Cuprins

1 Introducere **3**

2 Blockchain **5**
2.1 Introducere 5
2.2 Noduri 6
2.3 Permissionless & Permissioned 10
2.4 Elemente criptografice 12
2.5 Structura 14
2.6 Algoritmi de consens 16
2.7 Motivație pentru utilizare 19

3 BigchainDB **21**
3.1 Introducere 21
3.2 Proprietăți 22
3.3 Utilizare 24

4 Zero-Knowledge Succinct Non-Interactive Argument of Knowledge **27**
4.1 Introducere 27
4.2 Concepte criptografice 27
4.3 Zero-Knowledge Proofs (ZKPs) 30
4.4 zkSNARKs 31
4.5 Quadratic Programs 32
4.6 De la QAP la zkSNARKs 35

5 TrustNews **37**
5.1 Descriere 37
5.2 Modul de funcționare 40
5.3 Exemplu de funcționare 43
5.4 Planuri de îmbunătățire 45

6 Concluzii **46**

Listă de figuri **47**

Bibliografie **48**

Capitolul 1

Introducere

Noțiunea de știri false (*fake news*) a devenit de câțiva ani un fenomen din ce în ce mai răspândit pe toate mediile de comunicare. Efectele periculoase ale ei se întind de la o simplă preluare a unei informații greșite mai mult sau mai puțin voite până la adevărate campanii de dezinformare sau propagandă în favoarea cuiva.

Acest lucru a făcut ca și termenul de fake news să fie definit de ideea fabricării intenționate a unei știri sau informații care nu are o baza reală și care se poate dovedi falsă, scopul fiind derutarea cititorilor de la adevărata informație [31].

Încrederea revine companiilor de media ce descoperă informația și o împachetează și în final la oamenii ce o pun la dispoziția largă a publicului. Companiile sunt responsabile de informația pe care o prelucrează și de securitatea mediilor prin care aceasta este livrată.

Problema intervine atunci când vorbim de mediul online, care în acest context este format atât din paginile web respective unor trusturi de media cât și din influența mediilor de socializare. Oricine ar trebui să aibă acces la informație însă acest lucru poate să creeaze o portiță de exploatare atacatorilor. Dacă securitatea mediilor de publicare și stocare a informațiilor este slabă atunci ei pot altera conținutul și se pot folosi de mecanisme de analiză socială pentru a schimba esența informației.

Teza de disertație constă în propunerea și crearea unei faze incipiente a unei aplicații concept ce se folosește de mecanisme criptografice puternice precum blockchain și zero-knowledge proofs pentru a adăuga securitate celor două zone de impact (stocare și publicare).

În Capitolul 1 încep să prezint una din tehnologiile importante din componența aplicației TrustNews și anume blockchain. Capitolul definește conceptele de bază ale tehnologiei, tipurile de blockchain și diferențele între acestea, structura, motivul pentru care a devenit atât de rapid popular și care sunt situațiile în care el chiar ne poate ajuta.

Capitolul 2 merge un pic mai departe și prezintă BigchainDB, o tehnologie dezvoltată pe baza blockchain și care face trecerea ușoară de la clasicele baze de date către un nou concept. Voi vorbi despre ce facilități aduce și de ce este util în anumite aplicații.

Capitolul 3 curpinde în detaliu elementele ce alcătuiesc un Zero-Knowledge Succint Non-Interactive Argument of Knowledge(zk-SNARK). Prezintă proprietățile, componentele sale în detaliu și procesele prin care anumite componente trec de la o formă la alta.

În Capitolul 4 prezint aplicația concept prin care tehnologiile menționate pot fi folosite concomitent. Scopul este de a crea o formă mai sigură și mai eficientă decât un mediu online clasic de unde publicul poate lua informații și unde încrederea este oferită de instrumentele criptografice puternice.

Tehnologiile folosite sunt în continuă dezvoltare și astfel pot să apară modificări atât la modul de lucru cu anumite tehnologii cât și la motivația de a le folosi. Aplicația nu își propune să îmbunătățească securitatea actualelor medii online de unde preia informația ci se bazează pe încrederea și reputația companiilor de publicitate că nu publică o informație falsă.

Capitolul 2

Blockchain

2.1 Introducere

Blochain-ul poate fi privit ca un registru digital de dimensiuni foarte mari ce este rezistent la manipularea datelor ce îl compun. Are un caracter distribuit, lucru evidențiat de lipsa unui spațiu unic în care sunt stocate datele. De asemenea nu conține o singură entitate centrală, încrederea este oferită de procesele ce se desfășoară în interiorul lui, și la care participă, în anumite cazuri, toți utilizatorii din rețea.

Natura lui oferă posibilitatea unei comunități de utilizatori să efectueze trazactii și să le înregistreze în blockchain fiind apoi publice și verificabile de către oricine. Odată publicate tranzacțiile nu se pot modifica [25].

Conceptul de blockchain a început să devină din ce în ce mai popular odată cu apariția primei criptomonede numite Bitcoin în 2008. De atunci a început să se pună problema unei forme de stocare a banilor în mod electronic sigur și fără a depinde de o entitate centrală precum o bancă.

Vom vedea în capitolul despre algorimii de consens că siguranța și validitatea informațiilor stocate se păstrează datorită implicării multor persoane, fiecare având ca scop rezolvarea unor probleme dificile. Dovada muncii lor aduce beneficii de ambele părți pentru că el este răsplătit pentru munca depusă iar informațiile pe care a ales să le publice sunt gata să fie trimise pentru a fi verificate, urmând apoi să fie publicate pentru ca toate persoanele implicate să știe.

2.2 Noduri

Putem defini ca nod într-un blockchain orice dispozitiv conectat la rețea și care participă la anumite acțiuni [35]. Comunicarea dintre noduri se realizează prin rețele de tip Peer-To-Peer, comunicare ce se face constant cu un volum mare de date deoarece toate nodurile trebuie să fie capabile să spună ceva despre conținutul din blockchain.

În blockchain-urile populare cum este Bitcoin sau Ethereum putem face o diferențiere între tipurile de noduri bazată pe rolul pe care acestea le au în rețea. Rolurile presupun acțiuni ca stocarea întregii informații din interiorul blockchain-ului, validarea unui nou conținut și înștiințarea celorlalte nodurilor pentru a asigura caracterul distribuit.

1. **Full nodes.**

   Nodurile complete (*full nodes*) sunt nodurile ce stochează întreaga informație din blockchain. Le putem privi ca unități de management pentru că acționează ca puncte cheie de care depind celelalte tipuri de noduri.

   Nodurile complete sunt cele care vor răspunde la noile informații prin verificarea și stocarea lor în propria copie locală.

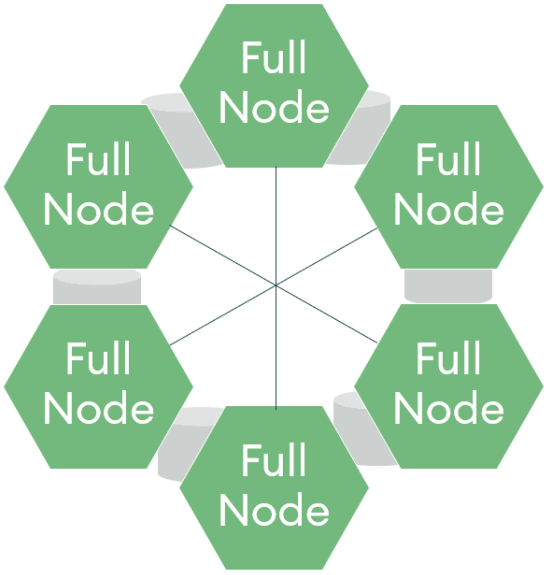


Figura 2.1: Noduri complete - sursă: [35]

În practică însă nu sunt atât de multe noduri complete. Acest fapt este datorat cantității uriașe ce are nevoie să fie stocată fizic pe fiecare nod. De exemplu rețeaua blockchain Bitcoin are în jur de 10000 de noduri complete iar conținutul a depășit pragul de 300 de gigabytes în septembrie 2019 [26] ajungând acum la puțin peste 350 gigabytes de informație [21].

Pe lângă rolul de unitate de stocare, ele sunt unitățile ce au autoritate asupra nodurilor secundare de tipuri diferite. Elementele definitorii nodurilor complete sunt validarea semnăturilor din fiecare tranzacție și alegerea noilor tranzacții ce doresc a fi procesate.

Vom vedea în secțiunile ce implică validarea unei tranzacții și ce probleme ar putea să apară dacă am omite acest pas.

Alegerea noilor tranzacții se reduce la profitul ce poate fi câștigat din urma procesării și publicării lor în registru.

Soluția nodurilor complete este favorabilă companiilor ce aleg să investească într-un anumit blockchain pentru a-i spori caracterul descentralizat și pentru a beneficia de răsplata oferită în urma muncii lor.

2. **Light nodes.**

Nodurile ușoare sunt nodurile care stochează doar o mică parte din întregul blockchain. Scopul lor este de a păstra doar elemente precum identificatori ai conținutului pentru a putea apoi verifica și accesa tranzacțiile la care fac referire.

Vom vedea în secțiunea despre structura cum acești identificatori ne pot garanta un istoric sigur al proceselor desfășurate pentru ca tranzacțiile să fie acceptate.

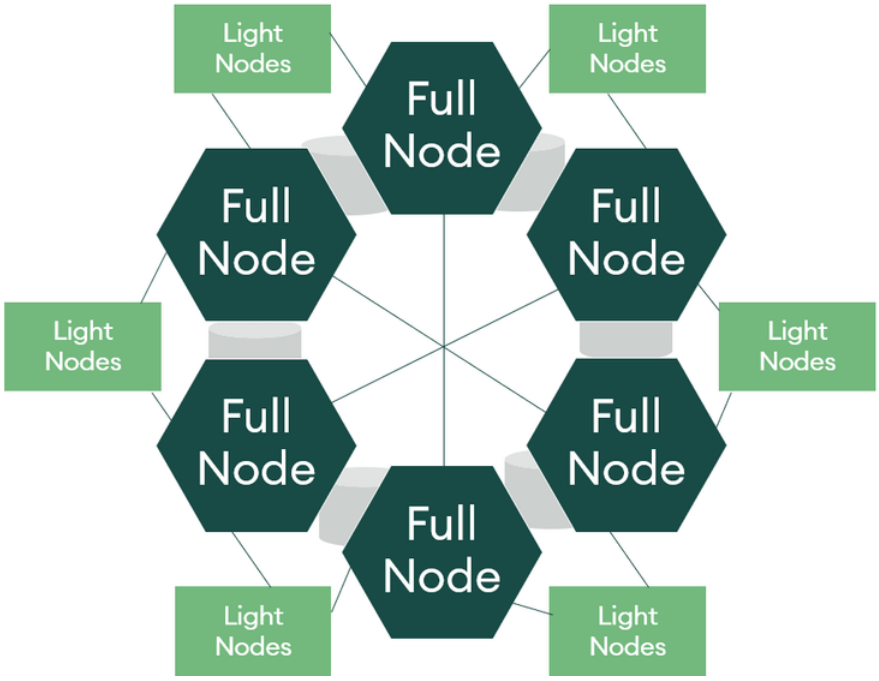


Figura 2.2: Noduri ușoare - sursă: [35]

Imaginea arată cum nodurile ușoare depind de cele complete numite, în cazul acesta, noduri părinți pentru a-și lua informația de care au nevoie. Securitatea revine nodurilor complete care trebuie să fie capabile să livreze informațiile valide cerute ce nodurile copil.

În marea majoritate a cazurilor nodurile ușoare se găsesc sub forma portofelelor electronice și au rolul de a stoca tranzacțiile și de calcula automat cantitățile de criptomonede avute.

3. **Worker nodes.**

Nodurile lucrătoare sunt cele care au nevoie de cea mai multă putere de procesare. Ele sunt cele care vor intra în competiția ce are ca scop rezolvarea unei probleme dificile, a cărei rezultat este definitoriu în pasul de publicare al unui nou conținut în blockchain.

Vom vedea mai pe larg în secțiunea despre algorimii de consens de ce această problema se păstrează a fi grea și de ce nodurile lucrătoare rezolvă marea parte din procesele desfășurate în blockchain.

Consumul de energie ridicat pentru anumite noduri lucrătoare ce participă la algoritmi de tip Proof of Work au generat cu timpul termenul de minare. Termenul provide din ideea muncii intense desfășurate ce este răsplătită la final în criptomonede sau tokeni.

În practică mai multe noduri lucrătoare comunică cu un nod complet deținut de o companie investitoare care strânge profitul colectiv și îl împarte după cât a lucrat fiecare. Acest proces a făcut ca minatul criptomonedelor să devină din ce în ce mai popular promovându-se ideea că orice o poate realiza de pe orice dispozitiv și poate și răsplătit pe măsura muncii lui.

Rolurile celor două tipuri de noduri sunt foarte bine delimitate. Nodul complet este gândit spre a spori gradul de distribuire al conținutului din blockchain stocând o clonă a lui și lucrând atât la verificarea și publicarea informațiilor cât și la extragerea și trimiterea unor noi tranzacții spre a fi procesate. El nu creează însă informație nouă.

Procesele ce stau la baza creării unor noi informații se desfășoară în nodul lucrător și carela final trimite ce a realizat către un nod complet pentru a i se valida munca. Dacă se dovedește a fi corect, nodul lucrător este recompensat.

De menționat este și faptul că nodul miner depinde de unul complet însă relația inversă nu este adevărată, un nod complet poate fi creat având niciun miner atașat și care participă doar la validarea colectivă a informației.

