Desenvolupament d'un canal de pagament sobre la xarxa de Bitcoin

Carlos González Cebrecos

5 de juny del 2017

Resum– Bitcoin és una criptomoneda descentralitzada que permet als usuaris que la utilitzen realitzar pagaments entre ells de forma pseudo-anònima sense cap figura central. Malgrat tot el seu potencial existeix un problema real d'escalabilitat a la xarxa: la seva implementació limita la ratio de transaccions per unitat de temps que poden ésser absorbides per la xarxa, ja que aquestes s'han de validar passant a formar part d'un registre públic comú a tothom anomenat Blockchain i aquest procés té certes limitacions. En aquest treball es donarà una solució a aquest problema d'escalabilitat: la construcció de canals de pagament bidireccionals entre dos usuaris que operen majoritàriament fora d'aquest registre públic. Concretament es durà a terme la implementació de la Lightning Network.
Paraules clau– Bitcoin, Blockchain, Off-Chain, Transaccions, Canals de Pagament, Lightning Network, Descentralització, Xarxa P2P, Proof-of-Work, Escalabilitat

Abstract– Bitcoin is a cryptocurrency without central authority that allows its users to make payments between them in a psedo-anonymous way. Despite its potential there is a real scalability problem, due to its internal implementation there are some restrictions affecting the ratio between transactions and time units that the network can process. Those transactions will have to be validated in a shared public ledger the Blockchain and such a process involves several restrictions. This project will give a solution for that scalability problem: the development of a bidirectional payment channel between two users that will operate on its majority outside the public ledger. Concretely it will implement the Lightning Network.
Keywords– Bitcoin, Blockchain, Off-Chain, Transactions, Payment Channels, Lightning Network, Decentralization, P2P Network, Proof-of-Work, Scalability

1 Introducció

BITCOIN és una divisa electrònica basada en la criptografia i la descentralització que proporciona pseudo-anonimat als usuaris que la utilitzen. La forma en que assoleix aquesta descentralització és mitjançant un únic registre públic anomenat Blockchain (nom que ve donat per la seva naturalesa, ja que esta formada per blocs encadenats els uns als altres). Aquest registre es construeix pels propis usuaris de la xarxa p2p, que competeixen per tal de posar el proper bloc a la cadena mitjançant una sèrie de càlculs. Aquest procés d'afegir un nou bloc s'anomena minat. Cada nou bloc té una mida màxima fixada a 1 MB, i la dificultat dels càlculs a realitzar s'ajusta automàticament en funció del total hashrate (o poder de càlcul) de la xarxa, d'aquesta manera s'intenta fer que el temps entre el procés de minat de dos blocs sigui constant i proper a 10 minuts.

Amb aquests paràmetres fixats trobem una limitació: a l'ecosistema de Bitcoin hi ha un flux de transaccions per unitat de temps limitat i molt baix. A diferència d'altres sistemes de pagament que poden suportar una gran ratio transaccions per unitat de temps, com pot ser el cas de VISA suportant fins a pics de 56000 transaccions per segon (tps), a Bitcoin en parlem de 7 tps [1]. Aquest fet deixa entreveure el gran problema d'escalabilitat que existeix actualment.

És en aquest punt on es plantejen solucions per tal d'augmentar aquesta ratio, una de les possibles solucions és fer canals de pagament que operin en la seva majoria fora de la cadena, fet que augmentaria de forma virtual el throughput de la xarxa per sobre de les seves limitacions de disseny.

El treball es centrarà en fer una implementació concreta de canals de pagament, en forma de la Lightning Network [3].

Per dur a terme el projecte s'usarà el llenguatje de programació Python, concretament la versió 3.5. La implementa-

• E-mail de contacte:mail@ccebrecos.com
• Tecnologies de la Informació
• Treball tutoritzat per: Sergi Delgado Segura (dEIC)
• Curs 2016/17

ció podrà ser trobada online al seu repositori oficial [2] sota llicència *Apache License 2.0*.

La resta del document s'estructura de la següent manera: a la secció 2 podem trobar tot el *background* necessàri per a una correcta compresió de la resta del treball, la secció 3 conté la creació de l'entorn del programa en Python, tant a baix com a alt nivell. A continuació, la secció 4 explica la creació d'un canal unidireccional que servirà de base per a la posterior creació de la Lightning Network, presentada a la secció 5. Finalment, la secció 6 conclou el treball amb un petit incís en les línies futures d'aquest.

