

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Žan Pižmoht

**Sistem za zagotavljanje zaupanja v
dobavni verigi zdravil z uporabo
verige blokov**

MAGISTRSKO DELO
MAGISTRSKI ŠTUDIJSKI PROGRAM DRUGE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: prof. dr. Vlado Stankovski

SOMENTOR: doc. dr. Petar Kochovski

Ljubljana, 2025

To delo je ponujeno pod licenco *Creative Commons Priznanje avtorstva-Deljenje pod enakimi pogoji 2.5 Slovenija* (ali novejšo različico). To pomeni, da se tako besedilo, slike, grafi in druge sestavine dela kot tudi rezultati zaključnega dela lahko prosto distribuirajo, reproducirajo, uporabljajo, priobčujejo javnosti in predelujejo, pod pogojem, da se jasno in vidno navede avtorja in naslov tega dela in da se v primeru spremembe, preoblikovanja ali uporabe tega dela v svojem delu, lahko distribuira predelava le pod licenco, ki je enaka tej. Podrobnosti licence so dostopne na spletni strani creativecommons.si ali na Inštitutu za intelektualno lastnino, Streliška 1, 1000 Ljubljana.



Izvorna koda zaključnega dela, njeni rezultati in v ta namen razvita programska oprema je ponujena pod licenco GNU General Public License, različica 3 (ali novejša). To pomeni, da se lahko prosto distribuira in/ali predeluje pod njenimi pogoji. Podrobnosti licence so dostopne na spletni strani <http://www.gnu.org/licenses/>.

ZAHVALA

Na tem mestu zapišite, komu se zahvaljujete za izdelavo magistrske naloge. V zahvali se poleg mentorja spodobi omeniti vse, ki so s svojo pomočjo prispevali k nastanku vašega izdelka.

Žan Pižmoht, 2025

Vsem rožicam tega sveta.

*"The only reason for time is so that
everything doesn't happen at once."*

— Albert Einstein

Kazalo

Povzetek

Abstract

1	Uvod	1
2	Pregled literature in ozadje	3
2.1	Zaupanje	3
2.2	Upravljanje zaupanja v razpršenih sistemih	6
2.3	Farmacevtske dobavne verige	9
2.4	Tehnologija veriženja blokov v dobavnih verigah	12
2.5	Semantični splet in ontološke metode	16
2.6	Analiza vrzeli in prispevek te naloge	18
3	Implementacija sistema	21
3.1	Uporabljene tehnologije	21
3.2	Arhitektura sistema	24
3.3	Tok podatkov v sistemu	34
4	Evalvacija	47
4.1	Predstavitev scenarijev delovanja	47
4.2	Varnostna analiza	57
4.3	Zmogljivost in razširljivost	57
4.4	Prednosti in omejitve	57
4.5	Primerjava z obstoječimi rešitvami	57

KAZALO

5	Zaključek in nadaljnje delo	59
5.1	Povzetek prispevkov	59
5.2	Nadaljnje raziskave	59
A	OWL ontologija dobavne verige	61
B	Izvillečki kode pametnih pogodb	77

Seznam uporabljenih kratic

kratica	angleško	slovensko
CA	classification accuracy	klasifikacijska točnost
DBMS	database management system	sistem za upravljanje podatkovnih baz
SVM	support vector machine	metoda podpornih vektorjev
...

Povzetek

Naslov: Sistem za zagotavljanje zaupanja v dobavni verigi zdravil z uporabo verige blokov

V vzorcu je predstavljen postopek priprave magistrskega dela z uporabo okolja L^AT_EX. Vaš povzetek mora sicer vsebovati približno 100 besed, ta tukaj je odločno prekratek. Dober povzetek vključuje: (1) kratek opis obravnavanega problema, (2) kratek opis vašega pristopa za reševanje tega problema in (3) (najbolj uspešen) rezultat ali prispevek magistrske naloge.

Ključne besede

zaupanje, tehnologija veriženja blokov, dobavna veriga zdravil

Abstract

Title: Trust System in Pharmaceutical Supply Chain Using Blockchain

This sample document presents an approach to typesetting your BSc thesis using L^AT_EX. A proper abstract should contain around 100 words which makes this one way too short. A good abstract contains: (1) a short description of the tackled problem, (2) a short description of your approach to solving the problem, and (3) (the most successful) result or contribution in your thesis.

Keywords

trust, blockchain, pharmaceutical supply chain

Poglavje 1

Uvod

Poglavje 2

Pregled literature in ozadje

To poglavje združuje temeljne koncepte, na katerih temelji sistem, razvit v okviru magistrske naloge. Začnemo z razumevanjem zaupanja in njegovih lastnosti ter s pregledom pristopov, ki se v razpršenih sistemih uporabljajo za njegovo modeliranje in ocenjevanje. V nadaljevanju se posvetimo farmacevtskim dobavnim verigam, kjer izpostavimo ključne težave, ki se pojavljajo pri zagotavljanju sledljivosti, varnosti in sodelovanja med akterji. Tu omenimo obstoječe rešitve, ki se že uporabljajo v praksi in predstavimo probleme, ki jih te rešitve imajo. Sledi pregled tehnologije veriženja blokov, pametnih pogodb in orakljev, ki danes predstavljajo pomemben del modernih decentraliziranih arhitektur. Poglavje zaključimo s predstavitvijo semantičnega spleta in ontoloških metod, ki so potrebne za formalno opisovanje akterjev in njihovih lastnosti. Ta poglavja skupaj strnemo v množico, v kateri lahko prepoznamo vrzeli obstoječih pristopov in pripravimo teren za rešitev, ki jo predstavimo v nadaljevanju naloge.

2.1 Zaupanje

Pojem zaupanja je temeljni gradnik pri zasnovi našega sistema za farmacevtsko dobavno verigo. Zaupanje je kompleksen, večdimenzionalen in interdisciplinaren koncept, ki ga obravnava različne vede. Obravnava se v sociologiji,

psihologiji, ekonomiji, računalništvu in še ostalih vedah. Prav zaradi te raznolikosti v literaturi ne obstaja enotna definicija, temveč več pristopov, ki poudarjajo različne vidike [1, 2].

2.1.1 Definicije zaupanja

V družboslovnem kontekstu je zaupanje pogosto opredeljeno kot stanje pozitivnih pričakovanj glede dejanj druge osebe v okoliščinah, kjer obstaja določena stopnja tveganja [3]. Gambetta [4] zaupanje definira kot subjektivno verjetnost, da bo agent opravil določeno dejanje, še preden je to mogoče preveriti. Mayer, Davis in Schoorman [5] pa ga opredelijo kot pripravljenost ene stranke, da se izpostavi ranljivosti glede na dejanja druge stranke, ob pričakovanju, da bo ta delovala v skladu s pričakovanji.

V digitalnem okolju se koncept zaupanja prenaša iz družbenega v tehnični kontekst. Denning [6] trdi, da je zaupanje v sistem lastnost, ki jo je mogoče formalno določiti in jo modelirati. Sistem je torej zaupanja vreden, če mu njegovi uporabniki zaupajo glede na opaženo skladnost z vnaprej določenimi standardi. Grandison in Sloman [2] zaupanje obravnavata kot kvalificirano prepričanje zaupnika o kompetentnosti, integriteti, varnosti in zanesljivosti zaupanja vrednega subjekta.

2.1.2 Lastnosti zaupanja

Zaupanje je po svoji naravi večdimenzionalen pojem, ki ga ni mogoče zajeti z eno samo definicijo. V literaturi se ponavlja, da zaupanje vključuje čustvene in vedenjske komponente, ki se med seboj prepletajo in vplivajo na odločanje posameznika ali sistema [7, 8]. V družbenem kontekstu gre za pripravljenost posameznika, da se izpostavi tveganju na podlagi prepričanja, da bo druga stran delovala predvidljivo in skladno s pričakovanji. V informacijskih sistemih pa to pomeni sposobnost sistema, da sprejema odločitve o sodelovanju na podlagi preteklih izkušenj in ocenjene zanesljivosti entitet.

Zaupanje obstaja le v razmerju med dvema ali več entitetami, kjer ena

zaupa drugi v določeni situaciji ali kontekstu. Takšno razmerje je dinamično, saj se lahko stopnja zaupanja sčasoma povečuje ali zmanjšuje. Poleg tega je zaupanje asimetrično: dejstvo, da entiteta A zaupa entiteti B, še ne pomeni, da bo tudi B zaupala A [5].

Ena od osrednjih značilnosti zaupanja je njegova povezanost s tveganjem in negotovostjo. Zaupanje je smiselno le v okoliščinah, kjer obstaja možnost, da drugi akter ne bo ravnal skladno s pričakovanji [3]. Če bi imeli popoln nadzor ali popolno informacijo, zaupanje sploh ne bi bilo potrebno. V kontekstu varnostnih in distribucijskih sistemov zaupanje dopolnjuje formalne zaščitne mehanizme, kot so avtentikacija in šifriranje, saj omogoča oceno vedenja akterjev tam, kjer tehnični ukrepi ne zadostujejo [6].

Zaupanje je tudi subjektivno in kontekstualno. Različni akterji lahko enake vedenjske vzorce interpretirajo različno, odvisno od svojih ciljev, izkušenj, regulatornih zahtev, prepričanj itn.

2.1.3 Atributi zaupanja

Da bi bilo zaupanje uporabno v digitalnem okolju, ga je potrebno izraziti z merljivimi atributi, ki opisujejo lastnosti ali vedenje zaupanja vrednega subjekta. Ti atributi omogočajo, da se konceptualna ideja zaupanja preslika v formalne modele. V literaturi se pojavljajo različne skupine atributov.

Klasični modeli, kot sta Mayer, Davis in Schoorman [5] ter McKnight in Chervany [7], opredeljujejo tri temeljne dimenzije zaupanja: kompetentnost, integriteto in dobrohotnost. Kompetentnost se nanaša na sposobnost entitete, da učinkovito izvede nalogo, integriteta na njeno zavezanost etičnim in profesionalnim načelom, dobrohotnost pa na odsotnost namere povzročiti škodo. Kasneje so bili tem trem dodani še predvidljivost in zanesljivost, ki poudarjata konsistentnost vedenja skozi čas.

V digitalnih in porazdeljenih okoljih so se pojavili dodatni atributi, ki izhajajo iz tehničnih vidikov zaupanja. Grandison in Sloman [2] zaupanje opišeta kot prepričanje o kompetentnosti, varnosti in verodostojnosti drugega subjekta. Denning [6] poudarja, da je zaupanja vrednost sistema mogoče oce-

njevati tudi z vidika varnosti, skladnosti s pravili ter preverljivosti delovanja. Ti atributi omogočajo formalno ocenjevanje zaupanja med akterji.

V farmacevtski dobavni verigi imajo atributi zaupanja izrazito regulatorno in sledilno komponento. Uddin in sodelavci [9] poudarjajo, da so pri zagotavljanju zaupanja ključni kazalniki sledljivost, pristnost in celovitost podatkov. Kayhan in sodelavci [10] izpostavljajo še pomen transparentnosti, nespremenljivosti in izvirne sledljivosti (angl. *provenance*), ki jih omogoča tehnologija veriženja blokov.

Predstavljeni atributi ne predstavljajo univerzalne množice meril, temveč zbirko značilnosti, ki jih lahko posamezen akter uporabi pri lastni presoji zaupanja. V sistemu, ki je predstavljen v tej nalogi, se atributi uporabljajo predvsem kot podatkovne lastnosti (angl. *data properties*) v ontologiji. Te omogočajo, da akterji formalno zapišejo svoje kriterije za ocenjevanje drugih entitet. S tem v sistemu omogočamo standardizirano predstavitev vhodnih metrik, ki jih posamezni akterji uporabljajo pri svojem algoritmu za izračunavanje zaupanja.

2.2 Upravljanje zaupanja v razpršenih sistemih

Huaizhi Li in Mukesh Singhal v članku *Trust Management in Distributed Systems* [11] opredelita upravljanje zaupanja kot proces zbiranja informacij, potrebnih za vzpostavitev zaupanja med entitetami, ter dinamičnega spremljanja in prilagajanja obstoječih razmerij zaupanja. Avtorja poudarjata, da v razpršenih okoljih kot so internet, sistemi enakovrednih entitet (angl. *peer-to-peer*) in mobilna omrežja ad hoc pogosto sodelujejo entitete, ki se med seboj ne poznajo. Zato zaupanje postane ključni mehanizem za zmanjšanje tveganja in zagotavljanje sodelovanja. Namen sistema za upravljanje zaupanja je ohranjati ažurne in skladne informacije o zaupanju med akterji v omrežju.

2.2.1 Dokazni modeli zaupanja

Dokazni modeli temeljijo na preverljivih dokazih, kot so digitalna potrdila, javni ključi ali kriptografski podpisi. Ti pristopi zagotavljajo preverjanje identitete in integritete entitet, ne pa tudi njihove dejanske zanesljivosti. Najpogostejša primera sta hierarhični sistem X.509 [12] in decentralizirani sistem PGP [13], kjer se zaupanje posreduje prek verige certifikatov oziroma prek subjektivnih ocen uporabnikov. Tak model se uporablja predvsem pri inicializaciji zaupanja in zagotavlja osnovno preverjanje pristnosti brez ocenjevanja vedenja.

2.2.2 Priporočilni modeli zaupanja

Priporočilni modeli gradijo zaupanje na osnovi izkušenj in posredovanih ocen drugih entitet. Vsaka entiteta ocenjuje druge glede na pretekle interakcije, te ocene pa se lahko delijo naprej kot priporočila. Model uporablja pogojno tranzitivnost zaupanja – entiteta A lahko zaupa entiteti C, če A zaupa B kot priporočitelju in lahko ovrednoti njegovo oceno.

Pri tem se uporablja zvezna vrednost zaupanja tv_T med 0 in 1, izračunana iz vrednosti priporočil po poti $A \rightarrow B \rightarrow C \rightarrow D$. Za končno vrednost se uporabi zmnožek delnih zaupanj na poti:

$$tv_T = \prod_{i=1}^n \frac{rtv(i)}{4} \times tv(T)$$

kjer $rtv(i)$ predstavlja stopnjo zaupanja v posameznega posrednika, $tv(T)$ pa končno vrednost zaupanja v ciljno entiteto. Če obstaja več poti med akterjema, se končna vrednost določi kot povprečje rezultatov posameznih poti.

2.2.3 Porazdeljeno ocenjevanje zaupanja

V omrežjih enakovrednih akterjev (angl. *peer-to-peer*) se zaupanje ocenjuje decentralizirano. Vsako vozlišče vodi lastno evidenco interakcij in vrednoti

druge glede na uspešnost sodelovanja. Xiong in Liu predlagata metriko, ki temelji na razmerju med pozitivnimi in negativnimi interakcijami:

$$T(u, t) = \frac{\sum_{v \in P, v \neq u} S(u, v, t) Cr(v, t)}{\sum_{v \in P, v \neq u} I(u, v, t)}$$

kjer $S(u, v, t)$ označuje zadovoljstvo uporabnika u z v do časa t , $Cr(v, t)$ je korekcijski faktor za filtriranje povratnih informacij, $I(u, v, t)$ pa število interakcij med u in v . Rezultat $T(u, t)$ je vedno v intervalu $(0, 1)$ in predstavlja trenutno stopnjo zaupanja v entiteto. Sistem lahko določi pragove C_1 in C_2 , pri čemer entiteta velja za zaupanja vredno, če velja $I(u, t) > C_1$ in $T(u, t) > C_2$.

Takšni sistemi so učinkoviti pri akumuliranju ocen, vendar občutljivi za napačne ali zlonamerne povratne informacije. Zato številne rešitve uvajajo uteževanje priporočil glede na zanesljivost virov ali časovno starost podatkov.

2.2.4 Dinamično posodabljanje zaupanja

Ker se vedenje akterjev sčasoma spreminja, se mora zaupanje prilagajati novim podatkom. Posodobitev vrednosti se pogosto izvede z amortizacijskimi ali diskontnimi faktorji, ki dajejo večjo težo nedavnim interakcijam. Primer takega mehanizma je izračun nove vrednosti zaupanja po zaključeni interakciji:

$$T_{\text{new}} = \frac{r + N T_{\text{old}} e^{-\beta \Delta t}}{1 + N e^{-\beta \Delta t}}$$

kjer T_{old} predstavlja prejšnjo vrednost zaupanja, r novo oceno po transakciji, N število izvedenih interakcij, β faktor dušenja, Δt pa čas od zadnje posodobitve. Na ta način sistem zagotavlja, da nove izkušnje hitreje vplivajo na oceno, stare pa postopoma izgubijo težo.

