Estudi de la blockchain RSK

Bernat Marcilla Jiménez

Resum– La blockchain de Bitcoin va ser dissenyada amb forces restriccions tècniques per tal d'aconseguir una plataforma sòlida i segura però a mesura que ha passat el temps i a causa de l'augment de plataformes que permeten la creació d'*Smart Contracts*, cada cop va agafar més força la idea de poder incorporar els *Smart Contracts* sobre el mateix ecosistema de Bitcoin. El tema a tractar en aquest TFG és RSK, un protocol que permet la creació i execució d'*Smart Contracts* i que funciona com una *sidechain* a la blockchain de Bitcoin, el que permet aprofitar tant el minat, còmput i energia que gasta la blockchain de Bitcoin. En aquest TFG es farà un estudi teòric de com funciona la blockchain d'RSK, s'exploraran conceptes més concrets com el *merged mining* o el protocol *Powpeg*. D'altra banda, com a aportació pràctica del treball, es presentarà el desenvolupament d'una Dapp basada en la blockchain d'RSK que permet afegir liquiditat i intercanviar tokens ERC20.

Paraules clau– RSK, Rootstock, blockchain, Bitcoin, Ethereum, contractes intel·ligents, Solidity, exchange descentralitzat, criptomonedes

Abstract– The Bitcoin blockchain was designed with strong technical constraints to achieve a solid and secure platform, but as time has passed and due to the increase in platforms that allow the creation of Smart Contracts, the idea of being able to incorporate Smart Contracts into the Bitcoin ecosystem became stronger. The topic in this work is RSK, a protocol that allows the creation and execution of Smart Contracts and is a Bitcoin sidechain, which allows to take advantage and reuse both mining, computing, and energy spent by the Bitcoin blockchain. This article comprehends a theoretical study of how the RSK blockchain works, a deeper approach of concepts such as merged mining or the Powpeg protocol, as well as presenting the practical part developed, which is a Dapp based on the RSK blockchain that allows to add liquidity and exchange ERC20 tokens.

Keywords– RSK, Rootstock, blockchain, Bitcoin, Ethereum, Smart Contracts, Solidity, Decentralized Exchange, cryptocurrencies

1 Introducció - Context del treball

La blockchain de Bitcoin va ser dissenyada amb forces restriccions tècniques per tal d'aconseguir una plataforma sòlida i segura. Però a mesura que ha passat el temps i a causa de l'augment de plataformes que permeten la creació d'*Smart Contracts*, cada cop va agafar més força la idea de poder incorporar els *Smart Contracts* sobre el mateix ecosistema de Bitcoin. Aquests *Smart Contracts*, tal com indica el seu nom, consisteixen en contractes amb termes i condicions sobre un cert acord, però amb la diferència que són intel·ligents, el que significa que són capaços d'executar-se i fer-se complir per si mateixos de forma autònoma i automàtica sense intermediaris ni mediadors. Així doncs, com que implementar els canvis necessaris perquè aquests *Smart Contracts* es puguin executar sobre l'ecosistema de Bitcoin és francament complicat a causa del nivell de consens dels usuaris de la xarxa que cal aconseguir, han aparegut projectes com RSK que tenen com a objectiu oferir aquesta possibilitat.

En aquest article s'examina concretament el projecte RSK, un protocol que permet la creació i execució d'*Smart Contracts* i que funciona com una *sidechain* de la blockchain de Bitcoin mitjançant la tècnica de *Merged Mining* per tal d'aprofitar el minat, còmput i energia que gasta la seva blockchain pare [1].

2 Objectius

L'objectiu principal del TFG és fer un estudi sobre la blockchain RSK. A partir d'aquest objectiu, es poden definir diferents subobjectius que permetran l'acompliment de l'objectiu principal. Un d'aquests subobjectius és explorar i entendre què és una *sidechain* i alhora, analitzar com funcionen aquestes implementacions respecte a la cadena pare.

- E-mail de contacte: bernat.marcilla@e-campus.uab.cat
- Menció realitzada: Tecnologies de la Informació
- Treball tutoritzat per: Jordi Herrera Joancomartí (dEIC)
- Curs 2020/2021

És important estudiar com es desenvolupa aquesta relació o interacció entre ambdues cadenes per tal d'entendre com s'aprofita tot el que implica el minat dels blocs de la blockchain pare.

Un altre subobjectiu és valorar i comparar tant els avantatges com els inconvenients d'aquestes noves implementacions respecte a altres implementacions blockchain més "tradicionals".

Per últim, es proposa un subobjectiu més pràctic que consisteix a desenvolupar i desplegar dos tokens ERC20 a la xarxa de proves juntament amb una Dapp que permeti afegir liquiditat i intercanviar els tokens creats. Aquest subobjectiu permetrà poder contrastar tota la part teòrica de l'estudi amb les funcionalitats reals que actualment proporciona la blockchain d'RSK.

Així doncs, la idea és explorar i fer un estudi teòric complet de com funciona l'associació de la blockchain RSK respecte a la seva blockchain pare i com, a diferència d'aquesta, la blockchain RSK permet l'execució d'*Smart Contracts*. Addicionalment, també es treballarà la part més pràctica plantejant i desplegant els diferents *Smart Contracts* així com la Dapp per interactuar amb aquests.

3 METODOLOGIA

Durant el desenvolupament del treball es va plantejar una metodologia pel seguiment i control del treball o feina que es va realitzant i una metodologia per la gestió i control de les tasques del projecte.

Sobre la metodologia per controlar el seguiment i tenir un control de treball, aquesta va consistir a fer una reunió de seguiment amb el tutor aproximadament cada dues setmanes per tal de tenir un seguiment constant, poder anar complint els objectius proposats a cada reunió i aconseguir un desenvolupament del projecte més consistent i a l'hora, poder reaccionar amb temps a possibles dificultats o contratemps que es puguin trobar.

Sobre la gestió i control de tasques, l'objectiu va ser treballar seguint la metodologia Kanban. I vaig decidir triar aquesta metodologia, ja que és una metodologia àgil que ja coneixia i de la qual vaig poder comprovar de primera mà tant els avantatges com els inconvenients.

Alhora, també s'ha utilitzat el sistema de control de versions Git a través de Github, per tal de poder gestionar, controlar i desenvolupar de la millor forma la construcció tant dels contractes com de la Dapp.

4 ESTAT DE L'ART

Una de les polèmiques que ha anat creixent més en els últims anys sobre el Bitcoin i les blockchains basades en l'algorisme de consens *Proof of Work* és el consum energètic que implica.

Aquests darrers anys el consum s'ha disparat arribant a màxims històrics, el que ha fet que es comencin a contemplar alternatives a aquest algorisme de consens. I l'alternativa que actualment sembla que pot plantar cara és l'algorisme de consens *Proof of Stake*, el qual es basa en proves de possessió de monedes en comptes de proves de treball, evitant així el consum energètic que implica buscar la prova de treball vàlida. Però tot i l'acceptació general de la comunitat que ha tingut aquest algorisme com ho avalen algunes blockchain que ja l'implementen com Cardano o la futura implementació d'Ethereum 2.0, aquest algorisme de consens segueix tenint algunes mancances que no té l'algorisme *Proof of Work*.

Un dels problemes principals és el perill de reproduir el model bancari actual, doncs aquesta solució en la qual hi ha una forta correlació entre el nombre de monedes que es posseeix respecte a la possibilitat d'obtenir noves monedes, s'apropa a les polítiques bancàries criticades per fer "els rics més rics i els pobres més pobres". En defensa, però d'aquest algorisme, cal indicar que s'està treballant en tècniques com el *coin-age* per tal del d'establir un valor corrector en funció de l'antiguitat d'una moneda. Un altre dels problemes que té algorisme és l'atac *nothing at stake*, un problema que apareix en implementacions en les quals l'intent fallit del minat d'un bloc no suposa cap penalització, permetent que en cas de *fork* de la cadena, els miners participin en el minat de cada una de les cadenes "branques"disponibles alhora, fent així possible que els atacs de doble despesa siguin més factibles [2].

Així doncs, la tècnica de *Merged Mining* es presenta no com a una alternativa al problema de l'algorisme de consens *Proof of Work*, sinó com a una forma per reaprofitar aquest alt consum que implica la cadena de Bitcoin utilitzant-lo també per assegurar la pròpia cadena que implementa aquesta tècnica i fent així aquest consum més eficient.

Fig. 1: Consum elèctric de la xarxa de Bitcoin segons el *Cambridge Bitcoin Electricity Consumption Index* (CBECI) [3].