2 BACKGROUND

En aquesta secció s'abordaràn els principals aspectes tècnics de Bitcoin donant una visió superficial a fi de poder entendre ben bé com funciona tot l'ecosistema i conèixer aquelles parts més importants. Amb això es pretèn explicar alguns aspectes clau i que posteriorment sigui més fàcil seguir les següents seccions.

2.1 Transaccions

Una transacció [10] a Bitcoin és senzillament una forma de moure els fons d'una o vàries adreces a una/es altra/es. Així doncs, una transacció es composa, principalment, d'una llista d'inputs i d'outputs. Aquests outputs generaran el que s'anomena UTXO (acrònim de l'anglès *unspent transaction output*). I cada input ha de gastar un UTXO anterior, referenciant-lo en un dels camps interns de l'input. D'aquesta manera, cada input gasta un output anterior en la seva totalitat. Si es vol fer un pagament per menys import, s'hauràn de crear més outputs, un d'ells amb l'import adequat al destinatari pertinent i un altre de retorn de la resta de fons. De la mateixa manera, si es vol fer un pagament d'un import superior al d'un input que es disposa s'hauràn de composar més d'un input per arribar a la suma desitjada. S'ha de tenir en compte que la suma dels inputs ha de ser superior o igual a la suma dels outputs per a que una transacció sigui vàlida, en cas de ser l'import superior, el que resta dels outputs es deixarà com a *fees* per al miner del bloc que inclogui aquella transacció.

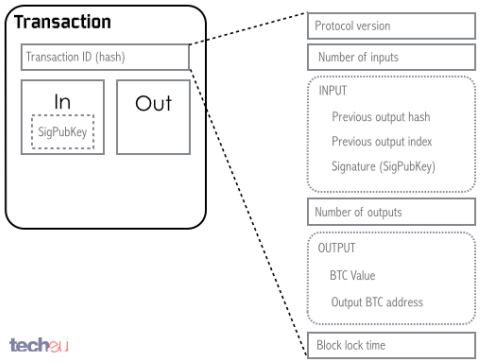

Fig. 1: Detall d'una transacció de Bitcoin [12]

Un dels problemes que hi ha ara mateix a Bitcoin és la maleabilitat de transaccions existent, l'identificador d'una transacció és el doble hash (sha-256) de la transacció serialitzada, això inclou tots els camps com, per exemple, les signatures que hi ha als scripts continguts a la transacció. Donat que l'algorisme de signatura està basat en corves elíptiques (ECDSA) existeix el càlcul d'un factor aleatori en l'elaboració de la signatura que pot fer que dues signatures siguin vàlides tot i ser diferents, ja que tindràn un valor aleatori diferent. Això fa que la signatura pugui variar i, en alterar un dels camps de la transacció el seu identificador serà diferent també.

Això és un problema si hem de tenir en compte pagaments fora de la cadena que s'han de referènciar entre ells, ja que s'ha de tenir en compte que l'identificador d'una transacció haurà de romandre immutable per tal de poder-lo referenciar a les transaccions posteriors.

Aquest problema es pot sol·lucionar amb el *Segregated Witness* o *SegWit* [11], tot i que encara no hi ha una adopció general a la xarxa i per tant, no s'usa.

2.2 Scripting

Per a permetre que els fons només puguin ésser gastats legítimament, cada nova UTXO portarà associat un script que els bloquejarà. Per a poder-los desbloquejar i gastar, l'input corresponent que gasta aquella UTXO portarà un altre script que els desbloquejarà. Amb aquests scripts s'assegura que cada UTXO només podrà ser gastat sota les condicions especificades.

Hi ha molts tipus d'scripts que creen certes condicions, amb un script es pot fer que qualsevol pugui gastar aquells fons, que només els pugui gastar aquell qui estigui en possessió la clau privada associada a una pública, alguns scripts més elaborats permeten condicions més complexes i fins i tot, es pot fer que ningú pugui gastar els fons.

