Prototipni sistem samovladne identitete s tehnologijo Ethereum

Rihard Marušič, Matevž Pustišek

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana
E-pošta: rm3612@student.uni-lj.si, matevz.pustisek@fe.uni-lj.si

A Self-Sovereign Identity Prototype System with Ethereum

Abstract. *There are many different ways and approaches to identity management in the digital world. This article focuses on a relatively new concept of identity management, called self-sovereign identity, which aims to give individuals more control over their identity and reduce dependency on centralised entities.*

The article includes a theoretical part, where the self-sovereign identity is presented in more detail, with descriptions of the basic building blocks and functionalities. The second part presents the creation of a simple prototype implementation of a self-sovereign identity, which contains only the basic functionalities and aims to present in some more detail the technological background of an example of such system.

1 Uvod

Članek se predvsem posveča še dokaj novemu konceptu samovladne identitete (ang. self-sovereign identity, SSI), katere cilj in potencial, če se tehnologija uveljavi, postane bolj popularna in dozori, je predstavljati nekakšen nov identifikacijski sloj interneta.

Trenutni uveljavljeni sistemi upravljanja z identiteto, ki so povečini centralizirani ali delegirani, niso prenosljivi in interoperabilni, nadzor nad podatki pa je v rokah ponudnikov storitev in identitete. Sistemi samovladne identitete pa stremijo k temu, da bi uporabniku dali čim več nadzora nad lastnimi podatki, ter komu, kako in kdaj jih bo predstavljal. Ponujali bi več zasebnosti in varnejše hranjenje identitete. Stremi tudi k decentralizaciji, ki postaja dandanes z večanjem popularnosti tehnologije veriženja blokov (ang. blockchain) vedno večji predmet pogovora, ko se tudi zdi, da ljudje dajo vedno več pozornosti ter skrbi na zasebnost in varnost njihovih podatkov na spletu.

Cilj članka je bralcu predstaviti koncept samovladne identitete ter ga na kratko seznaniti s tehnološkim ozadjem in različnimi tehnologijami, ki stojijo za implementacijo takih vrst sistemov identitet. Cilj je bil tudi izdelati prototipno, demonstracijsko rešitev samovladne identitete. Rešitev je zato tudi bolj poenostavljena, nekateri elementi so implementirani nekoliko drugače, kot bi bili sicer, saj rešitev ni namenjena produkcijski uporabi, temveč služi za zgolj še globlje in lažje razumevanje teh tehnologij.

2 Samovladna identiteta

Pri samovladni identiteti je uporabnik sam upravljalec nad lastno identiteto in ima nad njo veliko več nadzora kot pri drugih modelih upravljanja identitete. Uporabnik se lahko odloči do kdaj, komu in katere informacije bo delil, prav tako samovladna identiteta stremi k čim večji decentralizaciji ter čim večji moči in avtonomiji na strani uporabnika. Koncept samovladne identitete je v določenih pogledih zelo podoben današnjemu fizičnemu (nedigitalnemu) svetu upravljanja identitete, kjer ljudje povečini sami pri sebi v denarnicah hranimo na primer osebno izkaznico ali vozniško dovoljenje. To potem uporabimo za izkazovanje identitete v banki, če banka seveda zaupa izdajatelju dokumenta.[1]

Samovladna identiteta temelji na osnovnih gradnikih med katere spadajo decentralizirani identifikatorji (DID-ji), DID dokumenti, preverljiva potrdila oz. poverilnice ter digitalne denarnice. Za nekatere gradnike že obstajajo raznorazni standardi, tu je mogoče smotrno izpostaviti standarde w3c-ja (ang. World Wide Web Consortium)[2][3].

Za večino interakcij, kjer je potrebno izkazovati določene trditve, so pomembne tri entitete - imetnik, izdajatelj, preveritelj (ang. holder, issuer, verifier). Imetnik je entiteta, na katero se trditve nanašajo, izdajatelj je entiteta, ki potrjuje, da so trditve resnične. Preveritelj pa je entiteta, ki za raznorazne namene zahteva trditve, katerih imetnik mora dokazati legitimnost in verodostojnost, izdajatelju pa zaupa, da držijo.