5 EL MINAT EN UNA SIDECHAIN: MERGED MINING

El concepte de *Merged Mining* consisteix a minar dues o més criptomonedes al mateix temps, però sense que això impliqui que el miner requereixi un major poder computacional per minar els blocs. Això es tradueix en el fet que un miner pot utilitzar la seva potència computacional per minar blocs de múltiples cadenes de forma simultània.

Aquest concepte es va començar a implementar el 2014 a través del protocol que es coneix com a *Auxiliary Proof of Work* (AuxPoW) i va ser la primera implementació que introduïa aquesta idea de mineria combinada. Tal com indica el seu nom, aquest protocol permet que el treball computacional que s'ha realitzat per una cadena del tipus *Proof of Work*, sigui utilitzat pel minat d'una altra cadena.

Veiem doncs que una de les principals restriccions que té el Merged Mining és la necessitat que les cadenes compar-

teixin el protocol de validació de la cadena així com l'algoritme. En el cas de Bitcoin i RSK, el protocol de validació és *Proof of Work* i l'algorisme que implementen és el SHA-256.

El protocol *Proof of Work* és un sistema que es basa en algun tipus de treball que té un cert cost i que ha de realitzar un client per tal que la xarxa pugui validar aquest treball fet. I alhora, aquest permet evitar certs comportaments indesitjables a la xarxa com l'atac *double-spending*.

En el cas de Bitcoin, aquest treball consisteix a realitzar dos cops l'algorisme de hashing SHA-256 sobre la capçalera del bloc que es vol minar, el que suposa un cost computacional. La capçalera del bloc sobre la qual els miners realitzen el hash, inclou paràmetres com la versió actual del bloc, l'identificador del bloc anterior, un identificador que recull en forma d'arbre totes les transaccions del bloc, conegut com el *Merkle Root*, una marca de temps de quan el miner va començar a realitzar el hash sobre la capçalera, la dificultat actual de la cadena i per últim, el valor aleatori *nonce*. Aquesta dificultat que s'inclou al bloc, és un valor variable, que varia en funció del temps actual que s'està trigant a minar els blocs de Bitcoin. Aquest valor s'ajusta per aconseguir una mitjana del temps de minat dels blocs de Bitcoin cada 10 minuts. Si actualment els blocs es minen amb més rapidesa, aquesta dificultat augmenta i si es triga més, la dificultat disminueix.

Així doncs, la prova de treball que aplica el protocol de Bitcoin es basa a trobar una solució del doble hash de la capçalera del bloc que sigui igual o superior a la dificultat actual del protocol. És per això necessari la variable *nonce*, la qual és un valor que anirem definint aleatòriament per cada prova del hash de la capçalera del bloc que fem. La prova de treball per tant, consistirà a demostrar que hem trobat un valor de *nonce* que fa complir que el hash de la capçalera és igual o està per sobre de la dificultat actual del protocol, i indirectament estem demostrant el cost de tots els intents de prova i error que hi ha hagut darrere fins a trobar la solució.

5.1 Funcionament del Merged Mining a RSK

La idea fonamental del *Merged Mining* consisteix en primer fer la construcció del bloc de la cadena filla i aquest bloc, incloure'l d'alguna forma al bloc de la cadena pare. A partir d'aquest bloc de la cadena pare, podrem buscar la solució que compleixi amb una dificultat com a mínim igual a la dificultat de la cadena filla per demostrar la feina feta pel minat del bloc.

Aplicant aquesta idea a la blockchain d'RSK i a la de Bitcoin, la teoria ens diu que el miner de Bitcoin haurà de fer la construcció primer del bloc d'RSK i haurà d'afegir una referència d'aquest al bloc de Bitcoin que intentarà minar. A partir d'aquí, en funció de la solució que el miner vagi trobant i les dificultats de les diferents cadenes, podrem validar només el bloc d'RSK, el de Bitcoin i el d'RSK o cap dels dos i per tant, caldrà seguir buscant noves solucions.

A la pràctica, però, aquest procés no funciona exactament així, ja que apareixen 2 actors nous que són l'RSK merged-mining pluguin i el node d'RSKj. Aquests dos actors són bastant importants, ja que són els que permeten que els miners de Bitcoin puguin realitzar el *Merged Mining* i alhora, són els encarregats d'intentar que aquest procés suposi el mínim esforç possible als miners de Bitcoin.

Per una banda, l'RSK merge-mining plugin és un una interfície a partir de la qual els miners de Bitcoin poden interactuar amb el node d'RSKj a partir de crides JSON-RPC. Per altra banda, el node d'RSKj és un *full node* del protocol RSK i serà l'encarregat de crear el bloc RSK a minar, feina que s'estalvien de fer els miners de Bitcoin, ja que així aquests només hauran de demanar l'identificador del bloc. A més, quan els miners de Bitcoin trobin una solució vàlida per RSK, entregaran la solució al node d'RSKj perquè aquest la validi i si és vàlida, aquest s'encarregui de propagar el bloc d'RSK que ha creat anteriorment i sobre el que el miner de Bitcoin ha trobat una solució.

5.2 Diferents dificultats

Una de les principals diferències entre la blockchain de Bitcoin i la d'RSK és el temps mitjà de minat de bloc, doncs mentre que els blocs de Bitcoin es minen aproximadament cada 10 minuts, els d'RSK ho fan cada 30 segons. Això és possible gràcies a la dificultat del protocol, ja que reduint la dificultat, reduirem el temps mitjà del minat dels blocs.

Veiem clarament que la dificultat de la blockchain d'RSK serà inferior a la dificultat de la blockchain de Bitcoin i en conseqüència, moltes de les solucions que trobem que no compleixin amb la dificultat de Bitcoin, si ho faran amb la d'RSK.

Aquesta implementació permetrà que tots els blocs d'RSK es trobin referenciats a un bloc candidat de Bitcoin que no sempre ha acabat a la mateixa xarxa de Bitcoin. De fet, fent el càlcul a partir del temps mitjà del minat del bloc, per cada bloc candidat que si ha aconseguit la dificultat de Bitcoin, tindrem aproximadament 20 blocs candidats que no han aconseguit trobar una solució vàlida per la dificultat de Bitcoin però sí per la d'RSK.

En conclusió, com que la dificultat de la blockchain filla és menor, els miners trobaran una quantitat major de solucions per la blockchain filla i una quantitat menor per la blockchain pare. Alhora, com que la dificultat de Bitcoin és major, sempre que trobem un bloc que satisfà aquesta dificultat, el bloc també satisfarà la dificultat d'RSK i en conseqüència serà un bloc vàlid per la xarxa d'RSK [4].

5.3 Refresh time

El fet que la blockchain d'RSK tingui una dificultat més baixa i per tant hi hagi un minat dels blocs cada molt menys temps, també influeix en característiques com el temps cada quant els nodes actualitzen l'estat de la blockchain o de la *mempool*.

Aquesta actualització és fonamental pel que respecta als interessos dels miners, doncs a partir d'aquesta, poden estar sempre al cas de les transaccions que paguen més comissions per tal que les puguin incloure al bloc candidat que estan minant. Pel que fa a Bitcoin, aquesta actualització sobre les noves transaccions que arriben a la *mempool* es realitza aproximadament cada 30-50 segons.

Si el protocol d'RSK volgués aplicar aquesta implementació, ens trobaríem amb un clar problema, i és que a causa del fet que els blocs d'RSK es minen cada 30 segons aproximadament, fent una actualització de la *mempool* cada 50 segons estaríem minant treball vell.

La solució a aquest problema que es va plantejar i que actualment s'implementa és fer aquesta actualització cada cop que es realitza una nova transacció d'RSK. Això significa que cada cop que apareix una nova transacció del protocol RSK, el miner comprovarà si aquesta li interessa més que alguna de les transaccions amb les quals ja havia fet el bloc, ja sigui per un tema de comissions més altes o per qualsevol altre motiu. Tot i que aquesta implementació pot semblar un gran hàndicap cap als miners, doncs aplicar aquest procediment implica més exigència i moltes més interrupcions mentre s'intenta trobar una solució vàlida, els mateixos desenvolupadors del protocol RSK asseguren que s'han realitzat proves suficientment detallades de rendiment per assegurar que aquest canvi no generi cap impacte en el minat de Bitcoin. El refresc o actualització de l'estat de la *mempool* és molt més ràpid però no prou ràpid per a generar un impacte al rendiment.

Aquesta actualització es realitza a partir del *mining.notify message* que s'envia des de la *mempool* cap a tots els nodes d'RSK cada cop que es rep una nova transacció [5].

