Matej Urbas

Zagotavljanje zaupnosti osebnih podatkov v porazdeljenem sistemu s pomočjo varnih identifikatorjev

Diplomsko delo

Maribor, junij 2009

Diplomsko delo univerzitetnega študijskega programa

Zagotavljanje zaupnosti osebnih podatkov v porazdeljenem sistemu s pomočjo varnih identifikatorjev

Študent:	Matej Urbas
Študijski program:	univerzitetni, Računalništvo in informatika
Smer:	Programska oprema
Mentor:	red. prof. dr. Peter Kokol
Komentor:	izr. prof. dr. Milan Zorman

Maribor, junij 2009

Univerza v Mariboru

Zahvala

Zahvaljujem se mentorju in komentorju za pomoč in vodenje pri opravljanju diplomskega dela. Prav tako se zahvaljujem ekipi mimelT *za idejno pomoč.*

Posebna zahvala velja staršem, ki so mi omogočili študij.

Zagotavljanje zaupnosti osebnih podatkov v porazdeljenem sistemu s pomočjo varnih identifikatorjev

Ključne besede: programska oprema, obdelava podatkov, zaupnost, varnost, šifriranje

UDK: 004.42.056.5(043.2)

Povzetek

V okviru te raziskave smo preučili problem zaupnosti ob izmenjavi osebnih podatkov v posredniških sistemih. Posebno pozornost smo posvetili varnemu naslavljanju in hranjenju odjemalsko šifriranih vsebin. Osnovni rezultat tega diplomskega dela je zasnova arhitekture ogrodja, ki s pomočjo posebnih varnih identifikatorjev ponuja hranjenje šifriranih podatkov ob tajni izmenjavi informacij med obstoječimi sistemi. Uporaba identifikatorjev prav tako omogoča povsem anonimne načine delovanja, kar je v mnogih realnih transakcijah še posebej zaželeno. Ob koncu smo v okviru primerov uporabe iz zdravstva preverili še delovanje prototipskega sistema, kjer smo primerjali osnovne značilnosti z že obstoječimi rešitvami.

Ensuring confidentiality of personal information in a distributed environment through secure identifiers

Keywords: software development, data processing, confidentiality, security, encryption

UDK: 004.42.056.5(043.2)

Abstract

Within this diploma we have investigated the architectural shortcomings in middleware systems caused by the strict requirements of personal data confidentiality. Special attention has been paid to data addressing and client-side encryption. The main result is an architectural design of a framework that uses special identifiers for secure interchange of encrypted personal data between existing systems. The use of identifiers also allows anonymous usage, which is very beneficial in a large number of real-life scenarios. Consequently, a prototype system has been tested in the area of healthcare – we have compared main functional aspects of the prototype with some existing PHR solutions.

Kazalo

Slike

Izpisi

Kratice

SSL	Secure Sockets Layer
TLS	Transport Layer Security
SC	Smart Card
C-SDA	Chip-Secured Data Access
SQL	Structured Query Language
SUPB	Sistem za upravljanje s podatkovnimi bazami, *glej* DBMS
DBMS	Database Management System
CSS	Ciphertext Searching and Substitution
DAS	Podatkovna baza kot storitev
XOR	Binarni logični operator, ekskluzivni ali
MAC	Message Authentication Code
SSN	Social Security Number
B2C	Business-to-Consumer
E2E	End-to-End
ZVOP	Zakon o varstvu osebnih podatkov
IP	Internet Protocol
PDF417	Portable Data File, dvodimenzionalna črtna koda
WSIT	Web Services Interoperability Technology
MD5	Message-Digest algorithm 5

SHA	Secure Hash Algorithm
EJB	Enterprise Java Beans
JPA	Java Persistence API
WSDL	Web Services Description Language
PHR	Personal Health Record

Poglavje 1

Uvod

Zagotavljanje zaupnosti podatkov v računalniško podprtih transakcijah je le eden izmed osnovnih vidikov informacijske varnosti[1; 2]. Kljub temu je zaupnost mnogokrat težko zagotoviti v popolnosti ali celo v meri, ki je določena z zakoni[3; 4] – med drugim tudi zaradi konfliktnih lastnosti različnih vidikov implementacije, na primer: priročnost sistemov, stopnja zaupnosti ter težavnost same izvedbe[1].

Poudarek raziskav v okviru tega diplomskega dela je zagotavljanje zaupnosti podatkov in anonimnosti transakcij v sistemih, kjer si uporabniki (podjetja ali posamezniki) pogosto izmenjujejo osebne podatke oziroma podatke z visoko stopnjo občutljivosti. Naslavljanje podatkov lahko s pomočjo integracije poteka med različnimi sistemi, ob tem pa je potrebno ponuditi možnost anonimnega naslavljanja ter popolni nadzor nad dostopi izmenjanih podatkov.

[1]S povečevanjem stopnje zaupnosti je potrebno sprejemati kompromise v povezavi z uporabnostjo oziroma priročnostjo sistema. Kot primer lahko navedemo iskanje po šifriranih podatkih, kjer zaradi neberljivosti ni možno uporabiti klasičnih iskalnih prijemov. Natančnejša opredelitev in predlagani pristopi k reševanju problema sledijo v odseku 4.1.2.

Osnovni cilj je preveriti možnost izvedbe ogrodij, kjer osebni podatki niso varovani oziroma šifrirani zgolj na ravni komunikacijskih kanalov, temveč tudi v vmesnih fazah naslavljanja in shrambe. S tem bi onemogočili vpogled v podatke na kraju, kjer so le-ti začasno hranjeni[2]. Želimo pa zagotoviti dostop do vsebine zgolj in samo končnim naslovnikom ter onemogočiti vpogled nadzornikom sistema.

Morda najpomembnejše orodje, ki ga bomo uporabili ob raziskavi izvedljivosti in izvedbi prototipnega sistema, so tako imenovani varni identifikatorji AtomID, ki služijo kot naslovi uporabnikov, vendar le v mejah sistema. Predstavljeni so s kratkimi, preprostimi in berljivimi besedami – z namenom, da bi si jih uporabniki čim lažje zapomnili.

S pomočjo teh identifikatorjev želimo zagotoviti anonimnost, saj trdimo, da ob samem dejanju izmenjave identifikatorja, ki v tem primeru deluje kot naslov osebe, ne prihaja do izdaje osebnih podatkov o pošiljatelju oziroma naslovniku samem. Identifikator namreč ne vsebuje informacij o osebi kot takšni, temveč le o njenem naslovu znotraj sistema, kjer so vsi podatki zaščiteni. S tem uvajamo ključno inovacijo, saj razkritje naslova osebe s tem ne pomeni več nikakršne izdaje osebnih informacij. Do izmenjave osebnih podatkov pride šele, kadar se uporabnik tako odloči.

Aplikacija prototipnega sistema in empirična analiza bosta izvedeni v sklopu scenarijev uporabe osebnih podatkov v zdravstvu, kjer je zaupnost še posebej velikega pomena.

[2]Mnogo podatkovnih shramb sodobnih sistemov vsebuje podatke v nešifrirani obliki oziroma podatke, ki so nadzornikom sistema enostavno dostopni.

Poglavje 2

Obstoječe raziskave in tehnologije

Eden pomembnejših pristopov zagotavljanja zaupnosti podatkov je prav gotovo šifriranje s pomočjo klasičnih šifrirnih metod – bodisi simetrične ali asimetrične enkripcije. Vendar v posredniškem skladiščnem sistemu, kjer upravljamo z veliko množico šifriranih podatkov ter želimo zagotoviti dostopnost vsebin zgolj in samo ciljnim uporabnikom, naletimo na mnoge težave. Osnovna prednost šifriranja se pogosto prevesi v bistveno prepreko, saj je na primer sortiranje ali iskanje po hranjenih podatkih s tem močno oteženo.

Preostane še vprašanje, kako pravzaprav zagotoviti tajnost podatkov tudi v osrednji sistemski shrambi. Zaželeno je namreč, da so podatki neberljivi tudi administratorjem sistema [1]. Ena izmed možnosti je, da podatke šifriramo že na odjemalski strani – s ključem uporabnika. Vendar tudi s tem ponovno naletimo na težave, saj so na primer ob izgubi ključev podatki v shrambi izgubljeni.

Omeniti je potrebno, da tudi sprememba šifrirnega ključa ni preprosta. Ob

[1] S tem močno otežimo možnost fizične kraje podatkov iz shrambe.

spremembi je namreč potrebno prenesti prav vse podatke k uporabniku, saj jih le-ta edini lahko dešifrira. Šele nato jih z novim ključem šifrirane vrne v sistem. Očitno je, da verjetnost spodletelih transakcij s tem močno naraste. Prav tako je potrebno zagotoviti, da v kateri koli vmesni fazi prenosa podatkov ni možno razbrati in hraniti ključev oziroma nezaščitene vsebine.

Z omenjenimi in podobnimi težavami so se srečevali že v predhodnih raziskavah, zato je smiselno, da izsledke, kjer je le možno, vključimo v načrtovanje sistema.

2.1 Odjemalsko šifriranje

Hranjenje podatkov v digitalni obliki ter objavljanje informacij na spletu je že pred leti doživelo svoj razmah. Vse več podjetij v namene poslovanja uporablja mrežne storitve, s čimer se povečuje tudi dosegljivost podatkov v spletu. Izkaže se, da mnogokrat stranke nimajo več druge možnosti, kakor da zaupajo organizacijam, da le-te skrbno varujejo svoje podatkovne shrambe in da zaposleni osebnih podatkov ne zlorabljajo ali jih morda celo izgubljajo[2].

V splošnem so podatkovne shrambe podjetij zaščitene zgolj na ravneh komunikacijskih kanalov in poslovne logike[5] – kar vključuje mehanizme prijave uporabnikov z gesli, uveljavljanje nadzora dostopov (v srednjih plasteh večnivojskih sistemov ali celo odjemalskih programov dvo- ali enoplastnih sistemov) ter uporabo šifriranih tokov podatkov (na primer s pomočjo SSL oziroma TLS). Na

[2]Preprosto iskanje s pomočjo spletnih iskalnikov s ključnimi besedami *personal records lost* vrne kopico rezultatov, ki pričajo o visoki stopnji izgub osebnih podatkov po svetu; npr.: http://news.bbc.co.uk/2/hi/uk_news/politics/7103566.stm (obiskano 23.5.2009).

žalost omenjeni mehanizmi ne preprečujejo dostopov do podatkov iz notranjosti podjetja – upravitelji imajo večinoma neomejen dostop do sistema, s tem pa imajo možnost zlahka obiti varnostne mehanizme poslovne logike. Omenjeni mehanizmi prav tako ne preprečujejo možnosti zlorabe podatkov, kadar so shrambe podatkov fizično izgubljene ali zavržene v delujočem stanju.

Osnovna ideja varnostnih mehanizmov, ki preprečujejo omenjene težave, je šifriranje na strani odjemalskih programov. Uporabnik s pomočjo odjemalskega programa šifrira podatke, še preden so le-ti odposlani preko bodisi šifriranih bodisi nešifriranih komunikacijskih kanalov. Izvedljivost takšnega sistema sta raziskala že Bouganim in Pucheral[6], ki sta v ta namen preverila uporabo pametnih kartic (tako imenovanih kartic SC) – prototipni sistem sta imenovala C-SDA oziroma *Chip-Secured Data Access*.

Bouganim in Pucheral sta problem hranjenja zaupnih oziroma šifriranih podatkov razdelila v več razsežnosti:

- *Zaupnost in tajnost*: potrebno je zagotoviti zaščito pred vdori tretjih oseb, zaposlenih ali nadzornikov.
- *Kapaciteta shrambe*: sistem ne sme omejevati največje možne količine hranjenih podatkov.
- *Naslavljanje podatkov*: uporabnikom je potrebno ponuditi možnost, da lahko podatke naslavljajo tudi na ostale uporabnike v sistemu.
- *Izvajanje poizvedb*: omogočeno mora biti izvajanje poljubnih predikatnih poizvedb (nrp.: SQL) po podatkovni shrambi – neodvisno od razdrobljenosti ukazov.

