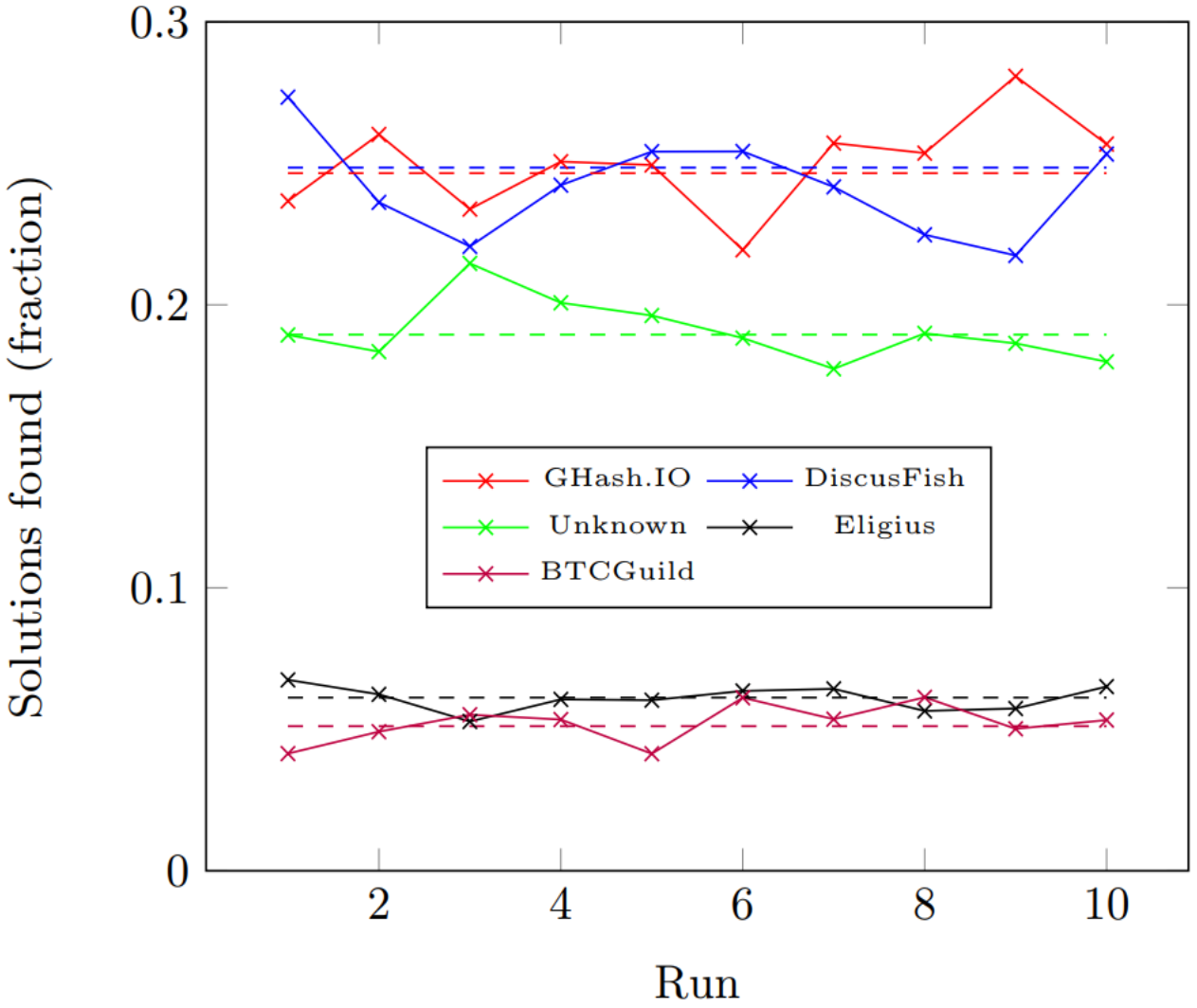
```
module a
   a_state : [0..3] init 0;

[found_y] a_state = 3 -> (a_state'=0);
[found_y] a_state = 2 -> (a_state'=0);
```

Analisi stocastica: equals Y-capacity

- La prima analisi di base è data da una run del modello descritto sopra.
- Ogni run è composta da 3000 step che corrispondono a circa 3 mesi di simulazione nel mondo reale.

