Razvoj platforme za izmenjavo multimedijskih vsebin z uporabo zasebne kriptovalute

Uroš Zoretič, Grega Jakus

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, Ljubljana, Slovenija
E-pošta: uros.zoretic123@gmail.com, grega.jakus@fe.uni-lj.si

Abstract. *Blockchain is a key element for implementation of cryptocurrencies. It ensures the integrity, authenticity, irreversibility, privacy and timestamp of the recorded data without the need for a third-party trusted authority. Because of their potential, blockchains have found their ways also into the fields of decentralized autonomous organizations, supply chains and the Internet of Things. Due to the increasing popularity of blockchains, developers also benefit from ever more ready-available functionalities in form of web services and programming libraries, which enables developers, among other things, to use the existing cryptocurrencies in their applications, ensure time traceability of their data, or create their own blockchains and smart contracts.*

In order to evaluate the effectiveness of using libraries for cryptocurrency creation, we developed a content sharing application where the user gets an amount of digital money in exchange for providing digital content to the community. The cryptocurrency was created using the Naivecoin library and can be interacted through a web service. The platform can be used through web or mobile application. Based on our experience the use of blockchain libraries speeds up the development. However, in order to obtain reliable applications thorough testing is needed due to frequent bugs in libraries.

1 Uvod

Tehnologija veriženja blokov je bila razvita za izvedbo porazdeljene javne glavne knjige transakcij pri kriptovaluti Bitcoin. Glavna knjiga je ključni element kriptovalut, saj vsebuje seznam vseh izvedenih prenosov denarja s podatki o plačniku, prejemniku in znesku posameznega prenosa ter zaporedju izvršenih prenosov, iz katerega je mogoče sklepati o stanju na posameznem računu.

Blokovna veriga je sestavljena iz blokov, v katere zapisujemo podatke v obliki transakcije, ki ima praktično enako vlogo kot v tradicionalnih podatkovnih zbirkah, kjer služi spreminjanju zapisov. V transakcijo lahko zapišemo različne podatke, odvisno od namena uporabe verige. Pri kriptovalutah so to virtualna denarna sredstva, lahko pa transakcija vsebuje tudi izvedljivo programsko kodo v obliki pametnih pogodb (ang. smart contracts).

Poleg podatkov v transakcijah vsebuje blok tudi časovno oznako njegovega nastanka, javni ključ lastnika bloka, ki ga je potrdil, dokaz dela (ang. proof of work), ki služi kot orodje za doseganje soglasja o tem, kateri vozel v omrežju lahko v verigo doda naslednji blok, in izvleček (ang. hash), ki zagotavlja nespremenljivost podatkov v bloku. Ko so posamezni bloki ustvarjeni in potrjeni, se zapišejo v blokovno verigo, v kateri vsak blok vsebuje kazalec na izvleček prejšnjega bloka.

Pomembna lastnost blokovne verige je nespremenljivost v njej zapisanih podatkov, kar je pri kriptovalutah ključno za odpravo problema večkratne uporabe istega denarja. Ker lahko poleg tega nad blokovnimi verigami uporabimo tudi mehanizme, ki zagotavljajo verodostojnost, avtentičnost, neovrgljivost, časovno opredeljenost in tajnost podatkov, odpade potreba po zaupanja vrednem posredniku pri transakcijah, saj za navedene lastnosti skrbijo kar sodelujoči člani v omrežju. Bitcoin in druge kriptovalute so se zato lahko uveljavile kot alternativa klasični izmenjavi denarja s posredovanjem zaupanja vrednih avtoritet (bank).

Zaradi omenjenih lastnosti blokovnih verig, predvsem pa ker ne potrebuje zaupanja vrednega posrednika, se tehnologija širi tudi na druga področja, kot so kibernetska varnost [1], decentralizirani splet [2] in shranjevanje podatkov v porazdeljenem oblaku [3]. S pomočjo pametnih pogodb [4] je mogoče ustvariti tudi decentralizirane avtonomne organizacije [5] in upravljati oskrbovalne verige podjetij [6]. Pilotni projekti potekajo tudi v zdravstvu [7] in internetu stvari [8].