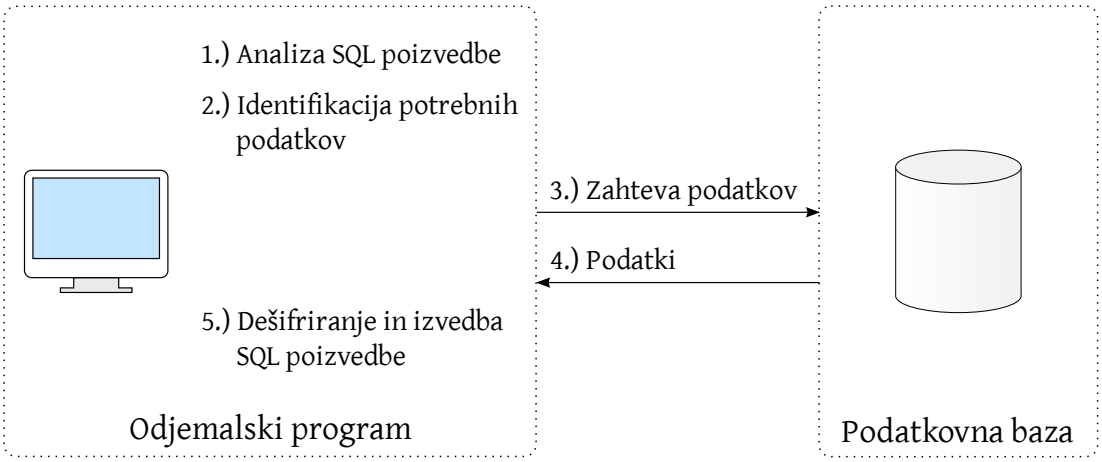
- *Hitrost izvajanja in skalabilnost*: sistem mora zagotavljati zadovoljiv odzivni čas, skalabilnost ter prilagodljivost najrazličnejšim scenarijem in primerom uporabe.

V okviru zastavljenih omejitev, ki jih narekujejo zgoraj navedene razsežnosti zaupnosti, sta avtorja lahko izbirala le med ozkim naborom rešitev. Med drugim sta predlagala rešitev, kjer odjemalski program sestavi SQL poizvedbo in jo v celoti izvaja lokalno. Podatki so v šifrirani obliki preneseni k uporabniku, kjer jih slednji dešifrira in obdela (glej sliko 2.1). Ocenjujemo, da je zahtevnost takšne izvedbe visoka, kljub obstoječim možnostim nadgradnje in optimizacije[7].

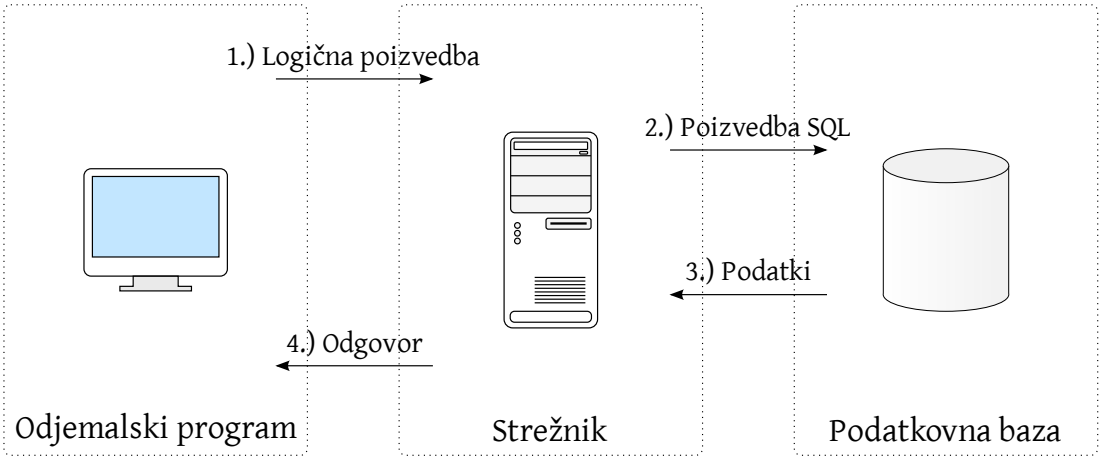
Kot alternativo predlagamo nekoliko drugačen pristop, ki sicer omogoča zgolj iskanje po ključu enakosti (tako imenovan *equality search*), vendar močno poenostavi implementacijo ter časovno in prostorsko zahtevnost algoritma. Osnovna ideja algoritma je iskanje s sekljanimi vrednostmi (*hashed value search*), ki ne primerja vsebine podatkov, ker je vsebina na strežniku šifrirana, temveč njihove sekljane vrednosti. V nasprotju s prvo metodo prenos celotnih podatkov v namene obdelave zahteve ni potreben, saj se le-ta lahko v celoti izvede kar na strežniku tronivojske arhitekture (glej sliko 2.2). V okviru te raziskave bomo ovrednotili še varnost takšnega sistema pred statističnimi napadi (glej podpoglavje 4.1.2).

Avtorja članka sta zaradi zastavljenih omejitev sistema predlagala uporabo simetrične enkripcije, saj je asimetrično šifriranje časovno in prostorsko zahtevnejše. V namene skupne rabe podatkov je tako najprej potrebno določiti skupine uporabnikov, ki dostopajo do istih podatkov, in nato skupne ključe razdeliti med vse uporabnike istih skupin.

Žal je v porazdeljenem večuporabniškem sistemu, kjer uporabniki naključno



Slika 2.1: Izvedba SQL poizvedbe v odjemalskem programu.



Slika 2.2: Izvedba logične poizvedbe na strežniku.

izbirajo naslovnike ter omejujejo dostope individualno, težko zagotoviti tajnost s pomočjo simetričnih ključev skupinskih pametnih kartic. V ta namen bi namreč potrebovali $n + \frac{n(n-1)}{2}$ ključev (kjer je n število uporabnikov sistema), saj bi vsak par uporabnikov ob lastnem ključu potreboval še skupen ključ za zaupno izmenjavo podatkov. Te ključe bi bilo potrebno ustvariti vnaprej in jih fizično posodabljati z naraščajočim številom uporabnikov. Seveda pa nizka pomnilniška kapaciteta pametnih kartic močno omejuje takšno rabo. Omenjeno težavo rešujemo v poglavju 3.4, kjer uvajamo ciljno naslavljanje podatkov s pomočjo identifikator-

jev AtomID in asimetričnega šifriranja simetričnih ključev. Šifrirane simetrične ključe hranimo v podatkovni bazi kar skupaj s šifriranimi podatki, kar lahko orišemo s pomočjo formule kriptografskega vektorja:

$$V = (E_{jk}(sk), E_{sk}(D)) \tag{2.1}$$

kjer je:

V – podatkovni par s šifriranim ključem in šifriranimi podatki,

sk – naključno ustvarjen simetrični ključ,

jk – uporabnikov javni ključ,

E_{jk} oziroma E_{sk} – šifrirna procedura s pomočjo asimetrične oziroma simetrične enkripcije,

D – podatki, ki jih želimo šifrirati.

2.2 Zaupne shrambe podatkov

2.2.1 Načrtovanje šifriranih podatkovnih baz

Podatkovne shrambe večinoma vsebujejo nešifrirane podatke, kar omogoča preprosto indeksiranje, pospeševanje najrazličnejših poizvedb, zagotavljanje referenčne integritete ter enostavnost podatkovne analize. Šifrirani podatki v veliki meri otežijo omenjene prijeme, ki so sicer vgrajeni v večino sodobnih sistemov za upravljanje z nešifriranimi podatkovnimi bazami (SUPB oziroma DBMS).

Menimo, da se zaradi naštetih omejitev načrtovanje šifriranih podatkovnih baz bistveno razlikuje od načrtovanja navadnih podatkovnih baz.

Wagner, Putter in Cain[8] so v svoji raziskavi preučili različne prijeme načrtovanja shramb s šifriraninimi podatki. Omenjenih je več pristopov:

- *Zaupna sita*: zaupna sita postavimo med SUPB in odjemalske programe. Ta metoda uporablja kriptografske sekljalne vsote (*checksums*[3]), da se zagotovi, da podatki niso dosegljivi uporabnikom, ki zanje nimajo dovoljenja[9].

- *Omejeni pogledi*: predlagana je ureditev podatkovne baze, kjer lahko uporabniki z nekega varnostnega nivoja pridobijo zgolj poglede na podatke, ki ne vsebujejo stolpcev višjih ravni varnosti. Za vsak nivo varnosti lahko uporabljamo poseben šifrirni ključ.

- *Bločno šifriranje ali šifriranje zapisov z enim ključem*: preprosta metoda, kjer s pomočjo enega šifrirnega ključa šifriramo vse oblike podatkov. Ta pristop je enostavno prilagodljiv za večnivojsko varnost z uporabo več ključev. Obdelava podatkov v tem načinu zahteva veliko število dešifrirnih posegov med izvajanjem predikatnih poizvedb.

- *Šifriranje zapisov z različnimi ključi za posamezna polja*: kombinacija šifriranja zapisov z dešifrirnimi postopki za posamezna polja. Predlagani so bili specializirani algoritmi[10], vendar so avtorji članka [8] mnenja, da ima ta metoda v primerjavi z npr. metodo šifriranja polj z več ključi bistveno več slabosti.

[3]Te procedure iz vektorja podatkov izračunajo eno samo število, tako imenovano sekljalno vsoto ali *checksum*. V kriptografiji obstaja zahteva po tem, da je karseda težko poiskati podobne vektorje podatkov, ki imajo enako sekljalno vsoto.

- *Šifriranje polj z enim ključem*: postopek, kjer vsako polje zapisa šifriramo posebej, vendar z istim ključem. Bistvena slabost teh algoritmov je visoka stopnja ranljivosti, saj je zaradi velike količine kratkih šifriranih vsebin možno uporabiti napad iskanja in zamenjave kriptogramov (CSS oziroma *ciphertext searching and substitution*).

- *Šifriranje polj z več ključi*: predstavlja nadgradnjo prejšnjega postopka, kjer vsako polje šifriramo z lastnim ključem (s tem nekoliko omejimo ranljivost metode pod napadi CSS).

- *Zaupni homomorfizmi*: s pomočjo homomorfnih preslikav bi lahko uporabili običajne prijeme za delo s podatki brez potrebe dešifriranja. Na žalost ima ta metoda teoretične omejitve in je tako kot šifriranje polj z enim ključem ranljiva pod napadi CSS.

- *Homofonska predstavitev podatkov v podpoljih*: poskuša uvesti prednosti, ki jih prinaša metoda zaupnih homomorfizmov. Metoda temelji na razdelitvi polj v več podpolj ter poskuša z vnosom umetnega šuma preprečiti ranljivost CSS.

Raziskava ponuja kopico rešitev, vendar je možno izluščiti lastnosti, ki so skupne vsem metodam. Običajni postopki urejanja zapisov so močno omejeni (z izjemo metod zaupnih homomorfizmov in homofonske predstavitve podpolj). Šifriranje celotnih zapisov prav tako izkazuje nižjo stopnjo ranljivosti pod določenimi kriptografskimi napadi kakor šifriranje posameznih polj. Predvidevamo, da je v naše namene smiselna uporaba šifriranja celotnih zapisov, saj ne potrebujemo popolnega nabora običajnih postopkov urejanja podatkov, ki jih sicer ponujajo standardni SUPB.

2.2.2 Upravljanje s ključi

Ob načrtovanju zaščite podatkov s pomočjo šifriranja je neizbežno tudi načrtovanje upravljanja s šifrirnimi ključi[11]. Pravilno gospodarjenje s ključi je vsaj tolikšnega pomena kakor upravljanje s podatki samimi. Izguba ključa lahko v temeljih izpodbije prav vse prednosti šifriranja, saj lahko najditelj ključa dostopa do podatkov prav tako, kakor lastnik sam. Še več, velikokrat izguba ključa pomeni tudi praktično izgubo podatkov, kar je posledica osnovne naloge šifriranja – napraviti odklepanje vsebine paketov brez ključa čim težje.