5.4 Inclusió del bloc d'RSK dins el bloc de Bitcoin

Així doncs, el primer pas que haurà de fer un miner de Bitcoin decidit a fer *Merged Mining* serà utilitzar l'RSK merge-mining plugin per tal de demanar el *Hash for Merged Mining* al node d'RSKj. Aquest hash serà l'identificador del bloc RSK que s'afegirà al bloc de Bitcoin.

A partir d'aquí, el miner es decidirà a crear un bloc de Bitcoin amb les transaccions de la *mempool* que més li interessin, crearà la capçalera amb els camps pertinents i podrà iniciar la cerca del *nonce* que faci complir la dificultat desitjada, de la mateixa forma que es faria amb el minat d'un bloc de Bitcoin que no implementi *Merged Mining*.

L'única diferència, però, d'aquest bloc de Bitcoin que permet el *Merged Mining* respecte a un bloc de Bitcoin normal i corrent és que aquest inclou la referència al bloc RSK que prèviament ha obtingut el miner a partir de l'RSK merge-mining plugin. Aquesta referència serà l'identificador *Hash for Merged Mining* que ha generat el node d'RSKj i s'afegirà a la *Coinbase Transaction* del bloc, doncs és l'única transacció que existeix per cada bloc de Bitcoin que es mina. I com que de la *Coinbase Transaction* l'únic important és l'adreça a qui es paga la recompensa del minat del bloc, no hi ha cap inconvenient en afegir informació extra a la transacció. Aquesta informació s'afegeix com a output de la *Coinbase Transaction* i segueix una sintaxi concreta que consisteix en:

"**RSKBLOCK:**" + hash de 20 bytes (*Hash for Merged Mining*) + informació de seguretat de 12 bytes

D'aquesta forma, tenim un bloc de Bitcoin que alhora té una referència al bloc d'RSK, i per tant ara es podrà buscar el *nonce* que faci complir la dificultat de Bitcoin o RSK perquè el bloc pugui ser enviat per la xarxa i es pugui validar la prova de treball.

- Exemple de transacció de Bitcoin que inclou la referència al bloc d'RSK: 0x449d99c34004f213983c9082552166546145ce23c0f37a8a2a43cca817

5.5 Com es valida el bloc d'RSK a partir del bloc de Bitcoin

Com hem vist, quan els miners de Bitcoin trobin una solució vàlida tant per la cadena de Bitcoin com per la d'RSK, enviaran aquest bloc al node d'RSKj, el qual serà l'encarregat d'extreure l'RSK tag que inclou el bloc de Bitcoin per poder recuperar el block d'RSK que té com a identificador el hash que inclou l'RSK tag.

Però abans de recuperar aquest bloc d'RSK a partir de l'RSK tag que inclou el bloc de Bitcoin, el node d'RSKj haurà de realitzar una sèrie de validacions:

1. Verificar que el bloc de Bitcoin que ha rebut compleixi amb la dificultat de la xarxa.
2. Mirar que el bloc inclou l'"RSKBLOCK:" amb el *Hash for Merged Mining* que se situa en algun output de la *Coinbase transaction*.
3. Validar que la capçalera del bloc realment inclou la*Coinbase transaction* que inclou l'RSK tag, és a dir, validar que fent el *Merkle tree* de totes les transaccions, obtinguem el *Merkle root* que s'ha utilitzat per generar la capçalera i el hash del bloc. D'aquesta forma, obtenim el *Merkle proof* respecte que la *Coinbase transaction* que té el *Hash for Merged Mining* efectivament pertanys al bloc que hem rebut.

Amb tot això validat, tindrem el *Proof of Work* del bloc d'RSK i per tant, el podrem recuperar perquè pugui ser propagat per la xarxa d'RSK.

5.6 Tot el procés en profunditat

Vistos els diferents processos intermedis amb una mica més de profunditat, fem una recapitulació de com funciona tot el procés: Tot comença amb el miner de Bitcoin interactuant amb el node d'RSKj a través de l'RSK merge-mining plugin i de la funció *getwork* per recollir la informació del bloc RSK a minar (Veure Figura 7).

Aquesta informació consisteix en l'identificador del bloc d'RSK quan encara no s'ha realitzat la prova de treball, el qual serà el hash de la capçalera d'aquest i s'identificarà com *Hash for Merged Mining*.

A continuació el miner recollirà totes les transaccions de Bitcoin que més li interessin incloure al bloc de Bitcoin que vol minar. Un cop les té recollides, crearà el bloc incloent l'identificador *Hash for Merged Mining* del bloc d'RSK a algun output de la *Coinbase transaction* del bloc de Bitcoin. A partir d'aquí, ja es podrà començar el procés de prova i error en busca del *nonce* que faci complir la dificultat d'alguna de les dues cadenes. Per cada prova de *nonce* que fem, ens podem trobar en diferents escenaris:

1. **El hash compleix la dificultat de Bitcoin:** Llavors el bloc de Bitcoin s'enviarà per la xarxa de Bitcoin. Alhora, com que la dificultat d'RSK és inferior a la de Bitcoin, el bloc d'RSK que inclou aquest bloc de Bitcoin també serà vàlid. Així doncs, també es proporcionarà el bloc de Bitcoin a l'RSK merge-mining plugin a través de la funció *submitWork()* perquè l'RSKj validi aquest bloc i si és vàlid, recuperi el bloc d'RSK a partir de la referència que té el bloc de Bitcoin, perquè pugui ser propagat per la xarxa d'RSK.

2. **El hash no compleix la dificultat de Bitcoin però si la d'RSK:** El bloc de Bitcoin no s'enviarà per la xarxa de Bitcoin, però sí al node d'RSKj a través de l'RSK merge-mining plugin perquè aquest el validi i recuperi el bloc RSK a partir de la referència que té el bloc de Bitcoin i pugui ser propagat per la xarxa d'RSK.

3. **No satisfà ni la dificultat de Bitcoin ni la d'RSK:** el bloc no s'envia a la xarxa de Bitcoin ni al node d'RSKj.

Aquest procés de recuperar el bloc d'RSK un cop s'ha trobat una solució vàlida, consisteix a agrupar el *Hash for Merged Mining* inclòs al bloc de Bitcoin, la capçalera del bloc de Bitcoin sobre el qual s'ha buscat la solució i informació per validar la prova de *Merged Mining*. Aquesta informació consisteix en l'identificador de la *Coinbase Transaction* del bloc de Bitcoin i el *Merkle path* que valida que aquesta *Coinbase Transaction* està inclosa al *Merkle tree* de les transaccions que inclou el bloc de Bitcoin, doncs d'aquesta forma es podrà validar que el *Hash for Merge Mining* del bloc d'RSK està inclòs al bloc de Bitcoin. Finalment, es podrà obtenir l'identificador final del bloc d'RSK un cop trobada la prova de treball vàlida. Aquest identificador sortirà de realitzar el procés de hashing sobre la capçalera del bloc d'RSK juntament amb la capçalera del bloc de Bitcoin mitjançant l'algorisme *keccak256* (Veure Figura 8).

6 RSK SMART BITCOIN A TRAVÉS DE POWPEG

L'RSK Smart Bitcoin (RBTC) és el token natiu de la plataforma RSK i és el que utilitza el protocol per pagar l'execució de les transaccions del mateix. Aquest token té una relació 1:1 amb el Bitcoin i per tant, comparteix atributs com per exemple, el subministrament total de 21 milions de tokens [6].

Així doncs, a causa d'aquesta relació 1:1 amb el Bitcoin, per tal de disposar d'RBTC sobre la cadena d'RSK, caldrà bloquejar la mateixa quantitat de tokens de la cadena de Bitcoin. Aquest procés actualment es realitza a través del Powpeg, el protocol que funciona com a pont bidireccional entre ambdues cadenes i que es basa en *Proof of Work* [6].

La solució que ofereix actualment el protocol Powpeg d'RSK ha evolucionat substancialment des de la primera implementació que es va plantejar cap al 2015, començant amb una idea més centralitzada però plantejant constants millores fins a arribar a una solució descentralitzada i alhora el més segura i confiable possible.

6.1 Evolució de les solucions per al POWPEG

6.1.1 2-way peg (2WP)

El *2-way peg* va ser el primer concepte a partir del qual es va començar a treballar sobre el procés de transferència d'un token d'una cadena a una altra. Aquesta implementació permet transferir el token natiu d'una blockchain a una cadena secundària i viceversa, tot i que aquesta transferència no existeix com a tal, ja que realment el que està passant és que el token natiu de la primera cadena queda bloquejat temporalment i alhora, es desbloquegen la mateixa quantitat de tokens a la segona cadena. De la mateixa manera, els tokens originals de la primera cadena podran ser desbloquejats quan la quantitat equivalent dels tokens de la segona cadena siguin bloquejats en aquesta. Aquest primer procés de transferència de tokens de la blockchain pare a la blockchain filla es coneix com a *Peg-in*, mentre que l'operació inversa es coneix com a *Peg-out*.

