Principles of Cryptocurrency Design

Esercitazione

Francesco Pasquale

9 aprile 2024

1 Il protocollo di Dolev-Strong

Consideriamo il protocollo di Dolev-Strong per il problema Byzantine Broadcast.

Algorithm 1 Dolev-Strong Protocol for Byzantine Broadcast

1: ROUND 0. Ogni nodo i inizializza un insieme vuoto $\mathcal{E}_i = \emptyset$.
2: La sorgente (il nodo 1) riceve in input b e invia $\langle b \rangle_1$ a tutti i nodi.
3: PER OGNI ROUND $r = 1, \ldots, f$. Ogni nodo i:
4: Per ogni messaggio $\langle \hat{b} \rangle_{1,j_1,j_2,\ldots,j_{r-1}}$ con r firme distinte (inclusa quella
5: della sorgente) ricevuto nel Round $r-1$:
6: SE $\hat{b} \notin \mathcal{E}_i$:
7: Aggiungi $\hat{b}$ a $\mathcal{E}_i$;
8: Firma il messaggio $\langle \hat{b} \rangle_{1,j_1,j_2,\ldots,j_{r-1}}$;
9: Invia il messaggio firmato $\langle \hat{b} \rangle_{1,j_1,j_2,\ldots,j_{r-1},i}$ a tutti i nodi;
10: ROUND $f+1$. Ogni nodo i:
11: Per ogni messaggio $\langle \hat{b} \rangle_{1,j_1,j_2,\ldots,j_f}$ con $f+1$ firme distinte (inclusa quella
12: della sorgente) ricevuto nel Round f:
13: SE $\hat{b} \notin \mathcal{E}_i$, aggiungi $\hat{b}$ a $\mathcal{E}_i$;
14: SE $\mathcal{E}_i$ contiene un solo elemento, allora OUTPUT l'elemento in $\mathcal{E}_i$,
15: Altrimenti OUTPUT 0

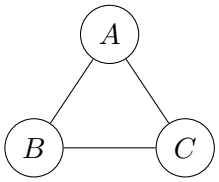
Esercizio 1. Alle linee 4 e 11 del protocollo ogni nodo onesto verifica, oltre al *numero* di firme distinte presenti (r firme per messaggi arrivati nel round $r-1$), che fra le firme ci sia anche la firma della sorgente. Descrivere un attacco al protocollo che i nodi corrotti potrebbero mettere in atto se i nodi onesti non verificassero la presenza della firma della sorgente.

Esercizio 2. Supponete che c'è un *bug* nell'implementazione del sistema di firme digitali usato nel protocollo che consente a un nodo corrotto di falsificare le firme dei nodi onesti. Descrivere un attacco al protocollo che i nodi corrotti possono mettere in atto in questo caso.

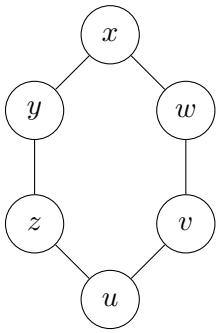
2 Lower bound per Byzantine Broadcast senza PKI

Esercizio 3. Considerate i sistemi G e H qui di seguito.

Sistema G



Sistema H



1. Eseguire il protocollo di Dolev-Strong con $f = 1$ nel sistema G, quando tutti i nodi sono onesti, A è il nodo sorgente e riceve in input 1.

2. Eseguire il protocollo di Dolev-Strong con $f = 1$ nel sistema H in cui x e u sono copie esatte[1] di A e ricevono in input 1, y e v sono copie esatte di B e z e w sono copie esatte di C (tutti i nodi sono onesti e seguono il protocollo).

3. Osservare che le esecuzioni dei due punti precedenti sono equivalenti (A, B e C non possono distinguere se sono nel sistema G o nel sistema H).

4. Eseguire il protocollo di Dolev-Strong con $f = 1$ nel sistema H in cui x e u sono copie esatte di A, ma alla copia di A in x viene dato in input 1 mentre alla copia di A in u viene dato in input 0; y e v sono copie esatte di B e z e w sono copie esatte di C (tutti i nodi sono onesti e seguono il protocollo). Quali sono gli output dei sei nodi?

5. È possibile simulare l'esecuzione in H del punto precedente con una esecuzione equivalente nel sistema G in cui

   (a) B e C sono nodi onesti e A è un nodo corrotto che simula il comportamento dei nodi x, u, v e w?

   (b) A e B sono nodi onesti e C è un nodo corrotto che simula il comportamento dei nodi z, u, v e w?

Esercizio 4. A lezione abbiamo dimostrato che, in assenza di PKI non esistono protocolli per il problema Byzantine Broadcast che soddisfano *validity* e *consistency* per tre nodi, se almeno uno dei tre è corrotto.

Generalizzare la dimostrazione al caso di un numero di nodi arbitrario: dato un n qualunque, in assenza di PKI nessun protocollo per Byzantine Broadcast può soddisfare *validity* e *consistency*, se almeno $n/3$ dei nodi sono corrotti.

3 Randomized Byzantine Broadcast senza PKI

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*. A lezione abbiamo dimostrato che il protocollo seguente per Byzantine Broadcast soddisfa *Validity* con probabilità 1 e soddisfa *Consistency* con probabilità almeno $1 - (2/3)^{k-1}$.