Damiani in ostali so v raziskavi [12] preverili upravljanje s ključi v večuporabniških podatkovnih bazah – v okviru tako imenovanih *podatkovnih baz kot storitev* (DAS oziroma *database-as-a-service*). Izluščili so štiri osnovne entitete in akterje takšnih sistemov: *lastnike podatkov*, *uporabnike*, *odjemalske programe* ter *strežniške programe*. Lastnik podatkov določa politiko dostopov do podatkov – s pomočjo odjemalskega programa šifrira podatke, še preden jih le-ta pošlje na strežnik preko komunikacijskih kanalov. Prav zaradi šifriranih vsebin strežniki, in s tem tudi ostali uporabniki sistema, nimajo možnosti vpogleda v podatke; ob tem je potrebno omeniti, da je lastnik podatkov tudi lastnik skritih ključev – zaradi tega je upravljanje s skritimi ključi popolnoma prepuščeno uporabniku oziroma odjemalskemu programu.

V sistemu AtomID želimo uporabnikom omogočiti preprosto izmenjevanje oziroma naslavljanje podatkov. Uporabniki bodo lahko dostopali do tujih podatkov šele takrat, kadar bodo lastniki šifrirali podatke s ključi, ki so znani naslovnikom. Seveda je o prispelih pošiljkah šifriranih podatkov potrebno ustrezno obvestiti naslovnike – naslovnik mora na primer vedeti, s kakšnim ključem so podatki

šifrirani; na srečo je obveščanje možno avtomatizirati s pomočjo odjemalskih in strežniških programov.

V namene naslavljanja podatkov med več uporabniki predlagamo uporabo asimetričnega šifriranja. Javni ključi vseh uporabnikov se hranijo v podatkovni bazi in so dostopni vsakomur. Lastniki podatkov lahko s tem po svoji presoji izbirajo, ali bodo zaupali svoje podatke komurkoli (ne le skupinam uporabnikov) – v ta namen je potrebno podatke zgolj šifrirati z javnimi ključi naslovnikov. V nadaljevanju bomo uporabili izsledke omenjenih raziskav in predstavili izvedbo sistema, kjer so skriti deli ključev nameščeni v odjemalskem programu (zaščiteni z geslom ali hranjeni na pametnih karticah) ter javni deli ključev prenešeni v podatkovno shrambo sistema.

2.2.3 Zagotavljanje celovitosti

Zagotavljanje določenih vidikov celovitosti v klasičnem okolju ponuja že večina SUPB sistemov in aplikacijskih strežnikov. Kadar želimo zagotavljati referenčno celovitost nešifriranega podpornega dela podatkovne baze, lahko seveda še vedno koristimo storitve SUPB – čeprav sicer hranimo šifrirane podatke. Prav tako lahko uporabimo sekljane vsote ali digitalne podpise za zagotavljanje vsebinske integritete vseh podatkov. Težava se pojavi, kadar želimo zagotavljati referenčno celovitost med vsebinami šifriranih zapisov.

Hacigümüş, Iyer in Mehrotra [13] so preučili možnosti zagotavljanja celovitosti povezanih podatkov, četudi so slednji šifrirani. Predlagali so rešitev, s katero ustvarijo digitalni podpis za vsako izmed šifriranih tabel. Bistvena omejitev takšnega mehanizma je v visoki stopnji prenosa podatkov med strežniškim pro-

gramom in odjemalskimi programi. Nesprejemljivo je namreč, da mora odjemalec ob vsaki spremembi (t.j. ob vnosu, izbrisu ali prepisu) vrstic v tabeli ponovno izračunati digitalni podpis vseh podatkov v tabeli. To pa pomeni, da je potreben prenos celotne vsebine tabele k uporabniku.

Da bi omejili potrebe po prenosu ogromnih količin podatkov, so dodatno predlagali postopek inkrementalnega podpisovanja. S pomočjo XOR MAC avtentikacije sporočil dosegajo bistveno nižje zahteve po prenosu podatkov, saj lahko izračunamo posodobljen podpis zgolj z obstoječim podpisom in s podatki zapisa, ki ga spreminjamo.

Celovitost podatkov je izjemnega pomena za pravilno delovanje storitev sistema. Predvidevamo, da za namene naše raziskave omenjen pristop popolnoma zadošča.

2.3 Šifriranje osebnih identifikatorjev

Eden izmed ciljev raziskave je oblikovati sistem, ki bo v celoti odpravil potrebe upravljanja z osebnimi identifikatorji. Uporabniki lahko koristijo sistem s popolno anonimnostjo, kar omogočamo s pomočjo varnih identifikatorjev AtomID. Osnovna zasnova identifikatorjev AtomID sledi izsledkom raziskave Eleanor Meux[14], ki za zagotavljanje anonimnosti našteje tri različne pristope enosmernega šifriranja osebnih številk (na primer številk SSN), vendar prvi metodi iz navedenih razlogov označi kot neprimerni:

1. uporaba obstoječih šifrirnih algoritmov (ki niso prestali strogih kriterijev,

potrebnih za zagotavljanje anonimnosti v zdravstvu),

2. prireditev popolnoma naključnih 9-mestnih števil vsakemu osebnemu identifikatorju (ta metoda je bila zavržena zaradi neoptimalnosti in težavnosti izvedbe), ter

3. izvedba posebne enosmerne preslikave, prilagojene posebej v namene šifriranja osebnih identifikatorjev.

Za ustvarjanje identifikatorjev AtomID uporabljamo metodo, ki temelji na prireditvi naključnih števil (pristop št. 2). Omeniti je potrebno, da se je izvedba s pomočjo indeksirnih metod sodobnih SUPB izjemno poenostavila. Menimo, da je metoda dovolj učinkovita, ob tem pa ponuja še dodatne prednosti.

V namene generiranja identifikatorjev uporabljamo proceduro brez parametrov, katere cilj je ustvariti število (sestavljeno iz številk in črk), ki ga lahko čim enostavneje izgovorimo in si ga čim lažje zapomnimo (opis postopka sledi v poglavju 3.2). Dodatna lastnost sistema AtomID je možnost prirejanja več identifikatorjev posameznim uporabnikom – slednje omogoča dejstvo, da AtomID ni vezan na kakršen koli drugi uporabnikov podatek.

Pomembna nadgradnja omenjene metode omogoča popolnoma anonimno naslavljanje podatkov – t.j. brez uporabe lastnih identifikatorjev. Osnovna ideja tega pristopa je, da ob izmenjavi podatkov ustvarimo nov AtomID brez lastnika (anonimen AtomID). Šele nato priredimo novo ustvarjenemu AtomID-ju podatke, ki jih želimo posredovati naslovniku. Uporabnik kljub vsemu ohrani možnost upravljanja anonimnih AtomID-jev s pomočjo skritih gesel. Natančnejši opis anonimnega naslavljanja sledi v poglavju 3.3.

2.4 OpenID

OpenID je odprt protokol za decentralizirano upravljanje identitet, avtentikacijo uporabnikov in nadzor dostopov v spletu[15]. Uporabniki lahko s pomočjo ogrodja OpenID upravljajo s svojim računom zgolj na enem mestu.

Naloga spletnih sistemov, ki želijo svojim uporabnikom ponuditi možnost prijave s pomočjo ogrodja OpenID, je, da v poslovno logiko posebej vgradijo podporo za avtentikacijo s pomočjo standardnega protokola. Preverjanje avtentičnosti uporabnika je s tem prepuščeno tujemu strežniku, kar seveda ni brez posledic v organizaciji varnosti sistema, kjer ob potrebnem zaupanju naletimo še na mnogo potencialnih ranljivosti[16]. Kljub vsemu je možno s pomočjo nadzorov dostopa, certifikatov in varnih komunikacijskih kanalov zagotoviti zadovoljivo mero varnosti in zaupnosti.

Bistvena razlika med ogrodji OpenID in AtomID je v količini oziroma načinu hranjenja podatkov ter v sami namembnosti sistemov. Primarna naloga platforme AtomID ni preverjanje avtentičnosti, temveč naslavljanje, hranjenje in varovanje šifriranih osebnih podatkov uporabnikov. Implementacija avtentikacije v sistemih, ki uporabljajo identifikatorje AtomID, je popolnoma neodvisna in prepuščena lastnikom sistemov.

Skupna lastnost sistemov je možnost decentraliziranega upravljanja z uporabniškimi računi. Eden izmed ciljev je namreč ponuditi uporabnikom možnost uporabe zgolj enega računa za vse AtomID podprte sisteme – kar je bistvena podobnost s protokolom OpenID.

2.5 Primerjava tehnologij

Izsledki predhodnih raziskav in obstoječih tehnologij ponujajo množico pristopov k realizaciji odjemalskega šifriranja podatkov. Prav tako smo spoznali več načinov varnega upravljanja z osebnimi identifikatorji.

Mnogo tehnologij smo lahko prevzeli in jih v okviru zahtev sistema AtomID tudi nadgradili. Bistvene nadgradnje vključujejo varno naslavljanje podatkov poljubnim naslovnikom s pomočjo asimetrične enkripcije ter možnost izvajanja večine poizvedb na strežniku – kar omogoča tronivojska zasnova sistema.

Za primerjavo omenjenih tehnologij glej tabelo 2.1.

	C-SDA	**OpenID**	**DAS**	**AtomID**
Šifrirani podatki	✓	N/A	✓	✓
Skupinsko šifrirano naslavljanje[4]	✓	N/A	×	✓
Poljubno šifrirano naslavljanje[5]	×	N/A	×	✓
Decentralizacija prijav	×	✓	×	×
Decentralizacija upravljanja uporabniških računov	×	✓	×	✓
Porazdeljena podatkovna shramba	×	N/A	✓	✓
Spletni identifikatorji	×	✓	×	✓

[4]Pošiljanje podatkov, šifriranih s skupnimi ključi.

[5]Pošiljanje podatkov, šifriranih z javnimi ključi naslovnikov – skupine in skupni ključi niso potrebni.

	C-SDA	**OpenID**	**DAS**	**AtomID**
Izvajanje šifriranih poizvedb[6]	odjemalec	N/A	odjemalec	strežnik

Tabela 2.1: Primerjava tehnologij C-SDA , OpenID, DAS in AtomID.

[6]Dešifriranje podatkov poteka v vsakem primeru na odjemalcu.

Poglavje 3

Tehnologija

3.1 Opis sistema AtomID

Z rastjo elektronskega poslovanja ter elektronskega hranjenja in obdelave podatkov se postopoma pojavljajo popolnoma nove oblike kriminala, kot sta kraja identitete in vdori v računalniška omrežja. Zaradi hitrega razvoja tehnologije in sočasnega strmega porasta novih vrst groženj se podjetja, katerih primarno področje poslovanja ne obsega informacijskih dejavnosti, težko soočajo z varnostnimi vprašanji.

Hkrati podjetja, ki poslujejo s fizičnimi osebami, za veliko večino svojih poslovnih dejavnosti potrebujejo vsaj nekatere osebne podatke svojih strank – te podatke morajo varovati v skladu z zakonom o varovanju osebnih podatkov[4], kar pomeni, da jih morajo hraniti v sistemih z relativno visoko stopnjo varnosti, ali pa tvegajo denarno kazen med 4.000 in 12.000 EUR. Žal organizacije ob poskusu razvoja lastnih ogrodij pogosto naletijo na težave, saj je razvoj varnih

sistemov:

drag – razvoj varnih informacijskih sistemov zahteva znatna vlaganja v znanje kadrov, revizije sistema, dosledna testiranja, certifikacije, ipd.

zapleten – razvoj varnih sistemov je bistveno bolj zapleten od razvoja povprečnih informacijskih sistemov.

specifičen – razvoj varnih sistemov zahteva specifična znanja, ki jih večina programerjev, ki se s tem ne ukvarja, nima.

Hkrati so podjetja, ki hranijo osebne podatke, v skladu z ZVOP dolžna zagotavljati določene storitve:

- vodenje zapisnika o dostopih do osebnih podatkov,
- vpogled v dnevnike dostopov na zahtevo upravičencev ter
- izbris in/ali spremembo vsebine v primeru zahteve s strani lastnika osebnih podatkov.

Na podlagi dejstev, da so osebni podatki jasno opredeljeni z zakonom, da so hranjeni v enaki ali vsaj podobni obliki v veliki večini podjetij in da je potrebno zagotavljati vsaj minimalni nabor storitev (vpogled, sledljivost in anonimnost), smo oblikovali sistem z naslednjimi značilnostmi:

- Podatki so hranjeni na enem mestu, s čimer je ponujena možnost deljene infrastrukture med podjetji in fizičnimi uporabniki,

- Varnostni sistemi in algoritmi so nadgradljivi (dopuščene so možnosti sprememb postopkov v upravljanju z osebnimi podatki, dodajanje novih šifrirnih algoritmov ipd.) – s tem je možno prilagajati sistem spremenljivim zakonskim zahtevam.