A grans trets es pot dir que cada script consta dues parts diferènciades: una de les quals bloquejarà els fons i l'altra que els allibera, anant cadascuna a un output i un input respectivament. El script situat a l'input permetrà gastar una UTXO anterior sempre que la execució de tots dos scripts junts sigui vàlida, l'output no hagi estat gastat abans i la transacció sigui vàlida. Aquestes parts són anomenades de la següent manera:

- ScriptPubkey: Aquest script és el que s'ubica a l'output i bloquejarà els fons.
- ScriptSig: Aquest script és el que permetrà gastar els fons bloquejats.

Comuntment reben aquest nom pel contingut que tenen al tipus de pagaments més usat a Bitcoin, on es paga al hash d'una clau pública, **P2PKH**.

En el moment de la validació es junten els dos scripts i s'executa el resultat de la concatenació de tots dos: ScriptSig—ScriptPubKey. A grans trets hi ha dos tipus d'operands, les dades i les ordres. El llenguatge d'scripting de Bitcoin funciona com una pila, les dades es van introduint a una pila i es van treient en ordre invers al quel han entrat, mentres que les ordres s'executen en el moment en que s'arriba a elles en temps d'execució a l'script.

En el cas d'una transacció del tipus **P2SH**, parlem de més scripts diferents:

- Locking Script: Aquest script és el que s'usarà per a bloquejar els fons, permetrà construir un ScriptPubKey amb el hash d'aquest.

- Unlocking Script: Aquest script és el que permetrà gastar els fons bloquejats, haurà de contenir el locking script a més a més de les dades necessàries per a poder validar la concatenació de tots dos.

D'aquesta manera, es construeixen els scripts esmentats anteriorment:

- ScriptPubKey: OP_HASH160 [20-byte-hash of {Locking Script}] OP_EQUAL
- ScriptSig: Unlocking Script {Locking Script}

Per a una correcta validació s'ha d'executar l'script sencer sense donar cap error i en la finalització d'aquest, ha de quedar alguna dada diferent de 0 o *FALSE* a la pila. A més a més, per a ser considerat estàndard, un ScriptSig no ha de contenir cap operació, només han de contenir dades.

3 CREACIÓ DE L'ENTORN

Per tal de dur a terme el projecte de forma satisfactoria s'ha decidit començar de baix a alt nivell. Partint de les estructures de dades bàsiques que seràn usades a posteriori per a crear tots els camps i estructures més complexes, donant lloc amb la seva unió a transaccions complertes. Això és així per tal de controlar tots els possibles factors i no haver de dependre d'altres llibreries o eines principalment, tot i que per a funcionalitats molt complexes s'han usat llibreries externes, con és el cas de pybitcointools [3], referènciada també en el codi.

3.1 Baix nivell

En primer lloc, s'ha començat per definir quina és l'estructura interna al projecte que s'usarà per representar els diferents camps, aquesta, per simplicitat a l'hora de crear les transaccions, ha estat el tipus *built-in* bytes de Python.

Amb això s'han creat classes Python per donar forma als diferents tipus de dades presents a les transaccions i scripts, tenint en compte les seves mides, si la seva codificació era little-endian o big-endian, l'existència de prefixos (en el cas de les dades als scripts) i s'han inclòs mètodes per serialitzar i de-serialitzar en hexadecimal.

Això ha permès una gran abstracció en endavant al projecte, ja que no ha calgut preocupar-se més de la codificació concreta de cada dada un cop definit el seu tipus.

3.2 Alt nivell

Un cop definits els diferents tipus de dades en classes Python, s'ha pogut donar pas a crear estructures més avançades, com per exemple un output d'una transacció, compost per la quantitat de satoshis a pagar, la llargada del script que l'acompanya i l'script mateix. S'ha anat construit des la base estructures cada cop més complexes fins arribar a poder composar scripts i finalment transaccions, composades per tota la resta dels camps.

És en aquest punt on s'han començat a crear els primers scripts de l'entorn del projecte.

3.2.1 Scripts

En un primer intent de crear un script s'ha implementat el més comú a Bitcoin, un **P2PKH**, amb el que s'ha fet la primera prova. El detall de les dues parts implementades és el següent:

- ScriptPubkey: OP_DUP OP_HASH160 <PubKeyHash>OP_EQUALVERIFY OP_CHECKSIG
- ScriptSig: <sig><pubkey>

Amb tot això s'ha aconseguit crear i enviar a la xarxa una transacció del tipus P2PKH de forma satisfactòria [1] que pot ésser trobada a qualsevol explorador de la testnet [5].

