Anàlisi del protocol SPV de Bitcoin

Marc Garcia Lavado

Resum– A causa de la gran popularitat de la criptomoneda Bitcoin, molts usuaris han decidit utilitzar moneders portables (lightweight wallets) per l'estalvi en capacitat d'emmagatzematge i la baixa potència de còmput que requereix aquest tipus de client. El seu desplegament requereix tant pocs recursos que pot ser utilitzat en dispositius de baix processament com telèfons intel·ligents i tàblets. Permet enviar i rebre transaccions i realitzar les mateixes funcionalitats que un client convencional (full client). Tot i les avantatges que aporta, la seva implementació és realment complexa i garantir un bon funcionament mantenint l'essència de la tecnologia i les propietats amb les quals va ser desenvolupada és tot un repte. En aquest treball s'investiga a fons el funcionament del protocol i la tecnologia que fa possible que des de un simple telèfon sigui possible interactuar amb la xarxa Bitcoin sense ni tan sols tenir descarregada ni sincronitzada la totalitat de la cadena de blocs. A més, s'ha descarregat un dels lightwallets més populars i utilitzats a la xarxa per comprovar de primera mà el seu funcionament i si aquest compleix les normes establertes per el protocol

Paraules clau– Paraules clau del treball, màxim 2 línies . Bitcoin, criptomoneda, Bloom filter, arbre, hash, merkle, privacitat

Abstract– Due to the great popularity of the Bitcoin cryptocurrency, many users have decided to use portable wallets (lightweight wallets) for savings in storage capacity and the low computing power required by this type of client. Its deployment requires so few resources that can be used in low-processing devices such as smartphones and tablets. It lets you manage to send and receive transactions and perform the same functionalities as a conventional client. Despite the advantages it offers, its implementation is really complex and guarantees a good operation while maintaining the essence of the technology and the properties which it was developed is a challenge. In this essay we investigate thoroughly the operation of the technology that makes it possible from a simple telephone to interact with the Bitcoin network without even having downloaded or synchronized the whole chain of blogs.In addition, one of the most popular and used lightwallets on the network has been downloaded to verify carefully how it works and if it fulfill the rules established by the protocol

Keywords– Bitcoin, crypto, cryptocurrency, Bloom, filter, privacy, merkle, hash, tree

1 INTRODUCCIÓ

La criptomoneda Bitcoin és la moneda virtual descentralitzada i dissenyada per un autor anònim japonès de pseudònim Satoshi Nakamoto per poder operar en una xarxa no confiable que ha esdevingut més popular en el món de les criptomonedes, on existeixen més de 1000 diferents.

El protocol Bitcoin està dissenyat per no confiar en ningú i tots els nodes de la xarxa realitzen verificacions quan reben nous blocs o transaccions per evitar ser enganyats.

• E-mail de contacte: marc.garciala@e-campus.uab.cat
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Jordi Herrera Joancomartí (DEIC)
• Curs 2017/18

Totes les transaccions realitzades no es poden cancelar ni revertir i són emmagatzemades en blocs encadenats que formen una gran cadena de blocs (blockchain). Això implica que el número de transferències registrades en tota la xarxa no deixa mai de créixer.

Per poder enviar i rebre bitcoins es necessari disposar d'un software client que haurà de descarregar completament tota la blockchain i verificar totes les transaccions (full client) abans de poder realitzar cap transacció. Degut a la gran quantitat de dades que cal descarregar i comprovar, es requereix una capacitat moderada de còmput i d'emmagatzematge de la qual molts dispositius actuals no disposen, com ara els dispositius mòbils. Per reduïr els requisits necessàris i fer més accessible aquesta tecnología es va desenvolupar el SPV (Simplified Payment Verification) i els wallets de tipus lightweight.

El SPV permet operar a la xarxa sense necessitat de

descarregar i verificar la blockchain sencera a canvi de crear una dependència amb un full client que si que disposi de la blockchain actualitzada. D'aquesta manera es dona lloc als moneders de tipus lightweight wallet, els quals seran els encarregats d'utilitzar el SPV amb nodes de tipus full client per aconseguir les transaccions del seu interès, garantint la privacitat i fiabilitat.

Totes aquestes aventatges són possibles gràcies a l'implementació de complexes estructures de dades que proporcionen mecanismes molt eficients per reduïr l'ús d'ample de banda i per implementar mecanismes de verificació de transaccions molt optimitzats.

2 OBJECTIUS

L'objectiu d'aquest treball és estudiar i conèixer el funcionament del SPV. Al tractar-se d'un subprotocol tan complex s'han establert objectius de diferent profunditat. Això ens ha permès obtenir una idea global sobre que és exactament el SPV, per a què serveix i si realment és útil. Un cop entesa l'idea principal hem pogut treballar amb objectius més concrets com les complexes estructures de dades que implementa per garantir un correcte funcionament.
De forma addicional, s'ha estudiat i revisat la implementació i ús en el software Bitcoinj i s'ha analitzat el seu funcionament fent especial esmena en si aquest aplica les implementacions necessàries per garantir la privacitat dels usuaris que l'utilitzen.
Per tant, han establert objectius específics centrats en funcionalitats concretes que, sigui per un error de disseny o d'implementació, poden ser més susceptibles a mostrar irregularitats en el seu funcionament. Son els següents:

1. Estudiar que és el SPV i per a què s'utilitza.
2. Estudiar i conèixer els diferents tipus de missatges utilitzats pel SPV.
3. Estudiar i entendre les comunicacions entre un lightwallet i un full client.
4. Entendre com es realitza una transacció de bitcoin utilitzant un lightwallet.
5. Entendre com es rep informació sobre una transacció utilitzant un lightwallet.
6. Entendre les estructures de dades que utilitza per obtenir un rendiment més òptim que un client clàssic.
7. Estudiar la privacitat que ofereix el lightwallet de Bitcoinj.
8. Extreure conclusions sobre el funcionament teòric del protocol
9. Extreure conclusions sobre el funcionament pràctic del software analitzat.
10. Proposar millores per augmentar la privacitat que ofereix el software

3 ESTAT DE L'ART

El protocol Bitcoin permet als seus usuaris realitzar pagaments al crear transferències de Bitcoin (BTC) des d'una o més adreces d'entrada cap a almenys una adreça de sortida. Una adreça Bitcoin corresponen a una clau pública mentre que la clau privada serveix per donar accés a l'adreça corresponent a la qual pertany i poder gastar els BTC. Per poder gastar BTC, un usuari crearà una transacció on típicament s'especificarà com a inputs els outputs de la transacció prèvia que serà d'on s'ha obtingut els BTC. Per enviar una transacció l'usuari la signarà i l'enviarà a varis peers de la xarxa peer-to-peer de Bitcoin.

Les transaccions són empaquetades en blocs de Bitcoin, aquests blocs són minats quan els miners resolen un problema de proof-of-work i troben els paràmetres adequats. Quan es resol el problema es diu que s'ha trobat un bloc i el miner l'envia per la xarxa perquè els altres peers el rebin i verifiquin si és correcte. Si finalment el bloc resulta ser correcte el miner serà recompensat amb una quantitat de BTC i el bloc serà referenciat al bloc anterior creant una cadena de blocs o blockchain.

El protocol Bitcoin està dissenyat perquè es verifiquin tots els blocs i transaccions. Aquest fet provoca que una instal·lació d'un full client requereixi una quantitat elevada tant d'espai d'emmagatzematge com de potència de còmput.

Tant el procés d'instal·lació de nous clients com de verificació de nous blocs resulta una tasca molt costosa per hardware amb poca capacitat de còmput com telèfons intel·ligents, tauletes o altres dispositius similars.

Per posar solució a aquest problema es va proposar un client lleuger (lightweight wallet) que funciona amb un subprotocol de bitcoin anomenat Simple Payment Verification (SPV). Els clients que utilitzin SPV no emmagatzemen tota la blockchain ni validen totes les transaccions, únicament verifiquen les capçaleres dels blocs. Quan un SPV vol calcular el seu saldo de BTC, els full nodes de Bitcoin li proveiran dels blocs que continguin transaccions rellevants l'usuari, estalviant processar gran quantitat d'informació sobre blocs i transaccions que no tenen cap rellevància pel client.

Aquesta sol·lució permet estalviar un excés de verificació innecessari i mentre la xarxa bitcoin tingui un gran nombre de full clients no hi haurà cap problema. En cas que el nombre de full clients es trobi molt reduït es correrà el risc que la xarxa pugui ser dominada pels atacants.

Mentre que els nodes poden verificar les transaccions pells mateixos, els SPV poden ser enganyats per transaccions creades per atacants si aquests dominen la xarxa. Per evitar aquest cas, existeix una estratègia per detectar aquest tipus d'atacs que consisteix en monitoritzar les alertes que es creïn quan es detectin blocs invàlids. Això sens dubte seria un avís de perill i és recomanaria descarregar un full client per confirmar possibles inconsistències.

4 EL PROTOCOL SPV

El protocol SPV (Simple Payment Verification) és un subprotocol de Bitcoin que va ser descrit a la secció 8 del paper original de Satoshi Nakamoto. El seu objectiu principal es reduïr les característiques tècniques dels dispositius que vulguin operar a la xarxa Bitcoin i d'aquesta manera fer

possible utilitzar aquesta tecnologia en dispositius mòbils. L'ús del SPV permet reduir el sobrecost computacional que es produeix a causa de la verificació constant de tots els blocs i transaccions que són minats.

A diferència d'un full client de Bitcoin, un lightwallet que utilitzi el protocol SPV només descarrega les capçaleres dels blocs, estalviant així gran quantitat d'espai al evitar la descàrrega d'un gran nombre de transaccions que no són rellevants. A canvi, el SPV necessita connectar-se amb un full client per realitzar qualsevol operació com rebre o enviar informació de transaccions. Per tant, es crea una dependència dels fulls clients a canvi de l'estalvi d'espai i de còmput.