- Dostopi do osebnih podatkov se beležijo, s čimer se poveča transparentnost sistema.

- Podatki so šifrirani, še preden prispejo v sistem AtomID (šifrirani so lokalno na strani uporabnika), kar zagotavlja, da jih ne morejo prebrati niti nadzornik sistema niti ostala podjetja, ki sistem uporabljajo (varnost je zagotovljena tudi v primeru, če bi bili podatki fizično odtujeni).

[17]

3.2 Identifikatorji AtomID

Identifikatorji AtomID so prav gotovo eden pomembnejših sestavnih delov sistema AtomID. Prevzemajo vlogo edinega logičnega in javnega naslova uporabnikov in so ustvarjeni popolnoma naključno. Z naključnostjo želimo zagotoviti, da niso izpostavljeni kakršni koli dejanski osebni podatki – mnogi sistemi namreč uporabljajo osebne podatke kakor javne identifikatorje v sistemu. Identifikator AtomID je torej edini kos informacij o uporabniku, ki je viden navzven – brez imen, priimkov ali podobnih informacij.

Identifikatorji AtomID so oblike osemznakovnih nizov, neobčutljivih na velikost posameznih znakov (*case-insensitive*). Sestavljeni so iz črk angleške abecede,

števk od 0 do 9 ter znakov - in / (izpis 3.1 prikazuje pripadajoči regularni izraz).

Izpis 3.1: Regularni izraz AtomID identifikatorjev.

```
AtomID  ::= ([A-Z]|[a-z]|[0-9]|-|/){8,8}
```

Upoštevajoč dejstvo, da velikost črk ni pomembna, lahko izračunamo, da dopušča omenjena oblika nizov približno $4,35 \cdot 10^{12}$ različnih identifikatorjev (glej izračun 3.1). Čeprav pričakujemo, da bo takšno število enoličnih nizov popolnoma ustrezalo vsem načrtovanim praktičnim primerom uporabe, dopuščamo možnost kasnejšega prilagajanja oblike ter povečevanja dolžine in nabora znakov.

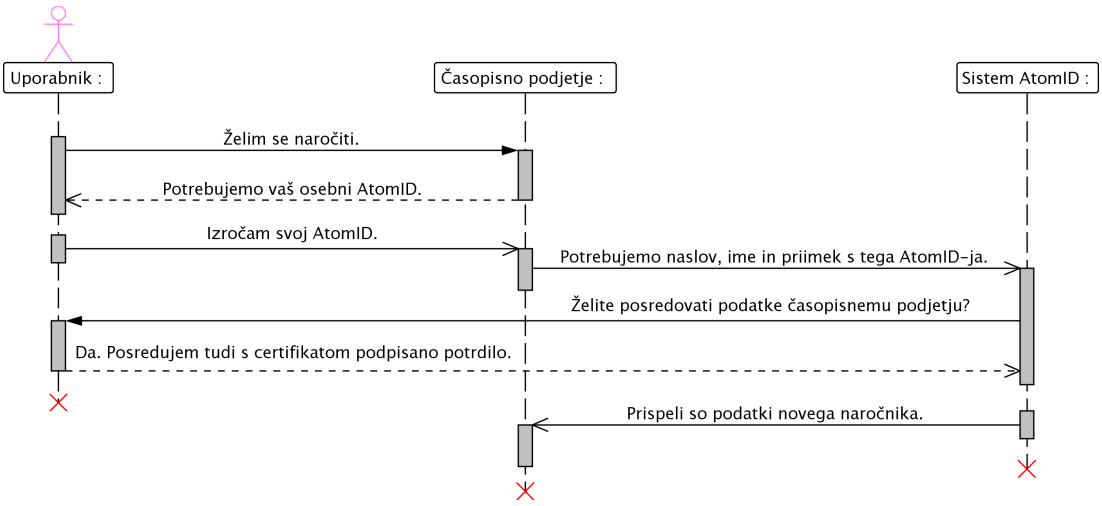
$$N = (26 + 10 + 2)^8 = 38^8 = 4.347.792.138.496 \tag{3.1}$$

V grobem delimo identifikatorje na dve kategoriji: *javni* oziroma *osebni identifikatorji* ter *anonimni identifikatorji*. Anonimni identifikatorji se od javnih razlikujejo zgolj v tem, da slednji pripadajo natanko enemu uporabniku, medtem ko anonimni nimajo pripisanega lastnika (glej diagram 3.1).

Slika 3.1: Prikaz relacije med entitetama *Uporabnik* in *AtomID*.

Javni identifikatorji: Uporabnik si lahko lasti več javnih identifikatorjev. Z njimi se lahko predstavlja ostalim uporabnikom v sistemu, naslavlja pošiljke ali povprašuje po osebnih podatkih.

V postopku naslavljanja in povpraševanja po podatkih sta potrebna zgolj identifikatorja udeleženih uporabnikov – s tem se povsem izognemo razkrivanju potencialno zaupnih podatkov že v času naslavljanja (slednje lahko nakažemo s primerom postopka naročanja na storitev, glej diagram poteka 3.2). Podatki potujejo iz pošiljateljevega AtomID računa na račun naslovnika (ob tem je ustrezno poskrbljeno za tajnost podatkov s pomočjo šifriranja, glej podpoglavje 3.4). Omeniti je potrebno, da lahko naslovnik dostopa do podatkov le s prijavo v sistem AtomID – s tem je zagotovljena možnost beleženja evidence o dostopih do podatkov.'

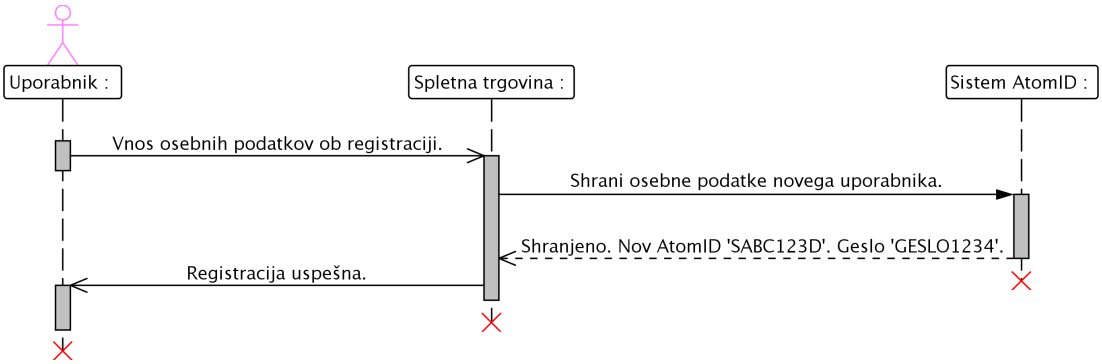


Slika 3.2: Primer poteka izmenjave osebnih podatkov.

Anonimni identifikatorji: Osnovni namen anonimnih identifikatorjev ni naslavljanje podatkov (v nasprotju z javnimi identifikatorji), temveč hranjenje podatkov oseb, ki niso uporabniki sistema AtomID (služijo kot odlagališče osebnih podatkov tujih sistemov podjetij – glej diagram poteka 3.3).

Anonimni identifikatorji se uporabljajo predvsem v scenarijih, kjer se uporabniki registrirajo v sisteme organizacij z običajnimi postopki – na primer z vnosom

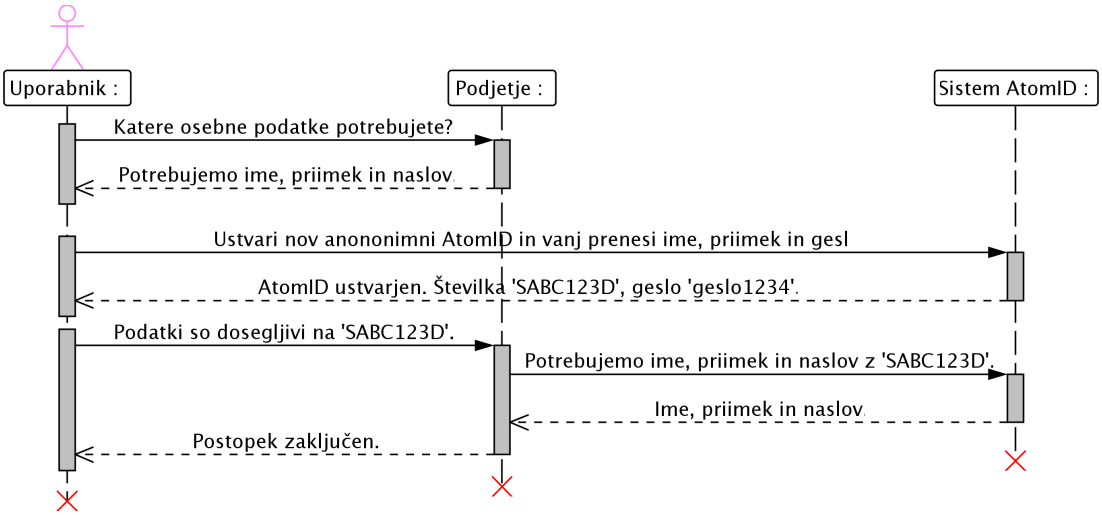
osebnih podatkov v spletni obrazec. Kadar se v sistem podjetja registrira nov uporabnik, se osebni podatki prenesejo v podatkovno shrambo AtomID – ob tem se ustvari nov anonimni identifikator in geslo, s katerim lahko uporabnik in podjetje urejata podatke (podjetje jih ureja neposredno s pomočjo knjižnic AtomID, uporabnik pa ima več možnosti: spreminja jih lahko v sistemu podjetja ali neposredno v sistemu AtomID). Razlogov, da bi podjetja hranila občutljive osebne podatke v podatkovni shrambi sistema AtomID, je več: predaja odgovornosti, zagotovljenost zaščite podatkov, beleženje evidence dostopov, dislokacija ter možnost vpogleda in urejanja podatkov – kakor predpisuje zakon o varstvu osebnih podatkov. Podjetjem s tem prav tako ni potrebno vlagati dodatnih sredstev v lastno implementacijo.



Slika 3.3: Hranjenje podatkov s pomočjo anonimnih identifikatorjev.

Kljub temu lahko anonimne identifikatorje uporabljamo tudi v namene naslavljanja podatkov. V tem primeru se postopek nekoliko razlikuje od postopka naslavljanja z javnimi identifikatorji (glej diagram 3.4).

Uporabnika sistema AtomID, ki prenese podatke na podjetje s pomočjo anonimnih identifikatorjev, načeloma ni možno izslediti, saj v podatkovni bazi sistema ni zavedeno, komu pripada takšen identifikator. Seveda lahko preneseni podatki kljub vsemu vsebujejo popolnoma zadostno količino informacij za identifikacijo



Slika 3.4: Primer poteka anonimne izmenjave osebnih podatkov.

lastnika kot takšnega (na primer s pomočjo naslova, imena in priimka ali davčne številke). Vsekakor je ključna lastnost tega pristopa, da se s pomočjo izmenjave podatkov z anonimnimi identifikatorji ne razkrije uporabnik, ki se sicer skriva za vsakim osebnim identifikatorjem.

Žal anonimno naslavljanje podatkov ni brez slabosti:

- Ker v sistemu ni informacije o asociiranem uporabniku anonimnega identifikatorja, tudi nadzorniki sistema AtomID nimajo možnosti izslediti izvornega lastnika podatkov. Zato je potrebno zagotoviti, da si odjemalec lokalno zapomni novo ustvarjene anonimne identifikatorje in pripadajoča gesla – zgolj z gesli je namreč uporabnikom omogočeno urejanje anonimnega AtomID računa in sledenje evidencam dostopov.

