

Principles of Cryptocurrency Design

Appunti ed Esercizi

Francesco Pasquale

19 maggio 2025

Il sistema Bitcoin è *by design* poco scalabile, nel senso che il limite imposto dal meccanismo di consenso sulla dimensione dei blocchi e sul tempo medio che intercorre fra la creazione di due blocchi consecutivi fa sì che il numero massimo di transazioni che possono essere incluse nella blockchain è molto piccolo (meno di una decina di transazioni per secondo), se confrontato con altri sistemi di pagamento elettronico centralizzati.

Dall'introduzione di Bitcoin sono stati proposti diverse soluzioni per ovviare a questa limitazione intrinseca [1], che hanno avuto più o meno successo. Quella di cui ci occupiamo oggi e nelle prossime lezioni è quella che consiste nella costruzione di una *rete di canali di pagamento* [3, 2] e nello specifico delle tecniche usate per progettare e implementare la Lightning Network [4] (per una implementazione si veda <https://github.com/lightningnetwork/lnd>).

Per comprendere il contenuto di questa nota è necessario avere ben chiara la struttura e il funzionamento delle transazioni Bitcoin.

1 Canali di pagamento: *Funding* e *Refunding* transactions

Supponiamo che *Alice* abbia *10btc* in un output non speso di una certa transazione con id *b8ff...11f0*

b8ff...11f0	
INPUT:	
OUTPUT:	10btc, spendibili da pkA0

(1)

dove *pkA0* è una chiave pubblica di cui lei ha la corrispondente chiave segreta *skA0*, e decida di costruire la seguente transazione, che chiamiamo *funding transaction* per ragioni che saranno chiare a breve, che spende quei *10btc*,

a5ba...34f9 (<i>FundingTx</i>)	
INPUT:	b8ff...11f0
OUTPUT:	10btc, spendibili da pkA & pkB

(2)

dove *pkA* è una chiave pubblica di cui Alice ha la corrispondente chiave segreta *skA*, mentre *pkB* è una chiave pubblica di cui un altro soggetto, *Bob*, ha la corrispondente chiave segreta *skB*.

Se Alice firmasse questa *FundingTx* in (2) con la chiave segreta *skA0* - quella autorizzata a spendere l'output nella transazione in (1) - e la inviasse alla rete Bitcoin, successivamente per spendere l'output contenuto nella transazione (2) Alice avrebbe bisogno della collaborazione di Bob, perché bisognerebbe inviare alla rete una transazione firmata sia con *skA*, che è nota solo ad Alice, sia con *skB*, che è nota solo a Bob. Quindi Alice starebbe bloccando i suoi *10btc* in un output che non potrebbe più spendere senza la collaborazione di Bob.

Prima di inviare la *FundingTx* in (2), Alice però genera una nuova coppia di chiavi *skAr0* e *pkAr0* e costruisce anche la transazione seguente, che chiamiamo *refunding transaction* per ragioni che dovrebbero cominciare a chiarirsi, e chiede a Bob di firmarla con la sua *skB*

e941...05c0 (<i>RefundingTX</i>)	
INPUT:	a5ba...34f9
OUTPUT:	10btc, spendibili da pkA, dopo n blocchi oppure da pkAr0 & pkB, subito

(3)

La transazione in (3) spende l'output della transazione in (2), che Alice non ha ancora inviato alla rete Bitcoin, e ha un unico output che è spendibile o da Alice, n blocchi dopo che la transazione sia stata inserita nella Blockchain, oppure da chi è conosce **skAr0** (che per il momento conosce solo Alice) e **skB** (che conosce solo Bob).

Esercizio 1. Usando gli *opcodes* opportuni, progettare un `locking_script` che implementi la condizione descritta nell'output della transazione (3).

Osservate che nel momento in cui Alice ha in mano la *refunding transaction* in (3) firmata da Bob con **skB**, può tranquillamente inviare alla rete Bitcoin la *funding transaction* in (2): Se Bob non dovesse collaborare nel momento in cui Alice volesse spendere i 10btc nell'output della *funding transaction* in (2), Alice potrebbe firmare la *refunding transaction* in (3) anche con **skA1** e inviarla alla rete Bitcoin, aspettare n blocchi dopo che la transazione sia stata inserita nella Blockchain e poi spenderne l'output.

L'invio della *funding transaction* alla rete Bitcoin costituisce l'*apertura di un canale di pagamento* fra Alice e Bob, finanziato con 10btc da Alice. L'esistenza della *refunding transaction* nelle mani di Alice con la firma di Bob è la garanzia, per Alice, che può tornare in possesso esclusivo dei suoi 10btc in qualunque momento. La condizione per spendere l'output della *refunding transaction* relativa alle chiavi pubbliche **pkAr0** e **pkB** fa sì che la transazione sia, di fatto, *revocabile* da Alice; nella prossima sezione vedremo come, e perché questo è necessario..