Naraščajoča priljubljenost in razširjenost tehnologije veriženja blokov pa ima še eno pomembno prednost, to je razširjenost že pripravljene funkcionalnosti, ki razvijalcem olajša in pohitri razvoj aplikacij, ki uporabljajo to tehnologijo. Namen prispevka je podati pregled razpoložljive funkcionalnosti in predlagati način uporabe enega izmed ogrodij pri izgradnji aplikacije, ki uporablja blokovno verigo za izvedbo zasebne valute.

2 Razpoložljiva ogrodja in storitve

Obstoječe funkcionalnosti zasledimo v obliki programskih knjižnic in spletnih storitev, ki omogočajo izgradnjo lastnih blokovnih verig ali pa integracijo obstoječih v končne aplikacije. Na ta način se razvijalcu ni potrebno ukvarjati z nizkonivojsko izvedbo blokovne verige, ampak pripravljeno ogrodje le prilagodi svojim potrebam.

2.1 Spletne storitve

Spletna storitev je storitev v skupni rabi, do katere odjemalci dostopajo z uporabo komunikacijskih protokolov, najpogosteje protokola HTTP (ang. HyperText Transfer Protocol). Storitev se izvaja ločeno od aplikacije, ki jo uporablja, pogosto na oddaljenem strežniku. Prednost uporabe storitve v primerjavi s programsko knjižnico je možnost sočasne interakcije z več odjemalci, ki storitev uporabljajo (vključno z njenimi podatki).

Uporaba kriptovalut v aplikacijah. Spletne storitve se najpogosteje uporabljajo za vključitev plačil s kriptovalutami v različne aplikacije. Med bolj uporabljenimi

tovrstnimi storitvami je Blockchain.info [9], ki omogoča plačila s kriptovaluto Bitcoin. Podobne storitve so še Coinbase [10], Gem [11] in BitGo [12].

Časovna sledljivost. Storitev Tierion [13] omogoča časovno sledljivost podatkov. Lahko se uporabi za shranjevanje zgodovine poslovnih procesov, časovno žigosanje in podpisovanje dokumentov ter zagotavljanje integritete podatkov, pridobljenih z naprav interneta stvari. Za zagotavljanje verodostojnosti podatkov uporablja protokol Chainpoint [14], ki je postal *de-facto* standard za časovno dokazljivost nastanka podatka.

Verižna omrežja v oblaku. Amazon Web Services (AWS) Blockchain [15] nudi oblačna ogrodja, ki temeljijo na priljubljenih verigah Ethereum ali Hyperledger, za izdelavo lastnih blokovno-verižnih omrežjih, ki se izvajajo znotraj okolja AWS. AWS Blockchain ponuja predloge, ki jih lahko razvijalec prilagodi in s tem ustvari lastne porazdeljene aplikacije.

2.2 Programske knjižnice

Programska knjižnica je deljena programska koda, ki se skupaj z izvorno kodo aplikacije, v kateri jo uporabimo, prevede v izvršno kodo. Razvijalec lahko do funkcionalnosti programske knjižnice dostopa s klici funkcij njenega aplikacijskega programskega vmesnika. Programska koda knjižnice se izvaja v istem izvajalnem okolju (računalniku) kot aplikacija, ki knjižnico uporablja.

Dostop do storitev. Programske knjižnice pogosto omogočajo le lažjo izvedbo klicev storitev, v okviru katerih je uporabljena funkcionalnost dejansko udejanjena. Veliko takih knjižic omogoča dostop do iste storitve v različnih programskih jezikih. Dostop do storitve Blockchain.info tako omogočajo knjižnice za python, PHP [16] in javascript [17]. Za interakcijo s storitvama Blockchyper in BitGo pa sta na voljo knjižnici za PHP [18, 19].

Zaradi njihove popularnosti se veliko knjižnic osredotoča na izmenjavo kriptovalut, še posebej Bitcoina. Za uporabo te kriptovalute sta tako na voljo knjižnici BitcoinJ [20] v javi in Bitcore-lib [21] v javascriptu.