- Naslovniku prav tako ni možno izročiti digitalno podpisanih podatkov s pomočjo uporabnikovega certifikata – vzrok za to je dejstvo, da anonimni podatki nimajo z njimi povezanega uporabnika sistema, žal pa je za za-

gotavljanje preverljivosti digitalnih podpisov potreben prav uporabnikov certifikat (ena izmed možnih rešitev je, da ob ustvarjanju anonimnih identifikatorjev ustvarimo tudi ad-hoc certifikat, s katerim lahko uporabnik kasneje spreminja podatke in potrjuje spremembe).

3.2.1 Zahteve

Identifikatorji so prav gotovo osrednjega pomena v okviru sistema AtomID – prav zaradi tega moramo opredelili nekaj zahtev, s katerimi bomo poenostavili delo s sistemom. Zagotoviti želimo, da so identifikatorji preprosto *berljivi*, da je glede na njihovo obliko možno določiti namen uporabe (slednje želimo doseči s pomočjo *segmentacije*) in da so *enolični* (kar pomeni, da se v sistemu ne moreta pojaviti dva enaka identifikatorja). Prav tako želimo zagotoviti *razširljivost* identifikatorjev – v primeru, da trenutno zastavljen nabor enoličnih nizov ne bi zadoščal.

Berljivost: Identifikatorji AtomID prevzemajo vlogo naslova v sistemu. Uporabniki jih lahko uporabljajo tako v tujih sistemih (kjer je vgrajena AtomID podpora), kakor tudi v namene izmenjevanja podatkov med posamezniki. Na podlagi omenjenih primerov uporabe predvidevamo, da je smiselno ustvarjati osebne identifikatorje, ki so preprosto berljivi in si jih je možno čim lažje zapomniti[18] (v podpoglavju 3.2.2 bomo predstavili algoritem za generiranje naključnih berljivih identifikatorjev).

Enoličnost: V sistemu se ne smejo pojaviti podvojena števila. Za vsak par identifikatorjev mora veljati naslednji izraz:

$$\forall I_i, I_j \in \mathrm{A} : i \neq j \Rightarrow I_i \neq I_j, \tag{3.2}$$

kjer je:

I_i – i-ti identifikator v sistemu,

A – seznam vseh identifikatorjev v sistemu.

Definicija. Identifikatorja sta enaka natanko tedaj, kadar imata enako dolžino in se ujemata v vseh istoležnih znakih:

$$I_i = I_j \overset{def}{\Longleftrightarrow} \mathrm{L}(I_i) = \mathrm{L}(I_j) \wedge \left[\forall k \in \{1, \ldots, \mathrm{L}(I_i)\} : \mathrm{U}(I_i(k)) = \mathrm{U}(I_j(k))\right], \tag{3.3}$$

kjer je:

L(I) – dolžina identifikatorja I (po trenutni specifikaciji je vrednost tega izraza enaka 8 za vsak identifikator, vendar dopuščamo možnost sprememb v prihodnosti),

U(a) – funkcija, ki preslika majhne črke v velike.

Segmentacija: Domnevamo, da bodo uporabniki sistema razdeljeni v manjše število osnovnih skupin. Ob začetku vzpostavitve sistema predvidevamo tri o-

snovne skupine uporabnikov: *fizične osebe*, *podjetja* ter *državni uradi* or. *vladne ustanove*.

V namene lažje prepoznavnosti posameznih skupin predlagamo, da se vsaki skupini pripiše enolična oblika identifikatorjev. Oblike posameznih skupin identifikatorjev imenujemo *razredi*. Vsak razred je določen s prvo črko identifikatorja. Tako na primer identifikatorji, ki se začnejo s črko O (na primer OABCD001, OFERI001 ipd.), pripadajo skupini organizacij. Tabela 3.1 prikazuje začetne znake identifikatorjev, ki določajo posamezne razrede.

Začetni znak	**Ime razreda**
P	Fizične osebe
O	Podjetja
G	Državni uradi/vladne ustanove
S	Anonimni identifikatorji
ostalo	Rezervirano

Tabela 3.1: Seznam razredov AtomID identifikatorjev.

S to spremembo se število enoličnih identifikatorjev v posameznih razredih nekoliko zmanjša. Sedaj je v vsakem razredu na voljo približno $1,14 \cdot 10^{11}$ različnih identifikatorjev (glej izračun 3.4). Domnevamo, da tudi to število povsem zadostuje trenutnim namenom.

$$N = (26 + 10 + 2)^7 = 38^7 = 114.415.582.592 \tag{3.4}$$

Razširljivost: V primeru, da začetni nabor identifikatorjev ne bi zadoščal potrebam v prihodnosti, želimo dopustiti možnost kasnejše nadgradnje. Predvsem pričakujemo potrebo po širjenju nabora dovoljenih znakov in povečevanju števila mest.

3.2.2 Ustvarjanje identifikatorjev

Glede na zastavljene zahteve želimo izdelati algoritem, ki bo ustvarjal berljive identifikatorje in ponujal čim večji nabor enoličnih izhodov. Prav tako želimo, da je algoritem enostavno prilagodljiv v primeru kasnejših razširitev.

Omeniti je potrebno, da potrebujemo višjo berljivost zgolj, kadar ustvarjamo javne identifikatorje. V primeru anonimnih identifikatorjev lahko uporabljamo celoten nabor znakov – kar pomeni uporabo popolnoma naključnega algoritma brez kakršnih koli pravil.

Algoritem mora ustvarjati enolične identifikatorje; zagotoviti je namreč potrebno, da v sistemu ni podvojenih identifikatorjev. Enoličnost lahko zagotavljamo s kombinacijo preverjanja že ustvarjenih identifikatorjev ter ponovnih poskusov (v navedenih primerih tega postopka ne bomo vključevali – čeprav končne različice algoritmov kljub vsemu uporabljajo takšen pristop).

Bistveno koristnejši algoritmi zagotavljanja enoličnosti so sekvenčni generatorji identifikatorjev s konstantnim časom – neodvisnim od odmika (primeri takšnih algoritmov so strogo monotone funkcije; na primer pretvorba številskih sistemov).

Berljivost je izrazito subjektivna lastnost identifikatorjev. Določiti cenilke

berljivosti ni povsem preprosta naloga[18]. Domnevamo pa, da so besede, kjer se paroma izmenjujejo samoglasniki in soglasniki, mnogo lažje berljive kakor popolnoma naključni nizi. Prav tako bomo v berljivih identifikatorjih izpuščali znaka - in /. Na primer:

- Naključni nizi: SVPINAVV, K3J4/5OX, EK50S-JP ipd.
- Izmenični soglasniki in samoglasniki[1]: LEGOROMO, DEGUBYKA, LAKESOPU ipd.

Preverili bomo izvedbe algoritmov, ki upoštevajo navedena načela ter kljub vsemu ponujajo čim večji nabor enoličnih identifikatorjev.

Izpis 3.2 prikazuje najpreprostejši primer. Algoritem sestavlja nize izmeničnih soglasnikov in samoglasnikov[2].

Izpis 3.2: Preprost algoritem za ustvarjanje AtomID identifikatorjev.

```
1 id := znakRazreda;
2 for i := 1 to 7 do
3     if Sodo(i) then
4         id := id + nakljucniSoglasnik;
5     else
6         id := id + nakljucniSamoglasnik;
7     end
8 end
```

Bistvena slabost tega algoritma je izjemno nizko število razpoložljivih kombinacij identifikatorjev v posameznih razredih (glej enačbo 3.5)[3]. Kljub vsemu ima algoritem to prednost, da je preprosto prilagodljiv za daljše nize.

[1] Primere smo ustvarili z algoritmom 3.2

[2] Znak y smo vključili v nabor samoglasnikov.

[3] V vseh nadaljnjih primerih predpostavljamo, da imajo identifikatorji natanko 7 prostih znakov, saj je prvi znak rezerviran za označbo razreda.

$$N = (20 \cdot 6)^3 \cdot 20 = 34.560.000 \tag{3.5}$$

V namen razširitve števila različnih kombinacij predlagamo nekaj očitnih razširitev:

- identifikator lahko vsebuje tudi števila (domnevamo sicer, da bi preveč raztresenih števk zmanjšalo berljivost identifikatorja, zato predlagamo, da smejo identifikatorji vsebovati največ eno strnjeno skupino maksimalno treh števk – skupine števk se lahko pojavijo kjerkoli v nizu),
- prvi znak identifikatorja je lahko tudi samoglasnik ali število.

Izpis 3.3 prikazuje nadgradnjo prejšnjega algoritma, ki ustvarja identifikatorje z upoštevanjem omenjenih razširitev. Ta algoritem ponuja približno $1,20 \cdot 10^9$ različnih kombinacij identifikatorjev (izračun v dodatku A.1).

Izpis 3.3: Nadgradnja algoritma 3.2.

```
1  id           := znakRazreda;
2  stStevk      := random() mod 4;
3  indeksGruce  := (random() mod (8 - stStevk)) + 1;
4  prviZnak     := random() mod 2;
5  for i := 1 to indeksGruce - 1 do
6      if Sodo(i) xor prviZnak then
7          id := id + nakljucniSoglasnik;
8      else
9          id := id + nakljucniSamoglasnik;
10     end
11 end
12 for i := 1 to stStevk do
13     id := id + nakljucnaStevka;
14 end
15 if stStevk > 0 then prviZnak = random() mod 2;
16 for i := indeksGruce + stStev to 7 do
```

```
17      if Sodo(i) xor prviZnak then
18          id := id + nakljucniSoglasnik;
19      else
20          id := id + nakljucniSamoglasnik;
21      end
22  end
```

Čeprav nadgrajen algoritem ponuja več kakor milijardo različnih izhodnih kombinacij – približno tridesetkrat več kakor prejšnja različica, ne smemo kar tako privzeti, da je tudi toliko boljši. Žal ima algoritem 3.3 bistveno pomanjkljivost, ki smo jo odkrili šele ob izračunu števila kombinacij. Algoritem namreč izbira velikostne skupine števk z enakomerno porazdelitvijo med 0 in 3, toda števila kombinacij, ki jih porodi vsaka izmed teh skupin, niso med seboj enaka (največ kombinacij identifikatorjev dobimo, kadar je skupina števk velikosti 1, najmanj pa, kadar v identifikatorju števk ni). Posledica je neenakomerna porazdelitev, ki pa je najpogostejši razlog za prezgoden nastop kolizij. Verjetnost kolizij je torej višja, kakor bi bila v primeru, če bi algoritem izbiral identifikatorje popolnoma enakomerno. Posledica velike verjetnosti kolizij pa je oteženo iskanje novih identifikatorjev.

Na podlagi omenjenih ugotovitev lahko ustvarimo uteži, ki so prilagojene algoritmu 3.3. Te uteži spremenijo verjetnost izbire posamezne skupine števk in s tem nekoliko zmanjšajo verjetnost kolizij. Skupinam, ki ponujajo največ kombinacij, bomo pripisali višje uteži.

Za uteži je smiselno uporabiti kar razmerja med številom kombinacij, ki jih ponuja neka skupina, in celotnim naborom enoličnih identifikatorjev[4] (glej tabelo 3.2).

[4]Ob tem smo zanemarili neenakomerne porazdelitve tudi znotraj skupin.

Velikost skupine števk	Nabor identifikatorjev	Utež
0	44.928.000	0,0376
1	499.392.000	0,4175
2	374.400.000	0,313
3	277.440.000	0,2319

Tabela 3.2: Seznam uteži za posamezno velikost skupine števk.

Primerjava uspešnosti algoritmov z utežmi in brez njih sledi v dodatku A.2.

Kot zanimivost navajamo še nekaj primerov identifikatorjev razreda P (ustvarjenih s pomočjo algoritma 3.3): PIRIB753, PUP45ITO, PBEMA1HI ipd.

Kadar želimo v sistemu gostiti več kakor milijardo uporabnikov, je nujno potrebno spremeniti tudi algoritem ustvarjanja identifikatorjev. Ukrepamo lahko na več načinov: razširimo lahko nabor znakov (obstaja možnost, da v celoti opustimo zahtevo berljivosti identifikatorjev), prav tako lahko poskusimo nadgraditi algoritem z dodatnimi pravili ali pa preprosto povečamo število znakov v identifikatorjih.

Uvedba dvoglasnikov (npr.: AE, AI, EI, OU ipd.) – ali kar možnost dveh zaporednih samoglasnikov dodatno razširi nabor berljivih enoličnih identifikatorjev (to razširitev vsebuje tudi končni algoritem, glej dodatek A.3).