2.1 Decentralizirani identifikatorji

Sistem temelji na asimetrični kriptografiji. DID je unikatni identifikator, ki predstavlja uporabnika, nadzor nad njim pa lahko uporabnik kriptografsko dokaže. DID je URI, ki vodi do DID dokumenta. Pridobivanju DID dokumenta zgolj z vedenjem zgolj DID-ja pravimo razreševanje (ang. resolving). v DID dokumentu so podatki, ki opišejo lastnika DID-ja s podrobnejšimi informacijami. Med najpomembnejše spadajo avtentikacijske metode s katerimi bo dokazal lastništvo nad DID-jem, končna točka storitve (ang. service endpoint - na primer omrežni naslov ali ostali načini za vzpostavitev komunikacije z

lastnikom DID-ja) in javni ključi s pomočjo katerih bo uporabnik kriptografsko dokazal lastništvo. DID-je lahko delimo na javne in zasebne. DID dokumenti javnih DID-jev, ki morajo biti razrešljivi so ponavadi objavljeni na javnem mestu - pogosto se za to uporabi veriga blokov. Uporabnik nima samo enega DID-ja, ponavadi ima za vsako novo razmerje drugi DID.[4]

2.2 Preverljiva potrdila

Preverljivo potrdilo je dokument, ki ga je mogoče kriptografsko preveriti in katerega izdajatelj je razviden. Uporabnik potrdila shranjuje sam pri sebi v svoji denarnici in jih uporablja za predstavljanje raznoraznim entitetam. Ker potrdilo hrani izključno uporabnik sam, ima tako več nadzora nad lastnimi podatki, komu jih bo predal in kdaj. Z implementacijo potrdil na način, da podpirajo dokaz brez znanja (ang. zero-knowledge proof), se lahko doseže še večjo zasebnost uporabnika ter prepreči korelacijo.[5][3]

2.3 Digitalne denarnice

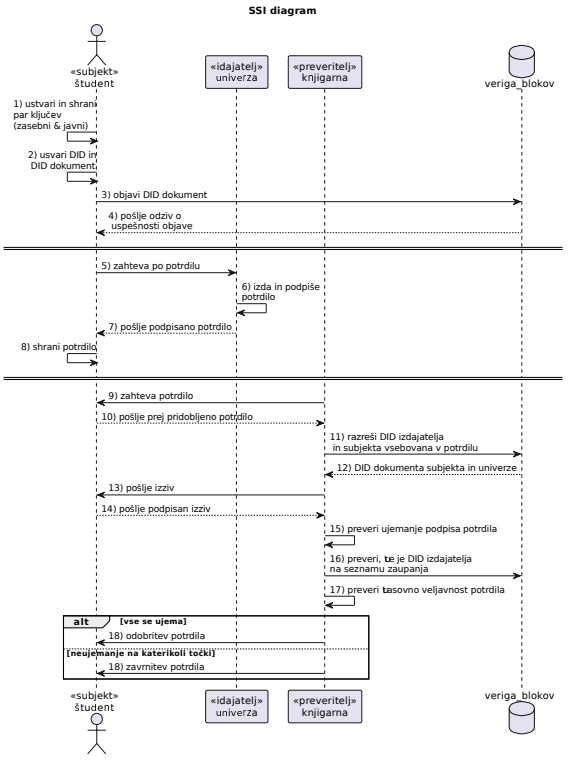
Digitalne denarnice so prostor, kjer se shranjujejo vsi podatki, potrebni za operiranje v ekosistemu samovladne identitete. Obstaja mnogo različnih vrst denarnic, od spletnih aplikacij, do mobilnih aplikacij, računalniških programov ali pa celo strojnih denarnic. Razlikujejo se tudi po namembnosti. Denarnica namenjena podjetju ali pa na primer napravi bo morala imeti dodatne funkcionalnosti, kot denarnica, ki je namenjena zgolj fizični osebi. Prav tako poznamo denarnice s skrbništvom ali brez skrbništva. Varnost denarnice je ključnega pomena, potrebna je dobra enkripcija podatkov, prav tako je pomembna enostavna kreacija varnostnih kopij, saj izguba nad dostopom do denarnice pomeni tudi izgubo identitete.[1]