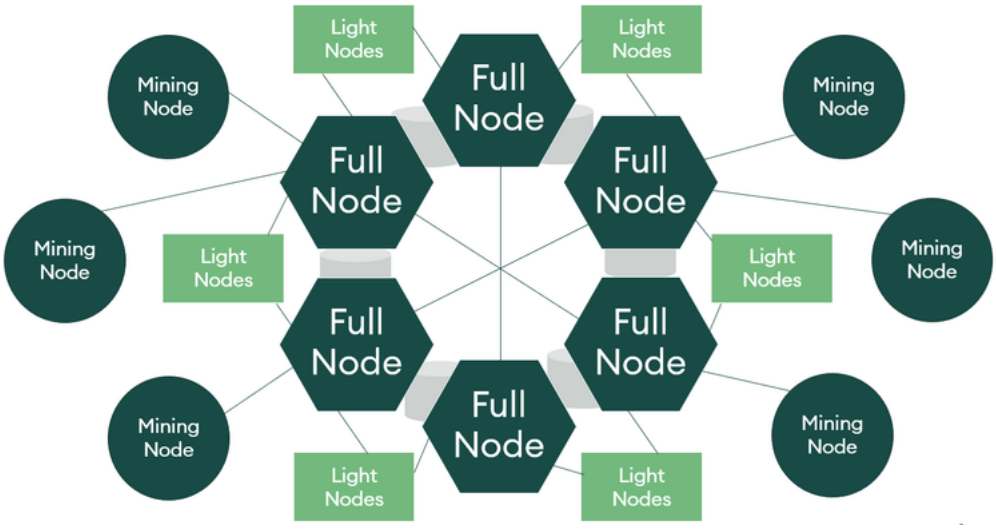


Figura 2.3: Noduri lucrătoare (Mineri) - sursă: [35]

2.3 Permissionless & Permissioned

În ceea ce privește o categorisire a blockchain-ului putem distinge două mari categorii: fără permisiune (*permissionless*) și cu permisiune (*permissioned*). În cele fără permisiune oricine poate intra în rețeaua blockchain-ului și contribuie la procesele ce se desfășoară în interiorul lui. Cele cu permisiune reduc numărul de utilizatori doar la cei aleși pe criterii specifice unui blockchain privat și ridică nevoia unei entități de control care să managerieze întreaga organizație.

1. **Permissionless.**

   Blockchain-urile fără permisiune sunt rețelele la care se poate conecta oricine având și posibilitatea de a publica nou conținut. Pentru acest tip de blockchain adesea se întâlnesc platforme open-source ce permit accesul facil la rețea și ușurează comunicarea cu acestea.

   Elementele ce trebuie să fie mereu disponibile sunt posibilitatea de citi și adaugă nou conținut. Acest lucru poate reprezenta o portiță prin care atacatorii pot introduce date incorecte sau preasamblate pentru a le aduce avantaje materiale. De aceea blockchain-urile permissionless au nevoie de ceea ce se numește algoritm de consens care în același timp dă regulile prin care un nou conținut ajunge să fie salvat în blockchain cât și forțează nodurile să îl respecte.

   Cele mai populare astfel de blockchain-uri sunt Bitcoin și Ethereum, însă există o serie lungă de produse, fiecare concurând cu diferite implementări ce aduc plus eficienței cu care se desfășoară evenimentele.

2. **Permissioned.**

   Blockchain-urile cu permisiune sunt cele în care există o autoritate care decide ce drepturi și roluri poate avea fiecare membru din rețea. Această autoritate poate lua decizii pe toate planurile din interiorul rețelei, poate restrânge dreptul unor noduri de a vizualiza conținutul din blockchain, poate limita ca doar anumite noduri să poată crea tranzacții sau doar să le verifice și să le publice.

   Astfel de rețele blockchain pot fi ca și în cazul precedent create și menținute folosind software-uri open-source cum este și cazul BigchainDB, tehnologia descrisă în capitolul următor.

Nu trebuie omis faptul că într-un blockchain permissioned se respectă toate evenimentele care se petrec în mod normal în formatul permissionless. Conținutul din blockchain este distribuit și respectă proprietatea că informația nu poate fi alterată. În ceea ce privește algoritmul de consens, el are loc însă nu este la fel de consumator de resurse și putere de procesare datorită faptului că nodurile membre se autentifică pentru participare iar factorii de decizie pentru validarea unui nou conținut pot fi mai relaxați.

În cazul acestui tip de blockchain nodurile trebuie să aibă un grad de încredere mai mare, atât între ele cât și față de autoritatea care îi manageriază. Încrederea este cea pe baza căreia se pot reduce etapele complicate și consumatoare de resurse. De asemenea, dacă un nod este considerat malițios, el își poate pierde toate drepturile pe care le are în rețea.

De obicei acest tip de blockchain este întâlnit la organizații ce doresc să beneficieze de proprietățile pe care le aduce tehnologia dar în același timp doresc un nivel de control asupra lucrurilor care se petrec. Situațiile pot varia în funcție de ce dorință are fiecare. Se pot, de exemplu, crea blockchain-uri private în care membrii să facă parte din organizații partenere iar tranzacționarea să se facă în acest format electronic. Se pot decide de comun acord obligațiile și drepturile pe care le are fiecare participant și se poate genera un plan prin care se iau deciziile.

Am văzut cum blockchain-urile permissionedse pretează unui cadru privat. De aceea nu este obligatorie transparența totală, tranzacțiile se pot procesa și publica în blockchain însă doar cel care tranzacționează și cel căruia îi este destinată tranzacția pot vedea conținutul complet.

Cum spuneam, gradul de securitate poate părea mai slab decât al celuilalt tip de blockchain și asta poate atrage mai mulți atacatori însă este mai ușoară detecția proceselor malițioase și a persoanelor din spatele lor datorită nevoii, în anumite cazuri, a autentificarii.

2.4 Elemente criptografice

Tehnologia blockchain are la bază un cumul de mecanisme și structuri criptografice foarte puternice care asigură securitate și persistență. Printre ele se numără funcțiile hash, semnăturile digitale și criptografia asimetrică.

Funcțiile hash sunt funcții criptografice ce asigură integritatea datelor. Ele sunt capabile să primească un input de aproape orice dimensiune și să întoarcă un rezultat unic relativ la ce date a primit. Ceea ce le face utile în diferite procese este usurita cu care pot fi verificate anumite informații primite în rețea. Orice modificare a conținutului de la care se pleacă va genera un cu totul alt rezultat și astfel putem vedea dacă toate părțile componente ale informației au fost primite sau dacă au fost alterate pe parcurs.

Aceste funcții respectă 3 proprietăți:

- **Rezistență la preimagine**. Sugerează că funcția este one-way și că este dificil să se calculeze valoarea input-ului dat fiind orice cod hash. În blockchain această proprietate nu este pusă în valoare datorită caracterului transparent al conținutului. Nodurile pot vedea atât tranzacțiile cât și rezultatul obtinut prin trecerea lor printr-ul algoritm de hash.
- **Rezistență la a doua preimagine**. Proprietatea ne garantează caracterul unic al rezultatului plecând de la o pereche input-output cunoscută. Dificultatea problemei rămâne dar ipoteza se schimbă în cazul acesta și reprezintă găsirea unui nou input astfel încât trecut prin funcția hash să întoarcă același cod ca al trecerii unui input cunoscut prin aceeași funcție. În cazul blockchain, unde mulțimea din care putem lua input-uri este colecția totală de tranzacții transmisă, proprietatea este utilă pentru a fi siguri că dacă știm o submulțime și știm codul hash făcut pe baza ei nu există o a doua submulțime care va avea același rezultat.
- **Rezistență la coliziuni**. A treia proprietate merge în aceeași direcție cu cea precedentă și spune că nu există nicio pereche de input-uri care să rezulte în același cod hash.

Fiecare blockchain poate alege să utilizeze o anumită implementare a unor funcții hash. Bitcoin alege să folosească în etape succesive Secure Hash Algorithm [2], pe când Ethereum alege Keccak [4], ambele întorcand un rezultat pe 32 de bytes. Încercarea de a găsi o coliziune poate depăși chiar și 30 de bilioane de ani și chiar dacă s-ar găsi este puțin probabil ca input-ul găsit să reprezinte o tranzacție validă.

Alte ingrediente criptografice folosite în cadrul blockchain sunt numerele folosite doar o singură dată (*numbers only used once - nonce*). Ele se folosesc în cadrul algoritmilor de consens de tip Proof of Work și reprezintă ceea ce se caută în problema consumatoare de energie. Nonce-urile se folosesc alături de alte date în funcții hash pentru a putea genera coduri diferite plecând de la același set inițial de elemente. Procesul poate fi rezumat de formula: *hash (date + nonce) = cod_hash*.

Mecanismele și elementele de criptare sunt cele care lucrează în blockchain pe două planuri, cel al autentificării și cel al dovedirii posesiei. Criptarea folosită este asimetrică iar aceasta implică două chei diferite, una privată și una publică într-o strânsă legătură. Cheia publică este cea care poate fi vizualizată de către oricine fără a se pierde din securitatea relației cu cheia privată. Simplu spus, având o cheie publică nu putem genera cheia privată.

În practică pentru criptare se folosește cheia privată, iar cea publică va folosită ulterior la decriptarea din procesul de verificare al tranzacțiilor. O semnătură digitală se realizează pentru fiecare tranzacție și reprezintă criptarea cu cheia privată a conținutului din tranzacție. Un utilizator alege să folosească semnătura pentru a atesta posesia cheii private și implicit a celei publice regăsite în tranzacții precedente.

De asemenea, în contextul blockchain-urilor cu permisiune se pot folosi alternative în ceea ce privește generarea și validarea cheilor asimetrice. Se pot folosi servicii precum Active Directory sau Lightweight Directory Access Protocol pentru a prelua certificate sau alte informații ce autentifică și autorizează un utilizator în rețea.

Pentru ușurință anumite blockchain-uri se folosesc de adrese în locul cheilor publice complete. Aceste adrese reprezintă șiruri de caractere scurte obținute prin trecerea cheilor publice prin funcții hash. O tranzacție poate astfel avea ca destinar o adresă despre care putem afirma că este unică deoarece este un cod hash derivat dintr-o cheie unică. Trebuie însă să reținem că din adresă nu putem reconstrui cheia publică și de aceea, în momentul tranzacționării, pentru ca semnătura digitală să poată fi verificată, trebuie să asociem și cheia publică reală.

2.5 Structura

Pe parcursul tezei a fost menționată ideea de tranzacție. Tranzacțiile sunt elementele prin care se realizează interacțiunea dintre membrii rețelei blockchain.

Deși termenul de tranzacție este universal folosit peste orice implementare de blockchain ea poate avea structură diferită de la caz la caz. În momentul în care cineva dorește să creeze o astfel de structură trebuie să țină cont de elementele ei definitorii ce sunt adresa transmițătorului, cheia publică a transmițătorului, o semnătură digitală, unul sau mai multe input-uri și unul sau mai multe output-uri.

Input-urile reprezintă informațiile ce se doresc a fi transferate. Dovedirea deținerii informației de către transmițător se poate realiza fie prin referențierea unei tranzacții anterioare în care el este pe rolul de destinatar, fie prin referențierea unei tranzacții prin care informația a fost creată în blockchain. Legătura către evenimente trecute limitează procesele cu ele dar în același timp le oferă siguranță. Informația unei tranzacții mai vechi nu poate fi schimbată în noul eventiment. De exemplu în cazul criptomonedelor, nu se poate schimba cantitatea de monezi primită. Securitatea este oferită și de semnătura digitală care atestă că cel care transferă informația este cel căruia ii aparține cu adevărat pentru că este posesorul cheii private asociate celei publice din tranzacție.

Output-urile sunt identificatori în blockchain precum adrese sau chei publice cărora se dorește a fi transmisă informația sau o anumită parte din ea. Cazul criptomonedelor permite ca mai multe tranzacții să fie folosite ca input-uri și astfel cantitatea totală să fie adunată și transferată către mai mulți destinatari. Valoarea sumată din input-uri nu trebuie obligatoriu să fie egală cu suma output-urilor, însă trebuie să fie mereu mai mare, diferența vom vedea că va juca rol în motivația minerilor din algoritmul de consens Proof of Work de a o lua și procesa [22].

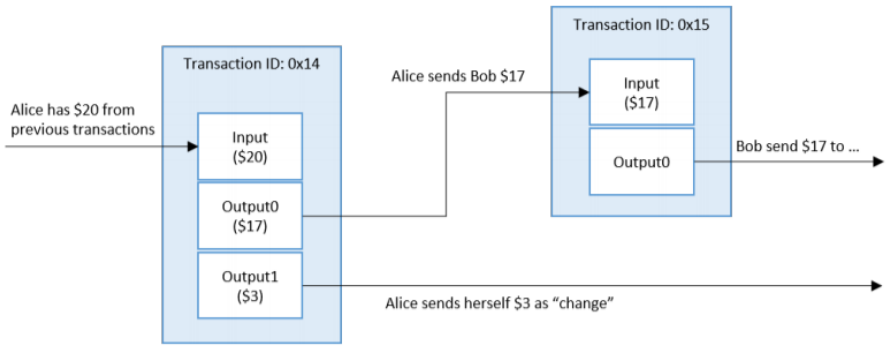


Figura 2.4: Exemplu de înlănțuire între tranzacții - sursă: [25]

Vom vedea că aplicația TrustNews folosește tehnologia descrisă în capitolul următor pentru a putea crea tranzacții ce au ca informații știri luate din diferite surse. Informația este creată și este deținută de nodul care o va prelua.

Următoarea și cea mai importantă structura în blockchain este block-ul, la care am făcut referire pe parcusul lucrării sub numele de conținut. În momentul în care o tranzacție este efectuată, ea este transmisă în rețea și ajunge în lista fiecărui nod care acum este responsabil să o proceseze sau mai simplu spus să o adauge într-un block.

Fiecare nod va alege o serie de tranzacții pe care le consideră profitabile și va încerca să creeze structura de block compusă din zona de date și zona de metadate (*header*). Datele reprezintă mulțimea de tranzacții alese mai devreme iar metadatele cuprind valori calculate pe baza datelor sau obținute din block-uri precedente.

- Numărul block-ului. Valoare cunoscută și ca înălțimea blockchain-ului reprezintă numărul total de block-uri stocate în lanț.
- Versiunea. Face trimitere la setul de reguli folosite pentru validarea conținutului din block.
- Dimensiunea. Este cantitatea exprimată în bytes a întregului conținut salvat în block.
- Codul hash al precedentului block.
- Codul hash al datelor. Acest cod este dat de rădăcina arborelui Merkle realizat folosind tranzacțiile alese. Procesul implică în prima etapă obținerea codulor hash pentru fiecare tranzacție, apoi mergând către vârf, se obțin codurile hash ale concatenărilor de câte două valori precedent aflate.
- Momentul de timp la care a fost creat block-ul
- Nonce. Este valoarea unică găsită ca rezultatul problemei dificile din algoritmul PoW și care poate fi omisă dacă blockchain-ul are un alt algoritm de consens.