2.2.5 Povezava z našo rešitvijo

Predstavljeni modeli zaupanja predstavljajo osnovo za delovanje razvitega sistema. Vsak akter v dobavni verigi lahko uporabi svoj način izračuna zaupanja, ki temelji na dokazih, priporočilih ali dinamičnih metrikah. Sistem ne

vsiljuje enotnega algoritma, temveč omogoča, da akter sam oblikuje pravila na podlagi podatkov iz ontologije. Rezultati teh izračunov se shranjujejo v verigo blokov, ki deluje kot skupen in trajen zapis ocen. S tem smo združili različne pristope v enoten sistem, kjer je zaupanje merljivo in preverljivo.

2.3 Farmacevtske dobavne verige

2.3.1 Značilnosti in pomen sledljivosti

Farmacevtska dobavna veriga je ena najbolj reguliranih in kompleksnih industrijskih verig. Vključuje proizvajalce, distributerje, prevoznike, lekarne, regulatorne agencije in končne uporabnike [14]. Zaradi velikega števila vmesnih akterjev in globalnega obsega poslovanja je nadzor nad izvorom in kakovostjo zdravil zahtevna naloga.

Sledenje zdravilom omogoča identifikacijo posamezne serije skozi celoten življenjski cikel izdelka, od proizvodnje do izdaje pacientu [10]. Natančno beleženje gibanja zdravil preprečuje vstop ponaredkov v sistem in omogoča hitro ukrepanje ob odpoklicih ali neskladjih. Sledljivost je zato neposredno povezana z varnostjo pacientov in zaupnostjo zdravstvenega sistema [15].

Za zagotavljanje sledljivosti so regulatorji uvedli stroge predpise. Evropska unija je sprejela Direktivo o ponarejenih zdravilih (2011/62/EU) [16] in Delegirano uredbo (EU) 2016/161 [17], ki zahtevata enolično označevanje embalaže z dvodimenzionalno kodo in uporabo evropskega sistema EMVS. V ZDA podobno vlogo opravlja zakon DSCSA, ki uvaja popolnoma digitalno sledenje zdravil do leta 2025 [18].

Ti ukrepi predstavljajo pomembne korake k večji sledljivosti, vendar še vedno temeljijo na centraliziranih bazah podatkov, kjer posamezni akterji nimajo popolnega vpogleda v celotno verigo.

2.3.2 Izzivi in tveganja

Kljub regulacijam farmacevtske dobavne verige ostajajo izpostavljene številnim tveganjem. Največji problem predstavljajo ponarejena in neustrezna zdravila, katerih število v zadnjih letih narašča. Po podatkih Svetovne zdravstvene organizacije je bilo med letoma 2017 in 2021 zabeleženih 877 incidentov ponarejenih ali neustreznih medicinskih izdelkov, kar pomeni povprečno letno rast za 36 % [19]. Ti izdelki so bili zaznani v vseh regijah sveta, tudi v državah z razvitimi regulativnimi sistemi. Ponarejena zdravila so lahko neučinkovita ali nevarna, njihova prisotnost pa neposredno ogroža zdravje pacientov in zmanjšuje zaupanje javnosti v zdravstvene sisteme.

Drugo tveganje izhaja iz razdrobljenosti informacijskih sistemov. Podatki o serijah, skladiščih in pošiljkah se pogosto hranijo v ločenih bazah, ki med seboj niso povezane [9]. To povzroča zamude, napake in oteženo preverjanje izvora zdravila. Centralizirane rešitve, kot je EMVS, sicer izboljšajo preverjanje avtentičnosti, vendar ne rešujejo težave zaupanja med različnimi organizacijami, ki podatke ustvarjajo in posredujejo.

Poleg ponarejanja in pomanjkljive sledljivosti se pojavljajo tudi tveganja, povezana z motnjami v dobavi zdravil. V kriznih obdobjih, kot so pandemije ali politične napetosti, se zaradi pomanjkanja podatkov o zalogah in zanesljivosti partnerjev pogosto pojavijo nepričakovane prekinitve dobav. Takšni dogodki razkrivajo, kako pomembno je zaupanje med organizacijami in pravočasna izmenjava podatkov.

Kljub tehnološkim in pravnim izboljšavam obstoječi modeli še vedno ne zagotavljajo celovitega vpogleda v dogajanje znotraj verige. Potrebne so rešitve, ki združujejo varno upravljanje podatkov, decentralizirano preverjanje in semantično usklajevanje informacij med akterji.

2.3.3 Regulativni okvir in trenutne rešitve

Evropska zakonodaja temelji na Direktivi o ponarejenih zdravilih (2011/62/EU) [16] ter Delegirani uredbi (EU) 2016/161 [17]. Skupaj uvajata obvezne var-

nostne elemente na embalaži, dvodimenzionalno kodo ter sistem EMVS (European Medicines Verification System). EMVS omogoča preverjanje serijske številke ob izdaji zdravila, sistemi NMVS pa sodelujejo prek skupnega evropskega vozlišča. Sistem zaznava poskuse ponarejanja v fazi distribucije, ne omogoča pa vpogleda v celotno zgodovino serije, saj temelji na centralizirani arhitekturi [10].

V ZDA sledljivost ureja zakon DSCSA (Drug Supply Chain Security Act), ki je del zakona DQSA iz leta 2013 [18]. DSCSA uvaja elektronsko izmenjavo treh vrst podatkov: transakcijske informacije, zgodovine transakcij in izjave o skladnosti. Sistem zahteva popolnoma interoperabilno sledljivost do leta 2025. Kljub napredku ameriški model ostaja pretežno centraliziran, podatki pa se prenašajo predvsem med neposrednimi poslovnimi partnerji [9].

Na mednarodni ravni Svetovna zdravstvena organizacija (WHO) pripravlja smernice za dobre distribucijske prakse in spremlja pojav ponarejenih in neustreznih izdelkov.

Za izpolnjevanje regulativnih zahtev podjetja uporabljajo različne informacijske platforme, med katerimi so najpogostejše centralizirane rešitve, kot so že omenjeni EMVS, TraceLink in SAP ATTP. Te rešitve izboljšujejo sledljivost serij in omogočajo avtomatizirano poročanje regulatorjem, vendar ne omogočajo skupnega vpogleda v podatke ali preverjanja zanesljivosti akterjev po celotni verigi [10]. Podatkovni tokovi ostajajo razdrobljeni, zanašajo pa se tudi na zaupanja vredne posrednike.

V zadnjem desetletju se pojavljajo projekti, ki preučujejo uporabo tehnologije veriženja blokov v farmacevtskih dobavnih verigah. Projekta MediLedger v ZDA in PharmaLedger v Evropi uporabljata porazdeljene zapise za izboljšanje integritete in sledljivosti podatkov [9]. Takšni pristopi dokazujejo, da blockchain omogoča nespremenljivost zapisov in enostavnejše preverjanje informacij, vendar večinoma rešujejo le problem integritete podatkov.

Kljub temu večina rešitev ostaja usmerjena v zaupanje v podatke, medtem ko je upravljanje zaupanja med posameznimi organizacijami pogosto implicitno, vezano na statične pravice dostopa ali vnaprej dogovorjene po-

godbeno-pravne odnose. Manj je pristopov, kjer bi lahko vsak akter izrecno definiral svoje kriterije zaupanja in iz njih izpeljal odločitve o sodelovanju. Prav to vrzel bo zapolnil sistem, predstavljen v tej nalogi, saj kombinira semantični model dobavne verige, subjektivne politike zaupanja in zapis rezultatov v verigo blokov.

2.4 Tehnologija veriženja blokov v dobavnih verigah

Tehnologija veriženja blokov je bila prvič predstavljena leta 2008 v sklopu kriptovalute Bitcoin [20]. Osnovna ideja je zasnova podatkovne strukture, ki povezuje zapise v verigo blokov. Vsak blok vsebuje podatke, kriptografski povzetek prejšnjega bloka in časovno oznako. Takšna vezava med bloki preprečuje naknadne spremembe podatkov, saj bi sprememba v enem bloku povzročila neujemanje celotne verige [21].

Veriga blokov deluje v porazdeljenem omrežju, kjer vsako vozlišče hrani kopijo celotne verige. Soglasje o pravilnem zaporedju blokov se doseže s konsenznim mehanizmom. V javnih verigah se pogosto uporabljata t. i. Proof of Work ali Proof of Stake, medtem ko dovoljena omrežja uporabljajo lažje mehanizme, ki temeljijo na znanih udeležencih. Zaradi porazdeljene narave sistem ne potrebuje centralne avtoritete, kar omogoča zanesljivo beleženje podatkov v okolju z različnimi deležniki.

Poleg javnih omrežij, kot je Ethereum, so v praksi pogosta dovoljena omrežja, kjer je dostop omejen na pooblaščne organizacije. Primer takšnega sistema je Hyperledger Fabric [22]. Ta pristop omogoča višjo prepustnost, večjo zasebnost podatkov in nadzor nad identitetami v omrežju. V dobavnih verigah so takšne lastnosti ključne, saj morajo podjetja varovati poslovne podatke, hkrati pa zagotavljati zanesljivo izmenjavo informacij.

Verige blokov omogočajo nespremenljiv zapis dogodkov, preverljivo zgodovino podatkov in enoten pogled na transakcije. To so pomembne zahteve farmacevtskih dobavnih verig, saj želimo omgočiti transparenten pogled na

transakcije v takšnem sistemu.

2.4.1 Pametne pogodbe

Pametne pogodbe predstavljajo enega ključnih konceptov sodobnih verig blokov. Gre za programe, ki se shranijo in izvajajo neposredno v verigi blokov ter se sprožijo, ko so izpolnjeni določeni pogoji. Ethereum je prvi uvedel splošno platformo za izvajanje pametnih pogodb prek navideznega stroja EVM (Ethereum Virtual Machine), ki omogoča deterministično izvajanje kode na vseh vozliščih omrežja [23].

Pametne pogodbe omogočajo avtomatizacijo pravil in procesov, pri čemer ni potrebno zaupati enemu samemu posredniku. Njihovo delovanje je predvidljivo, saj se vedno izvršijo natančno tako, kot je določeno v kodi. Zaradi te lastnosti se pogosto uporabljajo v scenarijih, kjer je treba uveljaviti dogovorjena pravila med neodvisnimi akterji. V dobavnih verigah se uporabljajo za potrjevanje dogodkov, registracijo premikov blaga ter preverjanje skladnosti podatkov.

V modelih, ki temeljijo na verigi blokov, pametne pogodbe služijo tudi kot enotna plast poslovne logike. S tem omogočajo, da vsi akterji delujejo na osnovi istega sklopa pravil. To je posebej pomembno v okoljih, kjer se morajo organizacije dogovarjati o interpretaciji podatkov ali izvaajanju procesov. Pametne pogodbe z nespremenljivo kodo zagotovijo, da se pravila ne spreminjajo brez soglasja deležnikov.

V predstavljenem sistemu ima pametna pogodba osrednjo vlogo, saj hrani rezultate ocenjevanja zaupanja in omogoča preverjanje vzpostavljenih razmerij med akterji. Pogodba vsebuje funkcije za zapisovanje rezultatov, njihovo posodabljanje in preverjanje stanja zaupanja. Tako tvori register zaupanja, ki je dostopen vsem udeležencem verige.

2.4.2 Orakli v pametnih pogodbah

Pametne pogodbe lahko zanesljivo izvajajo le pravila nad podatki, ki so že zapisani na verigi blokov. Ker so verige blokov zasnovane kot zaprt, determinističen sistem, nimajo neposrednega dostopa do zunanjega sveta. Da bi pametne pogodbe lahko reagirale na dogodke iz realnega okolja (cene na trgu, vremenske razmere, meritve senzorjev, podatke informacijskih sistemov), potrebujemo posrednika, ki te podatke pridobi in jih v preverljivi obliki posreduje na verigo. Tak posrednik se imenuje orakelj [24].

Caldarelli oraklje opiše kot celoten ekosistem, ki povezuje zunanje vire podatkov z decentralizirano aplikacijo [25]. Tipično ga sestavljajo trije deli: vir podatkov, komunikacijski kanal in pametna pogodba. Vir podatkov je lahko spletni vmesnik API, podatkovna baza, senzor v internetu stvari ali celo človek, ki potrdi nek dogodek.

Komunikacijski kanal predstavlja vozlišče oziroma niz vozlišč, ki podatke zbrane iz vira prevedejo v obliko transakcije in jih pošljejo pametni pogodbi. Pametna pogodba na drugi strani vsebuje pravila, s katerimi sprejme ali zavrne posredovane podatke in na njihovi podlagi sproži ustrezno logiko.

Ker oraklji ponovno uvajajo zaupanje v zunanje subjekte, se je v literaturi uveljavil pojem *problem oraklja* (angl. oracle problem) [24]. Blockchain sam po sebi nudi lastnosti, kot so nespremenljivost, deterministično izvajanje in odpornost proti cenzuri, vendar te lastnosti ne veljajo avtomatično za podatke, ki prihajajo preko oraklja. Če vir podatkov ni zanesljiv ali je komunikacijski kanal centraliziran, lahko napačni podatki povzročijo napačno izvedbo pametne pogodbe, ne da bi to bilo na verigi neposredno razvidno. Zato je v skupnosti poudarjena potreba po jasnem modelu zaupanja za vsak orakelj. Ta mora opisati, kakšen je motiv oraklja, da deluje pošteno, in kako sistem zazna ali kaznuje odstopanja [24].

V praksi se pojavlja več arhitektur, ki omogočajo delovanje orakljev. Najpreprostejši so centralizirani oraklji, kjer ena organizacija zbira podatke in jih pošilja pametnim pogodbam. Tak pristop je enostaven za implementacijo, vendar ponovno uvaja enotno točko zaupanja in napake. Sledijo jim poraz-

deljeni oziroma decentralizirani oraklji, kjer več neodvisnih vozlišč pridobiva in agregira podatke ter tako zmanjšuje tveganje manipulacije. Primer takšne zasnove je sistem ASTRAEA, ki uporabi glasovalno igro z vložki (angl. stake) in ločene vloge udeležencev (oddajatelji trditev, glasovalci, certificiranci), da z ekonomsko motivacijo spodbudi udeležence k poštenemu poročanju o resničnosti trditev [26]. Podobno pristopajo tudi druge rešitve, kjer se omrežja orakljev, kot je Chainlink, uporabljajo za zbiranje podatkov iz več API-jev in senzorjev ter za izračun dodatnih metrik (npr. reputacije) še pred zapisom v verigo blokov [27].

V okviru te naloge oraklje obravnavamo kot ločeno plast med semantičnim delom sistema in verigo blokov. Podobno kot pri Chainlinku smo zasnovali decentralizirano omrežje pametnih orakljev, kjer teče koda. Ta iz ontologije in pravil zaupanja prebere podatke, izvede izračun zaupanja ter rezultat posredujejo pametni pogodbi. S tem sledimo pristopu, ki ga Kochovski in sodelavci imenujejo *pametni oraklji brez zaupanja* (angl. trustless smart oracles), kjer so oraklji zasnovani tako, da so preverljivi in zamenljivi, veriga blokov pa služi kot nespremenljiv dnevnik rezultatov, ki jih izračunajo oraklji [27]. Na ta način odpravimo potrebo po centraliziranem zaupnem posredniku, hkrati pa omogočimo, da se različne implementacije orakljev oz. različni izračuni zaupanja enotno zapišejo v register zaupanja na verigi blokov. Podrobnejši opis delovanja decentraliziranega sistema orakljev si bomo pogledali v poglavju 3.

2.4.3 Decentralizirane identitete in preverljiva dokazila

Decentralizirane identitete (DID) predstavljajo mehanizem za dosledno in preverljivo identifikacijo akterjev v decentraliziranih okoljih. Standard W3C DID opisuje identifikatorje, ki niso vezani na centraliziranega ponudnika, temveč so pod nadzorom lastnika identitete [28]. DID je globalno enoličen identifikator, na primer `did:ethr:0x1234...`, pri čemer se podatki o tem, kako identiteto preveriti, hranijo v tako imenovanem DID dokumentu. Ta

dokument vsebuje javne ključe, metode preverjanja in dodatne attribute, ki jih sistem lahko uporabi pri avtentikaciji.