3 Predstavitev rešitve

Cilj je bil izdelati poenostavljeno, prototipno različico sistema samovladne identitete, ki bo omogočal podrobnejše razumevanje, kaj samovladna identiteta sploh je in kako deluje ter kaj se dogaja v ozadju, ni pa rešitev namenjena produkcijski uporabi. Rešitev ima še vedno vse jedrne sestavne dele in omogoča vse osnovne funkcionalnosti, ki so potrebni za pomembne korake interakcije med entitetami, ki sodelujejo v procesu primera dokazovanja nečesa s samovladno identiteto. V diagramu zaporedja sporočil na sliki 1 je opisan eden izmed najbolj pogosto predstavljenih primerov interakcij med entitetami, in sicer tak, kjer se uporablja tudi preverljiva potrdila in je izdajatelj tretja oseba oz. entiteta.

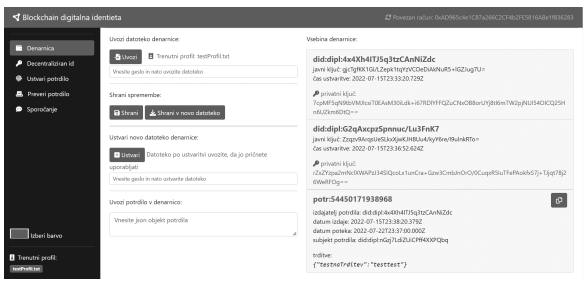
Prikazan je scenarij, kjer se uporabnik predstavlja s trditvami tretje osebe, za primer imamo študenta, ki študira na neki univerzi. Študent hoče v knjigarni kupiti knjigo, knjigarna pa za študente ponuja popust, ki ga študent hoče izkoristiti. V tem primeru bo moral študent knjigarni dokazati, da je resnično študent, knjigarna pa to preveriti, kar bo storil s pomočjo predstavitve potrdila, ki bo vsebovalo trditev tretje osebe – v tem primeru univerze. Študent predstavlja uporabnika, univerza predstavlja izdajatelja, knjigarna pa preverjevalca.

Namen diagrama je, da prikaže, kaj vse naj rešitev omogoča in s tem tudi funkcionalnosti rešitve.



Slika 1: UML diagram zaporedja sporočil.

Na sliki 2 je viden del končne rešitve. Razdeljena je na pet zavihkov, ki predstavljajo pet glavnih funkcionalnosti potrebnih za potek, prikazan v diagramu na sliki 1. Te so interakcija z datoteko denarnice (uvoz datoteke, uvoz podatkov datoteko, shranjevanje), ustvarjanje DID-ja ter DID dokumenta in zapis na verigo blokov, ustvarjanje potrdila, preverjanje potrdila (in razreševanje DID dokumenta z verige blokov s pomočjo DID-ja) ter sporočanje med uporabniki.



Slika 2: Primer datoteke denarnice v končni rešitvi.

4 Arhitektura rešitve

Rešitev je v obliki spletne aplikacije, zato bo uporabnik na brskalnik za popolno delovanje moral namestiti še vtičnik Metamask. Metamask je pri decentraliziranih spletnih aplikacijah, ki uporabljajo verigo blokov, postal

nekakšen standarden spletni vtičnik in večina spletnih decentraliziranih aplikacij ga uporablja za povezavo z Ethereum vozliščem. Ker je rešitev spletna aplikacija, pride do problema shranjevanja oziroma deljenja podatkov. Podatki, kot so javni ter zasebni ključi, DID-ji, DID dokumenti, potrdila, so zelo pomembni in praktično predstavljajo identiteto, zato je bolje, da jih strežnik sploh ne vidi.