Predvidevamo, da je prostora za raziskave v okviru ustvarjanja naključnih berljivih identifikatorjih še veliko[18; 19]. V prihodnje načrtujemo dodatne raziskave, kjer bomo s statistično analizo slovenskega ter angleškega jezika poskusili dodatno izboljšati algoritem.

Prav tako bomo s pomočjo anket dosledneje opredelili kriterije berljivosti posameznih oblik identifikatorjev. Izdelati želimo tudi merila, ki bi natančneje opredelila stopnje težavnosti pomnjenja računalniško generiranih nizov. Pričakujemo, da bomo s pomočjo izsledkov v prihodnje lahko še dodatno razširili in izboljšali omenjene algoritme.

3.3 Zagotavljanje anonimnosti ob naslavljanju

Omenili smo, da lahko uporabniki s pomočjo identifikatorjev AtomID med seboj izmenjujemo podatke. Tri osnovne primere uporabe smo že omenili (glej diagrame 3.2, 3.3 in 3.4), vendar smo se jih dotaknili zgolj površno. V tem poglavju bomo posvetili pozornost predvsem fazi naslavljanja, ki predstavlja prvi korak mnogih transakcij izmenjave podatkov. Poskusili bomo natančneje opredeliti formalne zahteve in omejitve. Prav tako bomo preučili zahtevnost izvedbe – predvidevamo namreč, da bo slednje še posebej koristno ob kasnejših arhitekturnih odločitvah.

Že v uvodu smo predstavili zahtevo, da naslovi v sistemu AtomID ne smejo izdati ničesar o uporabniku samem – določen pomen naj imajo zgolj v mejah sistema AtomID. Toda kako lahko zagotovimo takšno tajnost? Trdimo, da naslednje omejitve zadostujejo zastavljeni zahtevi:

- *anonimnost naslavljanja*: ob izmenjavi naslovov oziroma identifikatorjev med uporabniki ne smejo biti razkriti nikakršni osebni podatki,
- *zadostnost identifikatorjev*: naj bo identifikator AtomID edina potrebna informacija za omogočanje izmenjave osebnih podatkov ter

- *nepovezljivost identifikatorjev s fizičnimi osebami*: tako ostalim uporabnikom kakor administratorjem sistema AtomID naj bo onemogočeno povezovanje naročnikov (fizičnih oseb) z uporabniki v sistemu oziroma njihovimi identifikatorji[5].

V podpoglavju 3.2.2 smo omenili, da ustvarjamo identifikatorje popolnoma naključno ter ob tem ne uporabljamo osebnih podatkov naročnikov. Iz tega sledi, da zgolj na podlagi identifikatorja ni možno izluščiti nikakršnih informacij. Če torej privzamemo, da ob izmenjavi naslovov uporabimo le identifikator AtomID (s tem privzemamo zahtevo zadostnosti identifikatorjev), bo prva zahteva (t.j. anonimnost naslavljanja) nemudoma izpolnjena.

Nekoliko težja naloga je formalno opisati rešitev, ki zagotavlja zadostnost identifikatorjev v namene izmenjave podatkov:

Naj bosta A in B uporabnika sistema AtomID. Brez škode za splošnost lahko predpostavimo, da uporabnik A želi poslati podatke uporabniku B. Oba poznata le identifikator drugega. Opazujemo tri osnovne načine prenosov podatkov:

1. Kadar želi A poslati podatke v nešifrirani obliki, je naloga preprosta, saj je potrebno podatke zgolj ustrezno pripeti AtomID računu uporabnika B.

2. V primeru asimetričnega šifriranja prav tako zadostuje B-jev identifikator, saj je z njegovo pomočjo možno pridobiti B-jev javni ključ. Uporabnik A zgolj šifrira podatke in jih prenese na enak način kakor v prvem primeru.

[5]Naročniki sistema AtomID so stranke podjetja, ki ponuja storitve AtomID. Zahteva nepovezljivosti identifikatorjev narekuje, da tudi to podjetje ne sme posedovati informacij o tem, katere AtomID identifikatorje si lastijo njihove stranke.

3. Žal ob simetričnem šifriranju naletimo na težavo, saj morata oba uporabnika poznati še skrivnost, ki je znana samo njima – skupno geslo. V tem primeru zgolj identifikator ne zadošča več. Uporabnika si morata na nek način izmenjati še skupno skrivnost[6].

Opazimo lahko, da je druga zahteva izvedljiva – vendar le v primeru, kadar podatke naslavljamo v nešifrirani obliki ali s pomočjo asimetrične enkripcije. Trdimo, da to ni bistvena omejitev, saj lahko uporabnikom v vsakem primeru izročimo asimetrične ključe. Prav tako je potrebno omeniti, da v prvem in v drugem primeru tuj identifikator potrebuje zgolj še uporabnik A (ki pošilja podatke).

Obravnavati je potrebno zgolj še izvedljivost zahteve nepovezljivosti identifikatorjev s fizičnimi osebami. Menimo, da je ta lastnost ena bistvenih prednosti sistema AtomID. Kadar dovolimo odprto registracijo uporabnikov (brez potrebe po plačevanju storitev), je naloga preprosta. V tem primeru namreč ne potrebujemo nikakršnih osebnih podatkov[7]. Naloga je nekoliko težja v primeru komercialne uporabe sistema[8]. V tem primeru je namreč nemogoče v popolnosti ustreči tretji zahtevi – lahko pa se ji z naslednjim postopkom zelo približamo: ob registraciji podjetje ustvari identifikator in uporabniški račun ter povezave med naročnikom in uporabniškim računom nikjer ne zavede. Slabost tega pristopa je, da ni možno ukiniti računa, kadar uporabnik prekine pogodbo ali naročnino – uporabniški račun ostane veljaven toliko časa, kolikor ga je določala politika

[6]Omeniti je potrebno, da si lahko uporabnika izmenjata skrivnost s pomočjo javnih ključev – vendar takšen pristop uvrščamo v kategorijo asimetričnega šifriranja (druga točka). Seveda si lahko uporabnika izmenjata skrivnost kar s pomočjo sistema AtomID, vendar bo v tem primeru skrivnost znana tudi skrbnikom sistema.

[7]V vsakem primeru spoznamo IP naslov uporabnika, kar pa s strani ponudnika sistema AtomID ni možno povsem odpraviti. Kljub vsemu lahko izberemo možnost, da si IP naslova preprosto ne zabeležimo.

[8]Za izstavo računa so kljub vsemu potrebni določeni osebni podatki.

sistema v času registracije.

3.4 Zagotavljanje E2E zaupnosti

V prejšnjih poglavjih smo omenili, da sistem AtomID zagotavlja zaupnost podatkov v vseh fazah izmenjave podatkov. Prav tako smo izpostavili, da zaupnost dosegamo s pomočjo asimetričnega šifriranja na odjemalski strani – t.j. lokalno v odjemalskem programu. Kljub temu je v namene implementacije potrebna še nekoliko natančnejša razčlenitev.

V okviru zaupne izmenjave podatkov lahko izpostavimo dve temeljni zahtevi, ki predstavljata osnovno vodilo pri načrtovanju in implementaciji nadaljnjih postopkov:

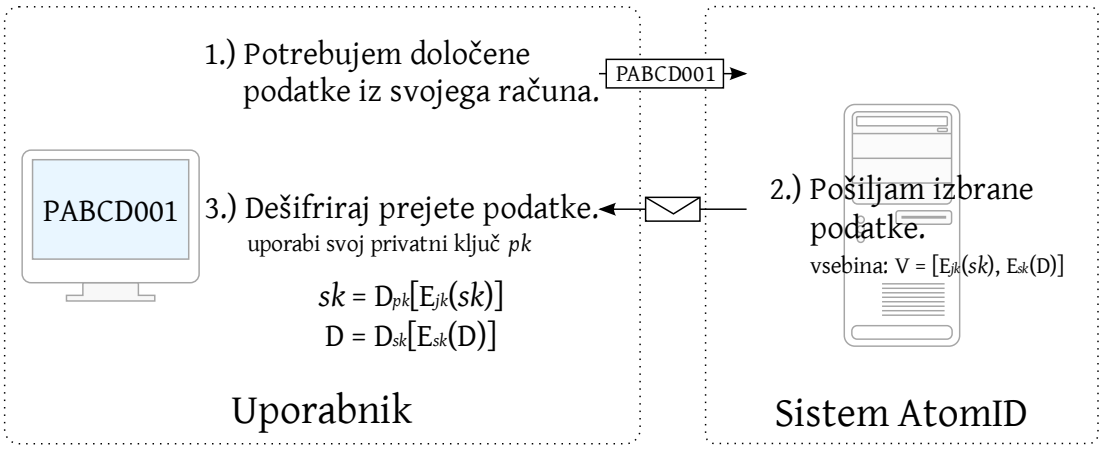
- podatki ne smejo v nešifrirani obliki zapustiti uporabnikovega računalnika,
- vsebina naj bo vidna zgolj in samo naslovnikom (ta zahteva zagotavlja, da vsebina podatkov ni dosegljiva nadzornikom sistema AtomID – seveda zgolj kadar nadzorniki niso med naslovniki).

Sedaj se lahko osredotočimo še na postopek izmenjave podatkov, ki ga prav tako lahko razdelimo na več delov:

1. *naslavljanje*: v tej fazi pošiljatelj pridobi naslov oziroma identifikator naslovnika (glej podpoglavje 3.3),

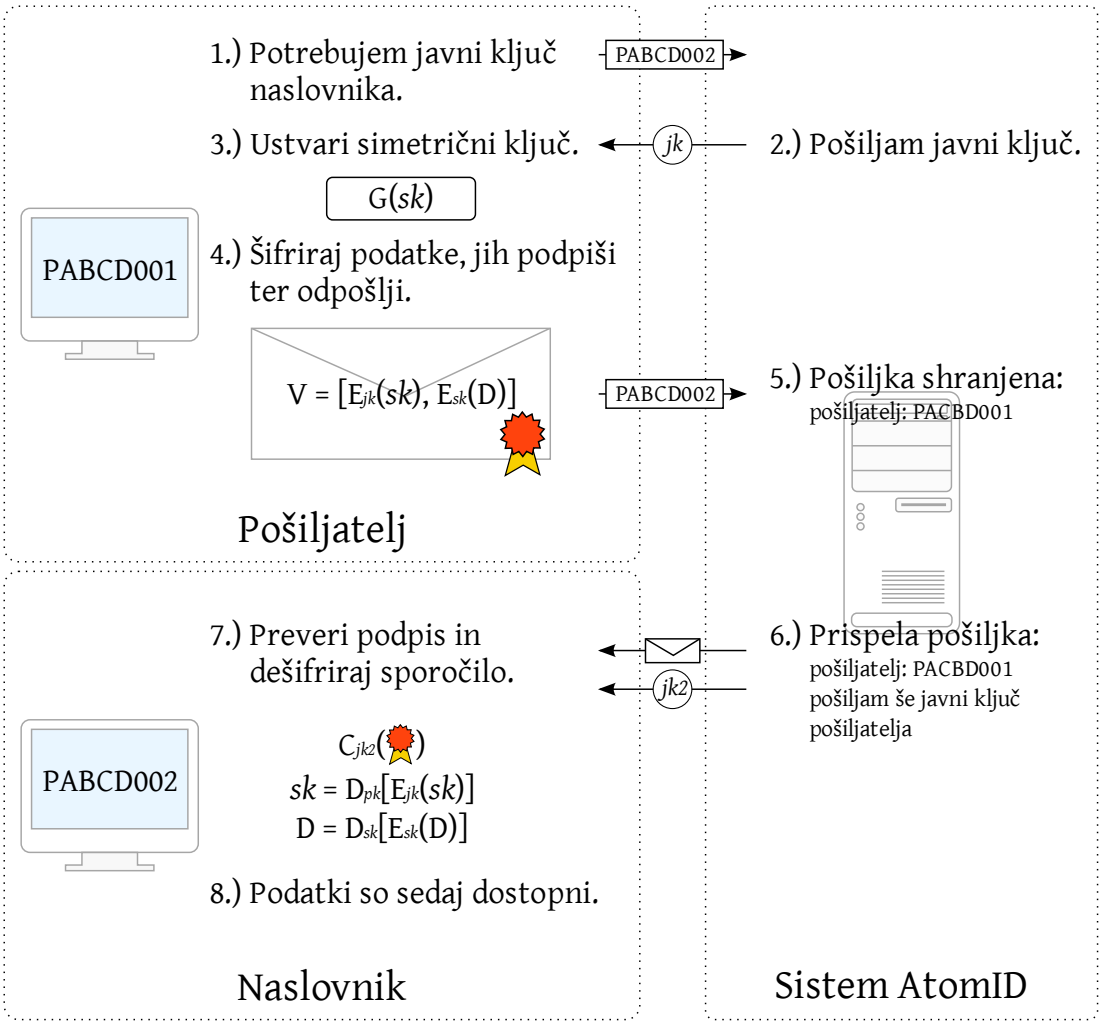
2. *pridobitev podatkov*: pošiljatelj s strežnika sname podatke, ki jih želi poslati naslovniku (če so podatki šifrirani, jih mora pošiljateljev odjemalski program najprej lokalno odkleniti, glej diagram 3.5) ter

3. *prenos podatkov*: odjemalec podatke po želji ponovno šifrira in jih s pomočjo sistema naslovi na izbran identifikator AtomID – postopek natančneje opisuje diagram 3.6.