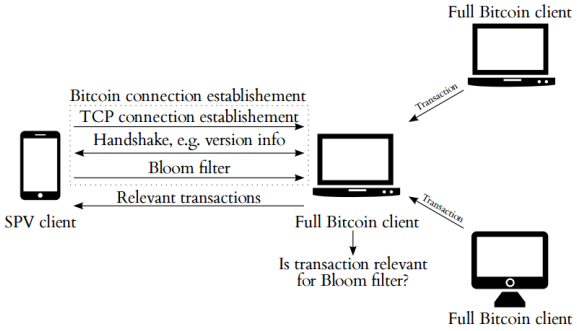

Figura 1: Il·lustració de com es produeix l'establiment de la conexió entre un client SPV i un full client

Existeix un mecanisme dissenyat per tal que el SPV obtingui la informació necessària sobre les seves transaccions sense sol·licitar-ho de forma explícita, ja que si un lightweight wallet demanés a un full client si aquest te informació sobre noves transaccions d'una adreça determinada, el full client coneixeria l'adreça o direccions del client SPV i aquest podrà obtenir informació sobre el seu saldo i els moviments que ha realitzat.

4.1 El Merkleblock

L'arbre de Merkle és una estructura de dades en arbre en què cada node que no és un full està etiquetat amb el hash de la concatenació de les etiquetes o valors dels seus nodes fill. Són una generalització de les llistes hash i les cadenes hash.

Permet que gran nombre de dades separades puguin ser lligades a un únic valor hash del node arrel de l'arbre. D'aquesta manera proporciona un mètode de verificació segura i eficient dels continguts de grans estructures de dades. En les seves aplicacions pràctiques normalment el hash del node arrel va signat per assegurar la seva integritat i que la verificació sigui totalment fiable. La demostració que un node fulla és part d'un arbre hash donat requereix una quantitat de dades proporcional al logaritme del nombre de nodes de l'arbre.

Els SPV de Bitcoin utilitzen aquesta estructura per reduir la quantitat de dades de la transacció que s'envien. Utilitzar els arbres de Merkle permeten als SPV realitzar una verificació de pagament simplificada, ja que no intenten verificar completament la cadena de blocs, sinó que només comproven que els encapçalaments de blocs es connectin correctament i confien que les transaccions que els hi ha enviat un full client són vàlides.

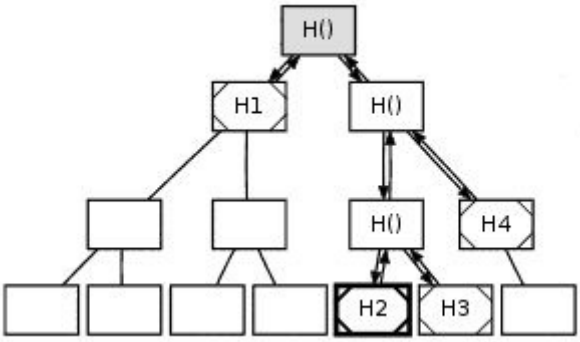

Figura 2: Il·lustració de com es produeix el merkletree a partir dels hash de les transaccions

Això evita als clients de SPV descarregar tot el contingut de blocs i totes les transaccions de difusió, només per llençar la gran majoria de les transaccions que no són rellevants per a les seves carteres.

Per tant, un missatge Merkleblock és un encapçalament de bloc que es pot utilitzar per extreure informació d'identificació per a les transaccions que coincideixen amb el filtre de Bloom i demostrar que les dades de transacció coincidents realment van aparèixer al bloc resolt. Els clients poden utilitzar aquestes dades per assegurar-se que el node remot no els està enganyant amb transaccions falses que mai han aparegut a un bloc real.

Un cop s'ha desxifrat el missatge de Merkleblock i s'ha comprovat l'autenticitat de la transacció s'envien les dades en format TX.

4.2 Els filtres de Bloom

El filtre de Bloom és una estructura de dades probabilística que permet conèixer de forma imprecisa si un element es troba dins d'un subconjunt o no. El resultat de l'operació obtindrà que l'element no es troba dins del conjunt o que aquest pot trobar-s'hi sense poder determinar-ho amb total certesa. Els falsos positius són possibles tot i que improbables si es gestiona la mida del filtre segons el nombre d'elements. Tot i que pot semblar ineficient, quan es treballa amb volums de dades molt grans no ens podem permetre assumir costos per consultar si cada element es troba o no dins d'un subconjunt, ja que si sabem amb certesa que no es troba estalviarem temps i recursos en fer una cerca.

Un filtre de Bloom consta d'un array d'm bits amb totes les posicions inicialitzades a 0. També ha d'haver-hi k diferents funcions hash definides. Per afegir un element s'aplicaran les funcions hash a l'element en qüestió, on cada funció indicarà una posició de l'array i aquesta serà escrita amb el valor 1. Per conèixer si un element es troba dins del subconjunt, s'aplicaran les funcions hash a l'element tal com pasa amb el cas de la inserció però aquest cop es comprovaran els valors dels camps resultants. Si el valor de tots els camps és 1, l'element pot existir dins del subconjunt, en canvi, si és 0 sabrem amb certesa que l'element no existeix.

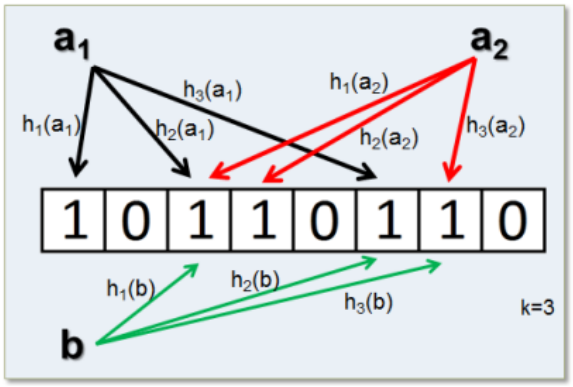

Figura 3: Il·lustració on es mostra com s'inicialitza un filtre de Bloom inserint els valors 1 a les posicions on els hash ho determinen.