Un cop minada i havent-se pogut gastar la primera transacció s'ha donat pas a la creació d'un canal unidireccional.

4 CANAL UNIDIRECCIONAL

Un primer pas per a donar una solució al problema d'escalabilitat esmentat anteriorment passa per reudir el nombre de transaccions que s'han d'afegir a la Blockchain. Per a entendre com es pot aplicar aquesta solució suposem el següent escenari: una usuària, Alice vol contractar un servei ofert de forma continuada en el temps per un altre usuari, Bob. D'aquesta manera, amb l'esquema actual de Bitcoin Alice haurà de fer més d'un pagament periòdic a en Bob a canvi del servei, traduint cada pagament en una transacció afegida a la Blockchain. Una alternativa és la creació d'un canal extern de pagament unidireccional on Alice pugués anar pagant de forma incremental a en Bob fins que s'esgotés la quantía del fons o es decidís tancar aquest canal, el que es traduiría en una transacció única final amb l'estat més recent de l'intercanvi.

L'Alice fundarà el canal amb fons seus, i cada cop que vulgui pagar a en Bob, crearà una transacció, amb la qual tots dos hi estaràn d'acord (que descartarà l'anterior si n'hi ha) actualitzant el valor del pagament i se la intercanviaràn. Això pot anar passant fins a exhaurir els fons del canal i serà en el moment en que en Bob o l'Alice facin pública la transacció que passarà a ser valida un cop sigui inclosa a la Blockchain.

Per aquest nou propòsit es crea una adreça multisignatura [6] que requerirà la signatura dels dos usuaris que en formen part per a poder gastar els fons. Això farà que totes dues parts hagin de signar i per tant, acceptin la transacció.

Els scripts d'aquests tipus d'adreces funcionen lleguream ent diferent, en aquest cas es tracta d'un pagament **P2SH** [7].

4.1 Implicacions

Donat que la majòria de transaccions estaran fora de la cadena s'han hagut de crear mecanismes per assegurar que cap usuari enganyarà a l'altre (enviant a la xarxa una transacció inesperada, per exemple). Això s'ha fet amb els scripts mitjançant teòria de jocs, a totes les parts els hi interessa complir amb la seva part del contracte ja que hi guanyen més complint que intentant enganyar a l'altra part. Aquell que intenti enganyar a l'altra part serà qui més perdi,

[1] 0ca139b55f5f6f8151d002a5d3b72a85006ad43d980a0091b3eb94dc781385bf

d'aquesta manera es pretèn que tothom faci el que se suposa que ha de fer.

Per a poder gastar una UTXO que pertany a una adreça multisignatura calen les tantes signatures com s'hagin especificat al script que acomapnya la UTXO de *funding*, això implica que si alguna de les parts decideix no participar i no donar la seva signatura, aquells fons mai més podran ser gastats.

S'han pres mesures per tal d'evitar aquesta situació. S'ha creat un script que permeti gastar en qualsevol moment els fons si es disposa de les dues signatures o bé, passat un cert temps [8] només amb la signatura de la part que hagi fundat el canal (per així poder recuperar els seus fons en cas de no participació de l'altra part). Però això imposarà una data de caducitat al canal, ja que un cop arribat aquest moment, la part fundadora podrà gastar els fons únicament amb la seva signatura.

4.2 Scripting

En aquest cas es tracta d'un pagament més complex com s'ha esmentat al principi d'aquesta secció. Cal dir que el l'script que realment bloquejarà els fons, pay-script, només contindrà el hash del Locking Script i que per a poder gastar aquests fons primerament s'haurà de comprovar que es coneix el script que dona aquest hash, redeem-script, d'aquesta manera el script només el revela en el moment de gastar la transacció, per a més claredat veure [7].

Per a complir amb totes les restriccions esmentades al punt anterior, s'ha fet un nou script amb estructura *if-else* que permetrà ser gastat de més d'una manera.

- Locking Script (corresponent al ScriptPubKey de la transacció de funding):

```
OP_IF
     <time> OP_CLTV OP_DROP
     <PubKeyFunder> OP_CS
OP_ELSE
     OP_2 <pubKeyAlice> <pubKeyBob>
     OP_2 OP_CMS
OP_ENDIF
```

- Unlocking Script (corresponent al ScriptSig de les transaccions d'opening del canal):
  - cas if:
    `<sigFunder> OP_1`
  - cas else:
    `OP_0 <sigAlice> <sigBob> OP_0`

4.3 Transaccions

Amb aquests scripts s'han fet diverses proves. S'ha fundat un canal i s'han pogut gastar en els dos casos [9].