Na tak način spletna aplikacija sicer ne deluje kot navadna spletna denarnica, ki ponavadi hrani te podatke (sicer zelo dobro enkriptirane), ampak bolj kot namenska računalniška aplikacija za končnega uporabnika. Tako bo v tem primeru strežnik uporabniku serviral aplikacijo, ki vsebuje kodo, potrebno za vse delovanje, hkrati pa bo podatke identitete videl samo uporabnik. Ker sem hotel, da bi aplikacija delovala tudi med različnimi entitetami na različnih računalnikih v različnih omrežjih, bo strežnik v rešitvi odgovoren tudi za pošiljanje sporočil iz ene instance aplikacije do druge. To bo sicer pomenilo, da bo strežnik videl poslane podatke, a je tak način pošiljanja sporočil zgolj namenjen demonstracijskim namenom in se v produkcijskem okolju ne bi uporabljal. Kljub temu pa strežnik teh podatkov v rešitvi ne shranjuje. Podatke o uporabnikov identiteti shranjujemo v preprosti *.txt* datoteki, v kateri bodo shranjeni vsi podatki, enkriptirana pa bo z geslom s pomočjo AES enkripcije. Na tak način bo uporabnik datoteko lahko preprosto kopiral in ustvaril varnostne kopije, v primeru, da do datoteke pride nekdo drug, pa je ta brez gesla ne bo mogel uporabljati. Uporabnik bo tako za uporabo rešitve najprej v spletno aplikacijo uvozil datoteko in jo dekriptiral z vnosom gesla. Vsi podatki datoteke bodo ostali zgolj na uporabnikovi strani in ne bodo šli na strežnik, s podatki bo aplikacija obratovala sama v brskalniku na strani uporabnika.

Pomembnejši deli razvoja rešitve so spodaj nekoliko bolj natančno opisani, večina ostalih delov rešitve pa sploh ni omenjenih, saj bi sicer razlaga bila preveč obširna za namen tega članka.

4.1 Pametna pogodba

Pametna pogodba je program, ki živi na Ethereum verigi blokov. Je skupek funkcij in podatkov in je vrsta Ethereum računa, ki ima lahko določeno število etra (kriptovaluta omrežja Ethereum), lahko tudi kliče ostale pametne pogodbe. Zažene oziroma kliče se jo s transakcijo na naslov pametne pogodbe z dodanimi parametri (če so ti potrebni). Vse interakcije s pametno pogodbo so ireverzibilne in pametne podobe z verige blokov ni mogoče izbrisati, ko je enkrat tja objavljena.

Ker rešitev ni namenjena produkcijski rabi, se bo DID in DID-dokumente shranjevalo kar na verigo blokov. Sicer konceptualno s tem ni nič narobe, a bi se v večini produkcijskih rešitev zgrajenih na omrežju, kot je na primer Ethereum, to hitro izkazalo kot problematično. Sami DID dokumenti bi v produkcijski rešitvi bili veliko večji, DID dokumentov bi bilo ogromno in posledično bi se shranjevanje tako velikega števila podatkov na verigo blokov izkazalo kot precej drago, saj veriga blokov ponavadi ni namenjena shranjevanju ogromne količine podatkov (z izjemo specializiranih verig blokov).

Pri rešitvi bo pametna pogodba odgovorna za zapisovanje DID-jev in DID dokumentov na verigo blokov, prav tako bo pametna pogodba odgovorna za razreševanje DID-jev do DID dokumentov. Kljub temu, da bo končna pametna pogodba dokaj kratka in preprosta, ta še vedno igra ključno vlogo v celotnem procesu samovladne identitete v tej implementaciji.

Pametna pogodba vsebuje dve glavni funkciji, eno za dodajanje novega DID-ja ter DID dokumenta na verigo blokov, drugo pa za branje informacij, oziroma razreševanje DID-ja. DID-ji in DID dokumenti so shranjeni v spremenljivki tipa *mapping*, ki podatke shranjuje po principu ključ-vrednost. DID-ji so v tem primeru ključ, DID dokumenti pa vrednosti. Uspešnost zapisovanja nam pametna pogodba sporoči preko spremenljivke tipa *event*, ki je shranjen v dnevniku transakcije.

Velja še omeniti, da je pametna pogodoba objavljena na testnem omrežju Goerli in ne na glavnem Ethereum omrežju. Funkcionalnost rešitve in objava pametne pogodbe bi sicer bili popolnoma enaki, edina razlika bi bila, da bi na pravem Ethereum omrežju za vsak zapis bilo potrebno plačati z etri, kar pa za prototipno rešitev predstavlja zgolj nepotreben strošek.