Els filtres disposen de molts paràmetres configurables que ens permetran augmentar o reduir l'anonimat que proveeixen. Els paràmetres més importants són:

- m: Correspon al nombre de bits que té l'array per emmagatzemar adreces. Al crear el filtre tots els bits es posicionen a 0 i s'aniran posant a 1 durant la inserció dels elements quan alguna funció hash apunti a qualsevol posició amb un valor de 0. El fet que aquest valor sigui molt elevat no garanteix un bon ús del filtre, ja que depèn en gran mesura de la relació entre el nombre d'insercions o el nombre de funcions k hash.
- n: Correspon al número d'adreces que conté el nostre filtre. No hi ha valors màxims ni mínims així que la mida mínim del nombre d'elements del filtre sería 1 (on no hi ha anonimitat, ja que el full client esbrinarà la nostra adreça de forma immediata) i el valor màxim que proporcionaria el nivell més alt de privacitat equivaldria a igualar la mida del filtre amb el nombre d'adreces actuals (la qual cosa no és viable en termes de rendiment). Així doncs, una bona pràctica seria adaptar la mida del filtre segons la connexió de banda ample de la que es disposés en aquell precís instant, proporcionant un nivell acceptable de privacitat a canvi de no comprometre el rendiment.
- k: Correspon al nombre de funcions hash utilitzades. Un major número de funcions hash utilitzades implicarà col·locar a 1 més posicions per cada adreça afegida al filtre. És a dir, si un filtre utilitza 12 funcions hash, el filtre inserirà un 1 a dotze posicions diferents per registrar l'adreça al filtre. Com ens podem imaginar, un valor k més elevat reduirà el nombre de col·lisions en un filtre que mantingui una bona proporcionalitat entre el nombre posicions totals i el nombre de posicions que estiguin iniciades 1.
- p: Correspon al nombre de falsos positius que es produeixin. Els falsos positius corresponen a aquells valors que el full client interpreta com adreces d'interès perquè així ho ha indicat el filtre de Bloom. Això pot tenir dos possibles causes.
  - El full client ha trobat una adreça de farciment que el SPV ha utilitzat per omplir el filtre.
  - El full client ha trobat una col·lisió, és a dir, una adreça que no està al filtre però que al haber estat computada amb les funcions hash aquestes han apuntat a posicions on es trobava un 1 i la han validat.

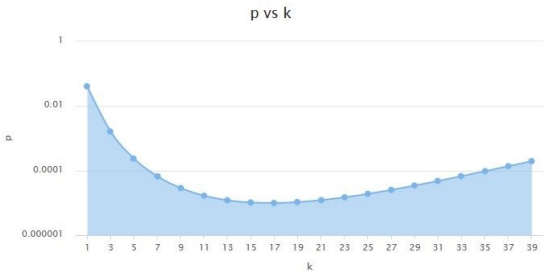

Figura 4: Probabilitat de falsos positius a l'eix Y (p) per nombre de funcions hash utilitzades a l'eix X (k).

Tal com s'indica a la figura, es produeix un lleuger augment en el nombre de falsos positius si el nombre de k funcions hash és superior a 23, trobant-se el mínim de falsos positius entre 13 i 19 funcions hash amb una taxa de falsos positius al voltant d'una col·lisió per milió

Els SPV de Bitcoin utilitzen aquest sistema per determinar si una transacció pot ser de l'interès del client SPV. Perquè el full client pugui determinar si pot ser de l'interès del client, agafarà totes les adreces contenides als blocs i les computarà amb les k funcions hash determinades pel filtre del client i comprovarà el valor de les posicions que aquestes indiquen. Es possible que es tracti de falsos positius i el client igualment acabi descartant l'informació però utilitzant aquest mètode el SPV estalvia realitzar una gran quantitat de còmput i estalvia capacitat d'emmagatzematge.

Els filtres de Bloom, si són implementats correctament, poden ser una molt bona eina per garantir la privacitat. Com més elevada sigui la taxa de falsos positius del filtre de Bloom, major serà el nivell de privacitat. Tot i que obtindrem una major taxa de falsos positius i un ús més intens de l'amplada de banda. Depenent de la configuració del lightwallet, els filtres de Bloom poden ser regenerats cada cert temps o cada cop que executem de nou el lightwallet. Depenent de la mida dels paràmetres del filtre de bloom i del nombre de filtres utilitzats amb el pas del temps pot ser possible amb major o menor dificultat esbrinar quina o quines poden ser les adreces dels clients de lightwallet.

Els programadors intenten anonimitzar les adreçes dels seus clients mitjançant aquesta eina i poder amagar les adreçes dels usuaris. Tot i així, hi han indicis de que es poden produir fuites d'informació si aquesta funcionalitat no s'implementa de forma correcte.