Preverljiva dokazila (angl. Verifiable Credentials - VC) so standardizirana digitalna dokazila, ki jih neka entiteta izda drugi entiteti. Model W3C sledi arhitekturi izdajatelj–imetnik–preveritelj. Izdajatelj (Issuer) ustvari in podpiše dokazilo, imetnik (Holder) ga hrani, preveritelj (Verifier) pa ga validira s pomočjo kriptografskih podpisov [29]. VC omogočajo prenos lastnosti ali certifikatov na način, ki je preverljiv, varen in ne potrebuje zaupanja v posrednika. V literaturi so prepoznani kot jedrni gradnik digitalnih identitet, ki združujejo suverenost uporabnika, interoperabilnost ter preverljivost podatkov [30].

V predstavljenem sistemu ima vsak akter v ontologiji lasten DID. DID služi kot osnovni identifikator, na katerega se vežejo atributi in podatki v ontološkem grafu. Na ta način sistem ohrani enotno identiteto akterjev med vsemi plastmi arhitekture.

Preverljiva dokazila uporabljamo za opis lastnosti, ki so pomembne za ocenjevanje zaupanja. Regulator, kot je EMA, lahko na primer izda VC, ki potrjuje veljavnost licence, certifikat GMP ali rezultat presoje kakovosti. Ta dokazila so digitalno podpisana in povezana z DID akterja. Orakelj lahko med izračunom zaupanja prebere VC, preveri podpis izdajatelja ter iz podatkov izlušči metrike, ki jih akter uporablja v svoji politiki zaupanja. Če akter v svojih pravilih določi, da mora imeti partner veljavno licenco ali ustrezno oceno presoje kakovosti, lahko te kriterije pridobi neposredno iz VC.

2.5 Semantični splet in ontološke metode

Semantični splet predstavlja razširitev svetovnega spleta, kjer podatki niso opisani le sintaktično, temveč tudi semantično. To pomeni, da so podatki strukturirani na način, ki omogoča strojno razumevanje in logično sklepanje. Temeljna ideja je, da lahko različni sistemi izmenjujejo podatke v enotnih formatih ter iz njih izpeljujejo nova dejstva, ne da bi se morali dogovoriti o

skupni implementaciji ali aplikaciji [31].

RDF (Resource Description Framework) je osnovni podatkovni model semantičnega spleta. Podatke predstavlja v obliki trojk subjekt–predikat–objekt, kar omogoča enostavno združevanje in razširjanje podatkovnih grafov [32]. RDF ne predpisuje sheme, zato se lahko prilagodi različnim domenam. Zapis se pogosto izvaja v obliki Turtle, RDF/XML ali JSON-LD.

Nad RDF je zasnovan OWL (Web Ontology Language), ki omogoča formalno opisovanje razredov, lastnosti in odnosov med pojmi. OWL temelji na deskriptijski logiki, zaradi česar omogoča avtomatsko sklepanje in preverjanje konsistence ontologije [33]. Z uporabo reasonerjev lahko sistem samodejno razvršča primerke v ustrezne razrede, zazna protislovja in izpelje nova dejstva, ki niso bila eksplicitno zapisana.

Ontologije so osrednji gradnik semantičnega spleta. Predstavljajo formalne modele znanja, kjer so pojmi, lastnosti in hierarhije jasno definirani. Tak pristop je posebej uporaben v kompleksnih domenah, kjer nastopa veliko različnih akterjev in množica podatkovnih razmerij. Ontologije omogočajo enotno semantično podlago, ki ni vezana na implementacijo posameznega sistema, temveč na konceptualni model podatkov.

V predstavljenem sistemu ontologija služi kot centralni model znanja za vse akterje farmacevtske dobavne verige. RDF omogoča predstavitev akterjev in njihovih lastnosti kot povezav v grafu, OWL pa dodaja formalne razrede in omejitve, ki omogočajo avtomatsko sklepanje o lastnostih ali vlogah akterjev. Reasoner iz RDF–OWL grafa izlušči podatke, ki jih orakelj nato uporabi pri izračunu zaupanja. Z uporabo semantičnih tehnologij je mogoče podatke strukturirati na način, ki je razširljiv, interoperabilen in neodvisen od posamezne aplikacije. Zato je ontološka plast primerna kot osnova za izmenjavo informacij med različnimi organizacijami in se dobro povezuje s preostalo arhitekturo sistema.

2.6 Analiza vrzeli in prispevek te naloge

V tem poglavju smo predstavili ključne gradnike, ki so pomembni za razumevanje razvitega sistema. Najprej smo opredelili pojem zaupanja in pokazali, da gre za večdimenzionalen koncept, ki ga ni mogoče zajeti z eno samo definicijo. Različni modeli poudarjajo kompetentnost, integriteto, dobrohotnost in zanesljivost, v digitalnem okolju pa se temu pridružijo še vidiki varnosti, verodostojnosti in skladnosti z zahtevami sistema. Nato smo opisali modele upravljanja zaupanja v razpršenih sistemih ter pokazali, da se ti modeli večinoma osredotočajo na izračun številčne vrednosti zaupanja na podlagi dokazov, priporočil ali zgodovine interakcij.

V nadaljevanju smo predstavili posebnosti farmacevtskih dobavnih verig. Te verige so močno regulirane in vključujejo veliko različnih akterjev. Regulativa, kot sta evropska direktiva o ponarejenih zdravilih in zakon DSCSA v ZDA, uvaža sledljivost serij in elektronsko izmenjavo podatkov. Kljub temu ostajajo pomembne vrzeli. Obstoječi sistemi večinoma temeljijo na centraliziranih bazah podatkov, kjer posamezna podjetja nimajo celovitega vpogleda v verigo. Ponarejena ali neustrezna zdravila se še vedno pojavljajo, podatkovni tokovi pa so razdrobljeni in vezani na zaupanja vredne posrednike. Trenutne rešitve tako izboljšujejo sledljivost, ne rešujejo pa dovolj dobro vprašanja, komu lahko posamezna organizacija zaupa pri sodelovanju.

Tehnologija veriženja blokov in pametne pogodbe ponujajo nov način upravljanja podatkov v takih okoljih. Verige blokov omogočajo nespremenljiv zapis dogodkov in enoten pogled na podatke v porazdeljenem omrežju. Pametne pogodbe omogočajo izvajanje skupnih pravil, ki veljajo za vse udeležence. Oraklji rešujejo problem povezave med verigo blokov in zunanjimi viri podatkov, vendar ponovno odpirajo vprašanje, komu zaupati pri posredovanju podatkov. Semantični splet in ontologije pa omogočajo formalno predstavitev znanja o akterjih in njihovih lastnostih ter avtomatsko sklepanje nad podatki. V literaturi se ti pristopi le redko združijo v enotno arhitekturo, kjer bi imeli akterji možnost, da na podlagi skupne ontologije definirajo svoje kriterije zaupanja, rezultat teh ocen pa se nato trajno zapiše v verigo blokov.

Na tej točki se pokaže osrednja vrzel. Obstoječe rešitve za sledljivost zdravil se osredotočajo na varnost in integriteto podatkov, upravljanje zaupanja med organizacijami pa je običajno implicitno. Temelji na statičnih pravicah dostopa, pogodbah in ročnih pregledih partnerjev. Po drugi strani številni modeli zaupanja in ontološki okviri ne upoštevajo posebnosti farmacevtske domene in niso povezani z verigo blokov. Manj je pristopov, kjer bi lahko vsak akter izrecno zapisal svojo politiko zaupanja, jo vezal na skupni semantični model ter rezultate ocenjevanja delil na način, ki ga lahko drugi akterji neodvisno preverijo.

Prispevek te naloge je predlog sistema, ki te elemente poveže v enoten okvir. Ontologija farmacevtske dobavne verige predstavlja skupno semantično plast, v kateri so akterji in njihove lastnosti opisani na strukturiran in razširljiv način. Vsak akter lahko nad temi podatki definira lastna pravila zaupanja v berljivi obliki JSON in tako določi, katere metrike so zanj pomembne. Mehanizem za sklepanje iz ontologije prebere lastnosti akterjev, uporabi lokalna pravila in za vsak par udeležencev izračuna rezultat zaupanja. Decentralizirani oraklji nato te rezultate prenesejo v pametno pogodbo, ki na verigi blokov hrani register relacij zaupanja. Sistem omogoča tudi vezavo na decentralizirane identitete in preverljiva dokazila, kadar akterji želijo izvor podatkov podpreti z digitalnimi certifikati.

Tak pristop neposredno naslavlja vrzeli, opisane v tem poglavju. Namesto enotne, vnaprej določene metrike zaupanja omogoča, da vsak akter oblikuje lastno politiko na osnovi skupnega modela podatkov. Rezultati teh različnih politik so kljub temu zbrani v enoten register, ki ga lahko delijo vsi udeleženci verige. S tem sistem podpira različne poglede na zanesljivost partnerjev, hkrati pa ohranja konsistentno predstavitev identitet in podatkovnih lastnosti.

Čeprav je sistem razvit na primeru farmacevtske dobavne verige, je zasnova splošna. Ontologijo in nabor atributov bi bilo mogoče prilagoditi drugim domenam, na primer pametnim mestom ali omrežjem interneta stvari, kjer številni akterji izmenjujejo podatke in storitve. Enak vzorec se ponovi

tudi tam: skupni semantični model entitet, subjektivne politike zaupanja, modul za sklepanje in register rezultatov na verigi blokov.

V nadaljevanju naloge bomo opisali, kako je tak sistem konkretno implementiran ter kako se obnaša v izbranih scenarijih uporabe.

Poglavje 3

Implementacija sistema

3.1 Uporabljene tehnologije

Pri razvoju sistema smo uporabili različna orodja in platforme, ki skupaj omogočajo obdelavo semantičnih podatkov, izvajanje izračuna zaupanja ter zapis rezultatov v verigo blokov. V tej sekciji podajamo pregled najpomembnejših tehnologij in njihovih vlog v sistemu.

Apache Jena Fuseki Semantični podatki o akterjih in njihovih lastnostih so shranjeni v strežniku Apache Jena Fuseki, ki predstavlja RDF podatkovno bazo z dostopom prek SPARQL vmesnika. Fuseki omogoča centralizirano hrambo ontologije in primerkov ter enoten način pridobivanja podatkov za vse oraklje. Zaradi stabilnosti, enostavne uporabe in dobre združljivosti s standardi RDF je bila izbira naravna. Sistem teče v Docker vsebniku, kar omogoča hitro postavitve in ponovljivost okolja.

Ontologija v OWL/RDF Ontologija farmacevtske dobavne verige je zapisana v jeziku OWL, ki definira razrede, lastnosti in odnose med akterji. Podatki so organizirani v RDF trojkah, kar omogoča enostavno obdelavo s knjižnico `rdflib` [34] v programskem jeziku Python. Na ta način lahko sklepanje temelji na aktualnih podatkih, zapisanih v ontološkem grafu, brez

potrebe po relacijski bazi ali ročnem povezovanju.

JSON pravila zaupanja Vsak akter lahko svoje kriterije zaupanja definira v JSON obliki. JSON je izbran zaradi enostavne berljivosti in neposredne integracije s Pythonom. Pravila določajo pragove, ki jih mora akter izpolniti, na primer minimalno točnost dostave ali največjo dovoljeno stopnjo temperaturnih odstopanj. Python orakelj ta pravila interpretira in preveri, ali podatki iz ontologije ustrezajo zahtevam ocenjevalca.

Ethereum, Solidity in pametna pogodba Za zapis rezultatov izračunanega zaupanja smo uporabili verigo blokov Ethereum. Pametna pogodba **TrustGraph**, napisana v jeziku Solidity, hrani relacije zaupanja med akterji. Ethereum je bil izbran zaradi široke podpore za orodja, stabilnega ekosistema in zmožnosti izvajanja pametnih pogodb brez zaupanja v enega posrednika. Razvoj in testiranje sta izvedena s pomočjo orodja Foundry, ki nudi lokalno testno omrežje, kompilator ter hitre skripte za izvajanje transakcij.

Pametna pogodba omogoča zapis rezultatov izračuna, preverjanje stanja zaupanja in beleženje dogodkov. Deluje kot nespremenljiv register, ki je dostopen vsem akterjem sistema.

Decentralizirane identitete in preverljiva dokazila Vsak akter v ontologiji ima dodeljen decentraliziran identifikator (DID). DID služi kot enoličen identifikator, ki povezuje podatke v ontologiji s predstavnikom akterja v decentraliziranem okolju. Poleg tega sistem podpira uporabo preverljivih dokazil (VC), ki jih lahko izdajajo regulatorji, kot je EMA. Ta dokazila vsebujejo metrike, kot so veljavnost licence, GMP skladnost ali rezultati presoje kakovosti. Orakelj lahko, kadar akter to določi v pravilih, med izračunom zaupanja prebere VC in uporabi podatke kot dodatne kriterije.

Python in osnovne knjižnice Del logike sistema je implementiran v programskem jeziku Python, kjer tečejo oraklji, agregator ter orodja za upravljanje ontologije. Python je bil izbran zaradi že razvitih uporabnih knjižnic

za obdelavo podatkov, delo z RDF in integracijo z verigami blokov. Spodaj so povzete ključne knjižnice, ki jih sistem uporablja:

- **web3** in **eth_account** – knjižnica **web3** omogoča povezavo z omrežjem Ethereum oziroma lokalnim vozliščem Anvil, klicanje funkcij pametne pogodbe in pošiljanje transakcij. Modul **eth_account** se uporablja za podpisovanje poročil orakljev in preverjanje podpisov, ki jih agregator prejme.
- **aiohttp** zagotavlja asinhroni spletni strežnik in odjemalca, ki ju uporabljata agregator in oraklji. Prek njega oraklji prejemaajo zahteve za izračun zaupanja, berejo podatke iz Fusekija in pošiljajo poročila nazaj agregatorju.
- **rdflib** omogoča delo z RDF grafi, razčlenjevanje OWL datotek in manipulacijo ontologije.
- **SPARQLWrapper** omogoča pošiljanje SPARQL poizvedb strežniku Apache Jena Fuseki in s tem branje lastnosti akterjev iz ontologije
- **requests** se uporablja pri pomožnih sinhronih HTTP klicih, kjer asinhrona izvedba ni potrebna ali bi otežila implementacijo.
- **click** se uporablja pri implementaciji ukaznih orodij, kot je **trustkb**, ki omogoča delo z ontologijo, uvoz podatkov in izvajanje SPARQL poizvedb iz ukazne vrstice.
- **pytest** knjižnica je bila uporabljena za enotsko testiranje ključnih modulov sistema, saj omogoča enostavno pisanje testov ter avtomatizirano preverjanje logike orakljev in agregatorja.
- **didkit** – knjižnica se uporablja v primerih, kjer se digitalna dokazila (VC) izdaja ali preverja neposredno iz Python okolja. Omogoča delo z DID metodami in verifikacijo podpisov, skladno s standardi W3C DID in VC.

Pametni oraklji Oraklji predstavljajo most med semantično plastjo in verigo blokov. Vsak orakelj zažene Python modul, ki prebere ontologijo, uporabi lokalna pravila akterja in izračuna vrednosti zaupanja. Nato preko `web3.py` pošlje transakcijo pametni pogodbi. Oraklji so zasnovani modularno, zato jih lahko organizacije poganjajo samostojno, kar omogoča decentralizirano izvajanje izračuna.

Docker Vse komponente sistema so zajete v vsebnikih Docker. To nam je omogočilo hitro postavitve razvojnega okolja, enostavno upravljanje odvisnosti in ponovljivost testov. Vsak orakelj, strežnik Fuseki, in lokalna Ethereum veriga tečejo v ločenih vsebnikih. Docker je uporabljen izključno za potrebe testiranja med razvojem.

3.2 Arhitektura sistema

Arhitektura razvitega sistema temelji na večplastnem pristopu, ki ločuje podatkovni model, mehanizme izračuna in zapis rezultatov zaupanja. Takšna zasnova omogoča jasne meje med posameznimi odgovornostmi, kar olajša razvoj, testiranje in morebitno razširjanje sistema na druge domene. V okviru magistrske naloge smo zasnovali arhitekturo, ki je dovolj splošna, da jo je mogoče uporabiti v različnih razpršenih okoljih, od farmacevtskih dobavnih verig do pametnih mest.

Osnovna ideja arhitekture je bila, da združimo tri ključne tehnologije: semantični splet, pametne oraklje in verigo blokov. Semantični splet omogoča opis akterjev in njihovih lastnosti v ontološkem grafu. Oraklji predstavljajo računsko plast, kjer posamezni akter izvede oceno zaupanja po svojih pravilih. Veriga blokov pa služi kot zanesljiva in nespremenljiva plast, kjer se hranijo rezultati teh ocen.

Sistem se razlikuje od ostalih rešitev, ki se osredotočajo zgolj na sledljivost ali upravljanje podatkov. Omogoča, da vsak akter izrecno definira svoje kriterije zaupanja in jih izvede nad skupnim modelom znanja. Rezultati teh

izračunov se zapišejo v decentraliziran register, ki je enak za vse udeležence.