Ustvarjanje pametnih pogodb. Naprednejše knjižnice omogočajo ustvarjanje pametnih pogodb, pri katerih namesto podatkov v blokovno verigo shranjujemo izvedljivo programsko kodo, ki olajša izvrševanja sporazuma med pogodbenimi strankami. Ukazi v programski kodi se izvedejo samodejno, ko so izpolnjeni določeni pogoji. S pametnimi pogodbami je tako mogoče ustvariti decentralizirane avtonomne organizacije ali pa upravljati oskrbovalno verigo podjetij.

Trenutno najbolj razširjena blokovna veriga na tem področju je Ethereum s programskim jezikom Solidity [22]. Programska knjižnica Web3j poenostavi pisanje pametnih pogodb za Ethereum [23] v javi in javascriptu. Pametne pogodbe je mogoče napisati tudi za blokovno verigo Hyperledger [24] v javi, pythonu in javascriptu. V primerjavi s knjižnicami za izvedbo transakcij v kriptovalutah je izbire pri ustvarjanju pametnih pogodb manj, saj zahtevajo več napora pri razvoju.

Zasledimo tudi bolj specializirane knjižnice, na primer takšne, ki izvajajo semantično in skladenjsko analizo pametnih pogodb. Primera sta pythonska knjižnica Mythril [25] in njena različica za javascript Solium [26].

Knjižnica Embark [27] po drugi strani služi kot orodje za razvoj porazdeljenih aplikacij. Temelji na verigi Ethereum in decentraliziranemu shranjevanju dokumentov IPFS (ang. InterPlanetary File System) [28].

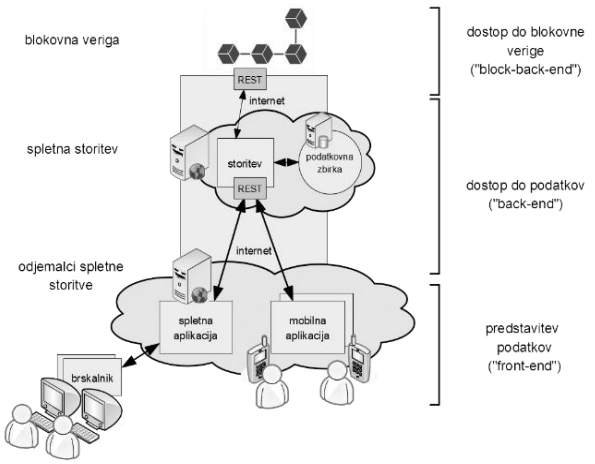
Knjižnice za izdelavo lastnih blokovnih verig. Večina knjižnic za izvedbo lastne blokovne verige je odprtokodnih in so na voljo v mnogih programskih jezikih. Mnogo tovrstnih programskih knjižnic je podanih v obliki spletnega tečaja. V programskem jeziku python sta tako na voljo "A simple Blockchain in Python" [29] in "Jbc" [30]. V javi je na voljo "NoobChain" [31], v javascriptu pa "BrewChain" [32] in "Naivecoin" [33]. Slednji služi tudi kot osnova za razvoj platforme z lastno kriptovaluto, predstavljene v naslednjem poglavju.

3 Platforma za izmenjavo vsebin z uporabo zasebne kriptovalute

Platforma MMShare je namenjena izmenjavi vsebin, pri čemer spodbuja uporabnike k sodelovanju. Naložene vsebine so zato dostopne samo tistim uporabnikom, ki so v skupnost že prispevali lastne vsebine. Uravnoteženost izmenjave dosežemo z uporabo zasebne kriptovalute, ki smo jo poimenovali MMCoin.

Ob registraciji dobi vsak uporabnik javni ključ, ki identificira njegov račun in se hrani v zalednem delu v podatkovni zbirki. Zasebni ključ, ki potrdi lastništvo prenesenih sredstev, se iz gesla vsakič sproti izračuna**.**

Uporabnik dobi ob registraciji pet tisoč kovancev. Nadaljnja sredstva pridobi z deljenjem lastnih vsebin in jih lahko porabi za dostop do vsebin drugih uporabnikov.