[1] Ossia entrambe hanno la stessa chiave privata

Algorithm 2 Randomized Byzantine Broadcast

1: ROUND 0. La sorgente (il nodo 1) riceve in input b e inizializza $\text{sb}_i = b$.
2: Ogni nodo i inizializza $\text{sb}_i = \perp$.
3: PER OGNI ITERAZIONE $r = 0, \ldots, k-1$:
4: ROUND $3r$. Il *leader* $\ell_r = H(r)$ dell'iterazione:
5: Se $\text{sb}_{\ell_r} \neq \perp$ allora invia a tutti sb_{ℓ_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 $\text{sb}_i \neq \perp$ allora invia a tutti sb_i;
9: Altrimenti invia a tutti il bit ricevuto nel round $3r$ dal leader ℓ_r (se non ha ricevuto nulla da ℓ_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 $\text{sb}_i = \hat{b}$;
12: Altrimenti imposta $\text{sb}_i = \perp$.
13: ROUND $3k$. Ogni nodo i:
14: OUTPUT sb_i

Esercizio 5. Si consideri il protocollo in Algorithm 2. 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 6. Alla linea 11 il protocollo stabilisce che ogni nodo i deve decidere come impostare il bit 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 7. Cosa succederebbe se non imponessimo che $H(0) = 1$ (ossia che il leader ℓ_0 dell'iterazione $r = 0$ è il nodo sorgente)?

4 Byzantine Broadcast e State Machine Replication

Si consideri un sistema distribuito con n nodi in cui ogni nodo può eseguire computazioni locali arbitrarie e inviare messaggi di lunghezza arbitraria a ogni altro nodo. Degli n nodi, f sono *corrotti* e $n - f$ nodi sono *onesti*. Assumiamo di essere in un sistema *sincrono* in cui c'è un *global clock* noto a tutti i nodi che scandisce il tempo in *round* discreti e ogni messaggio inviato in un round t arriva a destinazione prima dell'inizio del round $t + 1$.

Consideriamo il problema seguente, che chiamiamo *State Machine Replication*: ad ogni nodo $i \in [n]$, in ogni round $r \in \mathbb{N}$, può essere affidata una o più *transazioni* `tx` (stringhe binarie). Ogni nodo i mantiene un *log* che consiste in una concatenazione di transazioni, indichiamo con LOG_i^r il log del nodo i al round r. Vogliamo progettare un protocollo che faccia in modo che l'evoluzione dei log dei nodi nel tempo soddifi le due proprietà seguenti

- **Consistency**: Per ogni $i, j \in [n]$ e per ogni coppia di round $r, s \in \mathbb{N}$, se i e j sono nodi onesti allora $\text{LOG}_i^r \preccurlyeq \text{LOG}_j^s$ oppure $\text{LOG}_j^s \preccurlyeq \text{LOG}_i^r$, dove con la notazione $\text{LOG} \preccurlyeq \text{LOG}'$ intendiamo che LOG è un prefisso di LOG$'$.
- **Liveness**: Esiste un *confirmation time* $T_{\text{conf}} \in \mathbb{N}$ tale che, se una transazione `tx` viene affidata a un nodo onesto in un round $r \in \mathbb{N}$, allora per ogni nodo onesto $i \in [n]$, `tx` $\in \text{LOG}_i^{r+T_{\text{conf}}}$.

Esercizio 8. Mostrare che se abbiamo un protocollo Π_{BB} che risolve *Byzantine Broadcast* in R round, allora possiamo progettare un protocollo Π_{SMR} per *State Machine Replication* con confirmation time $T_{\text{conf}} = \mathcal{O}(nR)$.

5 Byzantine Agreement

Nel problema *Byzantine Agreement (BA)* abbiamo n nodi di cui f sono corrotti. A ogni nodo i viene affidato in input un messaggio b_i. Vogliamo progettare un protocollo, che termini in un tempo finito, in cui ogni nodo onesto i dia in output un valore y_i tale che

- **Consistency**: Se i e j sono due nodi onesti, allora $y_i = y_j$;
- **Validity**: Se tutti i nodi onesti hanno in input lo stesso bit b, allora $y_i = b$ per ogni nodo onesto i.

Esercizio 9. Modificando opportunamente il protocollo di Dolev-Strong per il problema Byzantine Broadcast, mostrare che nel modello sincrono con PKI esiste un protocollo per Byzantine Agreement che soddisfa consistency e validity qualunque sia il numero di nodi corrotti.

Esercizio 10. Modificando opportunamente la dimostrazione di impossibilità del problema Byzantine Broadcast con $n/3$ o più nodi, mostrare che nel modello sincrono senza PKI non esiste nessun protocollo per Byzantine Agreement che soddisfa consistency e validity in presenza di $n/3$ o più nodi corrotti.

6 Funzioni Hash Crittografiche

Con il programma seguente (o con qualunque altra implementazione della funzione hash crittografica sha256) potete verificare che l'hash espresso in esadecimale della stringa *Francesco 16114492071* inizia con una sequenza di 9 zeri.

```
from hashlib import sha256

nonce = '16114492071'
text = 'Francesco ' + nonce

print(sha256(text.encode('utf8')).hexdigest())
```

Esercizio 11. Scrivere un programma che trovi una stringa `<nonce>` tale che l'hash della stringa `<nome> <nonce>`, per il vostro `<nome>` inizi con una sequenza di 9 zeri. Stimare il *running time* del programma ed eseguirlo.