4.3 Missatges de xarxa específics de SPV

El protocol bitcoin utilitza una sèrie de missatges per garantir una comunicació eficient i efectiva entre un lightweight wallet SPV i un full node de Bitcoin. Els missatges són:

- Missatge Version: Al principi de la conexió s'envia aquest missatge per donar informació sobre el tipus de node que estem utilitzant, en el nostre cas es bitcoinj.

En cas que el destinatari rebi el missatge, ens contestarà amb el mateix missatge indicant quin tipus de node es ell (Satoshi per anar bé).

- Missatge Verack: Després que els dos peers hagin intercanviat els missatges de version, el full node enviarà un missatge de verack indicant que ja pot començar a enviar altres missatges perquè ja està preparat per operar.

- Missatge Addr: retransmet la informació de la connexió dels peers a la xarxa. Cada parell que vulgui acceptar connexions entrants crea un missatge addr que proporciona la informació de la connexió i, a continuació, envia aquest missatge als seus peers no sol·licitats

- Missatge Filterload: Indica al grup receptor que filtri totes les transaccions retransmeses i sol·liciti els blocs de Merkle a través del filtre proporcionat. Això permet als clients rebre operacions rellevants per a la seva cartera, a més d'una taxa configurable de falses transaccions positives que poden proporcionar una privacitat plausible-negativa.

  Utilitza varis paràmetres que es consideren rellevants:

  - nFilterBytes: Mida del filtre en bytes
  - filter: Camp de bit arbitrari on s'emmagatzema el filtre.
  - nHashFunctions: Número de diferents funcions hash que utilitza aquest filtre. Valor màxim es 50.
  - nTweak: Valor arbitrari per afegir al seed per a utilitzar la mateixa funció hash per tal de que proporcioni resultats diferents
  - nFlags: Conjunt de flags que controlen els parametres del filtre. Podem distingir entre:
    * BLOOM UPDATE NONE (0) el node no actualitza el filtre de Bloom.
    * BLOOM UPDATE ALL (1) Si el filtre coincideix amb qualsevol element de dades en un script de pubkey, el punt de sortida corresponent s'afegeix al filtre.
    * BLOOM UPDATE P2PUBKEY ONLY (2) Si el filtre coincideix amb qualsevol element de dades en un script de pubkey i aquest script és un script P2PKH o no P2SH de pagament a multisig, el punt de sortida corresponent s'afegeix al filtre.

- MemPool: El missatge mempool demana els TXID de les transaccions que el node receptor ha verificat com a vàlid, però que encara no han aparegut en un bloc. És a dir, les transaccions que es troben al grup de memòria del node receptor. La resposta al missatge mempool és un o més missatges inv. Que contenen els TXID en el format d'inventari habitual.

  L'enviament del missatge Mempool és útil quan un primer programa es connecta a la xarxa. Els nodes complets poden utilitzar-lo per reunir ràpidament la majoria o totes les transaccions no confirmades disponibles a la xarxa; això és especialment útil per als miners que intenten reunir transaccions per les seves tarifes de transacció. Els clients de SPV poden configurar un filtre abans d'enviar un mempool per rebre només transaccions que coincideixin amb aquest filtre; això permet que un client recentment iniciat obtingui la majoria o totes les transaccions no confirmades relacionades amb la seva cartera.

- Tx: El missatge tx transmet una sola transacció en el format de la transacció sense processar. Es pot enviar en una varietat de situacions

- MerkleBlock: El missatge Merkleblock és una resposta a un missatge getdata que va sol·licitar un bloc mitjançant el tipus d'inventari MSG MERKLE BLOCK. Només forma part de la resposta. si es donen les transaccions coincidents, s'enviaran per separat com a missatges tx.

- GetBlocks:El missatge getblocks sol·licita un missatge inv que proporciona les capçaleres de blocs a partir d'un punt concret de la cadena de blocs. Permet que un node que s'hagi desconectat o iniciat per primera vegada obtingui les dades que necessita per sol·licitar els blocs que no ha vist.

- Inv: El missatge permet a un full client anunciar que té informació relevant pel SPV. Pot ser en resposta a un missatge de getblocks o no sol·licitat si el node ho detecta gràcies al filtre.

- GetData: El missatge sol·licita un o més objectes de dades d'un altre node. Els objectes són sol·licitats per un missatge Inv.

  La resposta a un missatge getdata pot ser un missatge de tx, un missatge de bloc, un missatge de merkleblock, un missatge de cmpctblock o un missatge de no s'ha trobat.

  Aquest missatge no es pot utilitzar per sol·licitar dades arbitràries, com ara transaccions històriques. Els nodes complets ni tan sols poden proporcionar blocs més antics si han podat transaccions antigues de la seva base de dades de blocs. Per aquest motiu, normalment, el missatge getdata només s'hauria d'utilitzar per sol·licitar dades d'un node que anteriorment havia anunciat que tenia aquestes dades enviant un missatge inv.

5 METODOLOGIA

Per tal d'adquirir la informació per realitzar aquesta investigació s'ha utilitzat l'eina Wireshark per monitoritzar el tràfic generat per l'aplicació Bitcoinj. Els passos han estat els següents:

1. Obrir el Wireshark i iniciar la captura de paquets
2. Obrir el wallet de Bitcoinj
3. Escoltar la xarxa i monitorar les comunicacions. Depenent de la prova també han realitzat transaccions.
4. Tancar el wallet

5. Inspeccionar els paquets

El fet d'utiltizar l'eina Bitcoinj ens ha permès experimentar i comparar resultats jugant amb diversos paràmetres que l'aplicació utilitza per realitzar la comunicació. El fet de veure el fluxe del programa ens ha estat molt útil per conèixer de forma més exacte com funciona el software i determinar quins punts poden ser més interessants d'investigar.