Procesul de înlănțuire este dat de existența în fiecare block al codului hash provenit din precedentului block. Acest cod hash se realizează având ca date de intrare metadatele block-ului precedent. Procesul de hashing ne ajută să determinăm imediat dacă un block este format greșit, iar crearea continuă de noi block-uri valide face din ce în ce mai grea munca unui atacator care ar vrea să definească propriile block-uri care i-ar aduce beneficii materiale.

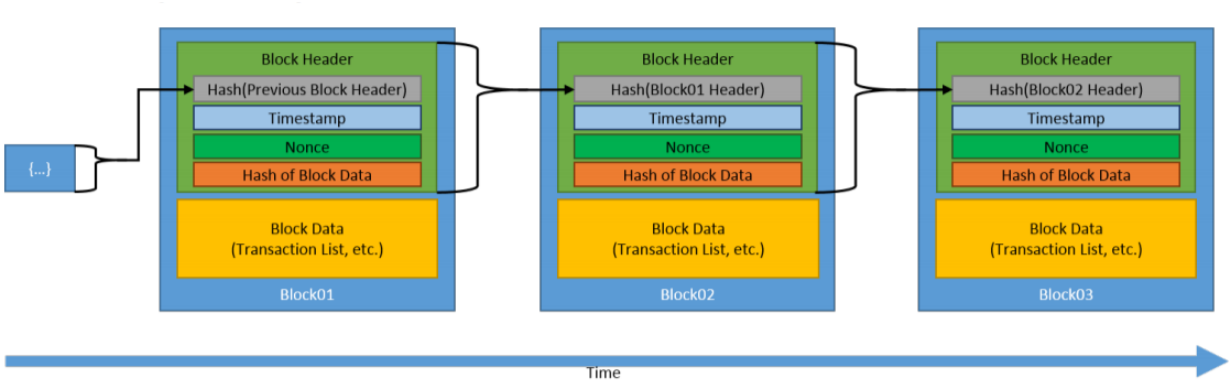


Figura 2.5: Exemplu de înlănțuire între block-uri - sursă: [25]

2.6 Algoritmi de consens

Un factor de decizie foarte important în blockchain este legat de cine are dreptul de a publica noi block-uri și ce reguli trebuie să respecte. Problema alegerii nodului care publică un nou block este rezolvată de implementarea unui algoritm de consens.

În cazul blockchain-urilor fără permisiune, unde toți participanții pot alege să publice informații, algoritmul poate fi privit ca o competiție având ca rezultat obținerea de beneficiu financiar sub formă de criptomonede. Algoritmul trebuie să fie capabil să gestioneze situații cu mulți participanți care doresc obținerea acelui bonus. În momentul în care un nod creează un nou block, îl trimite în rețea către toți ceilalți participanți pentru a-l valida. Dacă este construit corect, nodul primește răsplata cuvenită.

De asemenea algoritmul trebuie să poată decide cine este câștigător atunci când două sau mai multe noduri creează block-uri valide în același timp. De regulă, în astfel de situații numite *forks*, procesul de creare merge în paralel pe ambele ramuri până când o ramură conține mai multe block-uri decât cealaltă. În acel moment ramura mai lungă se consideră a fi cea principală pentru că a fost mai multă muncă depusă iar cea scurtă este eliminată. Tranzacțiile care au fost adăugate în block-urile de pe ramura eliminată și care nu au fost adăugate în block-uri din ramură câștigătoare se reîntorc în mulțimile fiecărui nod pentru a fi procesate ulterior.

De regulă fiecare blockchain are un block inițial numit geneză de la care se pornește. Orice nod care acceptă să între în rețea și să participe la algoritmul de consens acceptă implicit validitatea block-ului geneză și datorită proceselor de validare poate spune că și block-urile ulterioare au fost construite corect.

Fie că vorbim de blockchain-uri cu permisiune fie fără, algoritmul de consens este cel care protejează rețeaua de membrii malițioși care doresc să altereze conținutul. În cazul celor fără permisiune, cel care acționează malițios este oprit de numărul mult mai mare de utilizatori corecți, iar în cazul blockchain-urilor cu permisiune unde numărul nodurilor poate fi redus există acea autoritate de încredere care poate acționa chiar și legal atunci când observă că un utilizator nu respectă regulile la care s-a înscris inițial. Un alt element distinctiv atunci când vorbim de algoritmi de consens pentru cele două tipuri de blockchain-uri este gradul de utilizare al resurselor fizice. În cele cu permisiune adesea se folosesc algoritmi mai relaxați care pot fi reduși doar la simple validări.

O formă de algoritm de consens ce a devenit cunoscută odată cu apariția Bitcoin este Proof of Work (PoW). În acesta formă, nodul care va publica noul block este cel care reușește primul să găsească rezultatul unei probleme criptografice grele dovedind astfel munca depusă. Rezolvarea problemei este dificilă și mare consumatoare de energie dar verificarea rezultatului este ușoară și facilitează procesul de validare de către celalte noduri.

Un exemplu de astfel de problemă dificilă este găsirea unui cod hash ce conține pe primele poziții un număr țintă de zerouri. Pentru că procesul de hashing nu întoarce un rezultat previzibil dificultatea este dată de încercarea tuturor posibilităților până la găsirea unui rezultat favorabil. În momentul de față, în Bitcoin, problema se reduce la găsirea valorii nonce astfel încât codul hash realizat pe baza metadatelor din block să înceapă cu 19 zerouri. Această dificultate este crescută gradual pentru a asigura crearea block-urilor noi odată la aproximativ 10 minute.

Dificultatea găsirii acestei valori nonce limitează atacatorii care decid să investească și să dețină o foarte mare parte din puterea de procesare a rețelei pentru a fi singurii care primesc bonusul financiar. Atacul de genul acesta se numește atacul 51% și implică deținerea a peste jumătate din întreagă putere de procesare a rețelei blockchain. O astfel de putere poate, pe lângă efectul financiar descris anterior, să influențeze conținutul deoarece ar putea valida în mod intenționat un conținut malițios. O abordare al acestui atac implică crearea unui lanț paralel și mai lung decât cel principal în care unele tranzacții pot fi dublate. Când atacatorul decide poate publica ramura creată și, fiind mai lungă, o va înlocui pe cea reală procesată corect. Cu toate acestea, un astfel de atac este nefezabil financiar în momentul de față și devine din ce în ce mai greu de realizat cu cât mai mulți utilizatori aleg să intre în rețea.

O altă formă de algoritm de consens este Proof of Stake. Devenită cunoscută odată cu trecerea rețelei Ethereum la versiunea a doua, ea implică un proces similar cu cel al licitațiilor. În blockchain-urile care implementează acest tip de algoritm se presupune că fiecare utilizator are o sumă pe care o poate investi pentru a asigura buna funcționare a întregii rețele. O miză (*stake*) este o sumă pe care un nod este dispus să o dea pentru ca un block creat de el să fie publicat. Astfel, nodul câștigător care va publica un nou block este cel care oferă miza mai mare.

Spre deosebire de PoW, această formă nu implică un consum de resurse intens și din această cauză blockchain-urile care implementează soluția consideră că întreagă cantitate este distribuită și că nu mai este nevoie de o răsplată în urmă procesării block-ului. Răsplata financiară în această situație vine din taxele puse pe tranzacții pentru a fi procesate. Pentru că în această formă a algoritmului se poate pune problema unui monopol, există diferite forme prin care se alege cine este cu adevărat cel care publică un block.

Forma aleatoare este dată de numărul de noduri aparținând fiecărui utilizator. Un utilizator care deține 30% din totalul de noduri are 30% șanse să fie ales, șansă mult mai mare decât cineva ce deține doar 2% din totalul de noduri. Forma prin votare numită și Byzantine Fault Tolerance implică ca software-ul să aleagă o parte din nodurile care vor să publice noduri. Odată aleși acești participanți, ei participă la mai multe runde de votare asupra cărui block este publicat și implicit cine îl va publica. Alte forme pot conține voturi atât în favoarea cât și împotriva anumitor noduri, voturi ce cântăresc mai mult cu cât miza oferită este mai mare. Se creează astfel reputație pentru fiecare nod, reputație pe baza căreia se poate face distincția între un nod corect și unul malițios.

2.7 Motivație pentru utilizare

Cu toate că tehnologia blockchain aduce multe beneficii trebuie să ne punem problema dacă ne este cu adevărat utilă. Imaginea din pagina următoare preluată din articolul celor de la Național Institute of Standards and Technology (NIST) ne ajută să ne punem întrebările necesare și să determinăm dacă avem nevoie de toate facilitățile pe care le pune la dispoziție blockchain-urile în general.

Aplicația propusă TrustNews își propune să stocheze sub formă de block-uri știri preluate din diverse surse de încredere și are nevoie de proprietățile blockchain.

Este nevoie de o zonă distribuită a datelor. Caracterul distribuit ajută la menținerea în siguranță a valorilor introduse corect de noduri cinstite.

Aplicația propune existența mai multor noduri care să lucreze atât ca extractori de știri de pe diferite platforme online cât și ca noduri ce publică block-uri.

Odată publicate știrile sub formă de tranzacții în blockchain ele nu se mai pot modifica. Acest aspect este cel care ne garantează că știrile nu se pot altera pentru a deveni false sau cu un caracter derutant.

Deși software-ul permite, informații sensibile nu vor fi stocate. Conținutul va fi preluat doar din surse publice de media.

Pentru că nu vorbim de o soluție clasică ce implică o bază de date în care se gestionează conținutul este nevoie de un algoritm prin care participanții să poată transmite și primi informațiile noi și să le manageriezе în propria lor copie a întregului blockchain.

În plus, la orice moment, trebuie să existe posibilitatea de a verifica operațiile efectuate și de către cine au fost realizate. Identificarea unui nod ce poate fi echivalat cu un extractor de știri se va face folosind perechile de chei asimetrice generate de un software comun folosit de toți participanții.

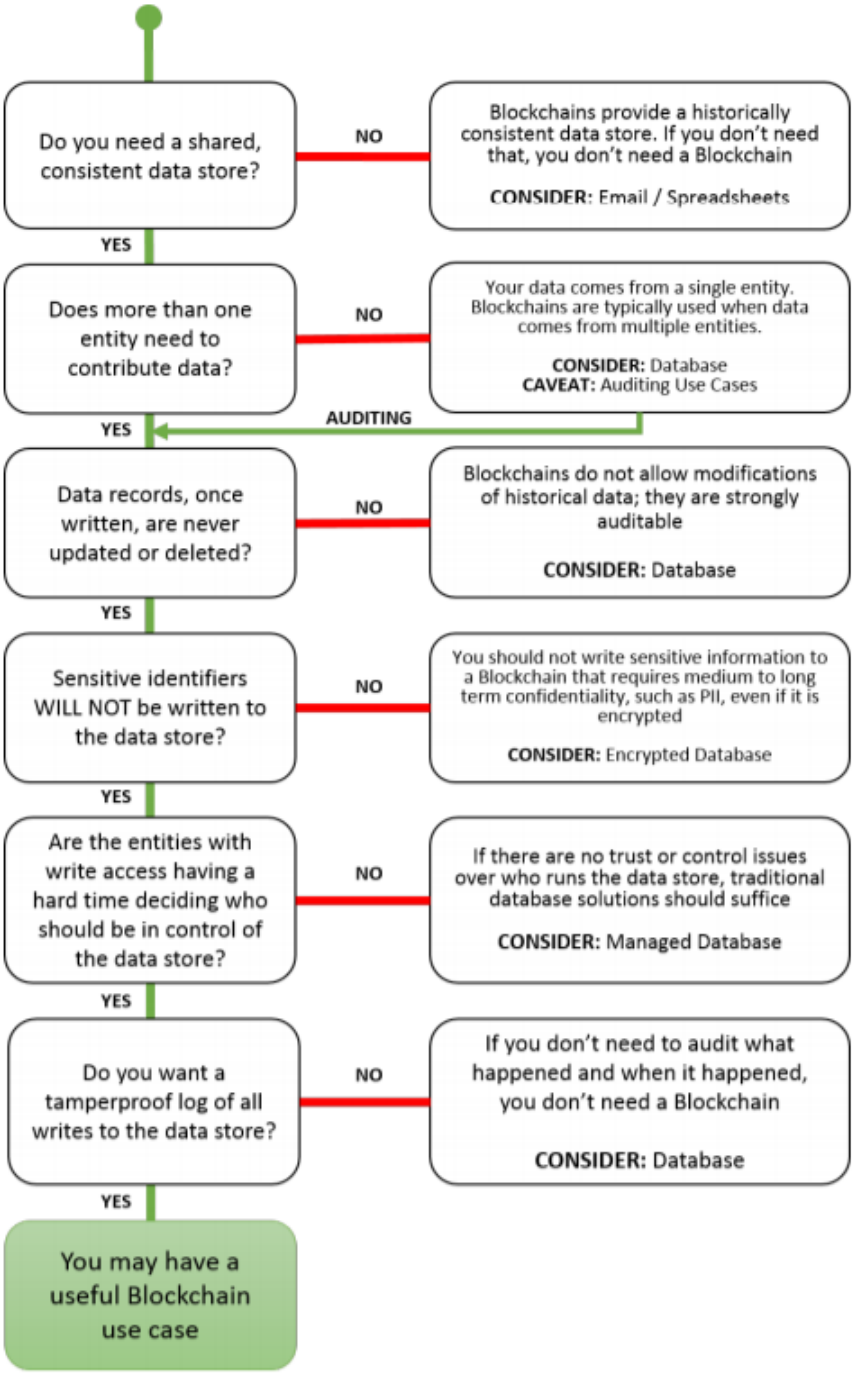


Figura 2.6: Blockchain Usage Flowchart - sursă: [25]

Capitolul 3

BigchainDB

3.1 Introducere

BigchainDB este tehnologia folosită pentru a răspunde nevoilor date de aplicația TrustNews, reușind să facă trecerea ușoară către noile moduri de procesare și stocare a datelor.

Tehnologia, ajunsă acum la versiunea 2.0, combină proprietățile blockchain (descentralizare, imutabilitate și management propriu asupra datelor deținute) cu cele ale bazelor de date (rată de tranzacționare mare, timp de răspuns mic și posibilitate de a interoga eficient conținutul). [24]

	Typical Blockchain	Typical Distributed Database	BigchainDB
Decentralization	✓		✓
Byzantine Fault Tolerance	✓		✓
Immutability	✓		✓
Owner-Controlled Assets	✓		✓
High Transaction Rate		✓	✓
Low Latency		✓	✓
Indexing & Querying of Structured Data		✓	✓

Figura 3.1: Proprietăți BigchainDB - sursă: [24]

3.2 Proprietăți

Structura descentralizată implică noduri deținute și menținute de persoane diferite. Astfel se elimina necesitatea unui singur punct de control sau a unei singure entități care să administreze întreagă rețea.