Ara doncs, el que toca es pensar: I si volem crear un canal que funcioni en ambdos sentits? Amb aquesta implementació per a que dos usuaris puguin realitzar-se pagaments en ambdos sentits cal crear dos canals, un en cada sentit, això no es gens pràctic, ja que cada canal funciona independentment de l'altre i no hi ha un fons comú que permeti tot el flux de moviment de fons. D'aquesta manera, si un usuari esgota els fons a un canal, però encara en té a l'altre, no podrà realitzar més pagaments en el primer, ja que no disposa de liquidesa.

És aquí on entra la creació del canal bidireccional, un canal que permetrà pagaments en tots dos sentits.

5 LIGHTNING NETWORK

Amb aquest nou canal, es permeten pagaments en ambdos sentits, dos usuaris que hagin creat un canal bidireccional podràn realitzar i rebre pagaments sota el mateix canal, reduint així la quantitat necessària de canals de pagament (ja que fins ara era necessàri un canal per cada sentit) i optimitzant el movimient de fons entre les dues parts.

Aquestes transaccions seràn també *off-chain* (excepte, com sempre, la primera i la última) de manera que s'aconsegueix alleugerir la xarxa.

Un cop es tingui la primera transacció, els usuaris crearàn noves transaccions per actualizar el flux dels fons del canal, la signaràn ells mateixos i la enviaràn a l'altra part per a que la mantingui. Aquestes noves transaccions aniràn reemplaçant a les anteriors en el temps.

Entrem en més detall.

5.1 Protocol

En primer lloc s'haurà de crear un canal multisignatura, com els vistos a la secció 4, amb els fons necessaris per dur a terme tots els moviments posteriors necessàris. Aquesta transacció, anomenada *opening transaction*, es llença a la xarxa per a consolidar el canal.

Per a actualizar el canal cada usuari calcularà el hash d'un valor secret i aleatori només conegut per ell mateix i enviarà el resultat del hash a l'altra part. Ara, cada usuari construirà una nova transacció que gastarà l'import de la transacció d'*opening* i el repartirà entre les parts involucrades amb l'estat actual del canal. De manera que tindrem dues noves transaccions reflexant el mateix estat, una nova creada per cada usuari.

Aquestes transaccions constaran d'un input, referenciant l'anterior transacció (la d'*opening* del canal) i un parell d'outputs:

- El primer output serà de l'emisor del pagament al receptor, aquest pagament serà un pagament del tipus P2PKH amb la quantía adient.
- El segon output, i és aquí on ve la novetat, enviarà els fons potencialment a tots dos usuaris, aquest cop no inclourà un script multisignatura com l'anterior a la transacció d'*opening*. Aquest nou script permetrà gastar els fons compartits sota dues condicions:
  - El receptor del pagament realitza una signatura i coneix la pre-imatge d'un hash, que haurà calculat el receptor i li haurà comunicat. És a dir, la pre-imatge del hash que ha calculat la part contrària.
  - L'emisor del pagament realitza una signatura i espera un cert temps, ja que aquesta opció conté un *locktime*.

D'aquesta manera, una imatge possible de les transaccions en aquest escenari és la que mostra la figura 2.

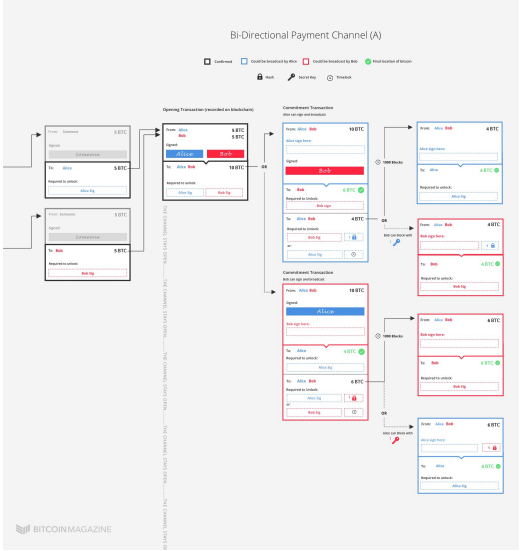

Fig. 2: Detall de les transaccions de la Lightning Network [3]

En aquest punt, cada un d'ells podria llençar la transacció a la cadena i tancar el canal, però no podrien rebre la seva part fins passat un cert temps, l'indicat per el *locktime* del script.