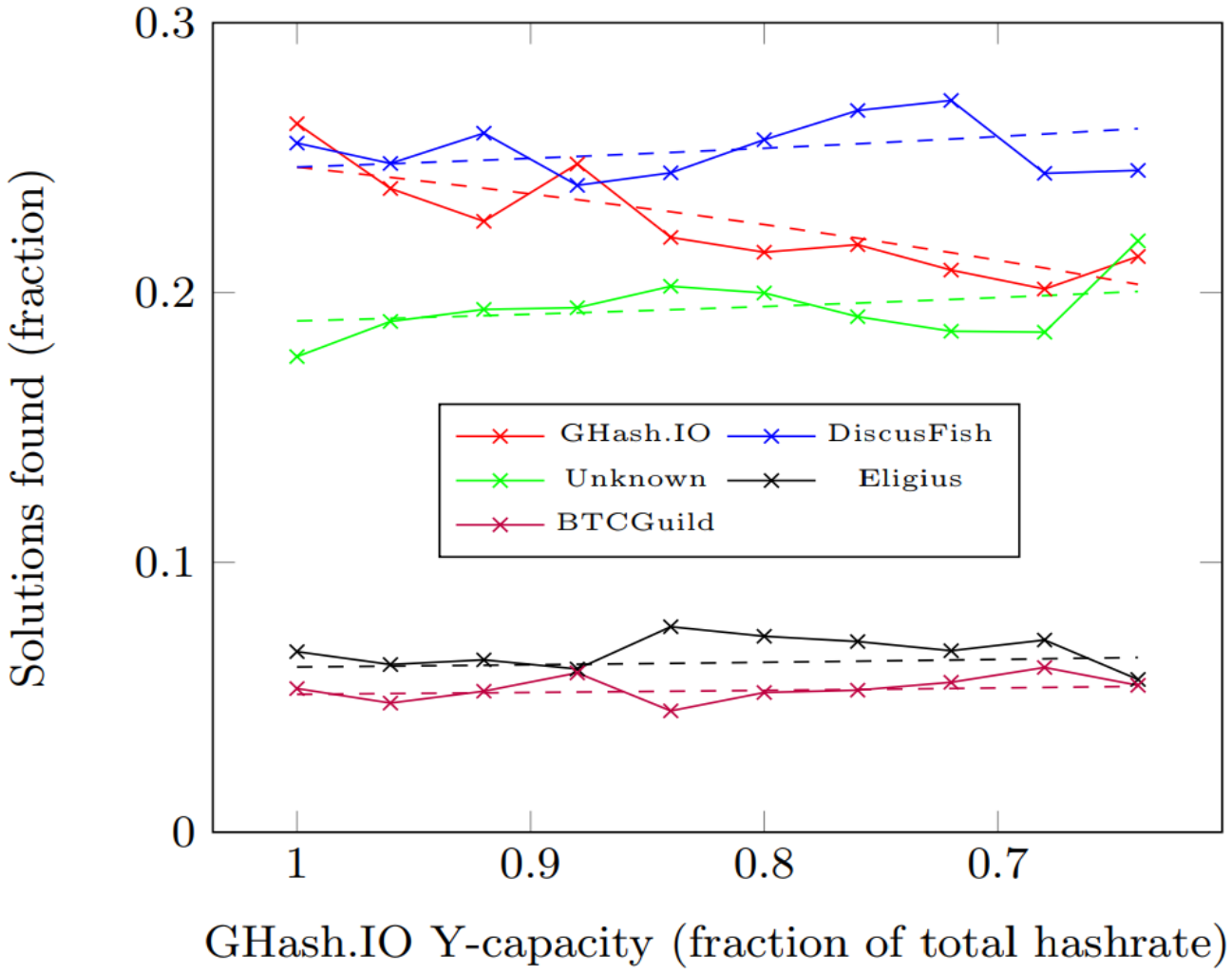
Analisi stocastica: equals Y-capacity



Analisi stocastica: equals Y-capacity

- La figura precedente mostra 10 run in cui tutti i pool hanno la stessa Y-solving capacity.
- Si nota che il rate delle soluzioni trovate è costante. Per cui l'introduzione della Y-challenge non influisce sul rate delle soluzioni trovate se tutti hanno la stessa Y-solving capacity.

Analisi stocastica: limiting Y-capacity



Analisi stocastica: limiting Y-capacity

- Avendo limitato la Y-capacity si vede come il rate delle soluzioni trovate decresce linearmente al diminuire della Y-capacity.

Conclusioni

- All'inizio si è parlato di ipotesi che andavano dimostrate per mettere in atto il two phase proof-of-work.
- Attraverso l'uso delle catene di markov continue è stato costruito un modello.
- L' ipotesi era che la decrescita delle azioni di un pool era lineare se si risolvevano poche Y-challenge(lo abbiamo notato dai grafici prodotti dalle simulazioni PRISM).

Conclusioni

- Questa ipotesi poteva anche essere dedotta in un altro modo:
- Si consideri un singolo minatore con

$$\alpha = Xrate, \sigma = Yrate$$

- Le challenge devono essere eseguite in serie poichè l'una dipende dall'altra.
- Il total rate sarà quindi uguale ad $\alpha * \sigma$
- Si nota come effettuando riduzioni lineari sul secondo coefficiente, il prodotto $\alpha * \sigma$ si riduce linearmente

Conclusioni

- Attraverso l'uso di PRISM, model checker formale, è stato quindi dimostrato che un 51% attack si può prevenire introducendo un two phase proof-of-work.