V nadaljevanju poglavja predstavimo zasnovo arhitekture in posamezne plasti, prikažemo njihov medsebojni tok podatkov ter utemeljimo vlogo uporabljenih tehnologij. Tak pristop daje sistemu modularnost, zamenljivost izračunskih enot in razširljivost na druge aplikacijske domene.

3.2.1 Pregled večplastne zasnove

Arhitektura sistema temelji na treh jasno ločenih plasteh, ki skupaj tvorijo celoten potek od zajema podatkov do zapisa rezultatov zaupanja na verigo blokov. Takšna razdelitev omogoča boljše razumevanje posameznih vlog in enostavnejše nadgrajevanje sistema, saj lahko vsako plast obravnavamo neodvisno od drugih.

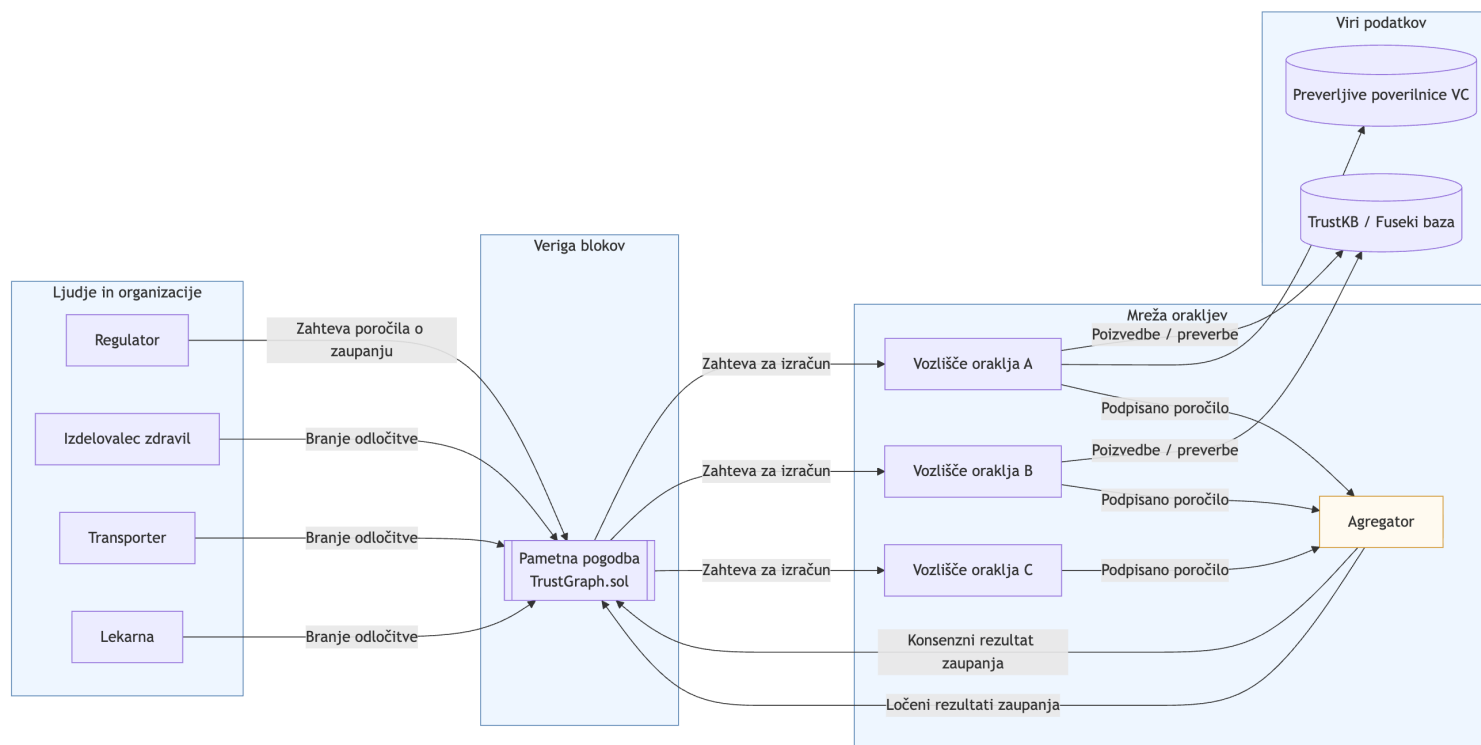
Na sliki 3.1 je prikazan kontekstni pogled celotnega sistema. Diagram združuje tri glavne sklope: akterje iz dobavne verige na levi, infrastrukturo orakljev in podatkovnih virov na desni ter verigo blokov v sredini. Akterji, kot so regulator, proizvajalec zdravil, prevoznik in lekarna, komunicirajo izključno s pametno pogodbo `TrustGraph.sol`. Ta beleži rezultate zaupanja in upravlja zahteve za ponovno oceno posameznih entitet.

Ko regulator ali drug akter sproži zahtevo za izračun zaupanja, pametna pogodba pošlje zahtevo omrežju orakljev. Vsak orakelj nato izvede svoj del naloge: iz ontologije in baze `TrustKB` pridobi podatke, po potrebi preveri pripadajoča preverljiva dokazila, nato pa izračuna rezultat zaupanja. Oraklji rezultate podpišejo in posredujejo agregatorju, ki pripravi končni konsenzni izračun in ga zapiše nazaj v pametno pogodbo.

V skrajnem desnem delu diagrama je prikazana plast znanja. Ta vključuje ontologijo (`TrustKB`), kjer so zbrani koncepti, njihove lastnosti in meritve, ter skladišče dokazil `VC`, ki zajema licence, certifikate in druge verodostojne podatke o akterjih. Oraklji pri izračunu združujejo oba vira: semantične lastnosti iz ontologije in verodostojne podatke iz dokazil.

Sistem tako deluje skozi jasno razmejene sklope, kar je razvidno iz celotnega toka na diagramu – od zahteve, preko izračuna, do zapisa končnega

rezultata zaupanja na verigi blokov.



Slika 3.1: Kontekstni pogled sistema za upravljanje zaupanja.

V nadaljevanju so podrobneje opisane tri plasti, ki sestavljajo arhitekturo:

1. **Plast znanja (Knowledge Layer)** V tej plasti se nahaja ontologija dobavne verige in opisne lastnosti akterjev. Ontologija predstavlja skupni referenčni model, ki omogoča enotno interpretacijo podatkov o akterjih. Poleg ontologije se tukaj nahaja tudi skladišče preverljivih dokazil (VC), ki dodaja podatke o licencah, certifikatih in presojah kakovosti.
2. **Plast izračuna zaupanja (Trust Resolution Layer)** Oraklji v tej plasti izvajajo izračune zaupanja. Podatke pridobijo iz ontologije in preverljivih dokazil ter jih povežejo s pravili, ki jih določi posamezen akter. Pravila so zapisana v JSON obliki in opisujejo pogoje, ki jih mora ocenjevana entiteta izpolnjevati. Oraklji izvedejo izračun, rezultat podpišejo in ga posredujejo agregatorju.
3. **Plast registra zaupanja na verigi blokov (Blockchain Trust Registry Layer)** Ta plast vključuje pametno pogodbo, ki hrani rezultate ocenjevanja zaupanja. Pogodba deluje kot decentraliziran register, ki omogoča vpogled v trenutna in pretekla razmerja zaupanja med akterji. Vanjo se vpisujejo tako posamezni odgovori orakljev kot konsenzni rezultat evalvacije.

V naslednjih poglavjih so opisane tehnične podrobnosti posamezne plasti ter njihove medsebojne povezave.

Plast znanja

Plast znanja predstavlja osnovni vir informacij, na katerem poteka izračun zaupanja. Vsebuje semantični model dobavne verige zdravil, podatke o posameznih akterjih ter dodatna preverljiva dokazila, ki opisujejo regulatorne lastnosti ali skladnost z zahtevami. V tej plasti se združujejo podatki iz dveh virov: ontologije, ki opisuje akterje in njihove lastnosti, ter preverljivih

dokazil (VC), ki dopolnjujejo semantični model z digitalno podpisanimi trditvami. Oraklji te podatke uporabljajo pri izračunu zaupanja v nadaljnjih plasteh sistema.

Ontologija (OWL/RDF) Ontologija farmacevtske dobavne verige je zgrajena v jeziku OWL in zapisana v formatu RDF. Vsebuje razrede, ki predstavljajo posamezne tipe akterjev, kot so *Proizvajalec*, *Prevoznik*, *Lekarna* in *Regulator*. Tem razredom so pripisane lastnosti, ki opisujejo njihovo vedenje ali skladnost s standardi, na primer:

- **hasDeliveryPunctuality** – delež pravočasnih dostav,
- **hasTempViolationRate** – delež temperaturnih odstopanj pri transportu,
- **hasLicense** – veljavnost poslovne licence,
- **hasGMP** – skladnost s standardi GMP [35],
- **hasAuditScore** – rezultat presoje kakovosti.

Ontologija vsebuje tudi posamezne primerke akterjev, na primer *Pfizer*, *DHL* in *MediPlus*, ki imajo pripisane konkretne vrednosti lastnosti. Spodaj je prikazan primer zapisa prevoznika v RDF/Turtle obliki:

```
ex:DHL a ex:Transporter ;
  ex:hasDeliveryPunctuality "0.94"^^xsd:decimal ;
  ex:hasTempViolationRate "0.02"^^xsd:decimal ;
  ex:hasLicense "true"^^xsd:boolean .
```

Ontologija je shranjena v strežniku Apache Jena Fuseki, ki omogoča izvajanje SPARQL 1.1 poizvedb preko spletnega vmesnika. Oraklji tako dostopajo do podatkov z neposrednimi poizvedbami, kot je na primer:

```
PREFIX ex: <http://example.org/trust#>

SELECT ?punctuality ?tempRate ?license
```

```

WHERE {
  ex:DHL ex:hasDeliveryPunctuality ?punctuality .
  ex:DHL ex:hasTempViolationRate ?tempRate .
  ex:DHL ex:hasLicense ?license .
}

```

Ker je ontologija zgrajena v OWL, omogoča tudi uporabo razumevalnika (angl. reasoner). Razumevalnik lahko samodejno razvrsti akterje v razrede (npr. **TrustedTransporter**), zazna nedoslednosti in izpelje lastnosti, ki niso zapisane neposredno. Vsak orakelj lahko razumevalnik po potrebi zažene lokalno in s tem pridobi izpeljane podatke.

Preverljiva dokazila (VC) Poleg podatkov iz ontologije sistem uporablja še preverljiva dokazila (VC), ki jih izdajajo regulatorji ali druge zaupanja vredne organizacije. Ta dokazila vsebujejo trditve o akterjih, kot so veljavnost licence, GMP skladnost ali datum zadnje presoje kakovosti. Dokazila so zapisana v formatu JSON-LD in digitalno podpisana.

Primer dokazila, ki ga izda regulator:

```

{
  "id": "vc-gmp-pfizer-001",
  "type": ["VerifiableCredential", "GMPCertificate"],
  "issuer": "did:example:ema",
  "credentialSubject": {
    "id": "did:example:pfizer",
    "hasGMP": true
  },
  "proof": {
    "type": "Ed25519Signature2018",
    "signatureValue": "3f9ac..."
  }
}

```

Preverljiva dokazila so shranjena v ločeni podatkovni bazi. Ob izračunu zaupanja orakelj preveri kriptografski podpis izdajatelja, iz podatkov izlušči relevantne lastnosti in jih združi z informacijami iz ontologije. Tako sestavi popoln nabor kriterijev, ki jih akter uporabi pri ocenjevanju zaupanja.

Plast znanja združuje semantične opise akterjev in njihove lastnosti z regulatorno potrjenimi podatki iz preverljivih dokazil. Oraklji pri izračunu zaupanja dostopajo do obeh virov: iz Fusekija pridobijo opisne metrike, iz VC-jev pa dodatne podatke, ki jih ontologija ne vsebuje. S tem plast znanja omogoča enoten, strukturen in preverljiv nabor podatkov, na katerem temelji celoten izračun zaupanja.

Plast izračuna zaupanja

Plast izračuna zaupanja predstavlja osrednjo procesno komponento arhitekture. Njena naloga je, da podatke iz plasti znanja pretvori v konkretne odločitve o zaupanju in jih posreduje verigi blokov. V tej plasti se nahaja omrežje orakljev in agregator, ki skupaj tvorita neodvisen mehanizem sklepanja. Oraklji izvajajo izračune, agregator pa skrbi za preverjanje rezultatov in zapis končnega stanja v pametno pogodbo.

Oraklji Ko sistem prejme zahtevo za izračun zaupanja, pametna pogodba sproži dogodek, ki služi kot signal orakljem, da morajo izvesti novo oceno. Oraklji ne delujejo kot pasivni odjemalci, temveč aktivno spremljajo verigo blokov in zaznavajo zahteve, ki jih je treba obdelati. Za podani subjekt nato pridobijo vse podatke iz plasti znanja: opisne lastnosti iz ontologije ter pripadajoča preverljiva dokazila iz baze poverilnic VC. Na ta način je izračun zaupanja vedno vezan na najnovejše in preverljive podatke.

Pridobljene podatke orakelj združi s pravili ocenjevalca. Pravila določajo, katere lastnosti mora ocenjevan akter izpolnjevati in pod kakšnimi pogoji. Orakelj izvede ocenjevanje in pripravi strukturirano poročilo, ki vsebuje identifikator subjekta, končno odločitev, numerično oceno, kriptografske povzetke uporabljenih dokazil ter čas izračuna. Poročilo je serializirano v JSON obliki

in digitalno podpisano z zasebnim ključem oraklja. S tem postane rezultat preverljiv in vezan na točno določen orakelj.

Agregator Podpisano poročilo orakelj pošlje agregatorju, ki deluje kot zbirni člen med oraklji in pametno pogodbo. Agregator preveri digitalni podpis, preveri, ali je poročilo prišlo od dovoljenega oraklja, ter ga zapiše na verigo kot posamično oddajo. Ko prejme zadostno število poročil, izvede konsenz nad prejetimi odločitvami in oceno zaupanja ter prek transakcije posreduje končni rezultat v pametno pogodbo **TrustGraph**.

Na visoki ravni potek izračuna poteka v naslednjih korakih:

1. pametna pogodba na verigi blokov prejme zahtevo za nov izračun zaupanja in sproži dogodek,
2. oraklji poslušajo te dogodke, za podanega akterja preberejo podatke iz ontologije in pripadajočih preverljivih dokazil,
3. podatke združijo s pravili ocenjevalca in izvedejo izračun zaupanja,
4. rezultat pretvorijo v strukturirano poročilo, ga digitalno podpišejo in pošljejo agregatorju,
5. agregator preveri prejeta poročila, izračuna konsenz in prek transakcij zapiše posamezne oddaje ter končni rezultat v pametno pogodbo.

Na ta način plast izračuna zaupanja povezuje semantične podatke z decentraliziranim zapisom rezultatov. Oraklji omogočajo neodvisno izvajanje izračunov, agregator pa poskrbi za preverjanje in enoten zapis v register zaupanja na verigi blokov. Plast tako tvorimo most med modelom znanja in verigo blokov ter predstavlja ključen element celotne arhitekture.

Plast registra zaupanja na verigi blokov

Plast registra zaupanja predstavlja končni sloj arhitekture, kjer se rezultati izračuna zaupanja trajno zapišejo. Njena vloga je zagotoviti podatkovno

plast, ki ni odvisna od posamezne organizacije in ki omogoča preverljiv, dosleden ter nespremenljiv zapis vseh odločitev o zaupanju. V tej plasti se nahaja pametna pogodba **TrustGraph**, ki deluje kot decentraliziran register razmerij zaupanja med akterji.

Pametna pogodba je zasnovana tako, da sprejema rezultate evalvacij orakljev in jih zapisuje v podatkovne strukture na verigi. Podpisi orakljev, konsenz agregatorja in mehanizmi preverjanja identitet poskrbijo, da v register pridejo le preverjene in verodostojne informacije. Vsak zapis je povezan z identifikatorjem akterja, časom vnosa in dodatnimi metapodatki, kar omogoča transparenten zgodovinski vpogled v spremembe zaupanja skozi čas.

Ko agregator prejme dovolj veljavnih poročil orakljev, izvede postopek združevanja in prek transakcije pokliče funkcijo v pametni pogodbi, ki zapiše tako posamične oddaje kot končni konsenzni rezultat. Pogodba sproži dogodke, ki služijo kot signal akterjem ali nadrejenim sistemom, da je za določen subjekt na voljo nova ocena zaupanja. Vsaka sprememba je trajno zapisana v verigo blokov, zato je mogoče rezultate neodvisno preveriti tudi za nazaj.

