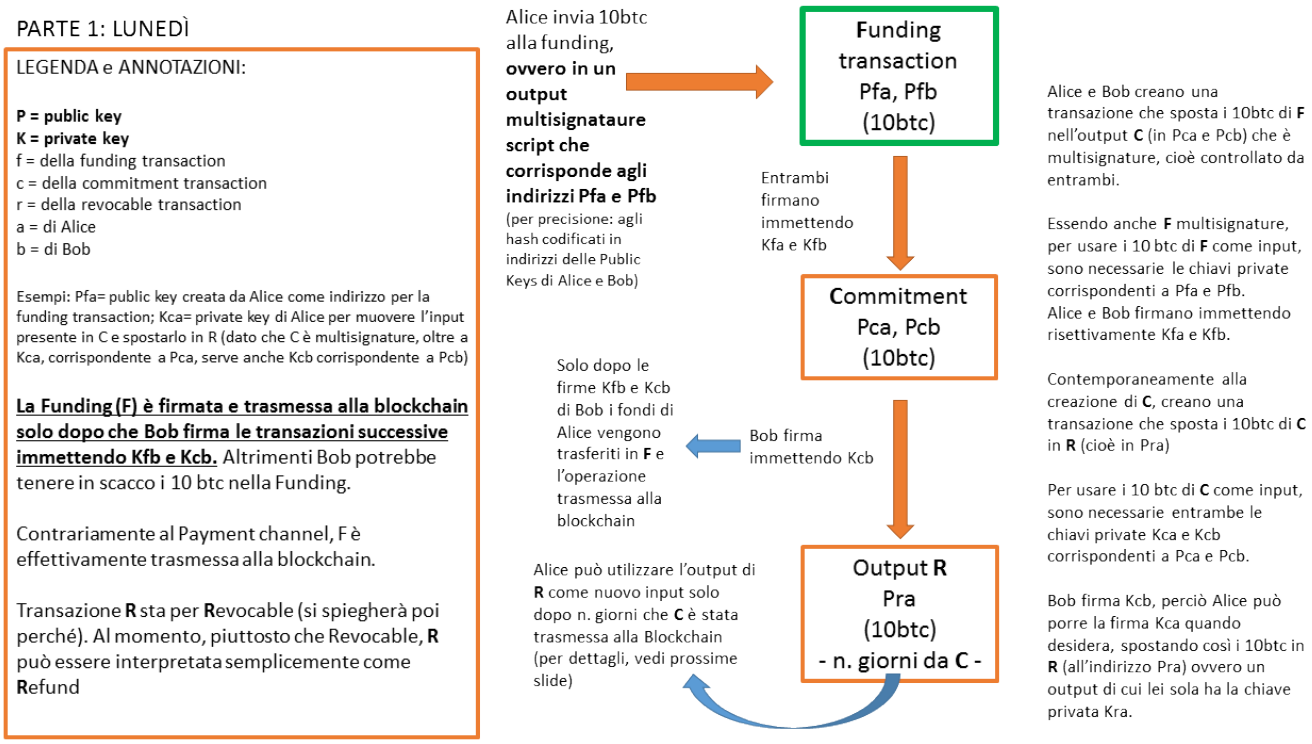
Lightning Network Parte 2

scritto da Alberto De Luigi | 30 Maggio 2016

< Torna alla Parte 1 Vai alla Parte 3 >

Rapporti a tempo indeterminato: il channel può rimanere aperto all'infinito e non richiede intermediari



F è la transazione «madre», C è figlia di F e R è figlia di C.

Dopo che C e R sono state create e firmate da Bob, F viene trasmessa alla Blockchain. Alice non la trasmette prima poiché perderebbe i propri bitcoin – o meglio – Bob potrebbe tenerli sotto scacco.

Dato che Bob ha già firmato inserendo le proprie chiavi private di F e di C, in qualunque momento Alice può firmare e trasmettere alla Blockchain C e R.

Affinché una transazione «figlia» venga creata nonostante

l’input presente nella «madre» non sia ancora stato firmato e trasmesso alla blockchain, è necessaria una soft fork, quindi un nuovo protocollo che ammetta Segregated Witness (“firma separata”), ovvero la trasmissione nella blockchain di una transazione valida senza il SIGNATURE SCRIPT, in gergo tecnico: «SIGHASH NO_INPUT». Il blocco insomma può contenere transazioni che non siano firmate.