Slika 1. Struktura platforme MMShare

Vsak prenos sredstev se zabeleži v obliki transakcije v blokovni verigi, s čimer je zagotovljeno, da nihče ne

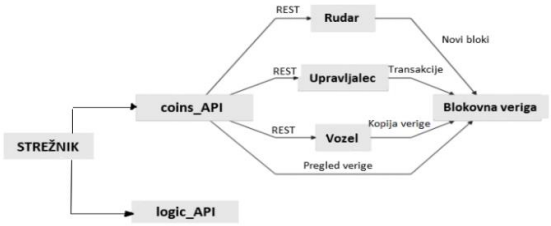
more porabiti istih sredstev večkrat ter tako dostopati do več vsebin, kot jih je sam dal na voljo ostalim. Digitalna sredstva so prenešena neposredno iz enega javnega naslova na drugega brez vmesnega posrednika.

Če uporabnik ne želi več uporabljati platforme, lahko svoj račun izbriše. Pri tem ostane zgodovina njegovih nakupov zapisana v blokovni verigi, se pa zaklenejo njegova digitalna sredstva, ki jih ne more porabiti nihče drug, saj nima dostopa do zasebnega ključa.

Slika 1 prikazuje strukturo platforme. Ta je sestavljena iz zalednega in čelnega dela. Zaledni del sestavljata blokovna veriga ter preostala aplikacijska logika, čelni del pa spletna in mobilna aplikacija.

3.1 Zaledni del

Zaledni del platforme skrbi za shranjevanje in zagotavljanje varnosti podatkov in dostop do blokovne verige. Za izvedbo slednje smo uporabili odprtokodno knjižnico Naivecoin, ki ponuja pripravljeno ogrodje blokovne verige. Ogrodje sestavljajo *Rudar*, ki skrbi za dodajanje novih blokov v verigo, *Upravljalec*, ki skrbi za javne ključe in bilance stanj na računih ter ustvarja nove transakcije za prenos sredstev med računi uporabnikov, in *Vozli*, ki predstavljajo sodelujoče člene omrežja. Struktura zalednega dela je prikazana na Sliki 2.



Slika 2. Struktura zalednega dela platforme MMShare

```
//ustvarimo novo transakcijo s podanimi parametri
let newTransaction =
  operator.createTransaction(walletId,
    fromAddress, toAddress, amount, changeAddress);
//preverimo veljavnost transakcije
newTransaction.check();
//transakcijo dodamo v blokovno verigo
blockchain.addTransaction
  (Transaction.fromJson(newTransaction));
```

Slika 3. Izvedba transakcije s knjižnico Naivecoin

Slika 3 prikazuje izvedbo transakcije z uporabo knjižnice Naivecoin. V prvem koraku transakcijo ustvarimo, pri čemer je potrebno določiti naslov denarnice prejemnika ter naslova pošiljatelja in prejemnika denarnih sredstev (uporabnik bi lahko imel v denarnici več naslovov ali pa bi imel več denarnic), količino nakazanih sredstev in naslov, kamor se prenesejo morebitni stroški transakcije. V drugem koraku preverimo veljavnost transakcije (pravilnost izvlečka, veljavnost podpisa itd.), na koncu pa transakcijo dodamo v blokovno verigo, v kateri je zapisana v formatu JSON (Slika 4).

Za ustvarjanje javnih in zasebnih ključev sodelujočih uporabnikov, namenjenih podpisovanju transakcij za prenos sredstev, uporablja Naivecoin asimetrično šifriranje z eliptičnimi krivuljami, imenovano ECDSA (Elliptic Curve Digital Signature Alghoritm).

Slika 4. Transakcija v blokovni verigi v formatu JSON

Blokovno verigo iz knjižnice Naivecoin smo preoblikovali v spletno storitev REST in jo od ostalega zaledja ločili s komunikacijskim vmesnikom. Če bi bilo to potrebno (na primer za čim večjo izkoriščenost strojne opreme za potrebe rudarjenja), bi se tako storitev, ki skrbi za blokovno verigo, lahko izvajala na lastnem fizičnem strežniku, dostop do verige pa se ne bi spremenil. Knjižnico smo prilagodili tudi tako, da lahko nove zapise v verigo dodaja (rudari) samo strežniški vozel. Vsak rudar je namreč nagrajen z nekaj virtualnimi sredstvi, če je on tisti, ki uspe dodati naslednji blok v verigo. To pa ni v skladu z namenom platforme, saj želimo, da lahko ostali uporabniki sredstva pridobijo le, če nalagajo vsebine.