Register zaupanja hrani dve vrsti podatkov. Prva vrsta so posamične oddaje orakljev, ki predstavljajo neobdelane rezultate izračuna. Druga vrsta so konsenzni rezultati, ki jih agregator izračuna na podlagi več oddaj in jih pogodba obravnava kot veljavno končno stanje zaupanja za posameznega akterja. Na ta način lahko sistem ohranja celovito sliko o tem, kako so se rezultati spreminjali in kako je bila sprejeta končna odločitev.

Zapis v register omogoča tudi poizvedovanje nad trenutnim ali zgodovinskim stanjem zaupanja. Akterji v dobavni verigi lahko kadar koli preverijo, ali je določen partner trenutno ocenjen kot zaupanja vreden ali ne. Ker podatki niso shranjeni v centraliziranem sistemu, temveč v porazdeljeni verigi blokov, se vnosov ne da naknadno spreminjati ali brisati. To daje sistemu lastnost dolgoročne preverljivosti, kar je ključno v okoljih, kjer se sprejemajo odločitve z regulatornimi posledicami.

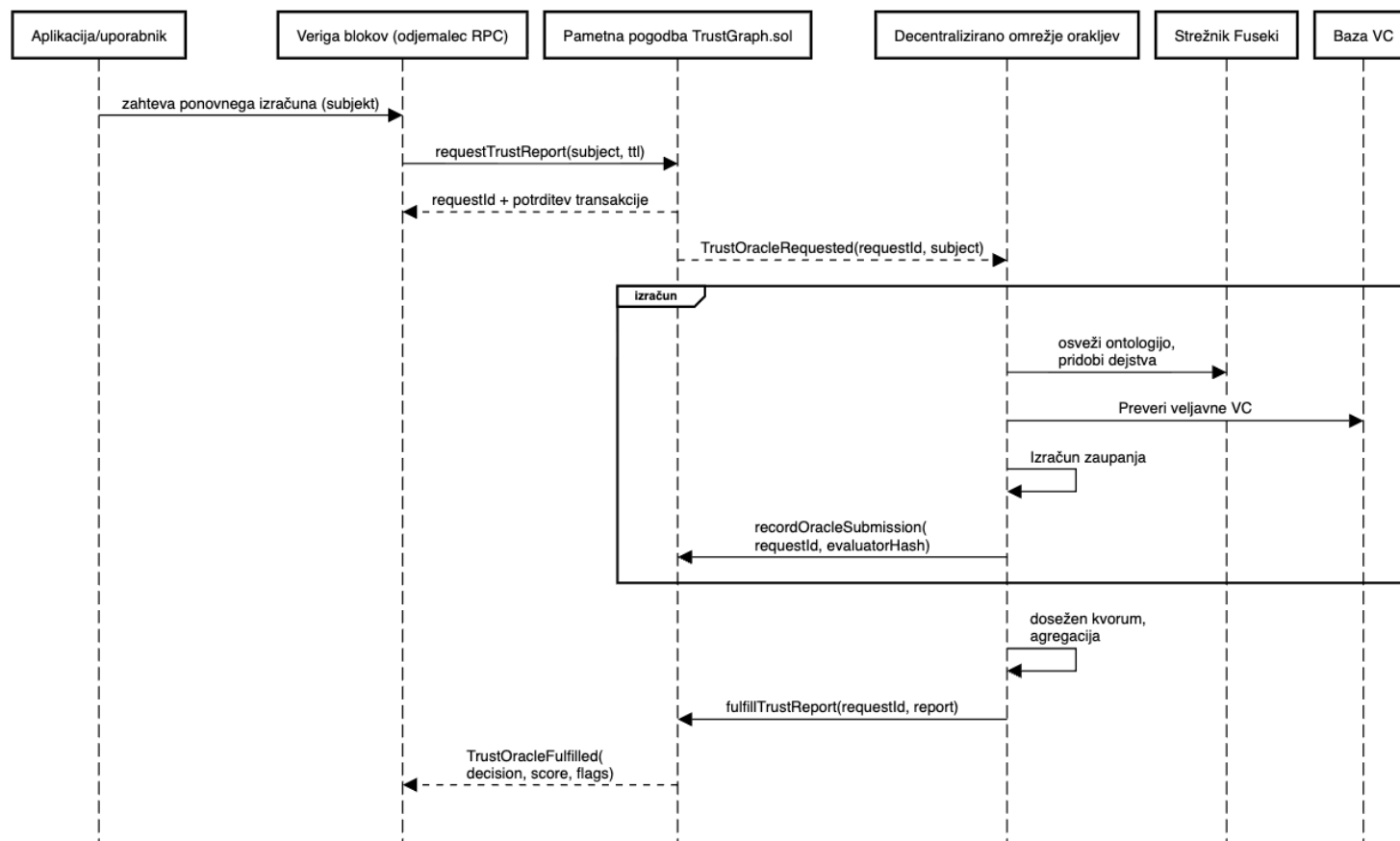
Plast registra zaupanja tako predstavlja stičišče celotnega sistema: prejme rezultate iz računske plasti, jih trajno zapiše in omogoča zanesljiv vpogled v

relacije zaupanja v vsakem trenutku. V nadaljevanju bomo opisali funkcije pametne pogodbe **TrustGraph**, podatkovne strukture, dogodke ter način, kako zunanji akterji dostopajo do registra.

3.3 Tok podatkov v sistemu

3.3.1 Celoten potek izračuna zaupanja

Na sliki 3.2 je prikazan celoten proces izračuna zaupanja, ki vključuje vse komponente sistema: aplikacijo, pametno pogodbo **TrustGraph.sol**, decentralizirano omrežje orakljev, strežnik Fuseki ter bazo preverljivih dokazil VC. V nadaljevanju se osredotočimo na funkcije uporabljene v procesu. Pri vsakem koraku so navedene ključne funkcije.



Slika 3.2: Diagram poteka ob izračunu zaupanja za subjekt.

1. Aplikacija ali uporabnik ustvari zahtevo

Uporabnik preko aplikacije ali klica API sproži postopek za izračun zaupanja za zelenega akterja. Akterjev DID pred klicem funkcije pretvorimo v binarno vrednost `bytes32` s pomočjo zgoščevalne funkcije Keccak-256, ki je značilna za Ethereumov ekosistem. Aplikacija preko odjemalca RPC pokliče funkcijo na pametni pogodbi:

```
requestTrustReport(subject, ttl)
```

2. Pametna pogodba obdela zahtevo in izda signal za izračun

Po klicu funkcije `requestTrustReport(subject, ttl)` pametna pogodba preveri, ali je za izbranega akterja že aktivna zahteva ter ali ima klicatelj ustrezna dovoljenja (če je seznam dovoljenih nastavljen). Ko so pogoji izpolnjeni, pogodba ustvari nov identifikator zahteve `requestId`. Identifikator temelji na zgoščeni kombinaciji DID akterja, časovnega žiga, naslova pošiljatelja in zaporedne številke zahteve. Pogodba shrani rok za oddajo (`deadline`) in nato sproži dogodek:

```
TrustOracleRequested(requestId, subject)
```

Ta dogodek služi kot edini mehanizem obveščanja decentraliziranega omrežja orakljev, da je treba izvesti nov izračun zaupanja. Obenem funkcija klicatelju tudi vrne `requestId`, na podlagi katerega lahko spremlja stanje in kasneje pridobi končni rezultat.

3. Orakelj zazna novo zahtevo in začne proces izračuna

Vsak orakelj neprestano posluša za nove dogodke, zapisane v verigi blokov (angl. *event logs*). Ko zazna dogodek `TrustOracleRequested`, preveri, ali mora obdelati podani `requestId`. Če je orakelj nastavljen za tovrstne izračune, začne postopek ocenjevanja. Najprej pridobi vse potrebne podatke iz plasti znanja: semantične lastnosti akterja preko SPARQL poizvedb strežniku Fuseki ter pripadajoča preverljiva dokazila iz baze VC. Orakelj tako zagotovi, da bo izračun temeljil na najnovejših, verodostojnih in digitalno podpisanih podatkih. Šele ko so vsi

podatki zbrani in preverjeni, orakelj preide na sam izračun zaupanja, kjer uporabi pravila ocenjevalca.

4. Orakelj osveži podatke v plasti znanja

Pred izračunom zaupanja mora orakelj pridobiti najnovejše podatke iz dveh ločenih virov:

- semantične lastnosti akterja iz ontologije prek SPARQL poizvedb strežniku Fuseki,
- veljavna preverljiva dokazila (VC), ki jih orakelj prebere iz lokalne baze dokazil.

Pridobivanje ontoloških podatkov poteka prek SPARQL končne točke, ki vrne lastnosti subjekta (npr. točnost dostav, licenčne statuse, GMP skladnost). Preverljiva dokazila se preverijo z validacijo digitalnega podpisa izdaje. S tem orakelj zagotovi, da izračun temelji na najnovejših in verodostojnih podatkih.

5. Izvedba izračuna glede na pravila ocenjevalca

Ko ima orakelj na voljo vse relevantne podatke, izvede izračun zaupanja. Pri tem uporabi pravila, ki si jih bomo v nadaljevanju pogledali na testnih scenarijih. Rezultat tega koraka je numerična ocena in binarna odločitev, ali akter izpolnjuje zahtevane pogoje. Poleg tega orakelj pripravi tudi metapodatke, kot so:

- čas izračuna (`asOf`),
- zgoščena vrednost pravil (`policyHash`),
- zgoščena vrednost uporabljenega VC (`credentialHash`).

6. Orakelj ustvari poročilo in ga podpiše

Orakelj pripravi končno poročilo v JSON obliki, ki vsebuje ključne informacije o izračunu zaupanja: identifikator subjekta, odločitev, oceno, zastavice in pripadajoče povzetke. Preden ga pošlje agregatorju, mora orakelj poročilo kriptografsko podpisati. Podpis se izvede z uporabo

metode `sign_message`, ki je del knjižnice `web3.py`. Ta temelji na Ethereumovi shemi EIP-191. Najprej se pripravi deterministična kanonična oblika JSON zapisa, ta pa se nato podpiše z zasebnim ključem oraklja. Podpis omogoča agregatorju, da preveri, ali je poročilo prišlo od zaupanja vrednega oraklja.

7. Agregator sprejme poročila in izvede preverjanje

Agregator deluje kot zbirno vozlišče, ki prejema poročila posameznih orakljev prek HTTP vmesnika. Pred prejemom poročila izvede naslednja preverjanja:

- ali podpis pripada dovoljenemu oraklju,
- ali poročilo ni podvojeno,
- ali se zgoščene vrednosti pravil in dokazil ujemajo z lokalnimi podatki.

Vsako veljavno poročilo agregator takoj zapiše v pametno pogodbo z uporabo funkcije:

```
recordOracleSubmission(requestId, evaluatorHash, report,
    credentialHash)
```

Na ta način je posamezna ocena oraklja shranjena na verigi blokov, neodvisno od končnega konsenza.

8. Agregator izvede konsenz in zapiše končni rezultat

Ko agregator prejme dovolj veljavnih poročil (dosežen kvorum), izvede konsenz nad rezultati. Pri tem kombinira odločitev, oceno in dodatne zastavice, nato pa pripravi končno poročilo. Končni rezultat se zapiše v pametno pogodbo z uporabo funkcije:

```
fulfillTrustReport(requestId, report)
```

Pametna pogodba preveri, ali je agregator pooblaščen, ali je `requestId` veljaven in ali zahteva ni pretekla. Če so vsi pogoji izpolnjeni, pogodba zapiše konsenzni rezultat v verigo blokov in sproži dogodek

`TrustOracleFulfilled`. Pogodba preveri identiteto klica, ujemanje ID in iztek roka, preden zapiše končni rezultat v:

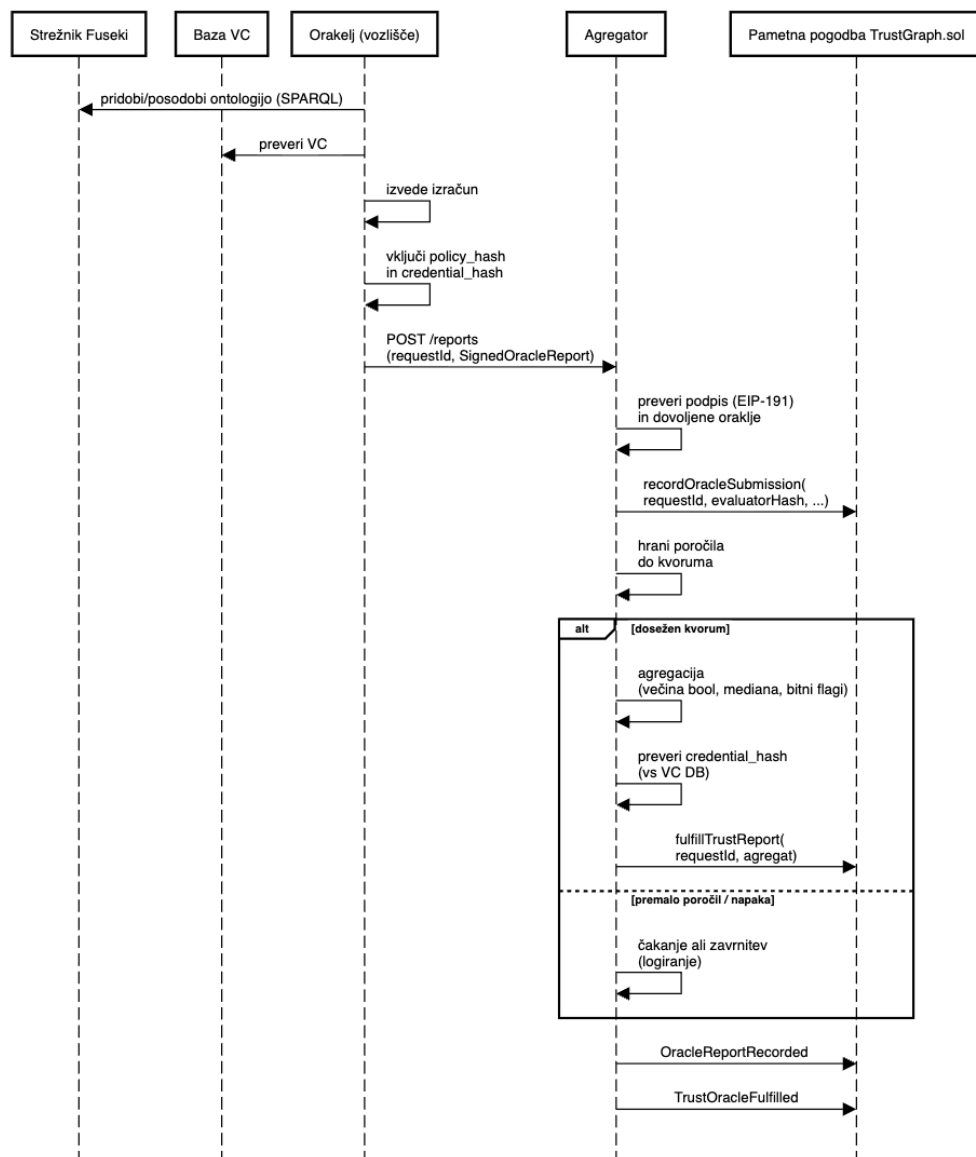
`subjectMetrics[subject]`

in sprožen je dogodek `TrustOracleFulfilled`, ki zaključi postopek.

Ker je celoten postopek sestavljen iz več medsebojno odvisnih korakov, se v naslednjih podpoglavjih osredotočimo na bolj podroben pregled posameznih delov. Najprej pogledamo notranje delovanje omrežja orakljev in agregacije rezultatov, nato pa še na branje podatkov iz registra zaupanja.

3.3.2 Agregiranje rezultatov v omrežju orakljev

Na sliki 3.3 je prikazan podroben potek obdelave poročil znotraj decentraliziranega omrežja orakljev (DON). Diagram prikazuje fazo po tem, ko oraklji prejmejo zahtevo za izračun zaupanja, in se osredotoča na izvajanje izračuna, podpisovanje poročil ter postopek agregacije v agregatorju. Ta proces je ključen za zagotavljanje verodostojnih rezultatov, saj združi poročila več neodvisnih orakljev v en sam, konsenzni izračun.



Slika 3.3: Potek agregiranja rezultatov v decentraliziranem omrežju orakljev

V nadaljevanju so opisani vsi ključni koraki, prikazani v diagramu.