Dal momento che questa modifica richiede una blockchain fork, oggi Lightning Network non è ancora funzionante. [nota a posteriori: SegWit è stato approvato nel luglio 2017]

Oltre a ciò, Lightning Network richiede altre caratteristiche che sono già state implementate mediante una soft fork:

Soft fork di luglio 2016, blocco 419328 (BIP68,112,113)

Grazie alla soft fork avvenuta nel luglio 2016, oltre a nLocktime, un altro parametro delle transazioni è stato introdotto: CheckLockSequenceVerify.

Grazie a questo parametro è possibile rendere disponibile l’output di una transazione x dopo un certo numero di blocchi conseguenti alla registrazione nella blockchain di una data transazione y.

Nell’esempio sopra, l’input di **R** può essere utilizzato come output soltanto dopo n. blocchi (parametro «nSequence») successivi a **C**.

Per esempio, se **C** è trasmessa lunedì, i 10btc in **R** possono essere usati soltanto mercoledì, se **C** è invece trasmessa mercoledì, i 10btc in **R** potranno essere usati solo venerdì, e così via.

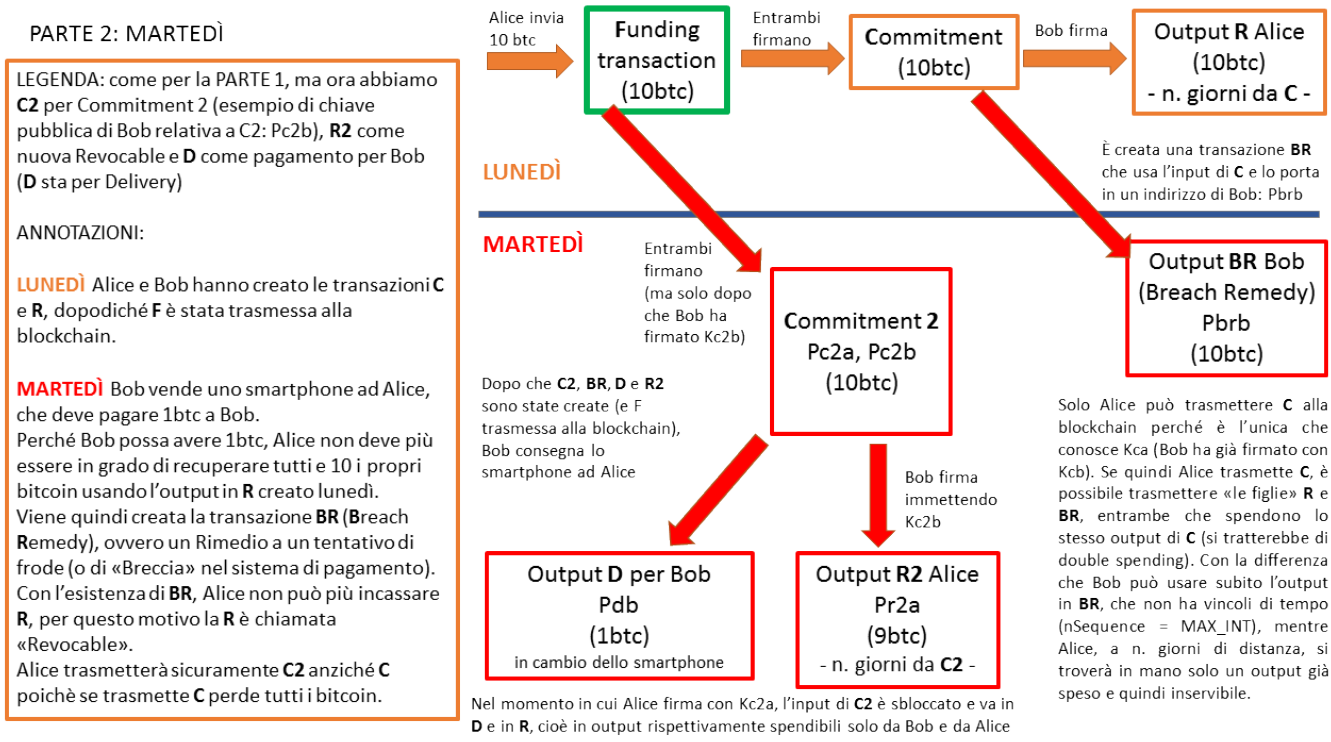
Finchè **C** non è trasmessa, il canale fra Bob e Alice può rimanere aperto a tempo indeterminato.

Come vedremo, né Bob né Alice avranno interesse a trasmettere

C (salvo casi di tentato furto/frode).