Nodurile sunt compuse din 3 elemente: nucleul Tendermint, server-ul BigchainDB și o bază de date MongoDB. Fiecare contribuie cu elemente specifice pentru a reuni toate proprietățile unui blockchain.

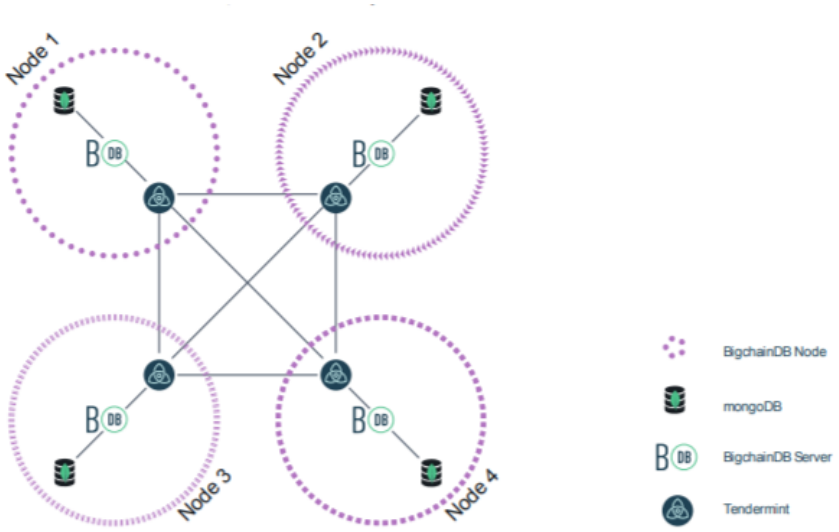


Figura 3.2: Noduri BigchainDB - sursă: [24]

Tendermint [6] este tehnologia de bază peste care se lucrează și care aduce elementele ce țin de rețea și de algorimul de consens. Mai exact Tendermint Core este componenta ce permite trimiterea de mesaje în rețea folosind protocoale specifice iar algormitmul de consens este Byzantine Fault Tolerant. Consensul se stabilește pe baza mai multor runde de votare la care participă toate nodurile și în care se decide ce block va fi publicat. Un block este valid dacă pentru fiecare rundă cel puțin două treimi din noduri au votat în favoarea lui.

Informația va fi stocată în final în bazele de date de pe fiecare nod. Natura descentralizată oprește atacatorii care ar vrea să corupă sau să șteargă datele. Ei ar putea să obțină privileglii crescute pe anumite noduri însă asta le va permite manipularea datelor doar de pe acele noduri. Încercarea de a acționa rău voit în procesul de votare poate fi nesemnificativ dacă numărul de noduri este unul foarte mare (implicit pragul de două treimi este mare).

De menționat este faptul că BigchainDB este deseori într-un context de blockchain cu permisiune. Astfel accesul în rețeaua dezvoltată pentru o anumită aplicație este dat de o autoritate care poate monitoriza acțiunile fiecărui membru și poate lua decizii în consecință.

Imposibilitatea de a șterge datele odată salvate în blockchain este realizată practic în software prin lipsa unui serviciu explicit care să permită acest lucru. În plus descentralizarea amintită anterior implică și salvarea în fiecare nod a conținutului din blockchain. Astfel dacă de pe un anumit nod se șterg date, celelalte noduri nu vor fi impactate.

Toate bunurile (*assets*) create în BigchainDB aparțin unor anumiți utilizatori. Aceștia pot dovedi posesia lor și le pot transfera atât timp cât pot arată deținerea cheilor criptografice cu care au fost create sau primite structurile de date. Interacțiunea se realizează prin tranzacții specifice de tip CREATE sau de tip TRANSFER descrise în secțiunea următoare. Software-ul se asigură că tranzacțiile sunt valide atât pe partea internă a conținutului cât și la nivel înalt verificând să nu existe situații de double-spending.

Rată crescută de tranzacții și timpul de procesare scăzut sunt date de performanța Tendermint. Se asigură timpi de sub o secundă pentru procesarea tranzacțiilor chiar și atunci când vorbim de mii de tranzacții. Trebuie totuși menționat că un scenariu real al conține un minim de 50 de noduri distribuite în mai multe zone geografice ale globului.

Indexarea și interogarea datelor se face asupra bazei de date MongoDB. Fiecare nod deține întregul conținutdin blockchain, de la block-uri la elementele particulare din interiorul fiecăruia, sub formă de structuri JSON și este în deplin control asupra propriei clone. Fiecare utilizator poate manevra datele în orice mod dorește și poate crea indexi sau interogări eficiente pentru diferite puncte de interes. Asupra lor se pot adăuga servicii precum apeluri HTTP API pentru a le folosi la o scară largă. De asemenea software-ul BigchainDB aduce câteva modalități predefinite prin care putem extrage informații rapid sub formă de apeluri HTTP API.

3.3 Utilizare

BigchainDB pune la dispoziție o serie de utilitare numite drivere cu care putem interacționa cu întregul ecosistem [19]. Există două tipuri, unul scris în Python [18] și unul în JavaScript [20] dar rămâne la latitudinea fiecărui utilizator să aleagă cu ce se simte confortabil pentru că ambele au aceleași capacități.

La nivel de bază aceste drivere se folosesc de anumite funcții implementate peste componenta din Tendermint numită Application BlockChain Interface (ABCI) [15], componentă care permite scrierea de cod ce acționează asupra componentelor blockchain în orice limbaj de programare. BigchainDB definește funcțiile din interfață pentru a putea iniția și finaliza acțiunile de creare a unui block (BeginBlock, EndBlock, Commit) și pentru a valida și trimite tranzacțiile (CheckTx, DeliverTx)[24].

Tranzacțiile sunt practic mesajele pe care dorim să le creem, mesaje ce conțin asset-urile pe care dorim să le salvăm în blockchain.

Tranzacțiile sunt structuri de tip JSON ce respectă anumite reguli de sintaxă și conținut [34]. Sunt compuse din:

- **ID**. Este codul hash SHA-256 calculat pe baza tuturor celorlalte date din tranzacție și reprezintă identificatorul unei tranzacții.
- **Versiune**. Reprezintă versiunea de software cu care a fost creată tranzacția și este utilă în preluarea corespunzătoare a regulilor de validare.
- **Operație**. Reprezintă tipul de tranzacție ce poate fi CREATE sau TRANSFER.
- **Asset**. În cazul tranzacțiilor CREATE, informația are o singură cheie principală numită "data" și care poate lua orice valoare. În cazul tranzacțiilor TRANSFER, câmpul conține perechea dată de cheia "id" și valoarea unei tranzacții la care ne referim pentru asset-ul din interior.
- **Metadate**. Conține, ca și câmpul anterior, o structură de tip JSON dar în care se pot adăuga direct informații ce țin de tranzacție sau de asset. Metadatele se vor stoca de asemenea în baza de date dar particularitatea este că se pot modifica prin tranzacții ulterioare, spre deosebire de câmpul asset unde datele devin imutabile.
- **Input-uri**. Un input este la rândul lui compus din mai multe câmpuri ce pot face legătură cu tranzacții precedente și care marchează posesia asset-ului.

  Câmpul "fulfills" este populat doar în cazul tranzacțiilor TRANSFER și în care se specifică explicit un output dintr-o tranzacție anterioară prin ID și prin indexul de legătură.

Câmpul "owners_before" marchează lista de chei publice a deținătorilor asset-ului până în momentul tranzacției curente. În cazul tranzacțiilor de tip CREATE valoarea câmpului va fi cheia publică a celui care creează asset-ul.

Câmpul "fulfillment" reprezintă semnătura digitală realizată pe baza tuturor cheilor private asociate cheilor publice marcate în câmpul anterior.

- **Output-uri**. Sunt structurile care indică condițiile ce trebuie respectate pentru ca un asset să fie transferat către un nou posesor. Un output este alcătuit de asemenea din trei câmpuri.

  Câmpul "amount" specifică cantitatea creată sau transferată de utilizator. Software-ul permite astfel divizarea asset-urilor precum în cazul criptomonedelor. Spre deosebire totuși de criptomonede în BigchainDB regula spune că suma cantităților din input-uri trebuie să fie egală cu suma cantităților din output-uri.

  În câmpul "condition", în proprietatea "details" se pot seta două tipuri de condiții. Dacă dorim o situație în care doar un utilizator este implicat atunci vom alege tipul ED25519-SHA-256 și vom specifica cheia lui publică. Dacă dorim o situație mai complexă putem alege tipul THRESHOLD-SHA-256 în care putem seta mai multe subconditii de tip anterior și putem seta un minim de condiții ce trebuie respectate.

  Câmpul final "public_keys" reprezintă reuniunea tuturor cheilor publice din condiție.

În momentul în care o tranzacție este trimisă în rețea, software-ul BigchainDB o verifică urmând anumite repere [34]:

- ID-ul tranzacției are formatul unui cod hash creat de algoritmul SHA-256.
- ID-ul tranzacției este unic la nivelul întregului blockchain. Verificare se poate face ușor prin interogarea bazei de date.
- Câmpul "fulfillment" trebuie să fie valid pentru fiecare input. Decriptarea folosind cheia publică asociată ar trebui să returneze o informație clară.
- În cazul tranzacțiilor TRANSFER, un input nu se poate asocia unui output deja preluat într-o tranzacție validă anterioară și nu pot exista două input-uri în aceeași tranzacție care să se asocieze cu același output.
- În cazul tranzacțiilor TRANSFER, un input trebuie pointeze către o tranzacție care există, care este validă și în care se face trimiterea către același asset.
- Conținutul trebuie să fie conform operațiilor care lucrează cu el (sintaxa json corectă, evitarea caracterelor speciale în zona de date).

```
{
  id: '5bf6d751911069f9fc9b0a0ce49150fa0e042a10acd4892fb2b9dd5dc9bd5553',
  operation: 'CREATE',
  outputs: [
    {
      condition: {
        details: {
          type: 'ed25519-sha-256',
          public_key: 'Dn96HxLYxfcdT6gJkqeUVE29h7qFmcZ9uZ4Y5CUSdS9o'
        },
        uri: 'ni:///sha-256;yxIhklNGgAXyFbALBiJby3LZHmp0Q3C96eo30meoqNw?fpt=ed25519-sha-256&cost=131072'
      },
      amount: '1',
      public_keys: [ 'Dn96HxLYxfcdT6gJkqeUVE29h7qFmcZ9uZ4Y5CUSdS9o' ]
    }
  ],
  inputs: [
    {
      fulfillment: 'pGSAIL3ch0Bndp2N6Fq31vtnw0TI-x_nBqx5rUReiRAwoSJSgUCi1oza4_-CdpPSc6usVqBdU5_V7Vu-aHXd4maP0
MeLSDZhxfAXL4VcsesrYyuUdbTjau7o-Bq5vUw8hIQSsXEK',
      fulfills: null,
      owners_before: [ 'Dn96HxLYxfcdT6gJkqeUVE29h7qFmcZ9uZ4Y5CUSdS9o' ]
    }
  ],
  metadata: null,
  asset: {
    data: {
      content: 'The content of a news story taken from a trusted site',
      datetime: 'Fri Jul 30 2021 22:04:47 GMT+0300 (Eastern European Summer Time)'
    }
  },
  version: '2.0'
}
```


Figura 3.3: Tranzacție exemplu de tip CREATE


```
{
  id: '4169a8aa7198351bd68e07b88e6bd9d81c8abce68c44399fd9da37f9813585c6',
  operation: 'TRANSFER',
  outputs: [
    {
      condition: {
        details: {
          type: 'ed25519-sha-256',
          public_key: 'CsyuFvftcS1z7GVJbuyqp3WDuzortdnYUbL62mbjT6in'
        },
        uri: 'ni:///sha-256;ivf63Q2D4aLHcnwnxh3mHeX4uDrcxeK0UOavhZ6lDgs?fpt=ed25519-sha-256&cost=131072'
      },
      amount: '1',
      public_keys: [ 'CsyuFvftcS1z7GVJbuyqp3WDuzortdnYUbL62mbjT6in' ]
    }
  ],
  inputs: [
    {
      fulfillment: 'pGSAIL3ch0Bndp2N6Fq31vtnw0TI-x_nBqx5rUReiRAwoSJSgUA6T_jf7l6LDcxoLSsU03TrF9JokE59N_ePxwdB7
6aRNcP9MC3RzN-T3F1r-800z7BMqGLZobpwDHEtAcnwPaAD',
      fulfills: {
        output_index: 0,
        transaction_id: '5bf6d751911069f9fc9b0a0ce49150fa0e042a10acd4892fb2b9dd5dc9bd5553'
      },
      owners_before: [ 'Dn96HxLYxfcdT6gJkqeUVE29h7qFmcZ9uZ4Y5CUSdS9o' ]
    }
  ],
  metadata: null,
  asset: {
    id: '5bf6d751911069f9fc9b0a0ce49150fa0e042a10acd4892fb2b9dd5dc9bd5553'
  },
  version: '2.0'
}
```


Figura 3.4: Tranzacție exemplu de tip TRANSFER

Capitolul 4

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge

4.1 Introducere

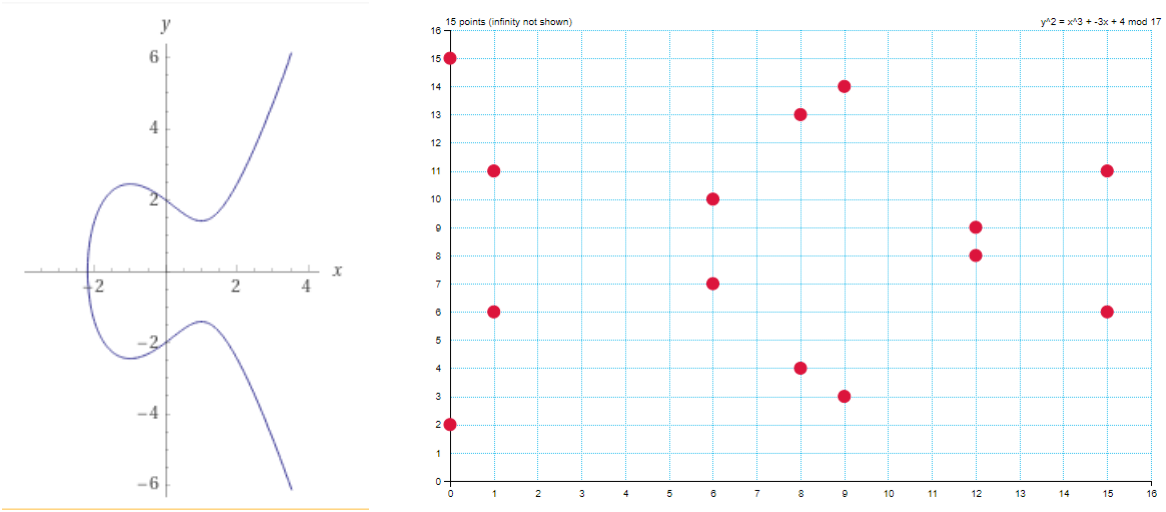
În acest capitol voi vorbi de următorul element principal care oferă încredere conținutului distribuit de aplicația TrustNews și anume despre Zero Knowledge Proofs (ZKPs). Voi aborda elementele criptografice de bază și etapele ce trebuie desfășurate pentru a putea genera și folosi o structură de dovedire.

