Principles of Cryptocurrency Design

Appunti ed Esercizi

Francesco Pasquale

22 maggio 2025

Nella nota precedente abbiamo visto come sia possibile in Bitcoin implementare dei *canali di pagamento*: due agenti, Alice e Bob, possono collaborativamente inserire nella Blockchain una transazione, la *funding transaction*, con un output spendibile solo da una successiva transazione che sia firmata da entrambi, e allo stesso tempo entrambi gli agenti possono mantenere *off-chain* delle transazioni di *committment* con le quali potrebbero spendere unilateralmente l'output della *funding transaction*, se l'altro agente dovesse smettere di collaborare. Usando gli opportuni protocolli crittografici, abbiamo visto come sia possibile costruire nuove *committment transactions* e "revocare" le precedenti, in modo che il *bilancio* del canale (quanti *btc* dell'output della funding transaction sono di Alice e quanti sono di Bob) possa essere aggiornato un numero arbitrario di volte, senza inviare nuove transazioni alla rete Bitcoin, dove l'unico limite al valore dei pagamenti è dato dall'ammontare complessivo di *btc* contenuto nell'output della *funding transaction*.

In questa nota vediamo come sia possibile implementare un sistema di *routing* di pagamenti all'interno di una "rete di canali": se Alice e Bob hanno un canale di pagamento e Bob e Carol hanno un canale di pagamento, allora Alice può chiedere a Bob di fare da intermediario per il suo pagamento a Carol: Alice pagherà Bob (vale a dire, ci sarà un aggiornamento del bilancio del canale fra Alice e Bob) e Bob pagherà Carol (vale a dire, ci sarà un aggiornamento del canale fra Bob e Carol).

1 Hash Time-Locked Contracts

In prima approssimazione, il meccanismo che nella Lightning Network garantisce che il *routing* del pagamento - da Alice a Carol passando per Bob come intermediario - sia eseguito in modo *trustless* è il seguente:

1. Carol genera un valore qualunque r, ne calcola l'hash $h = H(r)$ e dà h ad Alice;
2. Alice construisce una transazione con un output che può essere speso da Bob, se Bob esibisce una preimmagine di h;
3. Bob a sua volta costruisce una trasazione con un output che può essere speso da Carol, se Carol esibisce una preimmagine di h.

Siccome Carol conosce una preimmagine r di h, può inviarla a Bob e chiedergli di aggiornare il bilancio del loro canale. Se Bob non dovesse collaborare, Carol potrebbe spendere il suo output inviandolo sulla Blockchain. A sua volta Bob, una volta venuto a conoscenza della preimmagine r di h (o perché l'ha ricevuta da Carol, o perché Carol ha speso il suo output inviandolo sulla Blockchain e quindi rivelando r a tutti) può inviare r ad Alice e chiederle di aggiornare il bilancio del loro canale. Se Alice non dovesse collaborare, Bob spenderebbe il suo output e otterrebbe comunque il pagamento. Questo conclude il pagamento da Alice e Carol.

Esercizio 1. Si noti che il pagamento da Alice a Carol, anche se coinvolge due aggiornamenti di bilancio, è *atomico*: o entrambi gli aggiornamenti dei bilanci dei due canali possono andare a buon fine, oppure nessuno dei due può andarci.

Esercizio 2. Usando gli *opcodes* opportuni, progettare un `locking_script` che implementi la condizione per cui il corrispondente `unlocking_script` sia una firma valida relativa a una chiave pubblica `pk` e la preimmagine (diciamo rispetto a HASH160) un certo h.

Il meccanismo descritto nei punti 1,2,3 all'inizio di questa sezione ha il problema che, se per qualunque ragione Carol non dovesse rivelare r, Alice e Bob si troverebbero in una situazione di stallo in cui non potrebbero tornare in possesso dell'ammontare che hanno bloccato nell'output della transazione emessa. Per questo motivo, nei punti 2 e 3 Alice e Bob inseriscono un *time-lock*, ossia una condizione alternativa che gli consente di tornare in possesso dell'output nel momento in cui il numero di blocchi nella Blockchain raggiunge un valore prefissato x. Il tempo che intercorre fra il blocco corrente e il blocco x è quanto Alice e Bob dovranno aspettare prima di poter tornare in possesso in modo sicuro di quano avevano bloccato per il pagamento a Carol, se Carol dovesse non rivelare mai la preimmagine r.

1. Carol genera un valore qualunque r, ne calcola l'hash $h = H(r)$ e dà h ad Alice;
2. Alice construisce una transazione con un output che può essere speso da
   - Bob, se Bob esibisce una preimmagine di h;
   - Alice, se la Blockchain ha almeno x blocchi, dove x è un numero sufficientemente maggiore del valore corrente del numero di blocchi della Blockchain.
3. Bob a sua volta costruisce una trasazione con un output che può essere speso da
   - Carol, se Carol esibisce una preimmagine di h;
   - Bob, se la Blockchain ha almeno $x - \Delta$ blocchi, dove Δ è un numero piccolo (e.g., 6).

Esercizio 3. Riflettere sul perché se il *locktime* di Alice è x, quello di Bob deve essere di qualche unità inferiore $x - \Delta$.