# Principles of Cryptocurrency Design

## Appunti ed Esercizi

Francesco Pasquale

13 marzo 2025

Sia $H : \mathbb{N} \cup \{0\} \to [n]$ una funzione tale che $H(0) = 1$ e per ogni $r \in \mathbb{N}$ si comporta come un *random oracle* [1]. A lezione abbiamo dimostrato che il protocollo seguente per Byzantine Broadcast soddisfa *Validity* con probabilità 1 e soddisfa *Consistency* con probabilità almeno $1 - (2/3)^{k-1}$.

**Algorithm 1** Randomized Byzantine Broadcast

1: ROUND 0. La sorgente (il nodo 1) riceve in input $b$ e inizializza $\mathtt{sb}_1 = b$.
2: Ogni nodo $i$ inizializza $\mathtt{sb}_i = \perp$.
3: PER OGNI ITERAZIONE $r = 0, \ldots, k-1$:
4: ROUND $3r$. Il *leader* $\ell_r = H(r)$ dell'iterazione:
5: Se $\mathtt{sb}_{\ell_r} \neq \perp$ allora invia a tutti $\mathtt{sb}_{\ell_r}$;
6: Altrimenti sceglie un bit $\{0, 1\}$ u.a.r. e lo invia a tutti.
7: ROUND $3r + 1$. Ogni nodo $i$:
8: Se $\mathtt{sb}_i \neq \perp$ allora invia a tutti $\mathtt{sb}_i$;
9: Altrimenti invia a tutti il bit ricevuto nel round $3r$ dal leader $\ell_r$ (se non ha ricevuto nulla da $\ell_r$ o ha ricevuto entrambi i bit, invia 0 o 1 arbitrariamente)
10: ROUND $3r + 2$. Ogni nodo $i$:
11: Se c'è un bit $\hat{b}$ che ha ricevuto nel round $3r + 1$ da almeno $2n/3$ nodi distinti allora imposta $\mathtt{sb}_i = \hat{b}$;
12: Altrimenti imposta $\mathtt{sb}_i = \perp$.
13: ROUND $3k$. Ogni nodo $i$:
14: OUTPUT $\mathtt{sb}_i$



**Teorema 1.** Se il numero di nodi corrotti è $f < n/3$ allora il protocollo in Algorithm 1 soddisfa *Validity* con probabilità 1 e *Consistency* con probabilità almeno $1 - (2/3)^k$.

*Sketch of Proof.* Per *Validty* osservare che, se la sorgente è onesta, nell'iterazione $r = 0$ succede questo: nel ROUND 0 tutti i nodi onesti ricevono dalla sorgente il bit $b$ (linea 5); nel ROUND 1 ogni nodo onesto invia $b$ a tutti (linea 9); nel ROUND 2 ogni nodo onesto riceve $b$ da almeno $2n/3$ nodi e quindi fissa il suo $\mathtt{sb}_i = b$. Nelle iterazioni successive $r = 1, \ldots, k-1$ ogni nodo onesto $i$ continua a inviare $\mathtt{sb}_i = b$ (linea 8) e quindi il valore di $\mathtt{sb}_i$ rimane $b$ (linea 11). Per *Consistency* la dimostrazione procede come segue:

1. Dimostrare che in ogni iterazione non possono esserci due nodi onesti $i$ e $j$ tali che $\mathtt{sb}_i = 0$ e $\mathtt{sb}_j = 1$;
2. Definire una iterazione $r$ *lucky* se si verifica che (i) il leader $\ell_r$ è onesto e il bit inviato dal leader (linee 5-6) è lo stesso posseduto dagli altri nodi onesti $i$ che hanno $\mathtt{sb}_i \neq \perp$ all'inizio dell'iterazione $r$;

3. Osservare che, se esiste una iterazione *lucky* allora tutti i nodi onesti danno in output lo stesso bit;
4. Dimostrare che, qualunque cosa sia successa nelle iterazioni precedenti, una iterazione $r$ è *lucky* con probabilità almeno $1/3$;
5. Osservare che la probabilità che fra le $k$ iterazioni nemmeno una sia *lucky* quindi è minore di $(2/3)^k$.

□

**Esercizio 1.** Si consideri il protocollo in Algorithm 1. Dato $\delta > 0$, quanto deve essere grande il numero di iterazioni $k = k(\delta)$ affinché la probabilità che l'algoritmo soddifi la condizione di *consistency* sia almeno $1 - \delta$? Quanto deve essere grande $k$ se scegliamo $\delta = 1/n$? Confrontare il numero di round di questo protocollo con quello del protocollo di Dolev-Strong.

**Esercizio 2.** Alla linea 11 il protocollo stabilisce che ogni nodo $i$ deve decidere come impostare il bit $\texttt{sb}_i$ a seconda che abbia ricevuto o no almeno $2n/3$ "voti" per uno specifico bit $\hat{b}$ nel round precedente.

1. Mostrare un attacco che i nodi corrotti potrebbero mettere in atto se la decisione fosse presa in funzione di un numero di voti $h < 2n/3$;

2. Mostrare un attacco che i nodi corrotti potrebbero mettere in atto se la decisione fosse presa in funzione di un numero di voti $h > 2n/3$.

**Esercizio 3.** Cosa succederebbe se non imponessimo che $H(0) = 1$ (ossia che il leader $\ell_0$ dell'iterazione $r = 0$ è il nodo sorgente)?

## Riferimenti bibliografici

[1] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993. `https://dl.acm.org/doi/pdf/10.1145/168588.168596`.