1. Zajem podatkov iz plasti znanja

Orakelj pred izvedbo izračuna pridobi aktualne podatke iz dveh virov:

- ontološki podatki prek strežnika Fuseki (SPARQL poizvedbe ali lokalni OWL),

- preverljiva dokazila (VC), ki jih orakelj preveri glede veljavnosti in morebitnega preklica.

Ti podatki predstavljajo vhodne metrike, ki se uporabijo pri izračunu zaupanja.

2. Izvedba lokalnega izračuna zaupanja

Orakelj izvede hibridni algoritem ocenjevanja, ki združuje preverjanje pravil ter statistične metrike. V tej fazi orakelj pripravi tudi oba ključna povzetka:

- `policy_hash`, ki predstavlja zgoščeno vrednost pravil akterja,
- `credential_hash`, ki označuje VC dokazilo uporabljeno pri izračunu.

3. Pošiljanje podpisanega poročila agregatorju

Orakelj oblikuje strukturirano poročilo, ga serializira v kanonični JSON zapis in digitalno podpiše z metodo `sign_message` (EIP-191). Podpisano poročilo nato pošlje agregatorju prek zahteve HTTP (POST `/reports`), skupaj z identifikatorjem zahteve. Da agregator poročilo sprejme, mora orakelj poslati podatke v standardizirani obliki `SignedOracleReport`. Ta vsebuje dva dela:

- `OracleReport` – dejanski rezultat izračuna,
- `node_id` in `signature` – identiteto oraklja in kriptografski podpis poročila.

Struktura `OracleReport` mora vsebovati naslednja polja:

- `subject` – 32-bajtni identifikator subjekta (zgoščena vrednost DID),
- `decision` – končna odločitev (bool),
- `score` – številčna ocena zaupanja v odstotkih,
- `flags` – bitna opozorilna polja,
- `as_of` – čas izračuna.

Pred podpisom se poročilo pretvori v kanonično JSON obliko s stabilnim urejanjem ključev. Podpis se izvede z metodo `sign_message` (EIP-191) iz knjižnice `web3.py`. Agregator poročilo sprejme le, če je podpis veljaven, naslov `node_id` pripada dovoljenemu oraklju, poročilo ni podvojeno in `policy_hash` ter `credential_hash` se ujemata z lokalnimi podatki agregatorja. Standardizirana struktura omogoča, da lahko vsako poročilo neodvisno preverimo in da se rezultati več orakljev združijo v konsenzni izračun, ki se zapiše v pametno pogodbo.

4. Preverjanje podpisov in veljavnosti poročil

Agregator ob prejemu poročila preveri:

- ali podpis pripada dovoljenemu oraklju,
- ali je poročilo unikatno glede na kombinacijo `request_id` in `node_id`,
- ali se zgoščene vrednosti pravil in dokazil ujemajo z njegovimi lokalnimi podatki.

Veljavna poročila agregator takoj zapiše v pametno pogodbo s funkcijo `recordOracleSubmission`.

5. Doseganje kvoruma poročil

Agregator hrani prejeta poročila in spremlja, ali je dosežen kvorum (minimalno število poročil, potrebnih za izvedbo konsenza). Če kvorum ni dosežen, agregator čaka in nadaljuje zbiranje poročil dokler le ta ni dosežen ali do izteka časa čakanja (`ttl`).

6. Agregacija rezultatov znotraj DON

Ko je kvorum dosežen, agregator izvede agregacijo. Poleg odločitve in ocene agregator združi tudi `flags`, ki predstavljajo bitna opozorilna polja. Vsak bit ima posebej določen pomen, ki označuje posebne okoliščine ali tveganja, ki so se pojavila med izračunom:

- `FLAG_NO_DATA` (bit 0) – orakelj ni našel nobenih vhodnih podatkov za subjekt (npr. subjekt v ontologiji ne obstaja ali nima

nobenih lastnosti).

- FLAG_LOW_SCORE (bit 1) – verjetnostna ocena zaupanja je padla pod minimalni prag.
- FLAG_DISAGREEMENT (bit 2) – različni deli orakljevega notranjega izračuna so podali neskladne odločitve.
- FLAG_VC_REVOKED (bit 3) – preverljivo dokazilo (VC), uporabljeno pri izračunu, je označeno kot preklicano.

Ti biti se lahko nastavijo v orakljih funkcijah za izračun. Agregator pri združevanju uporabi *bitno večino*: vsak bit v končnem rezultatu je nastavljen, če ga ima vsaj polovica oddanih poročil. Zastavice tako predstavljajo povzetek opozoril, ki jih je zaznalo celotno omrežje orakljev.

7. Zapis konsenznega rezultata v pametno pogodbo

Končni agregirani rezultat se v verigo blokov zapiše s funkcijo:

```
fulfillTrustReport(requestId, report)
```

Pametna pogodba preveri identiteto klicatelja, veljavnost identifikatorja zahteve in rok izvedbe. Ob uspešnem zapisu sproži dogodek `TrustOracleFulfilled`, ki drugim komponentam signalizira, da je nova ocena na voljo.

Agregator povezuje izračune več orakljev v enoten rezultat, pametna pogodba pa poskrbi, da je končni izračun trajno in preverljivo zapisan v verigi blokov. V naslednjem podpoglavju si bomo ogledali, kako akterji v sistemu pridobijo te rezultate in kako poteka branje podatkov iz registra zaupanja.

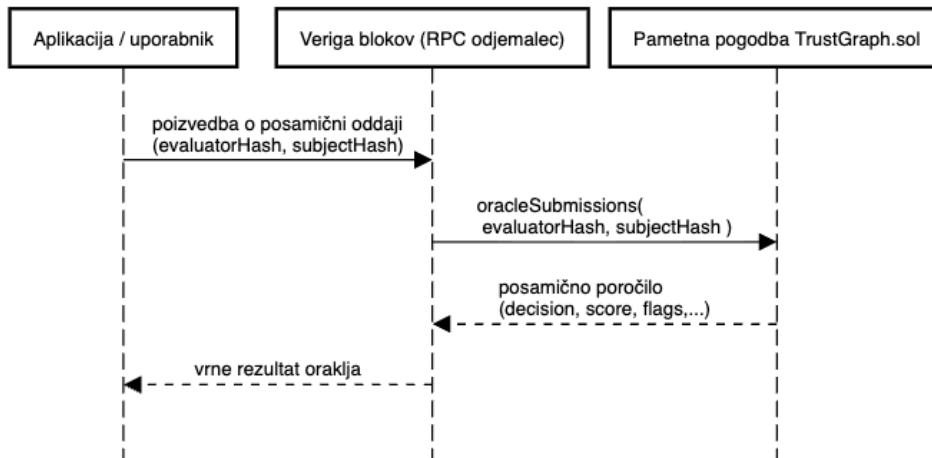
3.3.3 Branje rezultatov zaupanja

Po zaključenem izračunu in zapisu rezultatov v pametno pogodbo lahko akterji iz sistema pridobijo podatke o zaupanju na dva načina: (i) kot posame-

zne oddaje orakljev in (ii) kot agregiran, konsenzni rezultat omrežja orakljev (DON).

Branje posamičnih izračunov orakljev

Pametna pogodba `TrustGraph.sol` hrani posamične rezultate vsakega oraklja v podatkovni strukturi `oracleSubmissions[evaluator][subject]`. Ti podatki predstavljajo neagregirane rezultate, ki jih je posamezen orakelj izračunal za dani subjekt. Slika 3.4 prikazuje potek branja posamičnih oddaj.



Slika 3.4: Branje posamične oddaje oraklja iz registra zaupanja

Branje posamičnih oddaj je koristno za različne namene. Če je potrebno preveriti, kako je določen orakelj ocenil subjekt, ali pa analizirati morebitna neskladja med oraklji, lahko akterji neposredno dostopajo do teh podatkov. Do teh podatkov dostopamo z direktnim klicem funkcije za branje iz javnega slovarja:

```
oracleSubmissions(evaluatorHash, subject)
```

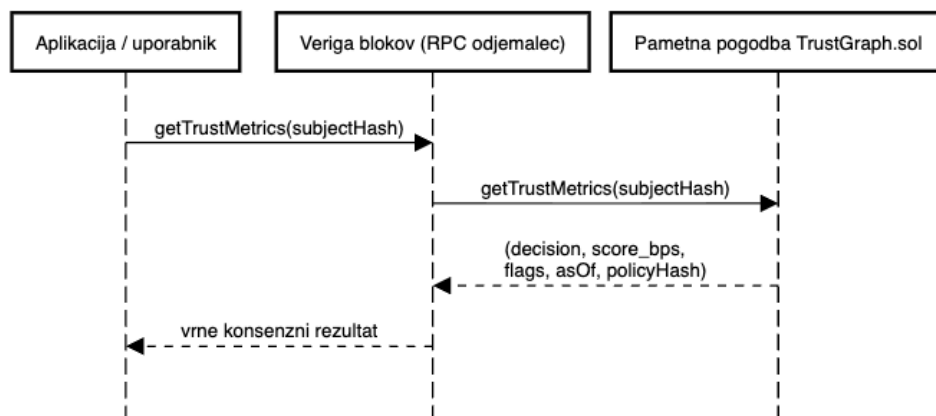
Rezultat vključuje:

- binarno odločitev,
- numerično oceno,

- zastavice (flags),
- čas izračuna,
- zgoščene vrednosti pravil in dokazil,
- ETH naslov oraklja.

Branje agregiranega (konsenznega) rezultata

Konsenzni rezultat, ki ga zapiše agregator, predstavlja končno in uradno odločitev omrežja orakljev.



Slika 3.5: Branje konsenznega rezultata ocenjevanja iz pametne pogodbe

Ta rezultat je v pametni pogodbi zapisan v strukturi:

```
subjectMetrics[subject]
```

Kot vidimo na 3.5, lahko do rezultatov dostopamo prek javne funkcije:

```
getTrustMetrics(subject)
```

Ta funkcija vrne:

- **odločitev** (decision) – končni izračun zaupanja,
- **oceno** (score_bps) – numerična vrednost v osnovnih točkah,

- **zastavice** (`flags`) – bitna opozorila, ki so bila prisotna v večini oddaj,
- **čas** (`asOf`) – čas izračuna konsenza.

Konsenzni rezultat je tisti, ki ga uporabi večina akterjev v sistemu, saj predstavlja stabilen in preverjen izračun celotnega omrežja orakljev.

Poglavje 4

Evalvacija

4.1 Predstavitev scenarijev delovanja

4.1.1 Eksperimentalno okolje in konfiguracija sistema

Preden predstavimo konkretne scenarije uporabe, najprej opišemo, kako je postavljen eksperimentalni sistem. Vse komponente zaganjamo z orodjem **Docker**, kar nam zagotavlja ponovljivo okolje in jasno ločitev med posameznimi storitvami. S tem zmanjšamo odvisnost od lokalne konfiguracije in omogočimo, da sistem testiramo kakor, da bi bile vse komponente že nameščene v oblaku.

Vsebniki in omrežje

Sistem sestavlja več vsebnikov **Docker**, ki so povezani v skupno virtualno omrežje in delujejo z orodjem **Docker Compose**. V nadaljevanju so opisani posamezni vsebniki in njihove vloge.

Veriga blokov Prvi vsebnik vsebuje testno Ethereum omrežje s pomočjo orodja **Anvil**. Na njem namestimo pametno pogodbo **TrustGraph.sol**. Na omrežju takoj dobimo tudi 10 prednastavljenih računov z velikimi količinami testnih ETH kovancev, ki jih oraklji in agregator uporabljajo za plačevanje

transakcijskih stroškov. Zato prve račune dodelimo agregatorju in orakljem. Potrebujemo tudi en račun za namestitev pametne pogodbe dodelimo le tega regulatorju EMA, kateri bo imel potem tudi pravico do sprožanja novih izračunov zaupanja.

Baza znanja V vsebniku Fuseki je naložena domenska ontologija. Ta je dostopna preko SPARQL končne točke, ki jo oraklji uporabljajo za pridobivanje semantičnih lastnosti akterjev. V eksperimentalnem okolju uporabljamo ontologijo, ki ga vsi oraklji delijo. Ontologija je predstavljena v prilogi A.

Oraklji Za evalvacijo smo pripravili tri neodvisne oraklje, ki izvajajo različne politike izračuna zaupanja. Vsak orakelj teče v svojem vsebniku Docker in je konfiguriran z ločeno datoteko, ki določa pravila ocenjevalca, nabor vhodnih podatkov ter težo posameznih kriterijev. Oraklji so implementirani v Pythonu in uporabljajo knjižnice za dostop do verige blokov (`web3.py`), baze znanja (`rdflib`, `SPARQLWrapper`) ter za kriptografske operacije.

Agregator , ki zbira poročila posameznih orakljev, preverja njihove podpise in objavi končno odločitev v pametni pogodbi na verigi.

Odjemalska aplikacija , ki v našem primeru nastopa kot ukazna vrstica oziroma skripta, s katero sprožimo zahtevo za izračun zaupanja in kasneje preberemo rezultat z verige. V realnem primeru bi bil to lahko npr. spletni vmesnik ali integracija v obstoječi ERP sistem akterja.

Kontejnerji si delijo notranje Docker omrežje, zato med seboj komunicirajo preko stabilnih DNS imen (npr. `fuseki:3030`, `ethereum-node:8545`).

Konfiguracija orakljev Vsi oraklji uporabljajo isti paket knjižnic za dostop do baze znanja, verige blokov in za kriptografske operacije, razlikujejo pa se v konfiguraciji in politikah izračuna zaupanja. Vsak orakelj ima svojo konfiguracijsko datoteko, v kateri so zapisani:

- referenca na *skupno bazo znanja* (isti Fuseki dataset),

- nabor *poverilnic in dokazov* (VC), ki jih pri izračunu obravnava kot vhod,
- *pravila in uteži* za posamezne kriterije zaupanja.

S tem modeliramo, da v omrežju sodelujejo neodvisni evalvatorji, ki uporabljajo različne poglede na iste podatke. Na primer, v tipični konfiguraciji:

- prvi orakelj poudarja regulatorni vidik in daje večjo težo veljavnim certifikatom in regulatornim odločbam,
- drugi orakelj se osredotoča na operativno zanesljivost, kjer imajo večjo težo incidenti pri transportu, temperaturna odstopanja in zanesljivost IoT senzorjev,
- tretji orakelj kombinira zgodovino sodelovanja med akterji in pretekle odločitve o zaupanju ter tako deluje bolj reputacijsko usmerjeno.

Agregator pri združevanju rezultatov ne pozna podrobnosti posameznih politik. Vsako poročilo obravnava kot podpisano izjavo oraklja, ki vsebuje identifikator subjekta, numerično oceno zaupanja, diskretno odločitev (npr. *zaupanja vreden* ali *nezaupanja vreden*) ter morebitne zastavice. Na osnovi konfiguriranega praga in števila prejetih poročil izračuna končno odločitev, ki jo zapiše v pametno pogodbo.

Povzetek konfiguracije. Tabela 4.1 povzema ključne parametre eksperimentalne postavitve, ki jih v nadaljevanju uporabljamo v obeh scenarijih.

Tabela 4.1: Povzetek eksperimentalne konfiguracije sistema.

Parameter	Vrednost
Okolje verige blokov	lokalni testnet (Docker kontejner)
Naslov pametne pogodbe <code>TrustGraph.sol</code>	<code>0x...</code> (primer v prilogi)
Baza znanja	Apache Jena Fuseki, en dataset
Število orakljev	3 (različne politike zaupanja)
Agregacija poročil	prag soglasja (npr. večina)
Odjemalski vmesnik	Python skripta / ukazna vrstica

4.1.2 Algoritmi izračuna zaupanja v orakljih

V poglavju 2.2 smo predstavili različne pristope k upravljanju zaupanja v razpršenih sistemih: dokazne modele, priporočilne modele, porazdeljeno ocenjevanje ter dinamično posodabljanje zaupanja. Ti koncepti tvorijo teoretično podlago za zasnovo naših orakljev. V nadaljevanju pokažemo, kako vsak orakelj uteleša drugačen pogled na zaupanje:

- Pfizerjev orakelj uporablja hibridni pristop, ki kombinira dokazne elemente (licence, certifikati) z dinamično metriko, podobno priporočilnim in porazdeljenim modelom.
- Orakelj Moderna je izrazito *dokazni* – odloča skoraj izključno na podlagi verifiable credentials, podobno kot hierarhični in PGP-sistemi iz podpoglavja o dokaznih modelih.
- Orakelj DHL je *telemetrijsko usmerjen* in se osredotoča na dinamično posodabljanje zaupanja preko operativnih metrik, kar je skladno z modeli, ki večjo težo dajejo nedavnim interakcijam.