Executar el software en mode Debug i veure el flux del programa amb tota mena de detall ha ajudat molt a la investigació d'aquest projecte i ha permès abocar resultats de gran valor.

6 ANÀLISI DEL FUNCIONAMENT DEL LIGHTWALLET BITCOINJ

6.1 L'eina bicoinj

Bitcoinj és un projecte opensource que implementa el software necessari per desplegar un lightwallet de Bitcoin completament funcional. Algunes de les funcionalitats de les quals disposa són:

- Mantenir una cartera.
- Enviar / Rebre transaccions sense necessitat d'una còpia local de Bitcoin Core.
- Xifratge per protegir l'accés a l'aplicació.
- Càlcul de tarifes.
- Signatura múltiple.
- Suport per a extensions i oients d'esdeveniments per mantenir als usuaris actualitzats amb els canvis de saldo.
- Suport per a canals de micropagaments que permeten establir un contracte de signatura múltiple entre el client i el servidor i, a continuació, negociar al canal, permetent micropagaments ràpids que eviten les tarifes de miners.

Està implementat a Java, però es pot utilitzar des de qualsevol llenguatge compatible amb JVM.

Per evaluar el funcionament del protocol SPV hem utilitzat el framework Bitcoinj, ja que es un dels SPV amb més activitat i que presenta més maduresa en el moment de realitzar l'anàl·lisi.

Moltes de les aplicacions SPV disponibles al mercat han desenvolupat a partir d'aquest lightweight wallet, ja que el codi per interactuar amb altre full node mitjançant SPV presenta un grau de maduresa superior a qualsevol dels altres projectes existents a la xarxa. Varis dels lightweight wallets més populars com Electrum o Copay estan basats en Bitcoin. Per tant, es molt probable que qualsevol mala implementació del codi pugui afectar directament a varis lightweight wallets disponibles al mercat.

La pàgina del projecte conté una documentació molt completa i ben estructurada que permet als desenvolupadors conèixer el software més fàcilment i adaptar-lo a les seves necessitats.

Tot i això, Bitcoinj és un treball en curs, i no té algunes característiques que probablement es considerin importants. També té peculiaritats estranyes i altres problemes que han de resoldre.

6.2 Establint la connexió

Quan iniciem el SPV el primer que farà serà establir una connexió amb varis nodes per connectar-se a la xarxa. El primer pas d'aquest procés consisteix a utilitzar el missatge version per indicar quina és la nostra versió de client (SPV en el nostre cas). El node al que haurem intentar establir la connexió ens respondrà amb un altre missatge version indicant la seva versió. En cas que es tracti d'un SPV la connexió es finalitzarà. Si el servidor ens contesta indicantnos que es una versió de full client el SPV enviarà un segon missatge verack per indicar-li que es troba preparat per iniciar la comunicació.

Passades varies hores, el SPV decideix tancar les connexions amb els nodes anteriors i buscar nous nodes, per tant, es tornen a reproduir els passos d'establiment de connexió.

han estudiat les conexions que estableix el SPV i s'ha detectat que es freqüent que el SPV es connecti de forma reiterada a algunes IP's concretes.

Durant l'estudi, s'ha executat el software de Bitcoinj varies vegades per monitorar a quins peers es conectava i enviava els filtres de bloom. Les captures han estat realitzades cada 10 minuts.

- El SPV s'ha connectat i ha enviat 4 filtres diferents a una sola IP.
- El SPV s'ha connectat i ha enviat 3 filtres diferents a 4 IP's diferents
- El SPV s'ha connectat i ha enviat 2 filtres diferents a 22 IP's diferents
- El SPV s'ha connectat i ha enviat 1 filtre a 79 IP's diferents

A partir d'aquest experiment podem destacar que durant un ús habitual del lightwallet es molt possible que un atacant amb un o varis nodes pugui interceptar gran quantitat de filtres i arribar a obtenir les adreces dels clients.

6.3 Creant un filtre de Bloom

El filtre de Bitcoinj es crea de manera estàtica, es a dir, el filtre sempre té la mateixa mida. Els paràmetres que pren el filtre per la seva creació són:

- Utilitzar 16 funcions hash diferents
- Establir la mida del filtre a 15384 bits
- Omplir el filtre amb 642 adreces.
- Generar un seed de 19 dígits de longitud. És un valor completament aleatori.
- Especificar quin flag es vol utilitzar per actualitzar el filtre de Bloom

Tal com recull la documentació i els comentaris dins del programa, la implementació dels filtres està dissenyada per obtenir un baix nombre de falsos positius, concretament amb els valors utilitzats la taxa es de 0.00001 o 1 fals positiu per cada 100.000.

Crida l'atenció el baix nombre d'adreces amb el que s'omple el filtre de Bloom si ho comparem amb la gran quantitat d'adreces que hi poden haver al protocol Bitcoin. Això provoca que el rendiment del filtre de Bloom sigui òptim, ja que les probabilitats de rebre transaccions amb informació de fals positius es realment baixa.