Davant d'aquesta idea, trobem un problema i és que si els miners de la cadena pare tenen més del 50% de la capacitat de càlcul de la cadena filla, aquests o una gran part d'aquests, tindrien poder suficient per censurar o presentar una prova de treball invàlida per tal d'aconseguir robar fons, atac conegut amb el nom de *51% Attack*. És necessari per tant, requerir el consens de tots els miners de la cadena pare per aconseguir un sistema segur.

Una possible evolució del *2-way peg* és un plantejament centralitzat, on tenim una entitat que emmagatzemarà i gestionarà els tokens d'ambdues cadenes per tal de bloquejar-los i desbloquejar-los quan es vulgui fer una transferència entre les dues cadenes. Però evidentment, aquesta solució centralitzada trenca amb tots els principis de la blockchain i per tant, no és una solució que s'arribi a contemplar.

6.1.2 Multi-signature Vault

Una altra possibilitat és realitzar aquest procediment de transferència mitjançant les transaccions amb multisignatura. La idea és tenir un sistema amb múltiples entitats, cadascuna d'elles controlant una única clau privada, de tal forma que aconseguim una descentralització i on necessitarem la majoria de les claus per poder realitzar una transferència. Aquesta implementació coneguda com *Multi-signature Vault*, va ser la primera implementació que va proposar RSK el 2015 mitjançant una federació de més de 15 nodes, cadascun d'ells en diferents parts del món, sobre diferents jurisdiccions governamentals, que no eren controlats per cap entitat i on, cadascun d'aquests nodes controla una clau privada.

6.1.3 Multi-signature Vault with HSM security

Sobre la proposta *Multi-signature Vault*, es va presentar una millora basada en la seguretat dels *Hardware Security Modules* (HSM), uns dispositius criptogràfics basats en hardware que generen, emmagatzemen i protegeixen claus criptogràfiques a més d'aportar una major velocitat per a realitzar operacions criptogràfiques. Aquesta proposta, coneguda com *Multi-signature Vault with HSM security*, va ser implementada per RSK a partir del 2019 i la idea era poder tenir una comunicació amb aquests dispositius per tal de demanar la firma d'una transacció a l'hora d'executar *Peg-outs*, però tenint la seguretat que no es podrà accedir a la clau privada d'aquests dispositius. A més, aquesta implementació permetia configurar els HSMs per tal de poder afegir restriccions com limitar el nombre de *Peg-outs*, protegint així el sistema en cas que s'hagi vulnerat la seguretat d'aquest.

6.1.4 Consensus-enforced HSM Multisig Vault

L'última proposta de millora del protocol es va presentar el 2020 i és la que actualment RSK implementa. A aquesta

se li ha donat el nom de *consensus enforced HSM multi-signature vote* o Powpeg. En aquesta implementació els mateixos dispositius HSM executen un node d'RSK (SPV node), i alhora, són capaços de seguir el *hash rate* del protocol. Per tant ara, en comptes de només requerir la signatura dels membres de la federació, també requerim una prova de treball del missatge perquè l'HSM el pugui firmar.

6.2 Característiques del PowPeg actual d'RSK

6.2.1 Defense-in-depth design

Una de les principals propietats del protocol és la filosofia del disseny que hi ha darrere. Aquest disseny, anomenat *Defense-in-depth* parteix de l'assumpció que cal estar preparat perquè les diferents capes del protocol puguin ser trencades a partir de vulnerabilitats. La idea doncs del *Defense-in-depth design* és treballar capa sobre capa per tal que el sistema mai depengui d'un simple component o una simple capa que en algun moment es pugui comprometre [7]. Diferenciem doncs, diferents participants de l'ecosistema:

- *Powpeg Pegnatories*: Es tracta d'uns usuaris especials de l'ecosistema que gestionen els HSM que guarden la clau privada. Els *Pegnatories* executen l'*RSK Powpeg node*, el qual és un *full node* d'RSK una mica més especial, ja que també es troba connectat amb el node de Bitcoin. És a través d'aquest *RSK Powpeg node*, a partir del qual els *Pegnatories* podran estar al corrent de les peticions i intents de *Peg-in* i *Peg-outs*.

- *Merge-miners*: Són els miners que produeixen els blocs i per tant, executen tant el node de Bitcoin com el node d'RSK.

- *Bridge contract*: El *Bridge contract* és un *Smart Contract* que s'executa a la cadena d'RSK i la seva funcionalitat es basa a verificar les peticions de *Peg-in* i organitzar els *Peg-outs*. Per tal de poder realitzar aquesta funcionalitat, és necessari que aquest *Smart Contract* gestioni una *wallet* de Bitcoin i en concret, gestiona una *wallet* de tipus *Simple Payment Verification* (SPV mode). Amb aquest tipus de wallet, les transaccions es validen a partir de l'identificador del bloc i alhora els blocs també són mínimament validats. Per tant, aquesta implementació valida la continuïtat de la cadena, però no valida la certesa d'aquesta i és per això, la necessitat del *SPV Proof* que cal proporcionar al *Bridge contract* quan es realitza una petició tant de *Peg-in* com de *Peg-out*.

- *Economic actors*: són usuaris individuals, *exchanges*, comerciants que voldran realitzar *Peg-in* i *Peg-outs* i que per tant, també necessiten executar l'*RSK full node* per tal de poder interactuar amb el *Bridge contract* per realitzar les transaccions tant de *Peg-in* com de *Peg-out*.

- *Armadillo Monitor*: Es tracta d'un sistema de monitoratge d'intents de *fork* sobre la blockchain d'RSK. Aquest sistema executa tant el node de Bitcoin com el d'RSK i és capaç de detectar i rastrejar qualsevol intent de *fork* sobre la cadena d'RSK, encara que aquest intent tingui una gran quantitat de *hash rate*. L'objectiu del sistema és d'informar del possible intent de *fork* que pugui detectar a tots els participants de l'ecosistema perquè aquests puguin actuar davant la situació. Com hem vist abans, el fet que la cadena pare tingui molt més poder de còmput que la cadena filla, posa sobre la taula l'opció de realitzar actes maliciosos i acabar guanyant. És per això que el sistema *Armadillo Monitor*, és un component clau de la filosofia *Defense-in-depth* que hi ha darrere el protocol Powpeg per tal de detectar possibles males intencions.

6.2.2 Code transparency through attestation

La idea d'aquesta propietat és afegir una capa de *firmware attestation* sobre tot el sistema, de tal forma que qualsevol usuari de la comunitat i per tant, no només els *Pegnatories*, podrà provar que les HSM realment estan executant el software que diuen que estan executant. Tot i que aquesta característica encara no està implementada, ja ha sigut acceptada per la comunitat. La idea darrere d'aquesta propietat és afegir un tipus de missatge de certificació especial a la blockchain perquè aquesta certificació assegurant que els HSM estan executant el software que diuen que estan executant sigui públic a la cadena [8].

6.3 Peg-in

Com hem vist abans, el procés de *Peg-in* consisteix a bloquejar el token natiu de la cadena de Bitcoin per poder desbloquejar la mateixa quantitat de tokens a la cadena d'RSK. Així doncs, aquest procés (veure Figura 2) comença amb un usuari de l'ecosistema de Bitcoin que vol convertir els seus Bitcoins en RSK Smart Bitcoins. Per tant, aquest usuari primer realitza una transacció en la qual envia una certa quantitat de Bitcoin a una adreça concreta. Aquest destinatari és l'anomenat *Bitcoin Powpeg Voult* i és controlat pels HSMs a través de multi signatures.

Paral·lelament, tenim els *Pegnatories* que al marge de gestionar els HSM, també tenen el rol d'actuar com a torres de vigilància per tal de controlar i estar al cas de totes les transaccions que envien fons al *Bitcoin Powpeg Voult*. Els *Pegnatories* estaran al cas d'aquestes transaccions, perquè quan aquestes transaccions que envien Bitcoin al *Bitcoin Powpeg Voult* tinguin 100 confirmacions (és a dir que s'han minat 100 blocs sobre el bloc que inclou aquesta transacció), els *Pegnatories* presentaran un *SPV proof* format pels identificadors dels 100 blocs que han servit de confirmació, l'identificador del bloc de Bitcoin que inclou la translació de Bitcoin que envia fons al *Bitcoin Powpeg Voult* i informació de la mateixa transacció.