Tabela 1. REST vmesnik blokovne verige

HTTP metoda	**URI**	**parametri**	**učinek**
POST	/wallets	geslo	dodajanje novega uporabnika
POST	/wallets/ *{walletID}* /transactions	geslo in naslov pošiljatelja, naslov prejemnika, količina prenesenih sredstev	izvedba transakcije za prenos sredstev med dvema računoma
GET	*{addressID}*/ balance	/	pridobivanje stanja na računu
POST	wallets/ *{walletID}*	staro in novo geslo	menjava gesla

Tabela 1 prikazuje dejanja, ki jih je mogoče preko vmesnika REST izvesti v blokovni verigi. Spletna storitev je dostopna na glavnem naslovu */coinsAPI*. Za prenos sredstev z enega uporabnikovega računa na drugega tako na naslov */coinsAPI/wallets/{walletID}/transactions* pošljemo HTTP zahtevek po metodi POST, v telo zahtevka pa vključimo geslo in naslov pošiljatelja, naslov prejemnika in količino prenesenih sredstev. Geslo pošiljatelja je pomembno, ker se iz njega izračuna pošiljateljev zasebni ključ, s katerim se transakcija digitalno podpiše. S tem se zagotovi pristnost transakcije, saj lahko digitalna sredstva porabi le njihov lastnik. Uspešno izvedena transakcija se nato zapiše v blok, ki se doda v blokovno verigo (Sliki 3 in 4).

Preostala aplikacijska logika je izvedena v okviru spletne storitve MMShare. Ta komunicira s storitvijo blokovne verige, obenem pa nudi dostop do funkcionalnosti platforme čelnemu delu s pomočjo lastnega vmesnika

REST, podobnega tistemu za interakcijo z blokovno verigo. Odjemalci lahko z zahtevki na naslov */logic_API* s predpisano HTTP metodo in dodatkom k URI naslovu shranjujejo, prenašajo, brišejo in kupujejo vsebine.

Zaledni del je napisan v javascriptu, izvaja pa se v izvajalnem okolju Node.js s pomočjo ogrodja Express.js. Za shranjevanje vsebin in ostalih podatkov smo uporabili entitetno-relacijsko podatkovno zbirko MongoDB, saj ta v primerjavi s klasičnimi relacijskimi zbirkami nudi večjo fleksibilnost strukture in povezav med podatki.

3.2 Čelni del

Za uporabo platforme MMShare lahko uporabnik izbira med spletno in mobilno aplikacijo. Funkcionalnosti obeh aplikacij vključujejo:

- registracijo in prijavo uporabnika,
- nalaganje novih vsebin in brisanje obstoječih,
- nakup vsebine in njen prenos na svojo napravo,
- filtriranje vsebin po temah ali lastniku.

Izsek iz uporabniškega vmesnika spletne aplikacije prikazuje Slika 5, na kateri so razvidne tudi pravice uporabnika nad posamezno datoteko.

Slika 5. Izsek iz uporabniškega vmesnika spletne aplikacije

Spletna aplikacija je razvita po principu aplikacije na eni strani (ang. Single Page Application, SPA) s programsko knjižnico ReactJS. Mobilna aplikacija uporablja v telefone vgrajen fotoaparat in tako omogoča hitro ustvarjanje in nalaganje datotek na strežnik.

4 Zaključek

Čeprav je tehnologija veriženja blokov razmeroma nova, v zadnjih letih pridobiva na popularnosti in razširjenosti. Njen razmah je spodbudilo zavedanje, da zaupanje v posrednika s seboj prinese tudi določena tveganja, če je ta kompromitiran. Veriženje blokov se poskuša uveljaviti na različnih področjih, vendar zaradi relativne novosti še ne moremo z gotovostjo trditi, na katerih se bo dokončno uveljavila. Na trgu se mesečno pojavi veliko novih aplikacij, ki ponujajo rešitve z uporabo blokovne verige. Novim aplikacijam je mogoče slediti in jih podpreti na platformi za množično financiranje ICO Drops [34].