Si un possible atacant volgués esbrinar l'adreça o adreces BTC del SPV podria utilitzar el filtre per processar totes les adreces amb les que han realitzat transaccions consultant la blockchain. Això reduiria considerablement el nombre de possibles adreces tot i que seria un gran nombre si tenim en compte les adreces utilitzades per farcir el filtre i els falsos positius que puguin ocasionar. Per tant, una alta taxa de falsos positius es considera bo des de el punt de vista de la privacitat, ja que augmentaria el nombre d'adreces candidates a ser les del SPV.

No obstant, si disposem de varis filtres de Bloom podem reduir encara més el rang de possibles adreces, ja que els filtres sempre contindran les adreces del SPV i seran omplertes amb adreces diferents cada vegada, sent fàcilment distingibles les unes de les altres.

A aquest fet, cal afegir que Bitcoinj genera nous filtres de Bloom cada vegada que es connecta a la xarxa per primera vegada o si ja fa molt temps que es troba conectat, buscant nodes diferents tal com hem detallat a la secció anterior.

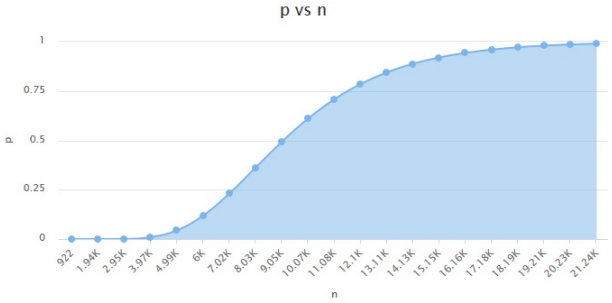

Figura 3: Percentatge de falsos positius a l'eix de les Y (p) per nombre d'elements inserits a l'eix de les X (n)

6.4 Rebent una transacció

Un cop el SPV ja ha establert connexió i ha creat el filtre de Bloom l'enviarà als nodes amb el missatge Filterload perquè el full client filtri totes les transaccions i envii al SPV els blocs de merkle. Quan el SPV ja ha enviat l'informació necessària respecte les seves adreces demanarà si existeixen transaccions seves mitjançant els missatges MemPool i GetBlocks. Amb el MemPool s'assegurarà de si existeixen transaccions vàlides que no haguin aparescut encara en cap bloc perquè encara no han estat minades i amb el GetBlocks sol·licitarà capçaleres de blocs que no coneix.

Si el full client té transaccions noves li farà saber al SPV enviant-li un missatge inv i el SPV li contestarà amb un getdata per rebre l'informació.

El full client contestarà amb un merkleblock perquè el SPV pugui comprovar l'autenticitat de la mateixa i després rebrà la transacció amb un missatge TX.

6.5 Realitzant una transacció

Sempre que el SPV es trobi connectat i estigui utilitzant un filtre de Bloom podrá enviar missatges TX amb la transacció signada. Les transaccions sempre s'envien per tots els peers per garantir una major fiabilitat.

7 Resultats

Bitcoinj genera els filtres de Bloom amb una longitud fixa i de mida reduïda. Els desenvolupadors han prioritzat el rendiment i l'eficiència en comptes de pensar en la privacitat, que és la finalitat per la qual va ser desenvolupada aquesta funcionalitat. Utilitzar la configuració de filtre que implementa Bitcoinj posa en risc la privacitat dels usuaris, ja que el disseny implementat està pensat per evitar falsos positius (factor que proporciona privacitat als usuaris).

Per altra banda, els filtres es guarden a la memòria volàtil del dispositiu per lo que cada cop que s'estableix una nova connexió o es busquen nous peers després d'estar un temps connectat es creen nous filtres permitent als atacants establir un rang de possibles adreces cada cop més reduït a base passar les adreçes pels filtres.

Aquest fet s'agreuja si observem que el software es propens a reconnectar-se als mateixos nodes proveint nous filtres i no garantint un nivell de privacitat suficient per protegir els adreces dels clients.

Per cada filtre de Bloom que un full node de Bitcoin pugui utilitzar es reduirà considerablement el número de potencials adreces que pot tenir el client SPV.

El resultat d'un bon funcionament del filtre de Bloom es basa en la correcta utilització del conjunt de tots els paràmetres.

Si volem un filtre de Bloom amb una baixa taxa de falsos positius obtindrem com a resultat un filtre on la possibilitat de produir-se una col·lisió serà propera al 0, de manera que podrem saber amb una certesa quasi completa que un element es troba dins del filtre. Aquesta implementació pot ser de molta utilitat per cercadors amb un gran volum de dades i podrà determinar de forma instantània si un element es troba o no dins del subconjunt.

Però quan es tracta d'altres implementacions, com per exemple per garantir un nivell de privacitat, un filtre de bloom amb una alta taxa de falsos positius serà beneficiosa per l'usuari tot i que haurà de processar transaccions que no seran del seu interès al tractar-se de col·lisions.