4.2 Concepte criptografice

Conceptul de la care se pornește este criptografia bazată pe curbe eliptice. O curbă eliptică este reprezentarea în plan a unei funcții non-singulare de forma $C : y^2 = x^3 + ax + b$. Câmpul de valori este grupul finit $\mathbb{F}_p$ unde p este un număr prim suficient de mare [23].

Criptografia în acest context se bazează pe ECDLP acronim ce provine de la *elliptic curve discret logarithm problem* (Problema algoritmului discret pe curbe eliptice). ECDLP presupune existența a două puncte P și Q pe curba C cu valori în câmpul finit $\mathbb{F}_p$ și a unui număr întreg k astfel încât kP = Q. Dacă P și k sunt cunoscute atunci găsirea punctului Q se poate realiza eficient, dar dacă se cunosc doar punctele P și Q problema găsirii numărului k este foarte dificilă.

O curbă eliptică în plan realizate peste numerele reale va arăta întotdeauna continuă și va fi simetrică față de axa Ox. În momentul în care restrângem la câmpul finit $\mathbb{F}_p$ graficul devine o colecție de puncte sau mai exact colecție de perechi de puncte, elementele din perechi vor fi simetrice față de axa imaginară dată de $y = p/2$.



(a) In $\mathbb{R}$ (b) In $\mathbb{F}_{17}$

Figura 4.1: Curba eliptică $C : y^2 = x^3 - 3x + 4$; Grafice realizate cu [16] și [5]

Înainte de a defini operațiile pe care le putem face trebuie să definim elementul neutru numit și punctul la infinit care satisface relațiile P + O = P și P + (-P) = O pentru orice P.

Negația unui punct $P(x_p, y_p)$ se definește ca simetricul său față de axa imaginară $y = p/2$ ($-P = (x_p, -y_p)$). Adunarea dintre două puncte P și Q se definește grafic ca negația punctului de intersecție dintre curbă și dreapta dată de cele două puncte ($P(x_p, y_p) + Q(x_q, y_q) = R(x_r, y_r) \Rightarrow -R$). În situația în care P = Q adunarea se numește dublare iar dreapta dată de punct este tangenta. Înmulțirea cu un scalar k poate fi acum definită ca suma repetată a aceluiași punct.

Peste punctele de pe curbe eliptice se pot defini perechi date de funcții de evaluare biliniare notate $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Aceste funcții trebuie să respecte biliniaritatea și non-degenerarea date de condițiile:

- $e(P, Q + R) = e(P, Q) \cdot e(P, R), \quad \forall P \in \mathbb{G}_1, \forall Q, R \in \mathbb{G}_2$
- $e(P + Q, R) = e(P, R) \cdot e(Q, R), \quad \forall P, Q \in \mathbb{G}_1, \forall R \in \mathbb{G}_2$
- $\forall P \in \mathbb{G}_1, P \neq 0, \quad \exists Q \in \mathbb{G}_2 : e(P, Q) \neq 1$
- $\forall Q \in \mathbb{G}_2, Q \neq 0, \quad \exists P \in \mathbb{G}_1 : e(P, Q) \neq 1$

Aceste perechi de puncte se pot folosi atât pentru atacuri ce demonstrează că anumite curbe eliptice sunt slabe criptografic cât și ca unelte pentru noua paradigmă de demonstrare Zero-Knowledge.

Un alt procedeu util în ZKPs este criptarea homomorfică, cunoscută în literatura de specialitate și ca schema de comitere, și presupune stabilirea unei informații și păstrarea ei sub formă criptată până la decizia celui care cripteaza de a o face publică.

Rolul criptarii homomorfice este de a ne asigura în primul rând ascunderea informații și în al doilea rând de a ne permite să folosim mai departe în operații valoarea criptată. Pentru ca o funcție HH(x) să satisfacă cele două nevoi trebuie să satisfacă următoarele criterii:

- Este ușor de calculat HH(x) pornind de la x, dar având HH(x) este foarte dificilă găsirea lui x.
- Două valori diferite x și y vor rezulta în două valori HH(x) și HH(y) diferite
- Liniaritatea trebuie păstrată: $HH(\alpha x + \beta y) = \alpha HH(x) + \beta HH(y)$

Pentru a răspunde tuturor cerințelor se apelează la o formă de criptare $HH(x) = g^x$. Aritmetica se realizează peste $\mathbb{Z}_p$, unde p este un număr prim mare. Valoarea g este aleasă ca generatorul grupului ciclic $\mathbb{Z}_p^*$, care ne poate da toate valorile din grup ridicând-o la puteri diferite. Primul criteriu este satisfacut de problema algoritmului discret, $HH(x) = g^x (mod\ p)$ va fi ușor de calculat dar aflarea lui x cunoscând HH(x) este grea. Criteriul liniarității este satisfacut, $HH(x+y) = g^{x+y} = g^x \cdot g^y = H(x) \cdot H(y)$ unde $x \neq y \rightarrow HH(x) \neq HH(y)$.

Alte elemente țin de problema criptării folosind polinoame. Acestea vom vedea că stau la baza programelor aritmetice care pentru a-și demonstra corectitudinea trebuie să satisfacă o relație de forma $t(x)h(x) = l(x)r(x) - o(x)$, unde t,h,l,r,o sunt polinoame.

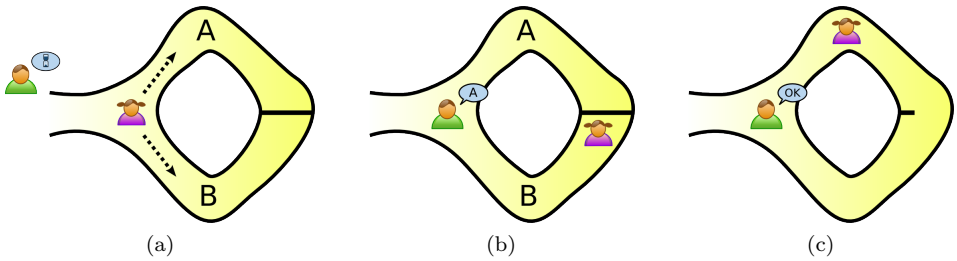
4.3 Zero-Knowledge Proofs (ZKPs)

Un mecanism de tip Zero-Knowledge Proof este un protocol criptografic între două părți comunicante și în care un doveditor convinge un verificator că o informație este adevărată sau că se cunoaște o anumită informație fără a o expune direct verificatorului [23].

Pentru ca interacțiunea dintre doveditor și verificator să se realizeze pe baza unui ZKP corect, acesta trebuie să respecte anumite proprietăți.

- **Completness**. Proprietatea asigură că dacă ambii participanți sunt onești atunci se respectă convenția protocolului, un doveditor vrea să demonstreze că știe o informație adevărată iar un verificator se va convinge că informația este întradevăr cum susține doveditorul că este.
- **Soundness**. Dacă doveditorul încearcă să păcălească verificatorul atunci acesta își poate da seama cu probabilitate mare și poate să refuze dovada primită.
- **Zero-Knowledgeness**. Caracterul Zero-Knowlegde se referă la incapacitatea verificatorului de a afla informația propriu-zisă pe care încearcă doveditorul să o demonstreze. El poate afla doar valoarea de adevăr pentru ceea ce susține doveditorul.

În sine un protocol ZKP standard este un proces interactiv în care cei doi membri pot transmite mesaje în mai multe etape consecutive și independente până când decid amândoi că ceea ce sustin este corect. Un celebru exemplu este peștera lui Ali Baba. Problema implică o peștera în formă circulară cu o singură intrare și cu o ușa magică la mijlocul formei circulare. Alice vrea să îl convingă pe Bob că știe parola pentru a deschide ușa fără să îi spună direct. Pentru a se convinge Bob alege să rămână în afară peșterii, în timp ce Alice alege să meargă pe oricare din cele două direcții posibile până ajunge la ușa. Apoi Bob intră în peștera până la bifurcatie și o roagă pe Alice să vină pe un drum ales de el. Dacă drumul ales de Bob este de partea opusă ușii, Alice va trebui să folosească parola și să iasă pe partea cerută demonstrându-i lui Bob că poate face acest lucru. Procesul se poate repeta de oricâte ori până când Bob este convins că Alice știe cu adevărat parola și nu doar ghicește.



(a) (b) (c)

Figura 4.2: Problema peșterii lui Ali Baba - sursă [17]

4.4 zkSNARKs

Abrevierea zkSNARKs provine de la succinct non-interactive arguments of knowledge și este construit peste conceptul prezentat anterior moștenind toate proprietățile acestuia [32].

Caracterul succint vine de la faptul că mesajele transmise între cele două părți comunicante au dimensiuni foarte mici. Practic computatia realizată pentru crearea unei dovezi ne permite ca în final să avem structuri mici care se pot verifica foarte ușor.

Se elimină nevoia de interacțiune între cei doi, va exista o etapă de preprocesare după care un singur mesaj va fi suficient pentru a trimite dovada. În plus verificarea va putea fi efectuată de oricine fără a fi nevoie de interacțiune în plus și ajută în principal atunci când este folosită în cadrul blockchain.

Argumentele fac referire la proprietatea de soundness și implică faptul că un verificator poate avea încredere doar într-un doveditor cu resurse de procesare limitate. Un doveditor malițios cu suficient de multă putere de procesare poate crea dovezi ce au la bază informații false și care pentru verificator par corecte.

Caracterul de cunoaștere este adresat doveditorului ce trebuie să cunoască setul de informații numit și witness pentru a putea construi o dovadă pe baza lui. Totodată verificatorul nu află nimic despre valorile acestor informații.

Dovezile de tip zkSNARKs se pretează, din câte se cunoaște până acum, problemelor din clasa NP. Această clasă de probleme este o generalizare a clasei P ce curpinde programe rezolvabile într-o complexitate polinomiala. Clasa de probleme NP se poate descrie ca mulțimea problemelor L care conțin la rândul lor un program V de complexitate polinomială și care este folosit pentru a verifica o informație cunoscând de asemenea un witness de dimensiune polinomială. Putem spune și că programul L având ca dată de intrare x se termină și întoarce o valoare de adevăr pozitivă dacă există un witness w astfel încât V(x,w) să întoarcă valoarea de adevăr pozitivă.

Un exemplu de astfel de problema NP este cea a verificării satisfiabilității unei formule SAT(f). Verificarea se poate realiza pentru orice funcție în timp polinomial dar doar cu ajutorul unui witness care în cazul acesta va cuprinde valorile de adevăr asociate fiecărei variabile astfel încât valoarea de adevăr a întregii formule să fie 1. De la acest exemplu se poate defini și proprietatea asociată oricărui program NP și care spune că un program L(x) poate fi echivalat cu SAT(f(x)). Verificarea unei tranzacții în blockchain poate fi un exemplu, problema se reduce la satisfacerea unei formule create pe baza criteriilor de construcție a unei tranzacții, witness-ul aici va consta din valori corecte incluse în tranzacție.

4.5 Quadratic Programs

Un Quadratic Span Programs (QSP) constă într-un set de polinoame iar problema pe care o rezolvă este găsirea unei combinații liniare astfel încât produsul rezultat să fie un multiplu al unui alt polinom țintă [32]. Mai mult decât atât coeficientii din polinoame vor fi dictați de input-urile programul și de witness.
Un program QSP peste un spațiu finit $\mathbb{F}$ cu input de lungime n conține o serie de polinoame $v_0, ...v_m$, o serie de polinoame $w_0, ..., w_m$, un polinom țintă t și o funcție injectivă $f : \{(i,j) \mid 1 \leq i \leq n,\ j \in \{0,1\}\} \rightarrow \{1, ..., m\}$. Validarea unui input u se face doar dacă există coeficienții $a_1, ..., a_m,\ b_1, ..., b_m$ astfel încât polinomul țintă t să dividă $(v_0 + a_1v_1 + ... + a_mv_m)(w_0 + b_1w_1 + ... + b_mw_m) = v_aw_b$ cu $a_k, b_k = 1$ dacă există perechea de forma $(i, u[i])$ pentru care $f(i, u[i]) = k$, u[i] = Valoarea bit-ul de pe poziția i și $a_k, b_k = 0$ în rest.

Procesul prin care doveditorul arată că știe o informație se transformă în dovedirea că el cunoaște polinomul h astfel încat $th - v_aw_b = 0$. Acțiunea verificatorului va fi validarea ecuației menționate anterior, însă fară a pierde generalitatea, se poate alege o valoare s, se pot evalua polinoamele obținând $t(s), v_a(s), w_b(s)$ și se poate verifica ecuația $t(s)h(s) = v_a(s)w_b(s)$.

Superioare programelor descrise anterior sunt Quadratic Arithmetic Program (QAP) care merg pe un drum similar însă pot valida construcții aritmetice și se folosesc de 3 polinoame în locul celor 2 (v_a și w_b) de la QSP.