2 Canali di pagamento: Aggiornamenti del bilancio del canale

Supponiamo che Alice voglia acquistare da Bob qualcosa del valore di 2btc. Vediamo come è possibile fare in modo di modificare il *bilancio* del canale fra Alice e Bob, che attualmente è 10 per Alice e 0 per Bob, in 8 per Alice e 2 per Bob.

Alice e Bob generano due nuove coppie di chiavi, rispettivamente (**skAr1**, **pkAr1**) viene generata da Alice e (**skBr1**, **pkBr1**) viene generata da Bob, e costruiscono le due transazioni seguenti, che chiamiamo *Commit Transaction 1*. Entrambe le transazioni sono progettate per spendere l'output della *funding transaction* in (2).

5e21...b39d (<i>CommitTx1 - Alice</i>)	
INPUT:	a5ba...34f9
OUTPUT 1:	8btc, spendibili da pkA, dopo n blocchi oppure da pkAr1 & pkB, subito
OUTPUT 2:	2btc, spendibili da pkB

c401...7a61 (<i>CommitTx1 - Bob</i>)	
INPUT:	a5ba...34f9
OUTPUT 1:	8btc, spendibili da pkA
OUTPUT 2:	2btc, spendibili da pkB, dopo n blocchi oppure pkA & pkBr1, subito

La transazione *CommitTx1 - Alice* è nelle mani di Alice e deve essere firmata da Bob con **skB**, la transazione *CommitTx1 - Bob* è nelle mani di Bob e deve essere firmata da Alice con **skA**. Si osservi che in questo modo sia Alice che Bob hanno in mano una transazione che potrebbero unilateralmente firmare e inviare alla rete Bitcoin, la transazione creerebbe due output, di cui 8btc potrebbero essere spesi da Alice e 2btc da Bob. Se Alice inviasse la sua *CommitTx1 - Alice*, allora Alice dovrebbe aspettare n blocchi prima di poter spendere i suoi 8btc, mentre Bob potrebbe spendere i suoi 2btc subito. Viceversa, se fosse Bob a inviare la sua *CommitTx1 - Bob* alla rete, Bob

dovrebbe aspettare n blocchi prima di poter spendere i suoi $2btc$, mentre Alice potrebbe spendere i suoi $8btc$ subito.

Si osservi che Alice ha ancora in mano la *RefundingTx* in (3) firmata da Bob che assegna $10btc$ ad Alice e 0 a Bob. Cosa impedisce ad Alice di inviare questa transazione alla rete Bitcoin? Qui entra in gioco il ruolo della seconda condizione dell'output: prima che l'update del bilancio da Alice:10 - Bob:0 a Alice:8 - Bob:2 si possa considerare concluso, Alice deve rivelare a Bob la chiave segreta $skAr0$ (quella che corrisponde alla chiave pubblica contenuta nell'output delle *RefundingTx*). Se Bob conosce $skAr0$, Alice non può più inviare la *RefundingTx* in (3) alla rete Bitcoin, perché se lo facesse dovrebbe aspettare n blocchi prima di poter spendere i $10btc$ nell'output, ma nel frattempo Bob potrebbe immediatamente spendere quei 10 btc inviandoli a una chiave sotto il suo controllo esclusivo e lasciando Alice senza nulla.

Esercizio 2. Riflettere con attenzione su qual è l'ordine con cui devono avvenire questi passaggi nell'update del canale fra Alice e Bob, e quale delle due parti potrebbe approfittarne e in che modo se gli scambi di informazione avvenissero in ordine diverso

- L'invio da Alice a Bob della firma di Alice della *CommitTx1* - Bob;
- L'invio da Bob ad Alice della firma di Bob della *CommitTx1* - Alice;
- L'invio da Alice a Bob della chiave segreta $skAr1$.

Esercizio 3. Supponiamo che successivamente all'update relativo alle transazioni *CommitTx1*, Bob debba fare un pagamento ad Alice di $1btc$. Spiegare i passaggi necessari ad aggiornare nuovamente il bilancio del canale a Alice:9 - Bob:1.

Si noti che dal punto di vista della rete Bitcoin, l'unica transazione visibile è la *FundingTx* che deve essere inserita nella Blockchain e segna l'apertura del canale. Tutti gli altri passaggi intermedi, finché una delle due parti o entrambe non decidono di *chiudere* il canale effettuando una transazione che spende l'output della *FundingTx*, restano scambi di messaggi privati fra le due parti.

Riferimenti bibliografici

- [1] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains: (a position paper). In *International conference on financial cryptography and data security*, pages 106–125. Springer, 2016.
- [2] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Stabilization, Safety, and Security of Distributed Systems: 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings 17*, pages 3–18. Springer, 2015. <https://tik-old.ee.ethz.ch/file/716b955c130e6c703fac336ea17b1670/duplex-micropaymentchannels.pdf>.
- [3] Mike Hearn and Jeremy Spilman. Bitcoin contracts. <https://en.bitcoin.it/wiki/Contracts>, 2012.
- [4] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. <https://lightning.network/lightning-network-paper.pdf>, 2016.