Aquest procés es realitza a través del *Bridge contract* i en concret, els *Pegnatories* utilitzaran les funcions *receiveHeaders()* i *registerBtcTransaction()* del contracte. Amb aquestes funcions, podran informar al contracte sobre l'existència del *Peg-in* per tal que el contracte pugui validar aquesta transacció i pugui desbloquejar RBTC a la cadena d'RSK.

Una de les restriccions que té aquest procés, és la necessitat que el receptor dels RBTC que es desbloquejaran a la cadena d'RSK, sigui el mateix usuari que ha realitzat

la transacció a la cadena de Bitcoin. A la pràctica, aquesta restricció es tradueix en el fet que els RBTC es desbloquejaran a l'adreça d'RSK que tingui la mateixa clau privada que l'adreça des d'on s'han enviat els BTC i viceversa, doncs la clau privada serà només coneguda per l'usuari que ha enviat els Bitcoins i per tant, la podrà utilitzar per accedir als seus RSK Smart Bitcoins.

Amb aquesta restricció, veiem clarament que un usuari que disposa de Bitcoin en un servei amb custòdia de claus, per exemple en un *exchange* i que vol realitzar un *Peg-in*, primer haurà de moure els seus fons a una *wallet* externa per tal que aquests siguin de la seva propietat i en conseqüència, disposi de la clau privada que li donarà accés als RSK Smart Bitcoin desbloquejats.

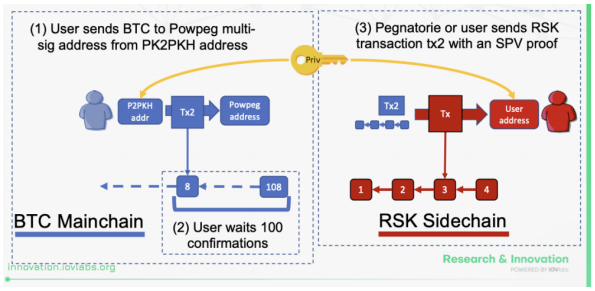

Fig. 2: Esquema procés de *Peg-in* [9].

6.4 Peg-out

Com que els tokens de les dues cadenes tenen una relació 1:1, és vital assegurar la finalitat de les transaccions de transferència entre una cadena i l'altra per tal de mantenir el total equilibri dels tokens bloquejats i desbloquejats. I l'única forma per tal d'assegurar aquesta finalitat de les transaccions és a partir d'un gran nombre de confirmacions.

El *Peg-out* (veure Figura 3) comença amb un usuari de la cadena d'RSK enviant els seus RSK Smart Bitcoin a una adreça concreta. Internament, el que està succeint és que s'està cridant al mètode *releaseBtc()* del *Bridge Contract* per tal de realitzar la petició de transferència de tokens de la cadena d'RSK a la cadena de Bitcoin. El contracte, generarà una transacció de Bitcoin, triant els UTXOs que serviran com a inputs de la transacció i triant els outputs, però no signarà aquesta transacció. Aquest procés s'identificarà com l'*event logReleaseBtcRequested* i guardarà la transacció de Bitcoin creada.

El següent pas serà esperar 4000 confirmacions d'RSK a aquest *event*, el que aproximadament són unes 30 hores. Passades aquestes confirmacions, els *Pegnatories* poden agafar aquest *event* i crear un *SPV proof* amb cada una de les confirmacions (és a dir, els identificadors dels 4000 blocs) i enviar-ho als PowHSMs juntament amb la transacció de Bitcoin que ha creat el contracte i que cal firmar.

Els PowHSMs validaran el *SPV proof*, i en cas de ser vàlid, oferiran la signatura als *Pegnatories* perquè aquests puguin cridar a la funció *addsignature()* del *Bridge Contract* per tal d'afegir al contracte la signatura i perquè aquest acabi de muntar la transacció. Aquesta transacció de Bitcoin ja firmada, es trobarà a l'*event logReleaseBtc* que ha creat el contracte i a partir d'aquest moment, qualsevol usuari ja podrà propagar aquesta transacció per la xarxa de Bitcoin.

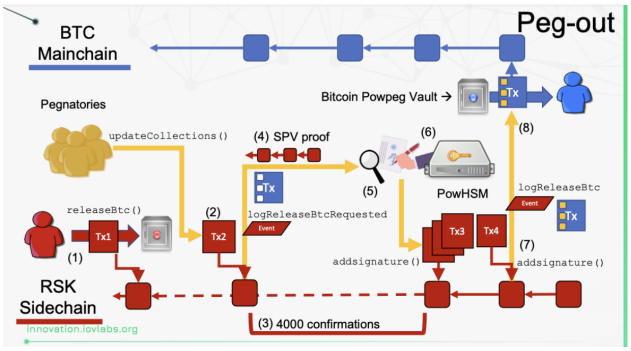

Fig. 3: Esquema procés de *Peg-out* [9].

7 RSK Virtual Machine

L'*RSK Virtual Machine* (RVM) és el nucli principal de l'execució d'*Smart Contracts* a l'ecosistema d'RSK. Els *Smart Contracts* són executats per cada un dels *full nodes* de la xarxa d'RSK i l'execució d'aquests ha de complir amb les següents propietats: Determinista: L'execució de l'*Smart Contract* ha de produir sempre el mateix resultat donat el mateix input. Terminable: Ha de tenir mecanismes que permetin parar la seva execució. Aïllat: Ha de ser executat en un lloc aïllat de tal forma que possibles efectes negatius no puguin afectar a l'ecosistema.

Per tal de complir amb aquestes propietats, els *Smart Contracts* són executats concretament en màquines virtuals, i en el cas d'RSK, en l'anomenada RVM. Aquesta RVM té un gran nombre de característiques compartides amb la màquina virtual d'Ethereum, l'EVM. Una d'aquestes característiques és la possibilitat que els *Smart Contracts* d'Ethereum puguin ser executats sobre l'ecosistema d'RSK i viceversa, doncs a nivell de com s'interpreten les instruccions del codi i les interfícies, trobem compatibilitat absoluta [10][11].

Gràcies a aquesta compatibilitat, la migració d'*Smart Contracts* d'un ecosistema a l'altre és un procés que no exigeix pràcticament cap feina extra i pot oferir no només beneficis econòmics, sinó també de velocitat i capacitat de processament. Alhora, un dels objectius principals d'aquesta característica és facilitar als desenvolupadors d'*Smart Contract* d'Ethereum la seva migració cap a l'ecosistema d'RSK sense que això impliqui un canvi en la seva forma de treballar. Per altra banda, mentre l'EVM utilitza el gas d'Ethereum per pagar l'execució i interacció amb els *Smart Contracts*, l'RVM utilitza l'RBTC, el token natiu de la plataforma d'RSK.

8 Desenvolupament pràctic:

Tal com s'indica al punt d'objectius del treball, una part d'aquests consistia en un desenvolupament pràctic desplegant certs *Smart Contracts* sobre la xarxa d'RSK així com una Dapp per poder interactuar amb aquests. La idea principal era la de replicar de forma senzilla les funcionalitats principals d'un DEX, que són *exchanges* de criptomonedes descentralitzats, que permeten als usuaris poder intercanviar fons entre ells sense haver de donar el control d'aquests a cap intermediari o custodi. I això és possible gràcies als usuaris que ofereixen liquiditat, anomenats *Liquidity Provi-*

ders, i que bloquegen la quantitat de fons que volen aportar enviant-los a *Smart Contracts* complexos capaços de gestionar totes aquestes operacions, anomenats *router contracts*. I a partir d'aquesta liquiditat aportada, es podran realitzar transaccions d'intercanvi de fons. En aquestes es cobrarà a l'usuari que vulgui realitzar l'intercanvi una petita comissió que servirà per recompensar als usuaris que aporten la liquiditat i que estan bloquejant els seus fons per tal de permetre l'intercanvi.

Així doncs, per poder desenvolupar aquestes funcionalitats, s'ha treballat amb el *router contract* del protocol Uniswap, el qual està desplegat tant en la xarxa de proves com la xarxa principal d'RSK i que inclou totes les funcions necessàries per poder aportar liquiditat, realitzar l'intercanvi de tokens o poder consultar la relació de preu entre diferents tokens, entre d'altres[12]. I al marge d'aquest contracte, l'objectiu era poder desplegar dos contractes més que fossin tokens ERC20 i que serien els quals implementaria la Dapp per tal de poder aportar liquiditat d'aquests així com realitzar intercanvis entre aquests.