Construcția acestor tipuri de programe se bazează pe circuite aritmetice. Aceste circuite nu sunt altceva decât grafuri orientate ce pornesc de la valori de input, care se unesc prin fire de porți reprezentând operațiile aritmetice efectuate. De exemplu, ceea ce formal se poate scrie ca $C : \mathbb{F}^2 \times \mathbb{F}^2 \rightarrow \mathbb{F}, C(a_1, a_2, a_3, a_4) = (a_1 + a_2) * a_3 * (a_3 - a_4)$ se poate reprezenta grafic în imaginea urmatoare.

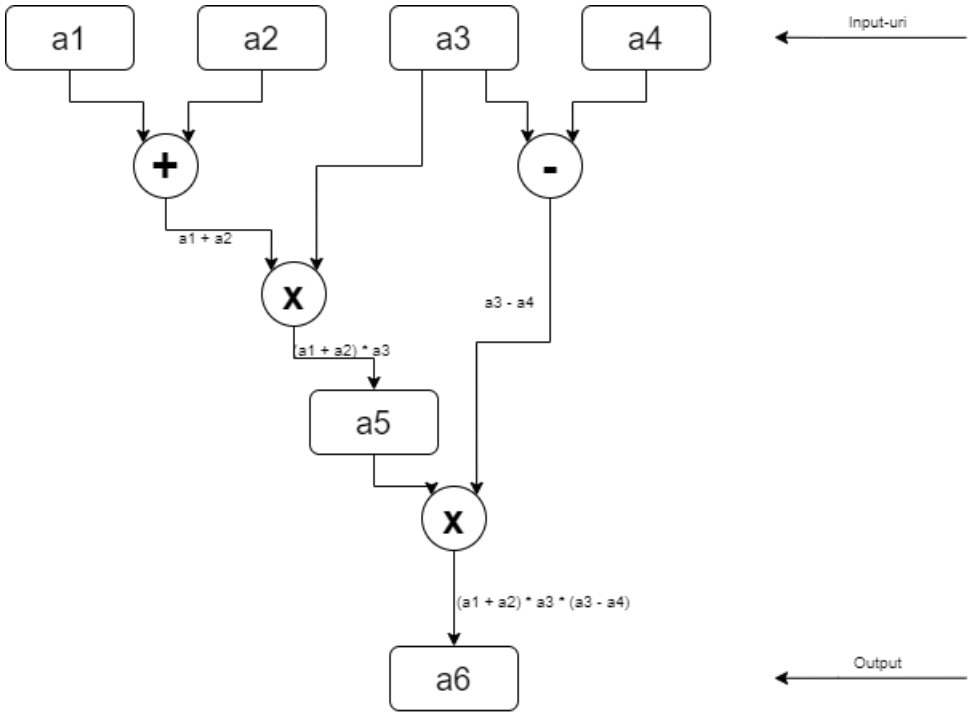


Figura 4.3: Exemplu circuit aritmetic - Grafic realizat cu [3]

Se definește mulțimea punctelor țintă M conținând indecși pentru fiecare poartă folosită pentru operația de înmulțire și mulțimea W ce conține fire folosite pentru input-urile programului și etichete output-urilor porților înmulțire [29]. Pe baza mulțimii M se definește polinomul țintă $T(x)$ de grad d ca fiind $\prod_{g \in \mathbb{M}}(x - r_g)$ unde r_i sunt rădăcini diferite. Pentru exemplul de sus $T(x) = (x - r_5)(x - r_6)$.

Pe lângă polinomul țintă se definesc 3 mulțimi de polinoame [27]:

- Polinoamele de la stânga. $L_i(r_g) = c_{g,L,i},\ i \in I_{g,L}$ și $L_i(r_g) = 0$ în rest
- Polinoamele de la dreapta. $R_i(r_g) = c_{g,R,i},\ i \in I_{g,R}$ și $R_i(r_g) = 0$ în rest
- Polinoamele de output. $O_i(r_g) = 1,\ dacă\ i = g$ și $O_i(r_g) = 0$ în rest

$I_{g,L}$ și $I_{g,R}$ sunt submulțimi ale lui W și reprezintă etichetele sau firele ce intră în poarta g venind din stânga, respectiv dreapta, și care nu au mai fost folosite în alte porți multiplicative înaintea lui g. $c_{g,L,i}$ și $c_{g,R,i}$ sunt coeficienții cu care eticheta i intră în poarta g venind din stânga, respectiv dreapta. $L_0(r_g)$ respectiv $R_0(r_g)$ vor fi valorile constantelor ce intră în porțile g. Se vor calcula valorile pentru fiecare poartă multiplicativă.
Valorile asociate evaluărilor polinoamelor pentru exemplul de sus sunt

$L_0(r_5) = 0$ $R_0(r_5) = 0$ $O_0(r_5) = 0$
$L_0(r_6) = 0$ $R_0(r_6) = 0$ $O_0(r_6) = 0$
$L_1(r_5) = 1$ $R_1(r_5) = 0$ $O_1(r_5) = 0$
$L_1(r_6) = 0$ $R_1(r_6) = 0$ $O_1(r_6) = 0$
$L_2(r_5) = 1$ $R_2(r_5) = 0$ $O_2(r_5) = 0$
$L_2(r_6) = 0$ $R_2(r_6) = 0$ $O_2(r_6) = 0$
$L_3(r_5) = 0$ $R_3(r_5) = 1$ $O_3(r_5) = 0$
$L_3(r_6) = 0$ $R_3(r_6) = 1$ $O_3(r_6) = 0$
$L_4(r_5) = 0$ $R_4(r_5) = 0$ $O_4(r_5) = 0$
$L_4(r_6) = 0$ $R_4(r_6) = -1$ $O_4(r_6) = 0$
$L_5(r_5) = 0$ $R_5(r_5) = 0$ $O_5(r_5) = 1$
$L_5(r_6) = 1$ $R_5(r_6) = 0$ $O_5(r_6) = 0$
$L_6(r_5) = 0$ $R_6(r_5) = 0$ $O_6(r_5) = 0$
$L_6(r_6) = 0$ $R_6(r_6) = 0$ $O_6(r_6) = 1$

Din aceste valori se pot crea polinoamele $L_i(x)$, $R_i(x)$, $O_i(x)$

$L_0(x) = L_3(x) = L_4(x) = L_6(x) = 0$
$L_1(x) = \frac{1}{r_5 - r_6}(x - r_6)$
$L_2(x) = \frac{1}{r_5 - r_6}(x - r_6)$
$L_5(x) = \frac{1}{r_6 - r_5}(x - r_5)$

$R_0(x) = R_1(x) = R_2(x) = R_5(x) = R_6(x) = 0$
$R_3(x) = \frac{1}{r_5 - r_6}(x - r_6) + \frac{1}{r_6 - r_5}(x - r_5)$
$R_4(x) = \frac{-1}{r_6 - r_5}(x - r_5)$

$$O_0(x) = O_1(x) = O_2(x) = O_3(x) = O_4(x) = 0$$
$$O_5(x) = \frac{1}{r_5 - r_6}(x - r_6)$$
$$O_6(x) = \frac{1}{r_6 - r_5}(x - r_5)$$

Polinomul final $P(x)$ va fi egal cu $L(x) \cdot R(x) - O(x)$, unde
$L(x) = L_0(x) + \sum_{i=1}^{|W|} a_i L_i(x)$,
$R(x) = R_0(x) + \sum_{i=1}^{|W|} a_i R_i(x)$,
$O(x) = O_0(x) + \sum_{i=1}^{|W|} a_i O_i(x)$,

Problema pe care o rezolvă un QAP poate fi acum formulată ca verificarea tuplurilor $(a_1, ..., a_n, a_{n+1}, ..., a_m)$ cu $a1, ..., a_n$ input-uri și $a_{n+1}, ..., a_n$ output-uri care anulează polinomul P în rădăcinile r_g, altfel spus $T(x)$ divide $P(x)$.

Tuplul $(1, 2, 0, 1, 0, 0)$ este valid și satisface programul QAP din exemplu.
$P(x) = [1 \cdot L_1(x) + 2 \cdot L_2(x)] \cdot [1 \cdot R_4(x)] - [O(x)]$
$P(x) = \frac{3}{r_5 - r_6}(x - r_6) \cdot \frac{-1}{r_6 - r_5}(x - r_5) - 0$
$P(x) = \frac{3}{r_5 - r_6} \cdot \frac{-1}{r_6 - r_5} \cdot T(x)$

4.6 De la QAP la zkSNARKs

Cadrul în care acționează protocoalele zkSNARK sunt unele noninteractive. De aceea primul pas în realizarea unor astfel de structuri de dovedire constă în etapa de setup în care se creează common reference string (CRS).

Această etapă se realizează o singură dată pentru o anumită problemă dată și constă în primul rând în transformarea problemei în QAP și aflarea polinoamelor specifice lui. Se aleg apoi o serie de numere random numite toxic waste cu care se vor evalua polinoamele. Aceste valori trebuie șterse imediat ce au fost folosite, aflarea lor de către cineva rău intenționat îi poate permite crearea unor dovezi corecte pentru informații false. CRS trebuie să fie public și conține aceste evaluări ale polinoamelor, cu alte cuvinte aceasta aduce cadrul matematic prin care se pot construi și verifica dovezi.

Se alege o funcție de criptare homomorfică $HE(x) = g^x$ peste curbe eliptice cu valori în $\mathbb{F}_p$ unde g este generatorul grupului ciclic $\mathbb{F}_p$. Se aleg două constante s și α, numere random care participă la calcularea seriei de valori $HE(s^0), HE(s^1), ..., HE(s^d)$ și seriei de valori $HE(\alpha s^0), HE(\alpha s^1), ..., HE(\alpha s^d)$, d este gradul maxim întâlnit în polinoame. Aceste două serii vor fi publicate în cadrul CRS.

Seria $HE(s^i)$ va fi folosită de doveditor pentru a arăta ca știe polinomul final P(x) și implicit elementele componente L(x), R(x) și O(x). El va trebui să evalueze $HE(P(s))$ și va putea face acest lucru deoarece calculul se bazează pe informații publice și nu pe valoarea s.

$$HE(P(s)) = HE(c_d s^d + ... + c_0 s^0) = g^{c_d s^d + ... + c_0 s^0} = g^{c_d s^d} \cdot .... \cdot g^{c_0 s^0} = HE(s^d)^{c_d} \cdot .... \cdot HE(s^0)^{c_0}$$

Seria $HE(\alpha s^i)$ va fi folosită la verificare pentru a fi siguri că polinomul doveditorului este corect și construit folosind schema comună. Calculul este asemănător celui precedent și se bazează tot pe polinomul P.

$$HE(\alpha P(s)) = HE(\alpha(c_d s^d + ... + c_0 s^0)) = HE(\alpha s^d)^{c_d} \cdot ... \cdot HE(\alpha s^0)^{c_0}$$

Verificarea propriu-zisă va fi realizată cu ajutorul unei funcții e de evaluare pentru perechi de puncte pe curba eliptică.

$$e(HE(P(s)), g^\alpha) = e(HE(\alpha P(s)), g)$$
$$e(g^{P(s)}, g^\alpha) = e(g^{\alpha P(s)}, g)$$
$$e(g, g)^{\alpha P(s)} = e(g, g)^{\alpha P(s)}$$

Un verificator poate totuși obține în forma aceasta informații despre dovadă. Dacă el ar acționa ca și doveditor și ar ști o altă informație corectă, ar putea crea o dovadă proprie și poate compara valorile $HE(P(s))$. Dacă sunt identice atunci verificatorul află exact informația știută de doveditor, iar dacă nu, află faptul că ce a verificat anterior nu este aceeași informație cu ce a dovedit el că știe.

Pentru a crea caracterul zero-knowlegde este adăugat un element random de către doveditor. El va calcula și va trimite $HE(\delta + P(s))$ și $HE(\alpha(\delta + P(s)))$. Verificatorul va efectua aceleași calcule fără să știe valoarea numărului random δ.

$$e(HE(\delta + P(s)), g^{\alpha}) = e(HE(\alpha(\delta + P(s))), g)$$

$$e(g^{\delta+P(s)}, g^{\alpha}) = e(g^{\alpha(\delta+P(s))}, g)$$

$$e(g, g)^{\alpha(\delta+P(s))} = e(g, g)^{\alpha(\delta+P(s))}$$

Capitolul 5

TrustNews

5.1 Descriere

TrustNews este aplicația care își propune să întregească toate elementele criptografice prezentate în teza de disertație pentru a crea o formă stabilă și sigură împotriva știrilor false. Ea își propune să acumuleze un număr mare de parteneri și cu timpul poate chiar să înlocuiască multitudinea de platforme pe care trusturile de știri publică informații.

Ea își propune să înlocuiască fundamental elementul clasic întâlnit în marea majoritate a platformelor de știri. Vorbesc despre bazele de date centralizate care salvează întreg conținutul într-un singur punct de control și de care depinde platforma de distribuire sub formă de website web.

Atacurile asupra website-urilor de acest tip pot avea la bază două scopuri ce vizează datele. Primul scop ar fi criptarea datelor printr-un atac ransomware iar al doilea ar consta în SQL Injection pentru a altera informația. Acest al doilea scop este principalul vizat în contextul știrilor false și poate face ca o simplă modificare a unei propoziții să schimbe sensul întregului text.

Pentru a elimina aceste riscuri TrustNews se folosește de blockchain pentru a stoca știrile în structuri bine definite. Blockchain-ul prin natura lui implică o rețea în care sunt mulți participanți fiecare putând să contribuie cu noi informații. Un atacator nu mai poate acum acționa asupra unui singur punct de control și ar fi nevoit să acționează asupra întregii rețele iar asta ar implica să dețină o putere de procesare mare. De asemenea datele sunt acum salvate pe fiecare nod participant în rețea și fiecare informație are un posesor. În acest caz unui atacator îi este foarte greu să altereze informația chiar și din propria copie regăsită pe nodul pe care lucrează.

Fiecare nod va putea avea două roluri în aplicație. Primul este cel implicit și implică faptul că pe orice nod se va regasi o copie a întregului conținut salvat în blockchain dat de totalitatea știrilor. Al doilea rol este cel de extractor (scraper) de știri. Aplicația TrustNews își va prelua informația doar din surse de încredere, trusturi de știri cu o reputație crescută despre care putem să afirmăm că nu publică știri false.