Skupna struktura izračuna

Ne glede na specifičen pogled vsak orakelj za dani subjekt izračuna:

1. deterministični del (pravila iz politik),

2. probabilistični del (številčna ocena),
3. kombinirano odločitev.

Deterministični del Deterministični del temelji na politikah, ki so zapisane v konfiguracijskih datotekah (`policy.json`). V njih so za posamezne tipe subjektov (proizvajalec, distributer, transporter) definirani kriteriji in pragovi, npr.

- `hasDeliveryPunctuality` $\geq 0,99$,
- `hasTempViolationRate` $\leq 0,05$,
- `hasAuditScore` $\geq 0,90$,
- `hasGDPVC` = `true`.

Če subjekt za vse zahtevane kriterije zadošča pogojem, deterministični del vrne odločitev *PASS*, sicer *FAIL*. Ta del je najbližje *dokaznim modelom zaupanja*, kjer se odločitve sprejemajo na podlagi preverljivih dokazov in trdih pragov.

Probabilistični del Probabilistični del pretvori iste kriterije v numerični score v intervalu $[0, 1]$, ki ga orakelj kasneje pretvori v basis points (0–10000). Za vsak kriterij se vodi Beta posterior, ki se posodablja po principu eksponentno uteženega drsečega povprečja (EWMA). S tem približamo obnašanje modelom iz podpoglavij o priporočilnih modelih in dinamičnem posodabljanju zaupanja.

Pri prvem branju ontologije se kriterij inicializira na osnovi trenutne vrednosti x in normalizirane zaželenosti $s_0 \in [0, 1]$ (višja vrednost pomeni bolj zaželen izid). Parametra porazdelitve Beta sta:

$$\alpha = 1 + s_0 \cdot \text{obs_scale}, \quad \beta = 1 + (1 - s_0) \cdot \text{obs_scale},$$

kjer je v implementaciji uporabljena lestvica $\text{obs_scale} = 9$. Za »popolno« izpolnjen pogoj ($s_0 \approx 1$) dobimo pričakovano vrednost Beta porazdelitve $\mathbb{E}[X] \approx 0,91$, kar predstavlja visoko, a ne absolutno zaupanje.

Ob novih opazovanjih se posterior posodablja z EWMA faktorjem $\gamma \approx 0,2$, kar zagotavlja, da nove informacije hitreje vplivajo na score, stare pa postopoma izgubijo težo. To je konceptualno podobno formuli za dinamično posodabljanje zaupanja v podpoglavju o amortizacijskih faktorjih, kjer nove interakcije s časom prevladajo nad starimi.

Kombinirana odločitev Za posamezen subjekt se probabilistični score izračuna kot uteženo povprečje srednjih vrednosti Beta posteriorjev aktivnih kriterijev. Uteži so zapisane v polju `_weights` v politiki in se normalizirajo tako, da zadošča:

$$\sum_i w_i = 1, \quad w_i \geq 0.$$

Vsak orakelj uporablja svoj prag *prob.threshold*, ki ga izražamo na lestvici 0–1 (v implementaciji pa v basis points). Kombinirana odločitev ima obliko:

$$combined = deterministic \text{ OR } (prob_score \geq prob_threshold),$$

kjer je *deterministic* deterministična odločitev (*PASS/FAIL*), *prob.score* pa izračunani probabilistični score. Na ta način se trda pragovna pravila (dokazni pogled) kombinirajo z mehko, numerično oceno, ki je bližje priporočilnim in porazdeljenim modelom zaupanja.

V nadaljevanju opišemo konkretne konfiguracije in primere za tri oraklje, ki utelešajo različne pristope iz poglavja 2.2.

Pfizer orakelj (hibridni pogled)

Pfizerjev orakelj uporablja hibridni pogled, ki združuje regulatorne kriterije, kakovost delovanja in informacije iz verifiable credentials. S tem v praksi kombinira dokazni model (digitalna dokazila, licence) z dinamičnim ocenjevanjem in uteženjem kriterijev preko Beta posteriorjev. Njegov prag za probabilistični score je nastavljen na:

$$prob_threshold^{(Pfizer)} = 0,75.$$

Kriteriji in uteži. Za posamezne tipe subjektov uporablja naslednje kriterije in uteži:

- **Transporter:**

- `hasDeliveryPunctuality` $\geq 0,99$ (utež 0,7),
- `hasTempViolationRate` $\leq 0,05$ (utež 0,3).

- **Distributor:**

- `hasLicense` = `true` (utež 0,4),
- `hasGDPVC` = `true` (utež 0,5).

- **Manufacturer:**

- `hasAuditScore` $\geq 0,90$ (utež 0,6).

Primer: MedLogix (distributer). Za distributerja MedLogix v trenutni ontologiji velja `license=true` in `GDPVC=true`. Deterministični del zato vrne *PASS*. Oba kriterija imata visoko zaželenost, zato se Beta posteriorja inicializirata z $\text{mean} \approx 0,91$. Po normalizaciji uteži (približno 0,44 in 0,56) dobimo probabilistični score:

$$\text{prob_score} \approx 0,91 \geq 0,75.$$

Kombinirana odločitev je:

$$\text{combined}^{(\text{Pfizer}, \text{MedLogix})} = \text{PASS}.$$

Primer: SkyFreight (transporter). Za transporterja SkyFreight sta za beleženi vrednosti `punctuality` = 0,90 in `tempViolationRate` = 0,08. Deterministični pogoji niso izpolnjeni, ker $0,90 < 0,99$ in $0,08 > 0,05$, zato deterministični del vrne *FAIL*. Tudi normalizirana zaželenost s_0 za oba kriterija je nizka, kar vodi v nizek Beta mean in probabilistični score $\text{prob_score} < 0,75$. Končna odločitev oraklja je:

$$\text{combined}^{(\text{Pfizer}, \text{SkyFreight})} = \text{FAIL}.$$

Moderna orakelj (VC-centričen pogled)

Modernin orakelj je zasnovan tako, da v ospredje postavlja verifiable credentials. Konceptualno je najbližje dokaznim modelom zaupanja iz poglavja 2.2, kjer imata veriga potrdil in kriptografska dokazila ključno vlogo pri inicializaciji zaupanja. Njegov prag za probabilistični score je strožji:

$$prob_threshold^{(Moderna)} = 0,8.$$

Kriteriji in uteži. Glavne kombinacije kriterijev so:

- **Distributor:**

- `hasGDPVC = true` (utež 0,6),
- `hasLicense = true` (utež 0,3),

kar po normalizaciji daje približno 0,67 in 0,33.

- **Manufacturer:**

- `hasAuditScore ≥ 0,92` (utež 0,4).

- **Transporter:**

- `hasRecallRate ≤ 0,02` (utež 0,5).

Ključna značilnost je, da brez ustreznih VC odločitev nagiba k *FAIL*, tudi če so druge operative metrike razmeroma dobre, kar odraža močan poudarek na formalnih dokazih in skladnosti.

Primer: MedLogix (distributor). Ker ima MedLogix veljavne VC in licenco, deterministični del vrne *PASS*. Za oba kriterija dobimo Beta mean $\approx 0,91$, uteži so po normalizaciji 0,67 in 0,33, zato:

$$prob_score \approx 0,91 \geq 0,8,$$

in kombinirana odločitev ostane:

$$combined^{(Moderna, MedLogix)} = PASS.$$

Primer: HealthChain (transporter). Za transporterja HealthChain je v bazi znanja zabeležen `recallRate` = 0,015, kar deterministično zadošča pogoju `hasRecallRate` ≤ 0,02 in daje *PASS*. Pri probabilističnem delu pa je zaželenost pri vrednosti blizu zgornje meje nizka (npr. $s_0 \approx 0,25$), kar vodi v Beta mean približno 0,30 in:

$$prob_score \approx 0,30 < 0,8.$$

Kombinirana odločitev je zato:

$$combined^{(Moderna, HealthChain)} = FAIL,$$

kar ilustrira strožji, VC-centričen pristop: deterministični *PASS* sam po sebi ni zadosten brez visoke probabilistične ocene.

DHL orakelj (telemetrija-prvi pogled)

DHL-jev orakelj uporablja telemetrijo transporta (časovne serije, agregirane metrike) kot primarni vir informacij. Konceptualno se približa porazdeljenim in dinamičnim modelom zaupanja iz poglavja 2.2, kjer se zaupanje gradi na osnovi kumulativnih izkušenj in se sproti prilagaja novim dogodkom. Njegov prag za probabilistični score je:

$$prob_threshold^{(DHL)} = 0,7.$$

Kriteriji in uteži. Za proizvajalce in distributerje uporablja naslednje kriterije:

- **Manufacturer:**

- `hasTempDeviationScore` ≤ 0,10 (utež 0,5),
- `hasGDPVC` = `true` (utež 0,4).

- **Distributor:**

- `hasDeliveryPunctuality` ≥ 0,95 (utež 0,5).

Pri transporterjih ta orakelj v trenutni konfiguraciji ne izračunava ocene, saj zanje ni definiranih kriterijev; v takem primeru subjekt v tem pogledu ni ocenjen.

Primer: BioPharm (proizvajalec). Za proizvajalca BioPharm sta zabeležena `tempDev = 0,08` in `GDPVC=true`. Oba pogoja sta deterministično izpolnjena (*PASS*), zaželenost je visoka in oba kriterija dobila Beta mean $\approx 0,91$. Probabilistični score je zato:

$$prob_score \approx 0,91 \geq 0,7,$$

kombinirana odločitev pa:

$$combined^{(DHL, BioPharm)} = PASS.$$

Primer: CanaryLabs (proizvajalec). Za proizvajalca CanaryLabs sta zabeležena `tempDev = 0,12` in `GDPVC=false`. Deterministični del zato vrne *FAIL*. Ker sta temperaturna odstopanja prevelika in ni VC o skladnosti, je tudi probabilistični score nizek ($prob_score \ll 0,7$). Kombinirana odločitev je:

$$combined^{(DHL, CanaryLabs)} = FAIL.$$

Takšna predstavitev treh orakljev neposredno gradi na modelih iz poglavja 2.2: Pfizerjev orakelj uteleša hibridni, semantično in dinamično podprt pristop, Moderna orakelj predstavlja strogo dokazno-kriptografski pogled, DHL orakelj pa dinamično, na meritvah temelječe ocenjevanje. V scenarijih, ki sledijo, bomo za izbranega subjekta pokazali, kako različni algoritmi privedejo do različnih, a razložljivih odločitev o zaupanju.

4.1.3 Scenarij 1: Ocena zaupanja akterja

V prvem scenariju si bomo pogledali, kako v sistemu pridobimo nove vrednosti zaupanja za določenega akterja. Predpostavimo, da želimo posodobiti oceno zaupanja za transportno podjetje DHL, ki sodeluje v naši dobavni verigi.

4.2 Varnostna analiza

4.3 Zmogljivost in razširljivost

4.4 Prednosti in omejitve

4.5 Primerjava z obstoječimi rešitvami

Poglavje 5

Zaključek in nadaljnje delo

5.1 Povzetek prispevkov

5.2 Nadaljnje raziskave

Dodatek A

OWL ontologija dobavne verige

Listing A.1: OWL ontologija za model zaupanja v farmacevtski verigi

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://example.org/trust#"
  xml:base="http://example.org/trust"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xml="http://www.w3.org/XML/1998/namespace"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:trust="http://example.org/trust#">
  <owl:Ontology rdf:about="http://example.org/trust"/>
  <owl:AnnotationProperty rdf:about="http://example.org/trust#trustedPartner"/>
  <owl:DatatypeProperty rdf:about="http://example.org/trust#hasAuditScore">
    <rdfs:domain rdf:resource="http://example.org/trust#Actor"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
```

```

</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://example.org/
  trust#hasDID">
  <rdfs:domain rdf:resource="http://example.org/trust
    #Actor" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/
    XMLSchema#string" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://example.org/
  trust#hasDeliveryDelayRate">
  <rdfs:domain rdf:resource="http://example.org/trust
    #Distributor" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/
    XMLSchema#float" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://example.org/
  trust#hasDeliveryPunctuality">
  <rdfs:domain rdf:resource="http://example.org/trust
    #Transporter" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/
    XMLSchema#float" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://example.org/
  trust#hasGDPVC">
  <rdfs:domain rdf:resource="http://example.org/trust
    #Distributor" />
  <rdfs:domain rdf:resource="http://example.org/trust
    #Transporter" />
  <rdfs:range rdf:resource="http://www.w3.org/2001/
    XMLSchema#boolean" />
</owl:DatatypeProperty>

```

```

<owl:DatatypeProperty rdf:about="http://example.org/
  trust#hasGMP">
  <rdfs:domain rdf:resource="http://example.org/trust
    #Manufacturer"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
    XMLSchema#boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://example.org/
  trust#hasIssuedCertifications">
  <rdfs:domain rdf:resource="http://example.org/trust
    #RegulatoryAuthority"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
    XMLSchema#integer"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://example.org/
  trust#hasJurisdictionLevel">
  <rdfs:domain rdf:resource="http://example.org/trust
    #RegulatoryAuthority"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
    XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://example.org/
  trust#hasLicense">
  <rdfs:domain rdf:resource="http://example.org/trust
    #Actor"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
    XMLSchema#boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://example.org/
  trust#hasPrescriptionComplianceRate">

```

```

    <rdfs:domain rdfs:resource=" http://example.org/trust
      #Pharmacy" />
    <rdfs:range rdfs:resource=" http://www.w3.org/2001/
      XMLSchema#float" />
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdfs:about=" http://example.org/
    trust#hasRecallRate">
    <rdfs:domain rdfs:resource=" http://example.org/trust
      #Transporter" />
    <rdfs:range rdfs:resource=" http://www.w3.org/2001/
      XMLSchema#float" />
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdfs:about=" http://example.org/
    trust#hasTempDeviationScore">
    <rdfs:domain rdfs:resource=" http://example.org/trust
      #Manufacturer" />
    <rdfs:range rdfs:resource=" http://www.w3.org/2001/
      XMLSchema#float" />
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdfs:about=" http://example.org/
    trust#hasTempViolationRate">
    <rdfs:domain rdfs:resource=" http://example.org/trust
      #Transporter" />
    <rdfs:range rdfs:resource=" http://www.w3.org/2001/
      XMLSchema#float" />
  </owl:DatatypeProperty>
  <owl:Class rdfs:about=" http://example.org/trust#Actor"
    />
  <owl:Class rdfs:about=" http://example.org/trust#
    Distributor">

```

```

    <rdfs:subClassOf rdf:resource=" http://example.org/
        trust#Actor" />
</owl:Class>
<owl:Class rdf:about=" http://example.org/trust#
    Manufacturer">
    <rdfs:subClassOf rdf:resource=" http://example.org/
        trust#Actor" />
</owl:Class>
<owl:Class rdf:about=" http://example.org/trust#
    Pharmacy">
    <rdfs:subClassOf rdf:resource=" http://example.org/
        trust#Actor" />
</owl:Class>
<owl:Class rdf:about=" http://example.org/trust#
    RegulatoryAuthority">
    <rdfs:subClassOf rdf:resource=" http://example.org/
        trust#Actor" />
</owl:Class>
<owl:Class rdf:about=" http://example.org/trust#
    Transporter">
    <rdfs:subClassOf rdf:resource=" http://example.org/
        trust#Actor" />
</owl:Class>
<owl:Class rdf:about=" http://example.org/trust#
    TrustedDistributor">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType=" Collection">
                <owl:Restriction>
                    <owl:onProperty rdf:resource=" http://
                        example.org/trust#hasDeliveryDelayRate" /

```

```

    >
    <owl:someValuesFrom rdf:resource="http://
      www.w3.org/2001/XMLSchema#float" />
  </owl:Restriction>
  <owl:Restriction>
    <owl:onProperty rdf:resource="http://
      example.org/trust#hasLicense" />
    <owl:hasValue rdf:datatype="http://www.w3.
      org/2001/XMLSchema#boolean">true</
      owl:hasValue>
  </owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://example.org/
  trust#Distributor" />
</owl:Class>
<owl:Class rdf:about="http://example.org/trust#
  TrustedManufacturer">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://
            example.org/trust#hasAuditScore" />
          <owl:someValuesFrom rdf:resource="http://
            www.w3.org/2001/XMLSchema#float" />
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://
            example.org/trust#hasGMP" />

```