Posteriorment, el proper cop que es vulgui actualitzar el canal, intercanviant fons sigui quin sigui el sentit, es tornarà a repetir el mateix procés amb una ètapa més: la revelació de la pre-imatge anterior. Cada usuari tornarà a calcular el hash d'un nou valor aleatori i secret només conegut per ell mateix i li enviarà a l'altra part i a més a més, revelerà la pre-imatge del hash inclòs a la transacció anterior. Igual que abans cada usuari construirà la transacció amb el detall del nou hash, la signarà i li enviarà a l'altra part.

Ara, cadascún d'ells podría llençar a la xarxa la transacció nova, però es repeteix la situació anterior. I podria llançar també la transacció anterior, però, com ha revelat la pre-imatge del hash, estaria donant tota la quantía de fons del canal a l'altre usuari, ja que l'altre usuari podria reclamar els fons per a ell amb la seva signatura i la pre-imatge ja coneguda abans que passi el temps especificat pel *locktime*.

D'aquesta manera s'assegura que totes les parts involucrades fan el que s'espera que han de fer, ja que és el que més els hi convé, si intenten alterar el transcurs del canal i tancar-lo abans d'hora, estaràn perdent fons.

5.2 Implicacions

Tot això implicarà doncs l'existencia de comunicació entre els usuaris, per un canal adicional, aliè al protocol de Bitcoin actual. Els usuaris hauran d'intercanviar-se transaccions no definitives, que no aniran a parar a la Blockchain, per tal de ser signades i guardades. Addicionalment, aquests canals auxiliars s'utilitzaran per l'intercanvi de les pre-imatges dels hashos inclosos a les transaccions per a cada actualització del canal.

5.3 Scripting

Cada transacció constarà d'un input, que provè de la creació del canal o d'una transacció anterior feta amb el canal i d'un parell d'outputs tal i com s'ha explicat al primer apartat d'aquesta mateixa secció.

Aquest script detalla totes les condicions necessàries per a garantir que cap de les dues parts llençarà a la xarxa una transacció sense el consentiment de l'altra part.

```
OP_IF
    OP_HASH160 <hash(val)> OP_EV
    <PubKeyB> OP_CS
OP_ELSE
    <time> OP_CLTV OP_DROP
    <PubKeyA> OP_CS
OP_ENDIF
```

Per a poder gastar aquesta nova UTXO, s'usaràn els següents *unlocking scripts*:

- cas if:
  `<sigBob> <val> OP_1`

- cas else:
  `<sigAlice> OP_0`
  (havent passat el temps estimat)

5.4 Juntant les peces

Amb tot això i juntant tots els scripts fets fins ara, donant una última volta amb noves millores com la presència d'un *hashlock* creuat, s'ha aconseguit crear totes les transaccions que permeten fer ús de la Lightning Network.

Finalment doncs, tot i tenir transaccions amb scripts i casuístiques força complexes i unes quantes transaccions *off-chain*, a la cadena només en llençaràn dues d'aquestes, la primera, per a fundar el canal i la última, per tancar-lo.

A ulls de la blockchain això serà tot:

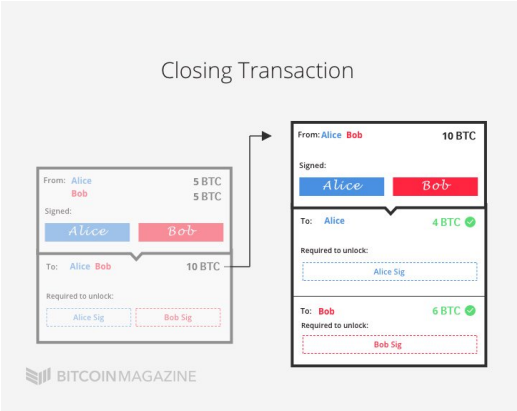

Fig. 3: Transaccions vistes a la blockchain [3]

6 Conclusions

Durant la primera ètapa del projecte s'ha fet molta recerca, fet que ha permès entendre a baix nivell Bitcoin i com funciona tot l'ecosistema.