Siguranța informației regăsită într-o sursă este în responsabilitatea celor care administrează respectivul website. TrustNews își propune astfel să fie pe lângă o zona mai eficientă din care să fie preluate știri și o formă de backup pentru totalul extras. Ea nu își propune să îmbunătățească securitatea website-urilor de unde extrage știri ci se bazează pe încrederea în trusturile alese. Aplicația va scana constant mediile și va prelua de fiecare dată prima formă în care știrile au fost publicate. Odată salvată informația în blockchain avem garanția că textul este cel cu adevărat real. Un atacator care ar reuși să atace website-ul și să modifice știrile dându-le un nou sens nu ar putea efectua acest lucru la fel de ușor și în forma nouă.

Tehnologia BigchainDB este folosită pentru a acoperi cadrul blockchain. Aplicația TrustNews constă într-o rețea privată în care va exista întotdeauna o autoritate de control în care membrii participanți vor avea încredere. El este de regulă primul nod din rețea care distribuie software-ul aplicației și informațiile publice lui altor participanți. Este util să fie un context privat deoarece nu ne dorim un număr foarte mare de participanți și ne dorim totodată un factor de decizie mare asupra a ce trebuie să se facă când cineva încalcă anumite reguli. De asemenea cadrul privat reduce nevoia alocării unei cantități de resurse mari pentru a desfășura procesele din algoritmul blockchain.

BigchainDB permite în ultimele versiuni ale software-ului tranzacții specifice de tip votare (election). Cu acestea se pot adăuga ușor membri noi în rețeaua blockchain. Tranzacțiile vor ajunge ca oricare alta la fiecare nod din rețea existent până în acel punct și fiecare va putea lua o decizie în algoritmul de consens. În acest mod putem fi siguri că fiecare nod cunoaște toate celelalte noduri.

O proprietate importanta a aplicației este că își extrage știrile dintr-o varietate foarte mare de platforme. Acest lucru este util din punctul de vedere al conținutului deoarece se salvează informații din toate domeniile dar și din punct de vedere al securității. Lista de platforme trebuie să fie publică pentru a crea un grad sporit de încredere cu cei ce citesc demonstrând că datele sunt luate de pe platforme cu o reputație crescută. De asemenea pentru ca un atacator să poată influența conținutul știrilor din blockchain ar însemna să reușească să treacă de mecanismele de securitate a unui număr mare de website-uri.

Cu toate că lista totală de surse este publică, nu este obligatoriu nevoie ca fiecărei știri salvată în blockchain să i se eticheteze și sursa de unde provine. Un atacator s-ar putea închipui ca un nod corect pentru a avea acces la procesul de tranzacționare și ar putea urmări pasiv ce conținut se extrage și de unde. Astfel el ar putea să își creeze surse de atac reprezentând platformele de interes pentru el, cu intenția de a adăuga știri contrafăcute știind că extractorul la un moment dat va ajunge să preia informația falsă.

Pentru a masca și totodată verifica sursa vom folosi structurile de dovedire de tip zkSNARKs. Conversia către un context matematic se poate face prin alocarea un număr prim diferit fiecărei platforme. În acest fel un nod extractor va trebui să calculeze produsul tuturor valorilor și va trebui să demonstreze că numărul prim asociat platformei de pe care a extras știrea divide produsul, fără a-l expune.

Pentru a extrage știrile aplicația TrustNews poate fi configurată să folosească orice utilitar de tip scraper. Acesta va fi responsabil de navigarea către anumite puncte pe website cât și de preluarea textului știrilor din conținutul HTML.
Având forma prin care se extrage textul unei știri și forma prin care se ascunde sursa de unde este preluată informația se poate acum lărgi structura unei tranzacții din blockchain. Fiecare tranzacție va avea acum câmpul de date compus din patru elemente titlul știrii, textul știrii, dovada sursei și momentul de timp când se creează.

5.2 Modul de funcționare

TrustNews este construită folosind limbajul de programare NodeJS pentru a putea integra ușor totalul de tehnologii folosite în aplicație. Totuși se poate converti la alt limbaj de programare dacă există module similare scrise în noul limbaj ales. Python este un astfel de exemplu ce permite utilizarea unor module similare pentru ca aplicația să aibă același comportament.

Funcția principală a aplicației numită *Main()* [12] este cea în care se programează extragerile de știri regulate la un interval de secunde ales de noi. Se poate seta în funcție de preferință un interval de 30 de secunde pentru a extrage cu o viteză foarte mare sau un interval de chiar câteva ore pentru a încetini fluxul.

Funcția care face extragerea propriu-zisă este *Extract()* [9] și cuprinde un plan bine definit de apelări de funcții din librării standard folosite pentru a interacționa cu tehnologiile alese.

În primul rând se folosește de structura publică ce mapează fiecare platformă de știri de un număr prim diferit. Pentru un exemplu restrâns am ales să folosesc

```
var news_platforms = [
  { name: "CNN", prime_nr: "3" },
  { name: "BBC", prime_nr: "5" },
  { name: "The Economist", prime_nr: "7" },
  { name: "The Wall Street Journal", prime_nr: "11" },
  { name: "World Health Organization", prime_nr: "17" },
];
```

, însă într-un context real această mapare poate conține zeci de astfel de perechi.
Aplicația, la fiecare iterație, va calcula produsul tuturor numerelor prime și va alege la întâmplare o sursă din mulțimea dată.

În funcție de sursa aleasă se merge mai departe în etapa de parsare/scraping a conținutului HTML de pe website-ul asociat platformei de știri. Pentru scraping am folosit utilitarul *scraperjs* [33] ce ușurează mult procesul având nevoie doar de identificatorii HTML pentru a ști la ce elemente să se uite. Se folosește un scraper static pentru a putea iniția un lanț de promisiuni de tip Javascript, cu alte cuvinte promisiuni că procese noi vor fi create și vor rula în paralel pentru a extrage conținutul de care avem nevoie din pagini.

Cu ajutorul lui extragem titlul și conținutul știrii pe care le întoarcem în funcția de extragere de la care am plecat.

În continuare se construiește structura de dovedire ce atestă că sursa știrii este una folosită în cadrul aplicației TrustNews. Am folosit utilitarul *ZoKrates* [28] pentru a putea crea întregul sistem ce conține atât etapa inițială de configurare cât și etapele ulterioare de creare a dovezilor și de verificare a lor.

În primul rând se consideră etapa de configurare deja efectuată de către nodul supervizor asupra întregii rețele blockchain. El va scrie codul programul cu care se vor putea crea dovezi, îl va compila și va crea pe baza lui CRS-ul, cunoscut și ca perechea de chei pentru dovedire și pentru verificare. Toate aceste trei etape vor fi efectuate o singură dată iar la finalul lor nodul supervizor este responsabil să publice în cadrul rețelei toate valorile calculate.

Compilatorul ZoKrates lucrează cu programe scrise într-un limbaj propriu de nivel înalt foarte asemanator cu Python. Codul conține verificarea logică că parametrul public reprezentat de produsul numerelor prime divide parametrul păstrat secret (witness) reprezentat de numarul prim asociat sursei de unde a fost extrasă știrea precedentă.

```
def main(private u32 chosen_site_prime_nr, u32 product_of_primes) -> bool:
    return product_of_primes % chosen_site_prime_nr == 0
```

Pentru a genera o dovadă folosim funcția *GenerateProof(witness)* [11] care va primi sub formă de șir de caractere valorile parametrilor din funcție. El se va folosi la bază de două dintre apelurile utilitarului, *compute-witness* pentru a genera o structură binară a valorilor cunoscute de doveditor și *generate-proof* pentru a construi fișierul dovadă în format JSON având ca argumente fișierul binar generat de comanda precedentă și cheia de dovedire cunoscută public.

Pentru verificare se poate folosi funcția *VerifyProof(proof)* [14] care va primi ca argumente fișierul dovadă, va prelua cheia de verificare publică și va întoarce rezultatul "PASSED" dacă verificarea se realizează cu succes.

Având cele trei elemente deja preconstruite, titlu, conținut și dovadă, funcția poate continua la asamblarea tranzacției care le va curpinde pe toate.

În primul rând se presupune că fiecare nod în momentul în care intră în rețeaua aplicației TrustNews își va genera o pereche de chei criptografice pe care le va folosi la crearea și semnarea tranzacțiilor. Se poate folosi codul *GenerateNodeKeys.js* [10] pentru generarea unui fișier ce conține perechea necesară.

Tranzacția realizată folosind utilitarul scris în Javascript pentru BigchainDB va fi de tip CREATE, va fi creată folosind funcția *CreateNews()* [8] și va avea ca parametri cele trei elemente generate de funcțiile precedente și perechea de chei. Câmpul de date marcat ca asset va conține structura

```
var asset = {
    title: title,
    content: content_of_news,
    proof: proof,
    datetime: new Date().toString(),
};
```

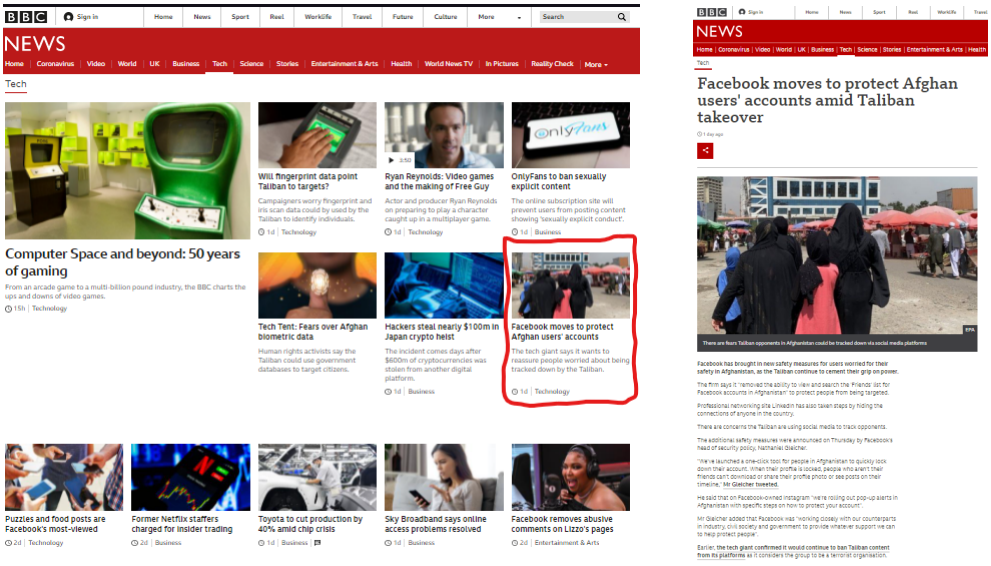
Nodul care va crea tranzacția va fi și cel carui îi va aparține știrea extrasă, de aceea cheia publică va fi folosită atât la câmpul issuers cât și la outputs. După ce tranzacția este semnată folosind cheia privată, este transmisă în rețeaua blockchain și un mesaj de tip "CREATE Transaction successfully posted." va fi afișat după ce își va termina ciclul de validare.

Pentru o dezvoltare în timp a aplicației a fost implementată și funcția *TransferNews()* [13]. Spre deosebire de funcția de creare, aceasta va primi ca parametru direct structura pentru input-uri conținând referințele către tranzacții deja procesate. Cheile folosite vor fi cea publică a nodului care va intra în posesia știrii și cea privată a actualului deținător pentru a dovedi că întradevar le poate accesa.

Ultimul pas din funcția de extagere este dat de afișarea conținutului din știre. Verificarea că o tranzacție a fost adaugată în blockchain se face folosind funcția *CheckTransaction(transaction_id)* [7] care apelează API-ul "/api/v1/transactions/${transaction_id}" și întoarce întregul conținut al tranzacției.

5.3 Exemplu de funcționare

Pentru testare am folosit un sistem de operare Ubuntu 18.04 având instalate Docker, NodeJS, ZoKrates CLI, npm și toate modulele asociate aplicației: scraperjs, child_process, axios, bigchaindb-driver, fs, util. Pentru BigchainDB am folosit un container cu imaginea de tip all-in-one.
În exemplul următor preiau o știre de pe platforma online BBC.com. Pe pagina principală regăsesc toate titlurile și toate URL-urile către paginile actualelor știri. Aleg o pereche la întâmplare și preiau conținutul text folosind o nouă instanță de scraper pentru URL-ul știrii alese.



(a) (b)

Figura 5.1: Știre de pe website-ul oficial - sursă [1]

```
danielpirvu@danielpirvu:~/Workdir/TrustNews$ node MainProgram.js

> The extractor has started

> Every 5 minutes a new transaction will be made on blockchain

> To stop the program press Ctrl + C

> ##########################################################################################
> CREATE Transaction fedda7d80170482dca8868c903f5384033b11493b2e1e9a6cfd464ca7bf8cbc4 successfully posted.
> ##########################################################################################
```

Figura 5.2: Output după procesarea tranzacției

Procesul de dovedire se folosește de convenția stabilită că fiecare platformă de știri are asociat numărul prim. Website-ul BBC.com are asociat numărul 5 iar produsul calculat pe baza tuturor asocierilor este 19635 = 3 * 5 * 7 * 11 * 17. În dovadă se văd doar constantele folosite pentru a arata că se cunosc polinoamele implicate în QAP de către

doveditor datorită valorii witness. Valorile de input publice vor fi de asemenea regăsite în dovadă. În cazul acesta, valoarea 0x4cb3, reprezentând produsul total în valoare hexazecimală, este prezentă.