Slika 3.5: Pridobitev lastnih podatkov.

S tem smo opredelili postopek pošiljanja podatkov zgolj enemu naslovniku. Preostane le vprašanje, kako omogočiti razpošiljanje paketov več naslovnikom hkrati. V ta namen potrebuje pošiljatelj javne ključe vseh naslovnikov – kar pomeni, da pošiljatelj večkrat ponovi korak 1 v diagramu 3.6. Nato ponovi še korak 4, in sicer tolikokrat, kolikor je naslovnikov. Seveda obstaja možnost, da pošiljatelj šifrira podatke s simetričnim ključem zgolj enkrat ter isti paket uporabi za vse naslovnike. Zagotoviti je potrebno zgolj, da v končni podatkovni vektor posameznega naslovnika vključimo še s pravilnim javnim ključem šifriran simetrični ključ. Prav tako moramo še vsako pošiljko posebej digitalno podpisati.



Slika 3.6: Potek zaupne izmenjave podatkov.

3.5 Samodejno ocenjevanje možnosti zlorabe

Ocenjevanje varnostnega tveganja v informacijskih sistemih je eden pomembnejših sestavnih delov načrtovanja in vzdrževanja ogrodij. V grobem delimo ocenjevanje ravni varnosti (glede na odsek sistema, kjer poteka obravnava) v tri večje skupine:

***ocenjevanje procesov in poslovne logike*:** poteka na razvojni in operativni

ravni v mejah organizacij oziroma podjetij. Varnostni nadzorniki ocenjujejo tako zasnovo in obratovanje sistema (v vseh fazah razvoja), kakor tudi poslovne procese oziroma poslovanje celotne organizacije[21].

Glede na dejstvo, da v sistemu AtomID hranimo v veliki meri le šifrirane podatke, ocenjujemo, da je tveganje zlorabe informacij bistveno nižje, kakor tveganje izgube podatkov iz malomarnosti – postopke, ki rešujejo omenjeni problem, bomo natančneje opredelili v podpoglavju 4.1.

***nadzor omrežja*:** nadzor omrežja je zelo efektiven način odkrivanja poskusov vdora. Za zagotavljanje sprotnega ocenjevanja ponuja kopico možnosti – uporabimo lahko na primer sprotno pregledovanje omrežnih podatkovnih paketov (tako imenovani nadzor omrežnega prometa oziroma *network traffic control*), analizo vzorcev uporabe omrežnih kanalov, preprečevanje vdorov s pomočjo učenja ipd.[22; 23]

***obravnava podatkovne shrambe*:** podatkovne shrambe so velikokrat najbolj občutljivi segment informacijskih sistemov. Glede na vsebino podatkovne shrambe lahko uporabimo več različnih pristopov sprotnega ocenjevanja varnosti: analizo vzorcev transakcij, preverjanje strukture poizvedb ipd. Kadar podatkovne baze vsebujejo nadzorne vnose (*audit entries*), si lahko pomagamo tudi s pomočjo različnih pristopov podatkovnega rudarjenja[24]. Omeniti je potrebno, da se v večini primerov poslužujemo algoritmov odkrivanja podatkovnih in transakcijskih anomalij.

V okviru te raziskave se bomo osredotočili predvsem na analizo podatkov v podatkovni shrambi. Uporabili bomo tri tipe sprotnega ocenjevanja varnosti: *ocenjevanje s pomočjo nadzornih vnosov*, *preverjanje razpršenosti neaktivnih*

osebnih podatkov in *analizo šifrirne moči.*

Sistem AtomID vzdržuje obsežne nadzorne vnose o sprotni rabi. Nadzorni zapisi vsebujejo podatke o uporabniku, o tipu transakcije ter o pričetku in trajanju celotne poizvedbe. S pomočjo analize pogostosti dostopov do podatkov lahko ocenimo, ali kakšen uporabnik lokalno hrani zaupne podatke. Prav tako lahko s pomočjo zaznanih anomalij (na primer izstopajočih dolžin ali tipov transakcij) obveščamo uporabnike o možni zlorabi osebnih podatkov, ki so jih zaupali uporabnikom, pri katerih zaznavamo izstopajoče načine uporabe sistema.

Drugi način ocenjevanja varnosti uvajamo s pomočjo analize razpršenosti in neaktivnosti podatkov. Pričakujemo, da bo razpršenost zaupanih osebnih podatkov s časom neprenehoma naraščala. Domnevamo pa, da višja stopnja razpršenosti doprinese k višji možnosti zlorabe podatkov. V namen krčenja razdrobljenosti podatkov v celotnem sistemu predlagamo algoritem, ki z uporabo analize pogostosti uporabe podatkov svetuje uporabnikom, katere podatke lahko zaradi daljše dobe neaktivnosti odstranijo. Izpis 3.4 prikazuje poenostavljeno osnovno strukturo algoritma.

Izpis 3.4: Algoritem za zmanjševanje razpršenosti neaktivnih podatkov.

```
1 zaupaniPodatki := najdiZaupanePodatke(uporabnik1);
2 for podatek in zaupaniPodatki do
3     seznamDostopov := preglejDostope(podatek);
4     if daljsaNeaktivnost(seznamDostopov) then
5         predlagajIzbris(podatek);
6     end
7 end
```

Prav tako je potrebno upoštevati še šifrirno moč ključev, s katerimi so podatki šifrirani. S tekom časa se namreč šifrirni ključi starajo in tudi varnostni standardi

se spreminjajo. Preprost algoritem (glej izpis 3.5) lahko bistveno pripomore k zagotavljanju varnosti skozi čas.

Izpis 3.5: Postopek preverjanja šifrirne moči podatkov.

```
1  zaupaniPodatki := najdiZaupanePodatke(uporabnik1);
2  lastniPodatki := najdiLastnePodatke(uporabnik1);
3  for podatek in zaupaniPodatki do
4      naslovnik := najdiNaslovnika(podatek);
5      simetricniKljuc := vrniSimetricniKljuc(podatek);
6      if nezadostnaMocKljuca(naslovnik) or ↩
          zastarelKljuc(naslovnik) then
7          obvesti(naslovnik);
8          obvesti(uporabnik1);
9      else if nezadostnaMocKljuca(simetricniKljuc) then
10         obvesti(uporabnik1);
11     end
12 end
13 if nezadostnaMocKljuca(uporabnik1) then
14     obvesti(uporabnik1);
15 else
16     for podatek in lastniPodatki do
17         simetricniKljuc := vrniSimetricniKljuc(↩
              podatek);
18         if nezadostnaMocKljuca(simetricniKljuc) then
19             obvesti(uporabnik1);
20         end
21     end
22 end
```

3.6 Fizično naslavljanje podatkov

V predhodnih poglavjih smo obravnavali zaupen in varen prenos podatkov s pomočjo računalniških komunikacijskih kanalov. Kljub velikemu razmahu digitalizacije domnevamo, da podjetja in njihove stranke pogosto poslujejo še s pomočjo tradicionalnih pristopov. V ta namen bomo predstavili še možnost fizičnega na-

slavljanja podatkov[9].

Prednost pošiljanja osebnih podatkov s pomočjo sistema AtomID je predvsem v tem, da lahko z digitalnim žigom časovno omejimo veljavnost podatkov ter zagotovimo avtentičnost in nespremenljivost vsebine[25].

Podobne lastnosti želimo ponuditi tudi v primeru, kadar naslavljamo podatke v fizični obliki.

Slika 3.7 prikazuje tako imenovan *podatkovni obrazec*, ki ga uporabnik izroči naslovniku (elektronsko ali fizično). Obrazec vsebuje dejanske podatke s komentarjem, informacije o pošiljatelju in prejemniku ter rok veljavnosti podatkov. Osrednjega pomena pa je dvodimenzionalna črtna koda[10], ki ob vseh podatkih v obrazcu vsebuje še pošiljateljev digitalni podpis. Izpis 3.6 prikazuje vsebino črtne kode iz prejšnjega primera.

Slika 3.7: Primer podatkovnega obrazca.

Čeprav sta v omenjenem primeru tako pošiljatelj kakor prejemnik uporabnika sistema AtomID, je dovoljeno naslavljanje tudi na osebe, ki niso lastniki računa

[9]Naslavljanje podatkov s pomočjo papirnatih obrazcev.

[10]Črtna koda v primeru 3.7 je bila izdelana s pomočjo javanske knjižnice *Barbecue*, glej spletno stran http://barbecue.sourceforge.net/. Tip črtne kode se imenuje PDF417

AtomID. Vsekakor pa je potrebno, da je vsaj pošiljatelj uporabnik sistema AtomID ter da uporablja svoj javni identifikator, saj je za preverjanje avtentičnosti žiga potreben javni ključ, ki ga lahko pridobimo zgolj z osebnimi identifikatorji v sistemu AtomID.

Izpis 3.6: Vsebina črtne kode s slike 3.7.

```
1 crtnaKoda = "PABCD001\tPABCD002\t1.7.2009\t" +
2             "i=Janez\tp=Novak\t" +
3             "n=Velika ulica 123\t" +
4             "ps=1234\tip=Srecna vas\n" +
5             base64(podpis);
```

S pomočjo podatkovnih obrazcev bi lahko preprečili nepooblaščeno uporabo osebnih podatkov tudi v primeru neelektronskega poslovanja. Podjetja ali državni uradi bi namreč lahko zahtevali od strank, da potrebujejo digitalna potrdila oseb, katerih pooblaščenci so.

Žal so takšni obrazci uporabni le, kadar ima organizacija, kjer želimo podatke vnovčiti, primeren čitalnik dvodimenzionalnih črtnih kod PDF417 in omrežni dostop do sistema AtomID. Prav tako ostane odprto vprašanje, kaj storiti z osebami, ki niso uporabniki sistema AtomID – od slednjih namreč podjetja ne morejo pričakovati, da bodo sposobni izstaviti digitalno podpisane podatkovne obrazce. Osnovna omejitev je torej, da bi morale prav vse potencialne stranke v tem primeru uporabljati sistem AtomID.

Seveda je možno predstavljen postopek fizičnega naslavljanja uvesti tudi s pomočjo manj kompleksnih sistemov, kakor je AtomID. S tem bi bilo mogoče nekoliko lažje uvesti omenjene varnostne ukrepe na nivoju celotne države.