8.1 Desplegament dels tokens

El primer pas d'aquesta part pràctica va ser la creació dels tokens ERC20. Per tal de seguir l'estàndard vaig utilitzar la llibreria que ofereix OpenZeppelin, la qual vaig poder importar des dels contractes. A partir de la llibreria, vaig definir l'herència del contracte ERC20 de la llibreria per tal de poder heretar els atributs i funcions que defineix l'estàndard com les funcions per obtenir el subministrament total del token, poder fer transferències d'aquests, o saber el balanç d'aquest token d'una adreça concreta, entre d'altres. Al marge d'aquesta herència, al constructor del contracte es defineix el nom del token, el símbol d'aquest i també es crida a la funció privada *_mint()* passant com a paràmetres l'adreça que desplega el contracte i el subministrament total del token. D'aquesta forma, es generaran el total dels tokens que defineix la variable de subministrament total a l'adreça que ha desplegat el contracte, fent així que aquesta adreça disposi del 100 % dels tokens disponibles [13] [14].

Per tal de poder fer proves d'interacció amb el contracte, comprovar que tot funciona correctament i desplegar-lo, vaig utilitzar l'eina Remix IDE, la qual ofereix un entorn per escriure, compilar i depurar *Smart contracts* de forma fàcil tant de forma local com a través de la blockchain que es configuri. Tenint el contracte desenvolupat i assegurant que compila sense errors, vaig desplegar-lo a la xarxa de proves pagant les comissions de la transacció amb el token natiu de la xarxa d'RSK com és l'RBTC. Per desplegar el contracte, cal indicar també el subministrament total que volem que tingui aquest token, doncs el constructor del contracte rep com a paràmetres aquest valor per poder generar el total dels tokens.

Listing 1: Codi del *Smart Contract* del token A

```
pragma solidity ^0.8.1;

import "@openzeppelin/contracts/token
/ERC20/ERC20.sol";

contract tokenA is ERC20 {

    constructor(uint256 initialSupply)
    public ERC20("tokenA", "TKA") {
        _mint(msg.sender, initialSupply *
        (10 ** 18));
    }
}
```

8.2 Decentralized Exchange (DEX)

L'altra part del desenvolupament pràctic va ser el desenvolupament d'una Dapp que repliqués certes funcionalitats de les DEXs, com ara afegir liquiditats dels dos tokens creats o bé poder intercanviar-los entre ells. Per desenvolupar aquesta Dapp, es va decidir utilitzar el framework Vue.js juntament amb una plantilla de Creative Tim [15].

La vista de l'aplicació consisteix en dues pàgines, la principal on s'agrupen les *pools* de liquiditat dels tokens i una segona pàgina que permet fer l'intercanvi dels tokens. Alhora, la vista de les diferents opcions de l'aplicació consisteix en un primer botó per aprovar l'acció i un segon per realitzar-la. Aquest botó per aprovar l'acció és necessari perquè l'usuari aprovi que l'aplicació pugui gastar els seus tokens d'entrada.
Totes aquestes accions com la d'aprovar els tokens a gastar o totes les crides a les funcions del contracte es realitzen a través de la llibreria Web3, la qual permet interactuar amb *Smart Contracts* desplegats juntament amb l'extensió de navegador Metamask que permet interactuar amb diferents blockchains com la d'RSK. Aquesta extensió permet gestionar una *wallet* per poder connectar-la amb diferents Dapps de l'ecosistema [16] [17].

8.2.1 Liquidity Pools

Pel desenvolupament d'aquesta funcionalitat, es va suposar l'escenari en el qual l'usuari que utilitzarà l'aplicació disposa o bé de tokens A o bé de tokens B i per tant, en cas que vulgui aportar liquiditat d'aquests tokens, ho farà juntament amb el token natiu de la blockchain d'RSK, l'RBTC. Així doncs, aquesta acció consistirà primer a indicar la quantitat que es desitgi aportar del token de la *pool* juntament amb la quantitat de valor equivalent del token RBTC. Seguidament caldrà aprovar la despesa d'aquests tokens a través de la crida a la funció *approve()* del contracte del token que es vulgui gastar, i finalment realitzar la crida a la funció que aportarà la liquiditat al contracte. Aquesta crida consisteix en la funció *AddLiquidityETH()* del *router contract* passant com a paràmetres les diferents quantitats a afegir, les adreces dels diferents tokens i l'adreça origen i destí d'aquesta transacció.

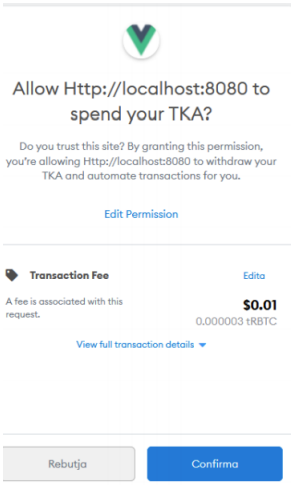

Fig. 4: Acció d'aprovar la despesa dels tokens d'entrada a través de Metamask.

8.2.2 Swap

Gràcies a la liquiditat aportada dels diferents tokens, és possible l'intercanvi entre ells i aquest, vindrà definit pel token d'entrada i sortida que indiqui l'usuari així com la quantitat d'entrada i de sortida que indiqui.
Aquesta acció es realitzarà a través de la funció *swapExactTokensForTokens()* del *router contract* passant com a paràmetres la quantitat d'entrada i sortida dels diferents tokens, l'adreça del contracte d'ambos tokens i l'adreça origen i destí de la transacció.
La vista d'aquesta funcionalitat consisteix en dues opcions d'entrada de la quantitat i el tipus de token tant d'entrada com de sortida, ja que la suposició de l'escenari de l'usuari que utilitzarà aquesta funcionalitat és un l'usuari que disposa o bé de tokens A o bé de tokens B i que voldrà intercanviar certa quantitat del token que disposa per obtenir quantitat de l'altre a canvi.

8.2.3 Càlcul Automatic de la conversió

Per tal que les transaccions d'interacció amb el contracte siguin correctes i no fallin, l'aplicació calcula la conversió dels diferents tokens en funció de la relació del preu actual entre ells que defineix la *pool*. Aquesta característica és una millora important ja que la crida a les funcions del contracte requereixen que els valors d'entrada que es passen com a paràmetres siguin proporcionals quant a valor pels diferents tokens, i en cas que no sigui així la transacció es rebutjarà. Així doncs, amb aquesta millora, quan un usuari vulgui afegir liquiditat del token A o del token B introduirà la quantitat de tokens d'entrada que vulgui afegir a la *pool* i automàticament es calcularà la quantitat d'RBTC que també caldrà afegir per tal de tenir una quantitat equivalent dels dos tokens. De la mateixa forma, en la funcionalitat que permet l'intercanvi de tokens l'usuari indicarà el token d'entrada, el token que vol de sortida i la quantitat d'entrada que voldrà aportar perquè l'aplicació calculi automàticament la quantitat de sortida que rebrà l'usuari. La idea principal d'aquesta característica és facilitar l'ús de l'aplicació evitant que l'usuari hagi de fer el càlcul de la relació de preu entre els tokens a més d'assegurar una correcta crida de la funció del contracte per tal de garantir que cap transacció es rebutjarà per un possible error de càlcul humà. Aquest càlcul automàtic es realitza a través de la crida a la funció *getAmountsIn()* del contracte que fa de router i que retorna l'equivalència dels tokens que es passen com a paràmetre en funció de la liquiditat actual de la *pool*.

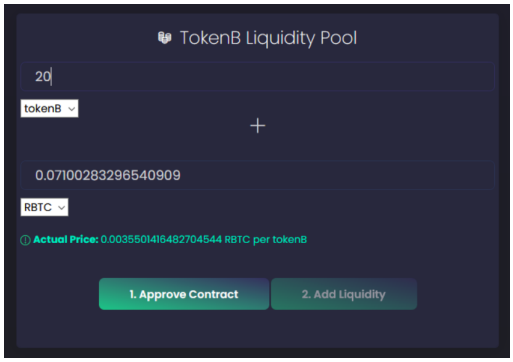

Fig. 5: Càlcul automàtic de la quantitat de RBTC equivalent a la quantitat de tokens B d'entrada.