```

        <owl:hasValue rdf:datatype=" http://www.w3.
            org/2001/XMLSchema#boolean">true</
            owl:hasValue>
    </owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource=" http://example.org/
    trust#Manufacturer" />
</owl:Class>
<owl:Class rdf:about=" http://example.org/trust#
    TrustedPharmacy">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType=" Collection">
                <owl:Restriction>
                    <owl:onProperty rdf:resource=" http://
                        example.org/trust#
                            hasPrescriptionComplianceRate" />
                    <owl:someValuesFrom rdf:resource=" http://
                        www.w3.org/2001/XMLSchema#float" />
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource=" http://
                        example.org/trust#hasLicense" />
                    <owl:hasValue rdf:datatype=" http://www.w3.
                        org/2001/XMLSchema#boolean">true</
                        owl:hasValue>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>

```

```

    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="http://example.org/
      trust#Pharmacy" />
  </owl:Class>
  <owl:Class rdf:about="http://example.org/trust#
    TrustedRegulatoryAuthority">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="http://
              example.org/trust#
                hasIssuedCertifications" />
            <owl:someValuesFrom rdf:resource="http://
              www.w3.org/2001/XMLSchema#integer" />
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="http://
              example.org/trust#hasJurisdictionLevel" /
            >
            <owl:someValuesFrom rdf:resource="http://
              www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="http://example.org/
      trust#RegulatoryAuthority" />
  </owl:Class>
  <owl:Class rdf:about="http://example.org/trust#
    TrustedTransporter">

```

```

<owl:equivalentClass>
  <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://
          example.org/trust#hasDeliveryPunctuality
        "/>
        <owl:someValuesFrom rdf:resource="http://
          www.w3.org/2001/XMLSchema#float"/>
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://
          example.org/trust#hasTempViolationRate"/
        >
        <owl:someValuesFrom rdf:resource="http://
          www.w3.org/2001/XMLSchema#float"/>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="http://example.org/
  trust#Transporter"/>
</owl:Class>
<owl:NamedIndividual rdf:about="http://example.org/
  trust#BioPharm">
  <rdf:type rdf:resource="http://example.org/trust#
    Manufacturer"/>
  <hasAuditScore rdf:datatype="http://www.w3.org
    /2001/XMLSchema#float">0.9</hasAuditScore>
  <hasDID>did:key:z6MkbioPharm987654321cdefg</hasDID>

```

```

    <hasGDPVC rdf:datatype="http://www.w3.org/2001/
      XMLSchema#boolean">true</hasGDPVC>
    <hasGMP rdf:datatype="http://www.w3.org/2001/
      XMLSchema#boolean">true</hasGMP>
    <hasTempDeviationScore rdf:datatype="http://www.w3.
      org/2001/XMLSchema#float">0.08</
      hasTempDeviationScore>
  </owl:NamedIndividual>
  <owl:NamedIndividual rdf:about="http://example.org/
    trust#CanaryLabs">
    <rdf:type rdf:resource="http://example.org/trust#
      Manufacturer"/>
    <hasAuditScore rdf:datatype="http://www.w3.org
      /2001/XMLSchema#float">0.81</hasAuditScore>
    <hasDID>did:key:z6Mkcanarylabs13579bdfhjlnp</hasDID
      >
    <hasGDPVC rdf:datatype="http://www.w3.org/2001/
      XMLSchema#boolean">false</hasGDPVC>
    <hasGMP rdf:datatype="http://www.w3.org/2001/
      XMLSchema#boolean">true</hasGMP>
    <hasTempDeviationScore rdf:datatype="http://www.w3.
      org/2001/XMLSchema#float">0.12</
      hasTempDeviationScore>
  </owl:NamedIndividual>
  <owl:NamedIndividual rdf:about="http://example.org/
    trust#DHL">
    <rdf:type rdf:resource="http://example.org/trust#
      Transporter"/>
    <hasDID>
      did:key:z6MkuruPZNP71MGVetJzof487d83iKXJgwK87Z1rDiAD9q6s
    </hasDID>

```

```

    <hasDeliveryPunctuality rdf:datatype=" http://www.w3
      .org/2001/XMLSchema#decimal">0.94</
      hasDeliveryPunctuality>
    <hasTempViolationRate rdf:datatype=" http://www.w3.
      org/2001/XMLSchema#float">0.02</
      hasTempViolationRate>
  </owl:NamedIndividual>
  <owl:NamedIndividual rdf:about=" http://example.org/
    trust#EMA">
    <rdf:type rdf:resource=" http://example.org/trust#
      RegulatoryAuthority"/>
    <hasDID>
      did:key:z6Mko5EueZnoZDAJVN MokNDLyPEsrwbrm4xobPjgqtDc6gur
    </hasDID>
    <hasIssuedCertifications rdf:datatype=" http://www.
      w3.org/2001/XMLSchema#integer">32</
      hasIssuedCertifications>
    <hasJurisdictionLevel>global</hasJurisdictionLevel>
  </owl:NamedIndividual>
  <owl:NamedIndividual rdf:about=" http://example.org/
    trust#EuroLogistics">
    <rdf:type rdf:resource=" http://example.org/trust#
      Distributor"/>
    <hasDID>
      did:key:z6MkmZ92uLrYfjM96ijWEQgv5FspCVAZwRjNwqw5zULcCxih
    </hasDID>
    <hasDeliveryDelayRate rdf:datatype=" http://www.w3.
      org/2001/XMLSchema#float">0.07</
      hasDeliveryDelayRate>
    <hasLicense rdf:datatype=" http://www.w3.org/2001/
      XMLSchema#boolean">true</hasLicense>

```

```

</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://example.org/
  trust#GreenPharmacy">
  <rdf:type rdf:resource="http://example.org/trust#
    Pharmacy" />
  <hasDID>did:key:z6Mkgreenpharm2468acegjlqs</hasDID>
  <hasGDPVC rdf:datatype="http://www.w3.org/2001/
    XMLSchema#boolean">true</hasGDPVC>
  <hasLicense rdf:datatype="http://www.w3.org/2001/
    XMLSchema#boolean">true</hasLicense>
  <hasPrescriptionComplianceRate rdf:datatype="http:
    //www.w3.org/2001/XMLSchema#float">0.94</
    hasPrescriptionComplianceRate>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://example.org/
  trust#HealthChain">
  <rdf:type rdf:resource="http://example.org/trust#
    Transporter" />
  <hasDID>did:key:z6Mkhealthchain123456789abc</hasDID>
  >
  <hasDeliveryPunctuality rdf:datatype="http://www.w3
    .org/2001/XMLSchema#float">0.97</
    hasDeliveryPunctuality>
  <hasGDPVC rdf:datatype="http://www.w3.org/2001/
    XMLSchema#boolean">true</hasGDPVC>
  <hasRecallRate rdf:datatype="http://www.w3.org
    /2001/XMLSchema#float">0.015</hasRecallRate>
  <hasTempViolationRate rdf:datatype="http://www.w3.
    org/2001/XMLSchema#float">0.03</
    hasTempViolationRate>
</owl:NamedIndividual>

```

```

<owl:NamedIndividual rdf:about="http://example.org/
  trust#MedLogix">
  <rdf:type rdf:resource="http://example.org/trust#
    Distributor"/>
  <hasDID>did:key:z6Mkmedlogix123456789abcde</hasDID>
  <hasDeliveryPunctuality rdf:datatype="http://www.w3
    .org/2001/XMLSchema#float">0.93</
    hasDeliveryPunctuality>
  <hasGDPVC rdf:datatype="http://www.w3.org/2001/
    XMLSchema#boolean">true</hasGDPVC>
  <hasLicense rdf:datatype="http://www.w3.org/2001/
    XMLSchema#boolean">true</hasLicense>
  <trustedPartner rdf:resource="http://example.org/
    trust#HealthChain"/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://example.org/
  trust#MediPlus">
  <rdf:type rdf:resource="http://example.org/trust#
    Pharmacy"/>
  <hasDID>
    did:key:z6MknrnXRoJwuj4LWF9rTpvH9WbEdXMbdHPRHdRiP1FraA9k
  </hasDID>
  <hasLicense rdf:datatype="http://www.w3.org/2001/
    XMLSchema#boolean">true</hasLicense>
  <hasPrescriptionComplianceRate rdf:datatype="http:
    //www.w3.org/2001/XMLSchema#float">0.93</
    hasPrescriptionComplianceRate>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://example.org/
  trust#Novartis">

```

```

<rdf:type rdf:resource="http://example.org/trust#
  Manufacturer"/>
<hasAuditScore rdf:datatype="http://www.w3.org
  /2001/XMLSchema#float">0.82</hasAuditScore>
<hasDID>
  did:key:z6MkqTxBfq3bDT1MHGZdGxyvjt3cJkZJsQ1ZD3zyZsAEY77q
</hasDID>
<hasGMP rdf:datatype="http://www.w3.org/2001/
  XMLSchema#boolean">true</hasGMP>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://example.org/
  trust#Pfizer">
<rdf:type rdf:resource="http://example.org/trust#
  Manufacturer"/>
<hasAuditScore rdf:datatype="http://www.w3.org
  /2001/XMLSchema#float">0.92</hasAuditScore>
<hasDID>
  did:key:z6MkqYUPZM9zRaA3YxZPh1RJLLDUrrV39Hpa7vLRRkvkxpZM
</hasDID>
<hasGDPVC rdf:datatype="http://www.w3.org/2001/
  XMLSchema#boolean">true</hasGDPVC>
<hasGMP rdf:datatype="http://www.w3.org/2001/
  XMLSchema#boolean">true</hasGMP>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://example.org/
  trust#SkyFreight">
<rdf:type rdf:resource="http://example.org/trust#
  Transporter"/>
<hasDID>did:key:z6Mkskyfreight112233445566</hasDID>
<hasDeliveryPunctuality rdf:datatype="http://www.w3
  .org/2001/XMLSchema#float">0.9</

```

```
      hasDeliveryPunctuality>
    <hasGDPVC rdf:datatype="http://www.w3.org/2001/
      XMLSchema#boolean">false</hasGDPVC>
    <hasRecallRate rdf:datatype="http://www.w3.org
      /2001/XMLSchema#float">0.035</hasRecallRate>
    <hasTempViolationRate rdf:datatype="http://www.w3.
      org/2001/XMLSchema#float">0.08</
      hasTempViolationRate>
  </owl:NamedIndividual>
</rdf:RDF>
```


Dodatek B

Izvlečki kode pametnih pogodb

Literatura

- [1] S. Holtmanns, Trust modeling and management: From social trust to digital trust, in: Computer Security, Privacy and Politics: Current Issues, Challenges and Solutions, IGI Global, 2008. doi:10.4018/978-1-59904-804-8.ch013.
- [2] T. Grandison, M. Sloman, A survey of trust in internet applications, IEEE Communications Surveys & Tutorials 3 (4) (2000) 2–16.
- [3] S. D. Boon, J. G. Holmes, The dynamics of interpersonal trust: Resolving uncertainty in the face of risk, Cooperation and Prosocial Behavior (1991) 167–182.
- [4] D. Gambetta, Trust: Making and breaking cooperative relations, Basil Blackwell, 1988.
- [5] R. C. Mayer, J. H. Davis, F. D. Schoorman, An integrative model of organizational trust, Academy of Management Review 20 (3) (1995) 709–734.
- [6] D. E. Denning, A new paradigm for trusted systems, Proceedings of the 1993 New Security Paradigms Workshop (1993) 36–41.
- [7] D. H. McKnight, N. L. Chervany, Trust and distrust definitions: One bite at a time, Trust in Cyber-societies 2246 (2001) 27–54.
- [8] S. Karthik, R. Shankar, N. Arunkumar, Ontology-based trust model for pervasive computing environments, Journal of Ambient Intel-

- ligence and Humanized Computing 8 (2017) 557–568. doi:10.1007/s12652-016-0442-y.
- [9] M. Uddin, K. Salah, R. Jayaraman, S. Pesic, S. Ellahham, Blockchain for drug traceability: Architectures and open challenges, *Health informatics journal* 27 (2) (2021) 14604582211011228.
- [10] H. Kayhan, Ensuring trust in pharmaceutical supply chains by data protection by design approach to blockchains, *Blockchain in Healthcare Today* 5 (2022) 10–30953.
- [11] H. Li, M. Singhal, Trust management in distributed systems, *Computer* 40 (2) (2007) 45–53. doi:10.1109/MC.2007.76.
- [12] R. Housley, W. Ford, W. Polk, D. Solo, Rfc2459: Internet x. 509 public key infrastructure certificate and crl profile (1999).
- [13] M. Elkins, D. Del Torto, R. Levien, T. Roessler, Mime security with openpgp, Tech. rep. (2001).
- [14] S. K. Panda, S. C. Satapathy, Drug traceability and transparency in medical supply chain using blockchain for easing the process and creating trust between stakeholders and consumers, *Personal and Ubiquitous Computing* 28 (1) (2024) 75–91.
- [15] W. H. Organization, et al., A study on the public health and socioeconomic impact of substandard and falsified medical products: executive summary (2017).
- [16] Council of European Union, Directive 2011/62/eu of the european parliament and of the council of 8 june 2011 amending directive 2001/83/ec on the community code relating to medicinal products for human use, as regards the prevention of the entry into the legal supply chain of falsified medicinal products text with eea relevance (2011).
URL <https://eur-lex.europa.eu/eli/dir/2011/62/oj/eng>

-
- [17] Council of European Union, Commission delegated regulation (eu) 2016/161 of 2 october 2015 supplementing directive 2001/83/ec of the european parliament and of the council by laying down detailed rules for the safety features appearing on the packaging of medicinal products for human use (text with eea relevance) (2015).
URL https://eur-lex.europa.eu/eli/reg_del/2016/161/oj/eng
- [18] U.S. Food and Drug Administration, Drug supply chain security act (dscsa) (2017).
URL <https://www.fda.gov/Drugs/DrugSafety/DrugIntegrityandSupplyChainSecurity/DrugSupplyChainSecurityAct/ucm382022.htm>
- [19] W. H. Organization, et al., WHO global surveillance and monitoring system for substandard and falsified medical products (2017).
- [20] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Available at SSRN 3440802 (2008).
- [21] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman, et al., Block-chain technology: Beyond bitcoin, Applied innovation 2 (6-10) (2016) 71.
- [22] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., Hyperledger fabric: a distributed operating system for permissioned blockchains, in: Proceedings of the thirteenth EuroSys conference, 2018, pp. 1–15.
- [23] G. Wood, et al., Ethereum: A secure decentralised generalised transaction ledger, Ethereum project yellow paper 151 (2014) (2014) 1–32.
- [24] G. Caldarelli, Understanding the blockchain oracle problem: A call for action, Information 11 (11) (2020) 509.

-
- [25] G. Caldarelli, Overview of blockchain oracle research, *Future Internet* 14 (6) (2022) 175.
- [26] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, A. Kastania, Astraea: A decentralized blockchain oracle, in: 2018 IEEE international conference on internet of things (IThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCoM) and IEEE smart data (SmartData), IEEE, 2018, pp. 1145–1152.
- [27] P. Kochovski, S. Gec, V. Stankovski, M. Bajec, P. D. Drobintsev, Trust management in a blockchain based fog computing platform with trustless smart oracles, *Future Generation Computer Systems* 101 (2019) 747–759.
- [28] D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, M. Sabadello, J. Holt, Decentralized identifiers (dids) v1. 0, Draft Community Group Report (2020).
- [29] M. Sporny, D. Longley, D. Chadwick, et al., Verifiable credentials data model 2.0, W3C Recommendation (2025).
URL <https://www.w3.org/TR/vc-data-model-2.0/>
- [30] C. Mazzocca, A. Acar, S. Uluagac, R. Montanari, P. Bellavista, M. Conti, A survey on decentralized identifiers and verifiable credentials, *IEEE Communications Surveys & Tutorials* (2025).
- [31] O. Lassila, J. Hendler, T. Berners-Lee, The semantic web, *Scientific american* 284 (5) (2001) 34–43.
- [32] R. Cyganiak, P. Reynolds, Rdf 1.1 concepts and abstract syntax, W3C Recommendation, (pridobljeno 10. 11. 2025) (2014).
URL <https://www.w3.org/TR/rdf11-concepts/>
- [33] W. W. W. Consortium, et al., Owl 2 web ontology language document overview (2012).

-
- [34] C. Boettiger, rdflib: A high level wrapper around the redland package for common rdf applications (2018). doi:10.5281/zenodo.1098478.
URL <https://doi.org/10.5281/zenodo.1098478>
- [35] J. D. Nally, Good manufacturing practices for pharmaceuticals, CRC Press, 2016.