Lightning Network: come funziona il channel

ovvero come Alice e Bob scambiano beni o servizi in cambio di bitcoin in modo continuativo attraverso un canale bilaterale aperto con un'unica transazione **F** trasmessa alla blockchain



Ma non è così semplice…

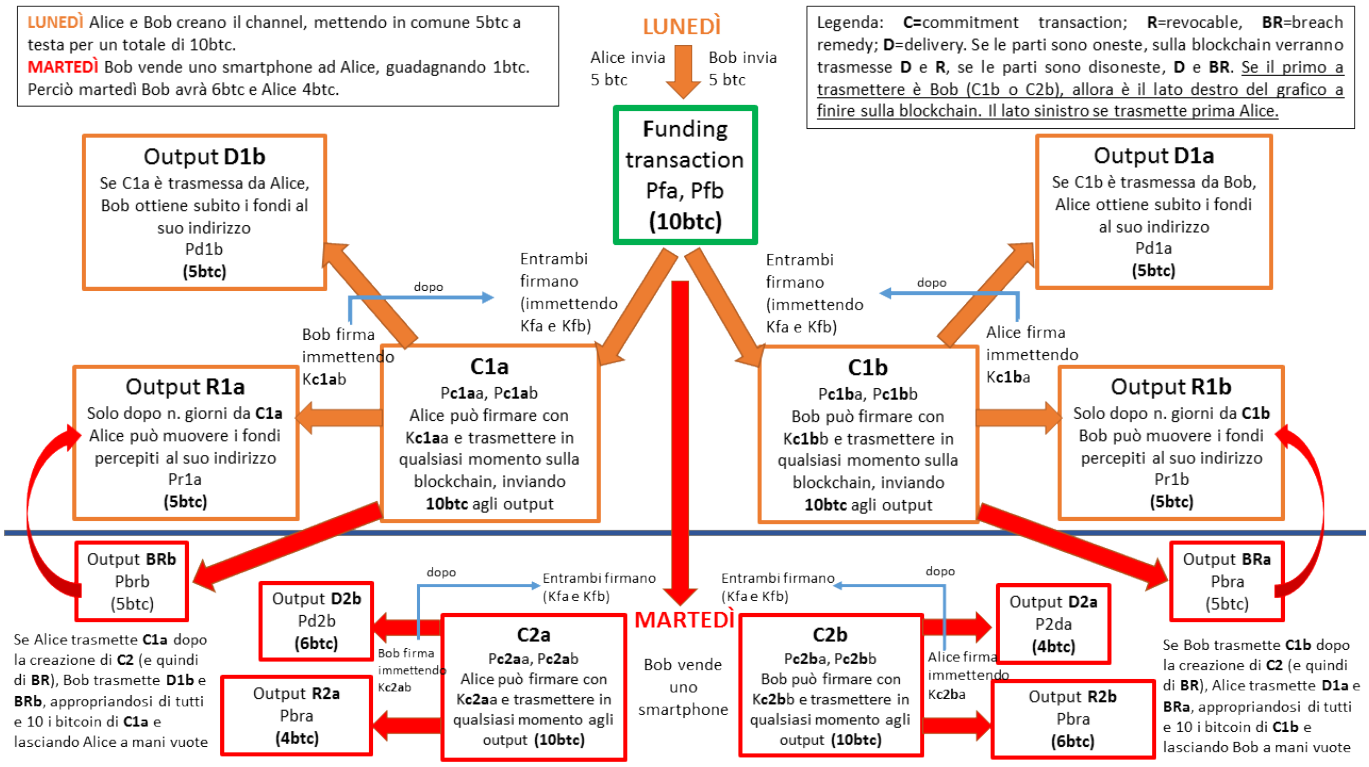
Nello schema appena visto, se Alice dovesse trasmettere alla blockchain, avrebbe convenienza economica a trasmettere **C2** anzicé **C1**: trasmettere uno status precedente comporta la perdita di tutti i propri bitcoin a causa della Breach Remedy. Tuttavia **C2** potrebbe non essere mai firmata da Alice se questa scompare (viene stirata da un bus) o se non agisce in modo razionale (ma potrebbe anche tentare di «ricattare» Bob con la minaccia di tenere in stallo l'output).

Inoltre, Bob avrebbe potuto partecipare alla funding transaction iniziale: per esempio Bob e Alice avrebbero potuto

mettere in comune 5 bitcoin a testa per aprire il channel. In questo caso anche Bob deve avere la garanzia di poter avere indietro i propri fondi prima di assentire alla creazione della **F**unding transaction.

Perciò viene creato uno schema a specchio, dove ogni transazione è raddoppiata. Per comprendere il meccanismo, si veda il prossimo grafico. Per comprendere la notazione, si tenga conot che, poiché C1 raddoppia in C1a e C1b, la chiave pubblica di Alice di C1a risulta «P**c1a**a», di Bob «P**c1a**b», mentre la chiave privata di Bob per muovere i bitcoin da C1b invece sarà «K**c1b**b» e così via).