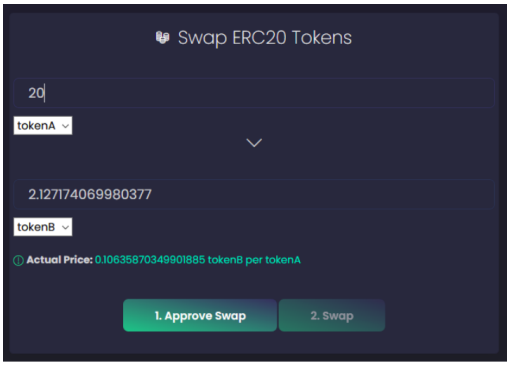

Fig. 6: Càlcul automàtic de la quantitat de tokens de sortida respecte a la quantitat de tokens d'entrada.

9 Conclusions

Pel que fa a les conclusions del treball desenvolupat, es poden diferenciar entre les conclusions extretes de la part teòrica del treball i les conclusions extretes de la part pràctica desenvolupada.
Per la banda més teòrica del treball com ha sigut l'estudi del protocol, he pogut assolir el coneixement pel que fa a què l'ecosistema d'RSK té estructura tecnològica ben desenvolupada, vist des del punt de vista de la millora que proposa sobre l'ecosistema base que conforma la blockchain de Bitcoin. Aquesta proposta, parteix de la seguretat i robustesa de la blockchain de Bitcoin mitjançant la tècnica de *Merged Mining* mentre que alhora aporta la millora tecnològica dels *Smart Contracts*. Aquesta bona estructura tecnològica, alhora es demostra a partir del fet que l'ecosistema d'RSK ha aconseguit el suport d'un gran volum dels miners de Bitcoin (actualment, aproximadament el 60% dels miners de Bitcoin participen en el *Merged Mining* d'RSK) garantint així, la integritat, estructura i continuïtat del protocol.
Una altra conclusió extreta de l'estudi del protocol, és el punt de confiança sobre la descentralització d'aquest, ja que la solució basada en la federació de *Pegnatories* ha sigut criticada per la comunitat de no ser del tot descentralitzada. Aquesta solució, tot i basar-se en poc més de 15 entitats úniques, depèn de dispositius *Hardware Security Modules* (HSM), els quals permeten emmagatzemar i protegir claus

criptogràfiques a més de fer operacions criptogràfiques com firmar transaccions, però garantint que ningú pugui accedir a les claus privades dels dispositius. Així doncs, amb aquesta solució, es garanteix que ni tan sols les mateixes entitats que gestionen els dispositius HSM poden actuar de forma malintencionada per intentar robar fons, per exemple.
Per la banda més pràctica, s'ha aconseguit desenvolupar una versió funcional de la Dapp, aconseguint el correcte funcionament de les diferents funcionalitats que s'han explicat en la secció anterior.

9.1 Línies Futures

Una línia futura interessant del projecte podria ser profunditzar en el funcionament del sistema de monitoratge *Armadillo monitor*. Segons el *Roadmap* d'RSK, durant aquest 2021 hi haurà una millora de versió d'aquest sistema, actualitzant així a la versió 2.0, el que aportarà una millora al protocol perquè aquest estigui actualitzat a noves tècniques i intents de *fork* de la cadena. Penso que pot ser bastant interessant profunditzar en el funcionament d'aquest per tal d'entendre com implementa les tècniques per defensar-se davant d'intents d'actes maliciosos.
Una altra proposta per donar continuïtat al treball o per seguir profunditzant en aquest, podria ser el de seguir de prop el desenvolupament de la característica *Garbage Collector v1.0* que RSK Labs també té intenció de fer pública aquest 2021 i que ja sembla estar funcionant en versió beta. Aquesta característica té com a objectiu millorar el node RSKj a partir de recuperar i desfer-se de dades històriques que ja no són crítiques per al funcionament del node, aconseguint així reduir l'espai en disc necessari. I aquesta característica sembla molt interessant, ja que un dels problemes actuals del desplegament d'un node complet d'una blockchain és que aquest implica una gran quantitat d'espai a disc, com ho demostren els gairebé 350 GB que ocupa la blockchain de Bitcoin, per exemple. Així que aquesta característica, pot ser una gran solució per millorar aquest problema [18].

AGRAÏMENTS

Al tutor d'aquest Treball de Final de Grau, Jordi Herrera, tant pel que fa al suport i guia durant en el desenvolupament del projecte com per introduir-nos de la millor manera en aquest meravellós món de la blockchain i les criptomonedes.
A la família, sobretot als meus pares, pel suport incondicional des del primer dia fent que això hagi estat possible.
I finalment als amics, els quals m'han donat els moments de desconnexió necessaris per seguir tirant endavant quan les coses es complicaven durant aquests quatre anys.

REFERÈNCIES

[1] R. Fund, "RSK - Merged Mining", Mining.rsk.co, 2021. [Online]. Available: https://mining.rsk.co/.

[2] "proof-of-stake", Ethereum Wiki, 2021. [Online]. Available: https://eth.wiki/en/concepts/proof-of-stake-faqs.

[3] Çambridge Bitcoin Electricity Consumption Index (CBECI)", Cbeci.org, 2021. [Online]. Available: https://cbeci.org/.

[4] "RSK: Bitcoin Merge Mining is Here to Stay — RSK - Smart Contract Platform Secured by the Bitcoin Network", Blog.rsk.co, 2021. [Online]. Available: https://blog.rsk.co/noticia/rsk-bitcoin-merge-mining-is-here-to-stay/.

[5] R. Portal, Ïmplementation Guide - RSK Developers Portal", Developers.rsk.co, 2021. [Online]. Available: https://developers.rsk.co/rsk/architecture/mining/implementation-guide/.

[6] R. Portal, "RBTC Token - RSK Developers Portal", Developers.rsk.co, 2021. [Online]. Available: https://developers.rsk.co/rsk/rbtc/.

[7] R. Portal, "Powpeg - RSK Developers Portal", Developers.rsk.co, 2021. [Online]. Available: https://developers.rsk.co/rsk/architecture/powpeg/.

[8] R. Portal, "Security model - RSK Developers Portal", Developers.rsk.co, 2021. [Online]. Available: https://developers.rsk.co/rsk/architecture/security/.

[9] "Webinar: Powpeg: the most secure, permissionless and uncensorable Bitcoin peg", Youtube.com,2021. [Online].

[10] R. Portal, "Turing complete - RSK Developers Portal", Developers.rsk.co, 2021. [Online]. Available: https://developers.rsk.co/rsk/architecture/turing-complete/.

[11] "RSK y los Smart Contract en Bitcoin con Diego Gutiérrez Zaldívar", Youtube.com, 2021. [Online]. Available: https://www.youtube.com/watch?v=FSg2j8w7vbI.

[12] Uniswap.org, 2021. [Online]. Available: https://uniswap.org/docs/v2/smart-contracts/router02/.

[13] E. Proposals, "EIP-20: ERC-20 Token Standard", Ethereum Improvement Proposals, 2021. [Online]. Available: https://eips.ethereum.org/EIPS/eip-20.

[14] "OpenZeppelin/openzeppelin-contracts", GitHub, 2021. [Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20 /ERC20.sol.

[15] "Vue.js", Vuejs.org, 2021. [Online]. Available: https://vuejs.org/.

[16] "web3.js - Ethereum JavaScript API — web3.js 1.0.0 documentation", Web3js.readthedocs.io, 2021. [Online]. Available: https://web3js.readthedocs.io/en/v1.3.4/.

[17] "MetaMask", Metamask.io, 2021. [Online]. Available: https://metamask.io/.

[18] "RSK - Development Roadmap", Rsk.co, 2021. [Online]. Available: https://www.rsk.co/development-roadmap.