8 Conclusions

El protocol SPV està molt ben dissenyat. Sobre el paper encaixen totes les peces i és una alternativa viable i avantatjosa respecte al full client. Utilitzen vàries estructures de dades que permeten implementar mecanismes per verificar transaccions, garantir la privacitat dels usuaris i estalviar amplada de banda i capacitat de processament. Han estat dissenyats per ser una alternativa amb garanties respecte al full client tot i dependre d'ells.

La seva utilització estalviar un excés de verificació innecessari i mentre la xarxa bitcoin tingui un gran nombre de full clients no hi haurà cap problema. En cas que el nombre de full clients es trobi molt reduït es correrà el risc de que la xarxa pugui ser dominada pels atacants.

Mentre que els nodes poden verificar les transaccions pells mateixos, els SPV poden ser enganyats per transaccions creades per atacants si aquests dominen la xarxa. Per evitar aquest cas, existeix una estratègia per detectar aquest tipus d'atacs que consisteix en monitorar les alertes que es crein quan es detectin blocs invàlids. Això sens dubte seria un avís de perill i es recomanaria descarregar un full client per confirmar possibles inconsistències.

Utilitzar un SPV suposa :

- L'ús dels SPV es desaconsella per negocis o plataformes que rebin gran quantitat de pagaments ja que el temps de verificació de les transaccions es major. En aquests casos es millor utiltizar un full client ja que proporciona més independència, seguretat i una verificació més ràpida.

- Més facilitat per operar a la xarxa BTC amb dispositius poc potents o antics

- No poder garantir la privacitat al desconèixer la implementació que ofereix el software, per tant podem dir que s'assumeix el risc de revelar informació rellevant a les adreces dels propis usuaris.

- Beneficiar-se d'un estalvi de còmput i capacitat d'emmagatzematge.

9 Propostes de millora

Actualment s'estan estudiant millores per resoldre aquest problema, com implementar filtratge de blocs en el client. Això permet que en comptes de que el client enviï un filtre a un full node, aquests generen filtres deterministes en les dades del bloc que se serveixen al client. Un client lleuger pot descarregar un bloc sencer si el filtre coincideix amb les dades que està mirant. Atès que els filtres són deterministes, només han de construir un cop i emmagatzemar-los al disc, sempre que un nou bloc estigui connectat a la cadena.

La privadesa del client es millora perquè els blocs es poden descarregar des de qualsevol origen, de manera que cap usuari no obtingui informació completa sobre les dades que requereix un client. Els clients molt conscients de la privadesa poden optar per obtenir blocs de forma anònima utilitzant tècniques avançades com la recuperació d'informació privada

Agraïments

Vull agrair al meu tutor de treball, el Dr. Jordi Herrera Joancomartí pel seu temps, dedicació i suport durant tot aquest treball. Ha servit de guia i mentor tant a l'hora d'enfocar el treball com a l'hora de resoldre dubtes.

També m'agradaria agrair als meus pares pels seus savis consells i la seva comprensió. Sempre heu estat aquí per a mi. Finalment, els meus amics. No només heu estat aquí per donar-nos suport entre nosaltres en els moments difícils, sinó que també hem tingut xerrades sobre altres coses no relacionades amb universitats i articles científics.

Referències

[1] Simplified Payment Verification (SPV) https://bitcoin.org/en/developer-guidesimplified-payment-verification-spv

[2] SPV as Implemented Today is EXACTLY As Described In the Bitcoin Whitepaper https://medium.com/@jonaldfyookball/spv-as-implemented-today-is-exactly-as-described-in-the-bitcoin-whitepaper-2a65265afbec

[3] Why Every Bitcoin User Should Understand "SPV Security" https://medium.com/@jonaldfyookball/why-every-bitcoin-user-should-understand-spv-security-520d1d45e0b9

[4] Connection Bloom filtering https://github.com/bitcoin/bips/blob/master/bip-0037.mediawiki

[5] Bitcoin: A Peer-to-Peer Electronic Cash System https://bitcoin.org/bitcoin.pdf

[6] BitcoinJ, disponible desde http://bitcoinj.github.io/.

[7] BitcoinJ limitations, disponible desde http://bitcoinj.github.io/limitations.

[8] BitcoinJ documentation, disponible desde https://bitcoinj.github.io/documentation

[9] On the Privacy Provisions of Bloom Filters in Lightweight Bitcoin Clients https://eprint.iacr.org/2014/763.pdf

[10] Bloom filter calculator https://hur.st/bloomfilter/

[11] Bloom filter https://en.wikipedia.org/wiki/Bloomfilter

Bloom filters by Jason Davies https://www.jasondavies.com/bloomfilter/

[12] Bloom filters by example http://llimllib.github.io/bloomfilter-tutorial/

[13] What are bloom filters https://blog.medium.com/what-are-bloom-filters-1ec2a50c68ff

[14] Arbres de Merkle https://bitcoin.org/en/developer-referencemerkle-trees

[15] Bitcoin protocol documentation https://en.bitcoin.it/wiki/Protocoldocumentation

[16] Bitcoin developer reference https://bitcoin.org/en/developer-reference

[17] What is merkle root https://bitcoin.stackexchange.com/questions/10479/what-is-the-merkle-root

[18] Missatges Filterload https://bitco.in/en/developer-referencefilterload

[19] Bloom filter com a estructura de dades probabilística https://hackernoon.com/probabilistic-data-structures-bloom-filter-5374112a7832