Nel prossimo grafico si ipotizza che sia Bob che Alice abbiano partecipato alla **F**unding con 5 bitcoin a testa.



Un sistema di «chiuse» dei canali d'acqua

Si guardi, per semplicità, solo alla parte sinistra del grafico (la destra è speculare, a parti invertite):

– **prima** della firma, da parte di Bob e Alice, della

transazione che **«fa fluire»** i 10 btc da **F** all'indirizzo di **C1a** (immettendo Kfa e Kfb), Bob **ha già firmato** con K**c1a**b la transazione che invia i bitcoin da **C1a** ai due output **D1b** e **R1a**

–allo stesso modo, **prima** della firma, da parte di Bob e Alice, della transazione **F->C2a** (immettendo Kfa e Kfb), Bob **ha già firmato** con K**c2a**b la transazione che invia i bitcoin da **C2a** ai due output **D2b** e **R2a**; inoltre entrambi hanno già firmato con Kc**1a**a e K**c1a**b la transazione che invia i bitcoin da **C1a** all'output **BRb** (che non è vincolato nel tempo, al contrario di **R1a**)

Tutto ciò è possibile grazie allo script «SIGHASH NO_INPUT» per cui è richiesto il Soft fork a SegWit (agosto 2017).

Il termine «far fluire» non è utilizzato a caso, ***il sistema richiama le chiuse di un canale d'acqua***: quando Bob firma una transazione apre la diga a valle, nonostante la diga a monte sia ancora chiusa. Una volta aperta la diga a monte, l'acqua (i bitcoin) fluisce automaticamente nell'output aperto che sta a valle. Nel caso di due output aperti per ricevere la stessa acqua (**R1a** e **BRb**) quello che non ha un vincolo temporale nSequence (**BRb**) riceve prima l'acqua, lasciando l'altro canale vuoto, nonostante sia aperto.

Perché tenere il canale aperto è conveniente

Dopo l'acquisto dello smartphone in cambio di 1btc, Alice avrebbe convenienza a frodare Bob, trasmettendo alla blockchain uno stato precedente (la commitment C1 anziché C2) poiché in C1 Alice ha 5btc e Bob 5btc, mentre in C2 Alice 4btc e Bob 6btc. Tuttavia, Alice oltre a poter trasmettere C2a può solo trasmettere C1a (non ha la chiave privata Kfb per inviare un output in C1b), e se trasmette C1a Bob si prende tutti e 10 i bitcoin: Bob potrà infatti trasmettere le figlie di C1a che sono D1b e BRb, mentre Alice potrà solo trasmettere la figlia R1a. In questo caso BRb e R1a rappresenterebbero un double spending, ma BRb non ha vincoli temporali, perciò viene spesa

molto prima di R1a che è, appunto, «Revocable».

Cosa succede se Alice riporta indietro lo smartphone perché non è più interessata ad acquistarlo? Semplice, viene creata la coppia di transazioni C3a e C3b, i cui output saranno 5btc per Alice e 5btc per Bob. Ovviamente, saranno create anche due nuove transazioni BR (Breach Remedy) non vincolate temporalmente, che «invalidino» le Revocable figlie di C2, ovvero R2a e R2b, assicurando così che nessuno trasmetta C2 anziché C3.

Tenere il canale aperto, piuttosto che trasmettere le transazioni sulla blockchain, è conveniente per entrambi. Infatti chi fra Alice e Bob per primo trasmette l'ultima transazione **C**ommitment (ipotizziamo sia C2 nel nostro esempio) avrà due svantaggi:

1 – paga le commissioni al miner per il caricamento della transazione nella blockchain

2 – dovrà aspettare che il vincolo temporale della Revocable scada prima di poter spendere i propri bitcoin (mentre l'altro potrà spenderli immediatamente, poiché la **D**elivery transaction non ha vincoli temporali).

In sintesi

Abbiamo visto come Alice e Bob possono scambiare smartphone con bitcoin senza dover mai trasmettere più di una transazione (la funding transaction) alla blockchain.

Infatti fra Alice e Bob è aperto un canale che può durare per sempre, finché uno dei due non tenta di derubare l'altro.

D'ora in poi dunque ogni transazione fra i due (a patto che non sia d importo maggiore di quanto depositato nella funding) si verificherà off-chain.

Questo può alleggerire la blockchain di molti dati, ma ogni

volta che Alice o Bob faranno una nuova transazione con un’altra persona, come Charlie o Dave, dovranno trasmettere una nuova funding transaction. La mole di dati sulla blockchain rimane comunque elevata.

< Torna alla Parte 1 Vai alla Parte 3 >