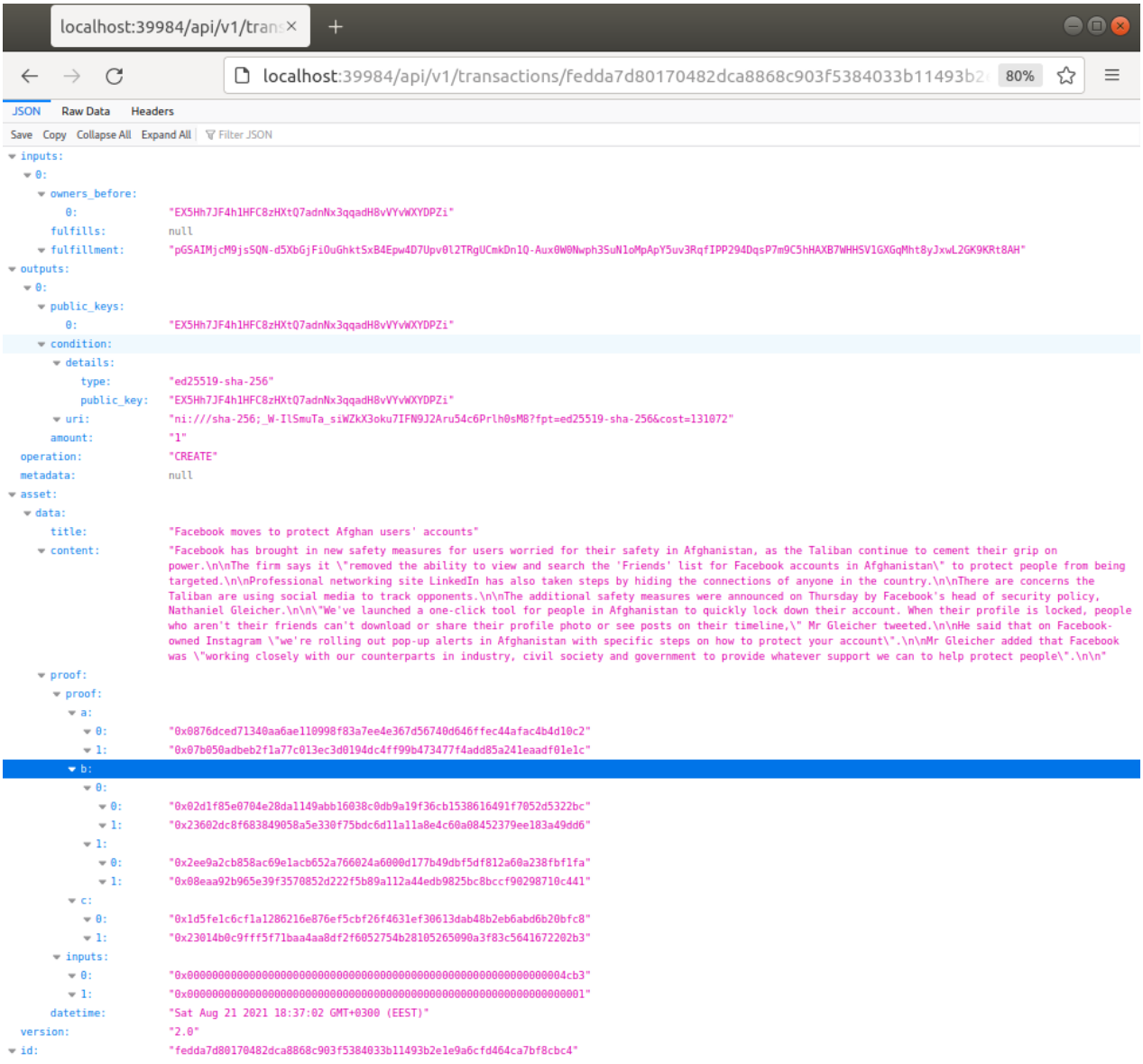


Figura 5.3: Tranzacție de tip CREATE pentru TrustNews

5.4 Planuri de îmbunătățire

În final, fiecare nod va ajunge să conțină o cantitate mare de știri preluate din foarte multe surse diferite. O primă îmbunătățire ar putea consta în expunerea știrilor pe o platforma nouă administrată tot în cadrul rețelei blockchain. Practic un nod va juca și rolul de server web care va primi request-urile din afara rețelei.

În continuarea acestei idei fiecare nod extractor va putea tranfera știrea creată către nodul ales pentru a fi server web. Acesta din urmă va putea interoga ușor toate output-urile din toate tranzacțiile ce au cheia lui publică specificată. Astfel va putea obține id-urile tranzacțiilor și poate ajunge la conținutul știrii din interior.

Aplicația am văzut că tratează cazul în care nodurile pot fi deținute de oricine primește acces în rețeaua privată. Odată ce un nod este prezent el poate să extragă informații de pe platforme deja existente și unde publicarea se face de către persoane autorizate. Pentru a îmbunătăți performanță aplicației se poate implementa o formă simplificată de nod care va fi deținută de trusturile de media care ar putea salva direct în blockchain conținutul nou.

Mai rămâne în picioare problema știrilor duplicat. Fiecare nod va trebui să mențină o proprie listă a știrilor pe care deja le-a extras cât și o listă cu știri pe care alte noduri le-au procesat. Acest din urmă aspect este greu de realizat deoarece nodurile funcționează în paralel și nu comunica decât în momentul în care au creat deja tranzacțiile. Comparând două tranzacții cu același text al știrii vom vedea că sunt multe elemente ce le disting precum cheile criptografice, momentul de timp și chiar și dovezile.

O altă formă de duplicat ține de natura semantică a informației. Știm că o informație poate să fie publicată pe două sau mai mai multe platforme toate fiind de încredere. Pentru că titlurile și nuanța cuvintelor poate fi diferită de la o platforma la altă ar fi nevoie de un modul de inteligență artificială care ar putea calcula un grad de similitudine, valoare după care putem decide dacă două știri sunt identice sau nu.

Capitolul 6

Concluzii

Vedem cum conceptul de fake news a început să ia amploare de la o zi la alta. Se poate ajunge la campanii ce vizează grupuri mari de persoane cu scopul final de a dezinforma sau de a influența cititorii să perceapă o informație falsă ca fiind corectă.

Este astfel nevoie de o putere care să contrabalanseze prin crearea de conținut adevărat și ușor verificabil de către oricine. Încrederea este oferită trusturilor de media mari care și-au clădit o reputație în acest sens de-a lungul anilor de activitate. Câteva exemple ar fi BBC, CNN, The Economist și The Wall Street Journal însă pe lângă acestea există mult mai multe în întreagă lume [30].

Cu toate că trusturile publică informații corecte ele pot fi alterate dacă platformele pe care sunt distribuite nu respectă anumite standarde de securite. Atacatorii ar putea ținti aceste platfome cu scopul bine definit de a schimba sensul textului pentru a aduce avantaje sau dezavantaje unei anumite părți.

Pentru a spori gradul de securitate aplicația TrustNews este gândită să acopere o foarte mare parte din probleme platformelor clasice. Tehnologia blockchain ne permite să avem un cadru în care informația este foarte greu de alterat. Caracterul distribuit și mecanismele criptografice împiedică atacatorii să acționeze în acest sens.

În plus TrustNews aduce un strat adițional de obfuscare folosind mecanisme de tip zkSNARKs. Ele limitează atacatorii, care ar putea ajunge în rețeau blockchain închipuind membri corecți, prin imposibilitatea de a corela știrile preluate cu sursele lor.

Trebuie însă menționat că aplicația nu își propune să actioneaze asupra securității platformelor web de unde își extrage conținutul. Idea pe care se bazează este că mediile sunt sigure și că software-ul extractor este rapid și poate prelua de fiecare dată prima formă a știrii imediat după ce aceasta este prezentă pe una din platforme.

Aplicația este la nivel de concept și necesită resurse semnificative și un set de reguli de funcționare bine definit pentru a putea fi realizată la o scară largă.

Listă de figuri

2.1 Noduri complete - sursă: [35] 6
2.2 Noduri ușoare - sursă: [35] 8
2.3 Noduri lucrătoare (Mineri) - sursă: [35] 9
2.4 Exemplu de înlănțuire între tranzacții - sursă: [25] 14
2.5 Exemplu de înlănțuire între block-uri - sursă: [25] 16
2.6 Blockchain Usage Flowchart - sursă: [25] 20

3.1 Proprietăți BigchainDB - sursă: [24] 21
3.2 Noduri BigchainDB - sursă: [24] 22
3.3 Tranzacție exemplu de tip CREATE 26
3.4 Tranzacție exemplu de tip TRANSFER 26

4.1 Curba eliptică $C : y^2 = x^3 - 3x + 4$; Grafice realizate cu [16] și [5] 28
4.2 Problema peșterii lui Ali Baba - sursă [17] 30
4.3 Exemplu circuit aritmetic - Grafic realizat cu [3] 33

5.1 Știre de pe website-ul oficial - sursă [1] 43
5.2 Output după procesarea tranzacției 43
5.3 Tranzacție de tip CREATE pentru TrustNews 44

Bibliografie

[1] BBC.com. https://www.bbc.com/news/technology. (Ultima accesare: 13.08.2021).

[2] Bitcoin Protocol Documentation. https://en.bitcoin.it/wiki/Protocol_documentation. (Ultima accesare: 18.07.2021).

[3] Draw.io. https://app.diagrams.net.

[4] Ethereum Protocol Documentation. https://eth.wiki. (Ultima accesare: 18.07.2021).

[5] Graui. https://www.graui.de/code/elliptic2/.

[6] Tendermint. https://tendermint.com. (Ultima accesare: 28.07.2021).

[7] TrustNews CheckTransaction(). https://github.com/PirvuDanielCatalin/TrustNews/blob/main/BigchainDBSingleNode/CheckTransaction.js#L6. (Ultima accesare: 21.08.2021).

[8] TrustNews CreateNews(). https://github.com/PirvuDanielCatalin/TrustNews/blob/main/BigchainDBSingleNode/Create%26Transfer.js#L11. (Ultima accesare: 21.08.2021).

[9] TrustNews Extract(). https://github.com/PirvuDanielCatalin/TrustNews/blob/main/MainProgram.js#L20. (Ultima accesare: 21.08.2021).

[10] TrustNews GenerateNodeKeys(). https://github.com/PirvuDanielCatalin/TrustNews/blob/main/BigchainDBSingleNode/GenerateNodeKeys.js. (Ultima accesare: 21.08.2021).

[11] TrustNews GenerateProof(). https://github.com/PirvuDanielCatalin/TrustNews/blob/main/ZoKratesCLI/GenerateProof.js#L7. (Ultima accesare: 21.08.2021).

[12] TrustNews Main(). https://github.com/PirvuDanielCatalin/TrustNews/blob/main/MainProgram.js#L95. (Ultima accesare: 21.08.2021).

[13] TrustNews TransferNews(). https://github.com/PirvuDanielCatalin/TrustNews/blob/main/BigchainDBSingleNode/Create%26Transfer.js#L61. (Ultima accesare: 21.08.2021).

[14] TrustNews VerifyProof(). https://github.com/PirvuDanielCatalin/TrustNews/blob/main/ZoKratesCLI/VerifyProof.js#L7. (Ultima accesare: 21.08.2021).

[15] What is Tendermint. https://docs.tendermint.com/master/introduction/what-is-tendermint.html. (Ultima accesare: 29.07.2021).

[16] WolframAlpha. https://www.wolframalpha.com.

[17] Zero-knowledge proof. https://en.wikipedia.org/wiki/Zero-knowledge_proof. (Ultima accesare: 07.08.2021).

[18] BigchainDB Python Driver. docs.bigchaindb.com/projects/py-driver/en/latest/index.html, 2019. (Ultima accesare: 28.07.2021).

[19] BigchainDB Documentation. https://bigchaindb.readthedocs.io/en/latest/index.html, 2020. (Ultima accesare: 28.07.2021).

[20] BigchainDB Javascript Driver Documentation. https://docs.bigchaindb.com/projects/js-driver/en/latest/index.html, 2021. (Ultima accesare: 28.07.2021).

[21] Bitcoin Blockchain Size. https://ycharts.com/indicators/bitcoin_blockchain_size, Iulie 2021. (Ultima accesare: 15.07.2021).

[22] K. Andersson A. Monrat, O. Schelén. A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8805074, Septembrie 2019. (Ultima accesare: 21.07.2021).

[23] J.H. Joancomarti A.B. Rodriguez. zk-SNARKs - Analysis and Implementation on Ethereum. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/120126/6/albertobrTFM0620memory.pdf, Iulie 2020. (Ultima accesare: 26.07.2021).

[24] Germany BigchainDB GmbH, Berlin. BigchainDB 2.0 - The Blockchain Database. https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf, Mai 2018. (Ultima accesare: 15.07.2021).

[25] N. Roby K. Scarfone D. Yaga, P. Mell. Blockchain Technology Overview. https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf, Octombrie 2018. (Ultima accesare: 13.07.2021).

[26] Liam Frost. Bitcoin Blockchain Grows to 300 Gigabytes in Size. https://decrypt.co/42427/bitcoin-blockchain-grows-to-300-gigabytes-in-size, Septembrie 2020. (Ultima accesare: 15.07.2021).

[27] Ariel Gabizon. Explaining SNARKs. https://electriccoin.co/blog/snark-explain/, Februarie 2017. (Ultima accesare: 07.08.2021).

[28] Stefan Tai Jacob Eberhardt. ZoKrates. https://zokrates.github.io/introduction.html. (Ultima accesare: 13.08.2021).

[29] Hartwig Mayer. zk-SNARK explained: Basic Principles. https://blog.coinfabrik.com/wp-content/uploads/2017/03/zkSNARK-explained_basic_principles.pdf, Martie 2017. (Ultima accesare: 09.08.2021).

[30] Dan Prince. The 12 Best News Sites You Can Trust for Credible Stories. https://www.makeuseof.com/tag/trust-news-sites/, Mai 2021. (Ultima accesare: 14.08.2021).

[31] Thorsten Quandt, Lena Frischlich, Svenja Boberg, and Tim Schatto-Eckrodt. The International Encyclopedia of Journalism Studies. https://www.researchgate.net/publication/332749986_Fake_News, Mai 2019. (Ultima accesare: 05.07.2021).

[32] Christian Reitwiessner. zkSNARKs in a Nutshell. https://chriseth.github.io/notes/articles/zksnarks/zksnarks.pdf, Decembrie 2016. (Ultima accesare: 07.08.2021).

[33] Kevin Ickes Jimmy Jea Emil Kjelsrud Chris Jolley Andrew Ho Rui Gil, Callum Macdonald. ScraperJS. https://github.com/ruipgil/scraperjs#readme, Octombrie 2016. (Ultima accesare: 13.08.2021).

[34] Abdiel Aviles Troy McConaghy, Vanshdeep Singh. BigchainDB ReadMe. https://github.com/bigchaindb/BEPs/tree/master/13/#readme, Septembrie 2018. (Ultima accesare: 28.07.2021).

[35] Ujjwal Mehra Yves Longchamp, Saurabh Deshpande. Classication and importance of nodes in a blockchain network. https://www.seba.swiss/static/8c6d3dfcbf3ce596bf82b60871ecf43e/the-bridge_20200827.pdf, August 2020. (Ultima accesare: 14.07.2021).