S'ha creat un software que permet, amb un gran nivell d'abstracció, crear transaccions de molts tipus i enviar-les a la xarxa sense cap problema (per a enviar-les s'usa un software extern, concretament el client oficial de Bitcoin [13] sobre Ubuntu 16.10).

Aquestes transaccions consten dels scripts necessaris per a poder complir totes les condicions desitjades. Van des dels més *simples* com és el cas d'un P2PKH, fins a la creació dels necessaris per a sustentar la Lightning Network, havent passat per la creació d'un canal unidireccional i solucionant totes les possibles solucions que poguessin sorgir, amb *locktimes*, protecció amb hashos i multisignatures.

6.1 Millores

La principal línia de treball futur del projecte, un cop implementada la Lightning Network entre dos usuaris, consisteix en anar més enllà i incloure la figura d'una tercera part. Amb això es pot aconseguir realitzar pagaments d'un usuari a un altre, sense haver de fundar un canal directament entre ells, trobant una combinació de nodes intermitjos que ja tinguin el canal creat entre ells. D'aquesta forma els usuaris finals podran realitzar pagaments segurs.

A més a més, es podría implementar no només els nodes, si no tot el protocol de comunicació necessari entre ells i d'aquesta manera donar forma a tota la xarxa que sustentarà els canals de pagament bidireccionals creats amb la implementació de la Lightning Network.

AGRAÏMENTS

A tots aquells que heu participat i m'heu ajudat d'alguna manera durant aquest projecte, en especial:

Al meu tutor, Sergi Delgado, per l'ajuda i la dedicació durant tot el procés.

A en Jordi Herrera, per la guía i la introducció a aquest meravellós món de Bitcoin.

I finalment, al meu company i amic David Lozano, per acompanyar-me durant el camí.

REFERÈNCIES

[1] "Scalability - Bitcoin Wiki", En.bitcoin.it, 2017. [Online]. Available: https://en.bitcoin.it/wiki/Scalability. [Accessed: 05- Jun- 2017].

[2] "btc-payment-channels", GitHub, 2017. [Online]. Available: https://github.com/uab-projects/btc-payment-channels.

[3] A. Wirdum, Ünderstanding the Lightning Network, Part 1: Building a Bidirectional Bitcoin Payment Channel", Bitcoin Magazine, 2017. [Online]. Available: https://bitcoinmagazine.com/articles/understanding-the-lightning-network-part-building-a-bidirectional-payment-channel-1464710791/.

[4] "Script - Bitcoin Wiki", En.bitcoin.it, 2017. [Online]. Available: https://en.bitcoin.it/wiki/Script.

[5] "Blockchain Explorer — Blockr.io", http://blockr.io, 2017. [Online]. Available: http://tbtc.blockr.io/

[6] "bitcoin/bips", GitHub, 2017. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0013.mediawiki.

[7] "Pay to script hash - Bitcoin Wiki", En.bitcoin.it, 2017. [Online]. Available: https://en.bitcoin.it/wiki/Pay_to_script_hash.

[8] "How is time encoded (BIP65) in scripts?", Bitcoin.stackexchange.com, 2017. [Online]. Available: https://bitcoin.stackexchange.com/questions/39119/how-is-time-encoded-bip65-in-scripts. [Accessed: 19- May- 2017].

[9] "Blockchain Explorer — Blockr.io", http://blockr.io, 2017. [Online]. Available: http://tbtc.blockr.io/address/info/2NCAQ1WUeKhBu133qhVD3SomyTVt1aT98GG.

[10] "Developer Guide - Bitcoin", Bitcoin.org, 2017. [Online]. Available: https://bitcoin.org/en/developer-guide#transactions.

[11] E. SegWit, "SegWit, Explained", CoinTelegraph, 2017. [Online]. Available: https://cointelegraph.com/explained/segwit-explained. [Accessed: 18- Jun- 2017].

[12] A. Barrera, "A Guide to Bitcoin (Part II): A deep dive into the Bitcoin ecosystem", Tech.eu, 2017. [Online]. Available: http://tech.eu/features/926/bitcoin-ecosystem/.

[13] "Descargar - Bitcoin", Bitcoin.org, 2017. [Online]. Available: https://bitcoin.org/es/descargar.