4.2 Datoteka denarnice

Ker bo datoteka denarnice preprosta *.txt* datoteka in ker so ponavadi DID dokumenti ter potrdila predstavljeni v JSON (ang. Javascript Object Notation) formatu, sem se odločil, da bo tudi sama datoteka denarnice v JSON formatu.

```
{
   "privatniKljuci": [ ],
   "DIDdokumenti": [ ],
   "potrdila": [ ]
}
```

Slika 3: Dekriptirana, prazna datoteka denarnice.

Kot videno na sliki 3, je datoteka denarnice zgolj en JSON objekt s tremi vrednostmi, ki so seznami. Prva vrednost je seznam vseh zasebnih ključev, druga seznam vseh DID dokumentov, tretja pa seznam vseh potrdil. Ker je potrebno vedeti, kateri zasebni ključ pripada kateremu DID-ju in posledično kateremu DID dokumentu, se oboje v sezname shranjuje na tak način, da imata zasebni ključ ter pripadajoč DID dokument v seznamih enak indeks.

4.3 DID dokument

DID dokument je prav tako predstavljen z JSON formatom, pirmer pa je viden na sliki 4

```
{
  context: "did-dokument",
  id: "did:dipl:anN7+fj0hzJvq6MX5t7ItY",
  datumUstvaritve: "2022-06-09T23:47:32.735Z",
  javniKljuc: "UaS6t5gcDdQG/Y..."
}
```

Slika 4: Primer DID dokumenta.

Znotraj objekta so štiri vrednosti. Prva vrednost *context* opisuje kontekst DID dokumenta. Tu bi ponavadi bila povezava do opisa strukture DID dokumenta ter formatov vrednosti v njem in raznorazni opisi standardov, ki bi se potrebovali zato, da lahko naprava razume strukturo DID dokumenta ter ga sprocesira. To je potrebno, ker so med DID dokumenti različnih rešitev vedno razlike. Ker ta rešitev ni namenjena za delovanje v produkcijskem okolju in bo moja aplikacija edina, ki bo delovala s takim formatom DID dokumenta, sem v kontekst preprosto samo napisal niz *did-dokument*, ki označuje, da je to DID dokument. V drugem polju pod ključem *id* imamo DID, ki pripada DID dokumentu in kateri vodi do tega DID dokumenta. DID je sestavljen iz niza *did:dipl:* in nato unikatnega niza DID-ja. Tretje polje vsebuje datum ustvaritve DID dokumenta, ki je niz v iso8601 formatu. Datum je zgolj informativne narave v tem primeru in ga nebi bilo potrebno vključiti za pravilno delovanje DID dokumentu. Na zadnjem mestu je še najpomembnejši podatek, in sicer javni ključ, zapisan kot niz v base64 formatu. Ta javni ključ pripada DID-ju, do katerega vodi ta dokument, zasebni ključ, pripadajoč temu javnemu ključu, pa je shranjen v zgoraj omenjeni tabeli *privatniKljuci* znotraj datoteke denarnice.