APÈNDIX

A.1 Comparació entre el minat de blocs de Bitcoin i RSK

TAULA 1: NOMBRE MITJÀ DE PROVES DE *nonces* DIFERENTS PER TROBAR UNA SOLUCIÓ DEL BLOC VÀLIDA

Protocol	Temps mitjà del minat del bloc	Nombre mitjà d'iteracions per trobar una solució	Supòsits
Bitcoin	10 minuts	2^{74}	100% Bitcoin hashrate
RSK	30 segons	2^{69}	50% merge-mining

TAULA 2: OBJECTIUS DE DIFICULTAT APROXIMATS PER AMBDUES BLOCKCHAIN EL MATEIX DIA

Blockchain	Target
Bitcoin	00000000000000000165exxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
RSK	0000000000000000db5a4xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

A.2 Actors que participen en el *Merged Mining*

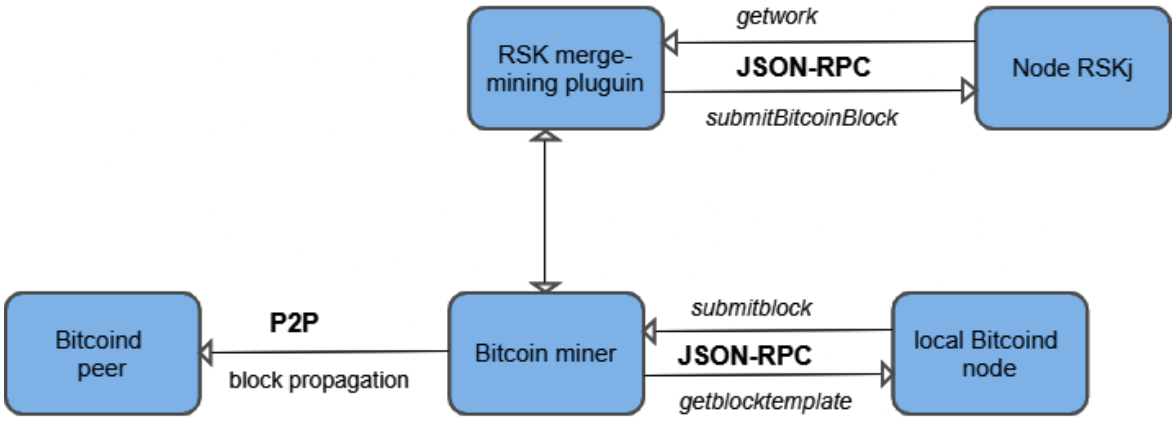

Fig. 7: Diagrama de la interacció del miner de Bitcoin amb el node RSKj i la xarxa de Bitcoin.

A.3 Esquema de la inserció de l'*RSK tag* a la *Coinbase transaction* del bloc de Bitcoin

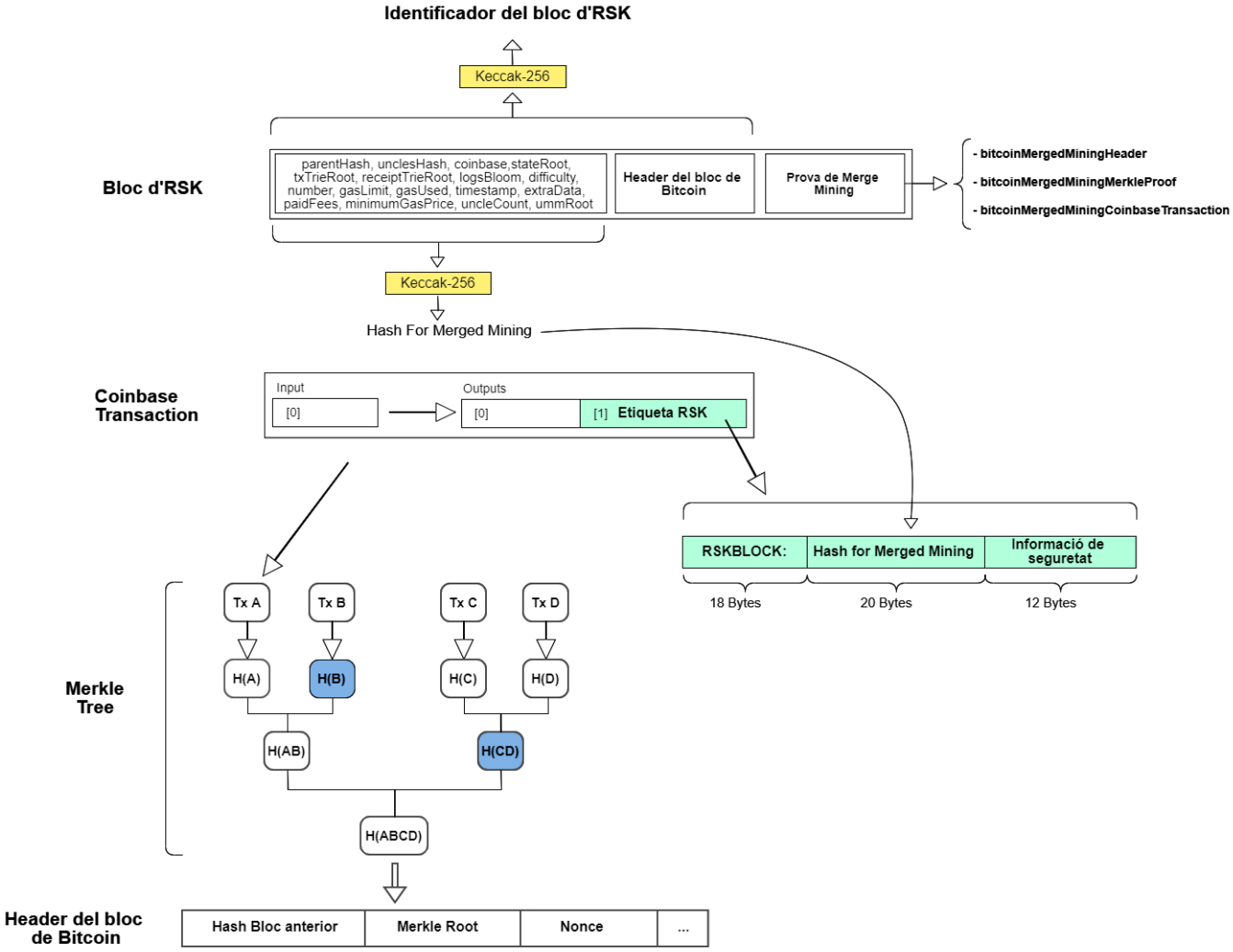

Fig. 8: Diagrama del *Header* del bloc de Bitcoin que inclou el *Merkle Tree* que inclou la *Coinbase transaction* que alhora inclou l'*RSK tag* a l'output de la transacció.

TAULA 3: TAULA DE CONCEPTES

Conceptes	**Descripció**
Mempool	És on totes les transaccions vàlides esperen ser confirmades per la xarxa Bitcoin.
sidechain	És una blockchain alternativa que intenta millorar les prestacions d'una o blockchain ja existent.
Proof of Work	És un mecanisme de consens descentralitzat que requereix que els membres d'una xarxa realitzin algun tipus de treball resolent un trencaclosques matemàtic arbitrari per evitar comportaments indesitjats.
Double-spending Attack	És una possible vulnerabilitat del sistema que sorgeix en fer transaccions amb la mateixa moneda d'origen més d'una vegada. Diverses transaccions que comparteixen la mateixa entrada i que es transmet a la xarxa poden ser problemàtiques i són un defecte exclusiu dels sistemes basats en criptomonedes.
Merkle tree	És una estructura de dades en forma d'arbre binari utilitzada en criptografia i en informàtica. Consta d'un node arrel, un conjunt ordenat de fulles, i, entremig, un conjunt de nodes.
Merkle root	És el node arrel de l'arbre que s'obté al realitzar el hash de cada conjunt de fulles de cada nivell del *merkle tree*. Aquest *merkle root* podrà validar que totes les fulles pertanyen a l'arbre.
Nonce	Es tracta d'un nombre aleatori que s'inclou a la capçalera del bloc de Bitcoin. Aquest nombre és el que busquen els miners per tal de complir que el hash de la capçalera del bloc que estan minant compleix amb la dificultat actual de la cadena.
Full node	És un node de la xarxa que valida completament les transaccions i els blocs. Gairebé tots els *full nodes* ajuden a la xarxa acceptant transaccions i blocs d'altres nodes, validant aquestes transaccions i blocs per a després transmetre'ls a altres nodes.
Coinbase Transaction	És un tipus especial de transacció que es produeix a cada bloc de la blockchain. Aquest tipus de transacció forma part del sistema per posar en circulació noves monedes que mai s'han gastat i els miners acostumen a cobrar-les per la feina feta durant el minat del bloc.
Hash rate	Fa referència a la potència de càlcul d'un miner de criptomonedes per trobar solucions en funció a un hash criptogràfic específic.
Blockchain fork	És un canvi en el protocol o una divergència de la versió anterior de blockchain. Quan un miner genera un bloc alternatiu amb fins fraudulents, el sistema consensua la invalidesa de l'-bloc, de manera que la resta de miners abandona aquest "bloc orfe"ràpidament.
Decentralized Exchange (DEX)	Són un tipus d'*exchanges* de criptomonedes que permet realitzar transaccions directes de criptomonedes entre usuaris de manera segura i sense la necessitat d'un intermediari.
Liquidity Pool	Són agrupacions de tokens bloquejats en *Smart Contracts* per proporcionar liquiditat en DEXs per intentar atenuar els problemes causats per la poca liquiditat.
51% Attack	Un atac del 51% consisteix en un atac a una blockchain per part d'un grup de miners que controla més del 50% del *hash rate* o potència de càlcul de la xarxa.