Za razvoj rešitev, predvsem pri manjših projektih, je smiselno uporabiti odprtokodne knjižnice, saj pohitrijo razvoj aplikacij in nudijo dodaten sloj abstrakcije tako, da se razvijalcem ni potrebno ukvarjati z delovanjem tehnologije veriženja blokov. Pri razvoju platforme za deljenje datotek MMShare smo tako enostavno uporabili knjižnico Naivecoin, jo prilagodili lastnim potrebam in se v nadaljevanju lahko osredotočili na funkcionalnosti same aplikacije.

Po drugi strani so programske knjižnice pogosto podvržene programskim hroščem in je zato potrebno veliko testiranja in prilagajanja programske kode. Pri uporabi odprtokodnih knjižnic v industriji je zato potrebno biti posebej pazljiv, saj mora biti njihovo delovanje brezhibno. Razvijalci zahtevnejših projektov se tako pogosto odločajo za razvoj lastnih komponent ali pa njihov nakup, da pridobijo vso potrebno podporo.

Literatura

[1] Namecoin, https://namecoin.org/
[2] Blockname, https://github.com/telehash/blockname
[3] Storj.io, https://storj.io/
[4] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, "Step by step towards creating a safe smart, contract: Lessons and insights from a cryptocurrency lab," v International Conference on Financial Cryptography and Data Security, 79-94, Springer, 2016
[5] Decentralized Autonomous Organisation – Organisations on the Blockchain, https://blog.codecentric.de/en/2017/09/decentralized-autonomous-organization-blockchain/
[6] IBM Pushes Blockchain into the Supply Chain, https://www.wsj.com/articles/ibm-pushes-blockchain-into-the-supply-chain-1468528824
[7] MedRec, https://medrec.media.mit.edu/
[8] IBM Reveals Proof of Concept for Blockhain-Powered Internet of Things, https://www.coindesk.com/ibm-reveals-proof-concept-blockchain-powered-internet-things/
[9] Blockchain.info, https://www.blockchain.com/en/explorer
[10] Coinbase, https://www.coinbase.com/
[11] Gem, https://gem.co/
[12] BitGO, https://www.bitgo.com/info/
[13] Tierion, https://www.bitgo.com/info/
[14] Chainpoint, https://chainpoint.org/
[15] AWS Blockchain, https://aws.amazon.com/about-aws/whats-new/2018/04/introducing-aws-blockchain-templates/
[16] Blockchain.info, https://github.com/blockchain/
[17] blockchain.info, https://libraries.io/npm/blockchain.info
[18] Blockcypher, https://github.com/blockcypher/php-client
[19] BitGOSDK-PHP, https://github.com/neto737/BitGoSDK-PHP
[20] The Top 3 Blockchain Libraries for Java Devs, https://dzone.com/articles/the-top-3-blockchain-libraries-for-java-devs
[21] Bitcore-lib, https://libraries.io/npm/bitcore-lib
[22] Solidity, http://solidity.readthedocs.io/en/v0.4.24/
[23] Web3j, https://github.com/ethereum/web3.js/
[24] HyperLedger Fabric, https://hyperledger-fabric.readthedocs.io/en/release-1.2/
[25] Mythril, https://libraries.io/pypi/mythril
[26] Solium, https://libraries.io/npm/solium
[27] Embark, https://libraries.io/npm/embark
[28] IPFS, https://ipfs.io/
[29] Blockchain, https://github.com/dvf/blockchain
[30] Jbc, https://github.com/jackschultz/jbc
[31] Noochain, https://github.com/CryptoKass/NoobChain-Tutorial-Part-2
[32] BrewChain, https://github.com/dbjsdev/BrewChain
[33] Naivecoin, https://github.com/conradoqg/naivecoin
[34] ICO Drops, https://icodrops.com/