4.4 Potrdilo

Potrdilo ima tri vrednosti, prva je *context*, ki služi istemu namenu kot kontekst v DID dokumentu, le da je tu vrednost preprosto "potrdilo". Pod vrednostjo vsebina pa imamo nov JSON objekt in ta nov objekt vsebuje šest vrednosti. Vsebina je posamezen JSON objekt, da se lahko za podpisovanje preprosto podpiše zgolj objekt in nato podpis uvrsti pod tretjo vrednost podpis. V objektu vsebina je na prvem mestu vrednost *id*, ki predstavlja unikaten id potrdila. Informacija je zgolj informativne narave, sestavljena je iz izraza *potr:*, ki je preprosto okrajšava za potrdilo, ter unikatnega niza številk, ki predstavlja potrdilo. Vrednost *izdajateljPotrdila* vsebuje DID entitete, ki je potrdilo izdala, *subjektPotrdila* pa DID entitete, na katero se potrdilo nanaša. *datumIzdaje* je ravno tako kot datum v DID dokumentu predstavljen kot niz v iso8601 formatu in predstavlja natančen čas ter datum, kdaj je bilo potrdilo izdano. V tem primeru datum ni zgolj informativen, temveč služi preprostemu preverjanju časovne veljavnosti potrdila. Če je trenutni datum preverjanja potrdila manjši od datuma izdaje, potem je potrdilo neveljavno. Podobno je z vrednostjo *datumPoteka*, ki predstavlja datum poteka potrdila, s katerim lahko preveritelj enostavno preveri, če je trenutni datum preverjanja manjši od datuma poteka, in če je to res, potem je potrdilo še veljavno. Ravno tako kot ostali datumi je to predstavljeno z nizom v iso8601 formatu. Zadnja vrednost je še *trditve*, ki ravno tako vsebuje JSON objekt. V tem JSON objektu lahko izdajatelj potrdila med ustvaritvijo naniza več vrednosti, odvisno, za kaj se bo potrdilo uporabljalo in kaj hoče o subjektu potrdila trditi. V produkcijskem okolju bi trditve bile implementirane drugače, a je tak preprostejši način zadosten za prikaz delovanja, prav tako je priročnejši, saj lahko izdajatelj zelo enostavno trdi karkoli o subjektu potrdila oziroma naniza več trditev.

```
{
  "context": "potrdilo",
  "vsebina": {
    "id": "potr:1234456",
    "izdajateljPotrdila": "did:primer:km3561f...",
    "datumIzdaje": "2018-03-04T00:12:04Z",
    "datumPoteka": "2024-01-01T00:06:70Z",
    "subjektPotrdila": "did:primer:ebfeb1f...",
    "trditve": {"studiraNa": "Univerza v Ljubljani"}
  }
  "podpis": "3ghj43g5jk42gj45g4jh5g3..."
}
```

Slika 5: Primer potrdila oz. poverilnice.

5 Zaključek

Implementacija načrtovane prototipne rešitve je namenjena lažjemu razumevanju osnovnega tehnološkega ozadja sistemov samovladne identitete. S tega stališča je namen uspel, saj rešitev omogoča predvidene osnovne funkcije, ki so pomembne za razumevanje takih sistemov. Še vedno pa ostaja veliko možnih izboljšav. Glede na to, kako je v rešitvi zasnovana denarnica (obratovanje preko *.txt* datoteke), bi najprimernejša in najbolj očitna izboljšava bila, da bi rešitev bila implementirana v obliki računalniškega programa ali mobilne aplikacije. Na tak način bi rešitev še bolj ustrezala splošnemu principu, da naj ima uporabnik čim več nadzora in da naj bo čim manj odvisnosti od centraliziranih tehnologij. Bi pa v tem primeru komuniciranje z Ethereum vozlišči bilo potrebno izvesti na drugačen način, saj to preko Metamask vtičnika, ki je namenjen brskalnikom, ne bi bilo več mogoče. Zelo primerna zboljšava bi bila tudi sprememba sistema pošiljanja sporočil. Trenutno sporočanje poteka s pomočjo Websocket tehnologije, kjer je seveda strežnik vmesni člen. Tak način je za princip samovladne identitete neprimeren. Potrebno bi bilo implementirati "peer-to-peer" sistem, ki se ponavadi uporablja pri takih sistemih. Možnih izboljšav je seveda še več, a kljub temu menimo, da je zastavljeni cilj dosežen.

Literatura

[1] A. Preukschat in D. Reed, *Self-sovereign identity*. Manning Publications Co., 2021.

[2] *Decentralized Identifiers (DIDs) v1.0*. spletni naslov: https://www.w3.org/TR/did-core/ (pridobljeno 5. 7. 2022).

[3] *Verifiable Credentials Data Model v1.1*. spletni naslov: https://www.w3.org/TR/vc-data-model/ (pridobljeno 7. 7. 2022).

[4] A. Tobin, "Sovrin: What Goes on the Ledger?", en, 2018.

[5] *EBSI Verifiable Credentials Playbook - EBSI Documentation* -. spletni naslov: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/EBSI+Verifiable+Credentials+Playbook (pridobljeno 24. 6. 2022).

[6] R. Marušič, diplomska naloga, Univerza v Ljubljani, Ljubljana, 2022.