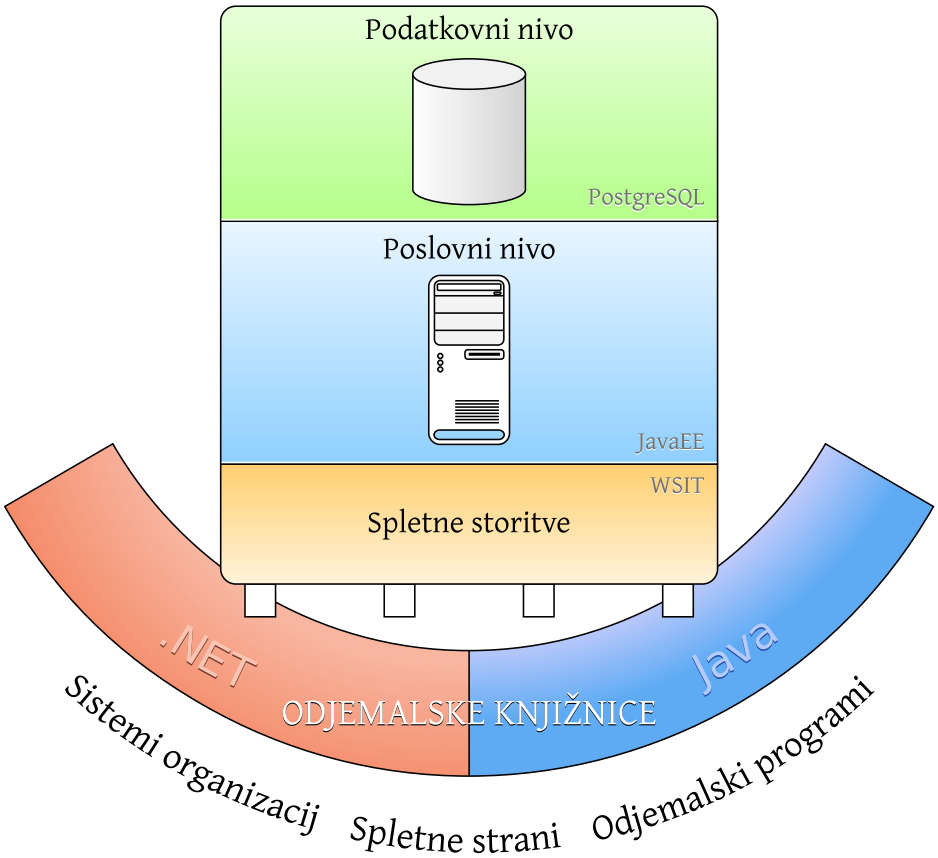
Poglavje 4

Arhitektura

Sistem AtomID smo zasnovali s pomočjo tronivojske arhitekture (glej diagram 4.1). Podatkovni nivo skrbi za hranjenje in celovitost šifriranih podatkov ter nadzornih vnosov. Poslovni nivo izpostavlja logične procese (naslavljanje, dostop in prenos podatkov ter urejanje računov) s pomočjo spletnih storitev.

Predstavitveni oziroma odjemalski nivo je glede na svojo vlogo v sistemu AtomID nekoliko nekonvencionalen (v primerjavi s svojo ustaljeno vlogo v večini večnivojskih sistemov). Razlog je odjemalsko šifriranje podatkov, kar pomeni, da je vsebina dostopna zgolj odjemalcem. Posledica je, da se vse procedure za obdelovanje vsebin podatkov izvajajo v odjemalskem programu (v tradicionalnih večnivojskih sistemih to vlogo načeloma prevzemata poslovni ali podatkovni nivo). Atomarnost, celovitost, izolacijo in trajnost podatkov in operacij kljub vsemu še vedno ponujata poslovni in podatkovni nivo.

Odjemalski programi in sistemi organizacij lahko uporabljajo vnaprej pripravljene odjemalske knjižnice, ki so prav tako sestavni del sistema AtomID. Pred-



Slika 4.1: Pregled arhitekture sistema AtomID.

stavljajo abstrakcijo nad spletnimi storitvami – s tem želimo poenostaviti delo s sistemom. Za višjo stopnjo prenosljivosti so odjemalske knjižnice izvedene v dveh različicah: s pomočjo ogrodij .NET in Java.

4.1 Podatkovna shramba

Osrednje orodje, ki ga uporabljamo za implementacijo podatkovne shrambe sistema AtomID, je relacijska podatkovna baza. Vsebuje seznam vseh naročnikov sistema z njihovimi javnimi ključi ter s soljo sekljanimi gesli. S tem ponujamo dva načina prijave v sistem AtomID: s pomočjo gesla ali asimetričnih ključev.

4.1.1 Upravljanje s podatki

Šifrirani ali nešifrirani osebni podatki so v vsakem primeru vezani na nek identifikator AtomID – s tem je določen dejanski lastnik podatkov. Prav tako smo omenili, da je naslovnik podatkov tudi identifikator AtomID. Pomembna lastnost sistema pa je v tem, da lahko nek identifikator pripada tudi organizacijam – kjer ima pravico do dostopa podatkov poljubno število članov organizacij. Slednje predstavlja problem, saj je ob prenosu podatkov k enemu samemu identifikatorju potrebno večkratno šifriranje – vsak uporabnik ima namreč svoj ključ (zgolj s tem pa lahko zagotovimo vzdrževanje nezanikljive evidence dostopov v nadzornih zapisih).

Omenjeno težavo lahko rešimo s pomočjo enkratnega šifriranja podatkov s simetričnim ključem in nato z večkratnim šifriranjem simetričnih ključev s certifikati posameznega uporabnika. V podatkovni shrambi je to možno rešiti s pomočjo povezav mnogo-mnogo med šifriranimi podatki in vsemi naslovniki (kjer je atribut povezave šifriran simetrični ključ).

Toda kaj storiti v primeru, kadar identifikator AtomID pridobi novega lastnika? Oziroma, kaj lahko storimo, kadar organizacija pooblasti novega člana za vpogled v podatke? Žal je glede na zahteve edini sprejemljivi način ustvarjanje dodatnih povezav med novim uporabnikom in vsemi podatki, povezanimi z identifikatorjem. Toda za to je potrebna intervencija katerega koli izmed naslovnikov ali celo lastnika podatkov, saj so zgolj ti sposobni dejansko dešifrirati šifrirni ključ in ga ponovno šifrirati z javnim ključem novega uporabnika. Drugi način, ki ga še ponuja sistem AtomID, je šifriranje s tako imenovanim skupnim javnim ključem. Ta ključ pripada AtomID identifikatorju in je dosegljiv vsem pooblaščenim la-

stnikom oziroma pooblaščenim članom organizacije, ki jim je zaupan privatni ključ.

Toda primarna vloga skupnih javnih ključev leži v anonimnih identifikatorjih, kjer se imenuje ad-hoc ključ. Kadar uporabnik anonimno naslavlja podatke, se ustvari kopija podatkov in nov identifikator AtomID, ki nima lastnika. Odjemalec prav tako ustvari še asimetrični par ključev, ki ga pripiše novo ustvarjenemu identifikatorju. Na tak način je zagotovljeno, da lastnik podatkov še vedno lahko dostopa do podatkov in jih spreminja, čeprav so podatki dejansko brez lastnika. Seveda se mora uporabnik ob tem izkazati za dejanskega lastnika – to lahko dosežemo s pomočjo digitalnega podpisa podatkov, ki ga strežnik pošlje kot izziv odjemalcu.

4.1.2 Iskanje podatkov

Iskanje po vsebini šifriranih podatkih je možno le, kadar jih lahko iskalni algoritem dešifrira. Žal so v sistemu AtomID vsi zaščiteni podatki šifrirani na odjemalski strani in so s tem nedostopni iskalnim algoritmom na strežniku. Da bi lahko kljub vsemu filtrirali podatke na strežniški strani, predlagamo rešitev s pomočjo zgoščevalnih algoritmov (*hash function*).

Metoda iskanja s pomočjo izvlečkov zahteva, da uporabnik ob šifriranih podatkih shrani še izvleček nešifrirane vsebine, ki ga izračuna s pomočjo določene zgoščevalne funkcije – na primer z algoritmom MD5 ali z družino SHA algoritmov. Iskanje sedaj ne poteka več s pomočjo primerjave vsebin, temveč s pomočjo preverjanja enakosti izvlečkov – računamo namreč na izredno veliko verjetnost, da sta ob enakih izvlečkih tudi izvorna niza enaka. Izpis 4.1 prikazuje primer

takšnega načina iskanja.

Izpis 4.1: Iskanje s pomočjo izvlečkov.

```
1 iskalniIzvlecek := sha256(iskalniNiz);
2 zadetki := [];
3 for podatek in najdiVsePodatke() do
4     if izvlecek(podatek) = iskalniIzvlecek then
5         dodaj(zadetki, id(podatek));
6     end
7 end
```

Toda takšen način iskanja močno omeji koristnost filtriranja – iščemo lahko namreč zgolj po enakosti dveh nizov. Dodatna razširitev bi bila, da nešifriran izvorni niz razdelimo na besede, izračunamo njihove zgoščene vrednosti in jih vse shranimo v podatkovno shrambo – s tem omogočimo iskanje po besedah iz vsebine (*full-text search*). Žal ima ta nadgradnja algoritma bistveno slabost. Kadar vsebino razdelimo na besede in njihove izvlečke shranimo v sistem, močno poenostavimo povratno sestavljanje vsebine besedila. V tem primeru namreč napadalcu ni potrebno ugibati vsebine celotnega niza, temveč le vsebino posameznih besed – kar je bistveno lažja naloga. Napadalcu seveda ne bo znan vrstni red izluščenih besed in število njihovih ponovitev, toda velikokrat je vsebina v zadostni meri razumljiva ne glede na vrstni red ali število ponovitev besed.

4.2 Strežniški sistem

Arhitektura nivoja poslovne logike je zasnovana s pomočjo spletnih storitev. Nivo je razdeljen v več segmentov, vsak prevzema določen tip funkcionalnosti:

- *upravljanje z uporabniškimi računi in pravicami*: prevzema administrativno

upravljanje uporabniških računov, kot je na primer ustvarjanje, brisanje in urejanje uporabnikov, uporabniških skupin in njihovih AtomID računov. Ta segment prav tako prevzema odgovornost za avtentikacijo uporabnikov v sistemu ter skrbi za dodeljevanje in upoštevanje uporabniških pravic.

- *sistem za varnost in upravljanje s ključi*: ta modul skrbi za samodejno ocenjevanje varnosti oziroma možnosti zlorabe ter skrbi za varno delo s ključi (preverjanje veljavnosti ključev, pridobivanje ključev, preverjanje digitalnih podpisov, spreminjanje ključev ipd.).

- *evidenca in nadzorni vnosi*: vodi evidenco o dostopih podatkov in splošni aktivnosti uporabnikov v sistemu.

- *delo s podatki*: skrbi za naslavljanje in varni prenos podatkov, ustvarjanje novih podatkovnih vnosov, filtriranje in pridobivanje podatkov.

Vsak izmed navedenih segmentov vsebuje množico spletnih storitev, ki predstavljajo vmesnik z zunanjim svetom – za izvedbo tega dela sistema uporabljamo tehnologijo WSIT. Implementacija poslovne logike je izvedena s pomočjo tehnologije EJB 3.0, komunikacija s podatkovnim nivojem poteka z uporabo objektno/relacijskega sistema JPA.

4.3 Odjemalske knjižnice

Enostavnost integracije v obstoječe sisteme je v veliki meri odvisna od odjemalskih knjižnic, ki jih razvijalci uporabljajo v namene komunikacije s sistemom AtomID. Pomembno je torej, da so odjemalske knjižnice čim preprostejše za

uporabo, da ponujajo čim večji nabor funkcionalnosti in da so prilagodljive za vsak scenarij uporabe. Seveda morajo knjižnice ob tem ohraniti tudi čim večjo kompaktnost in hitrost izvajanja – ponuditi želimo lahke knjižnice (*lightweight libraries*), ki ne zahtevajo veliko virov (prostorskih ali časovnih).

Sistem AtomID ponuja možnost neposrednega dostopa do funkcionalnosti s pomočjo spletnih storitev (*Web Services*). Kljub temu predvidevamo, da je smiselno ponuditi še dodatni nivo abstrakcije. Osnovni razlog za takšno odločitev je, da ob razvoju okolij s pomočjo direktnih dostopov do spletnih storitev potrebujemo množico dodatnih korakov, ki se jim je sicer možno izogniti.

Sledi natančnejša primerjava bistvenih razlik med direktno uporabo spletnih storitev in uporabo dodatne plasti abstrakcije:

Lastnost	**Spletne storitve**	**Abstrakcijska plast**
razvojni koraki	Uvažanje oddaljenih vmesnikov (s pomočjo spletnih naslovov in WSDL dokumentov).	Uvoz aplikacijske knjižnice.
inicializacijski postopki	Potrebni dodatni koraki za vsako invokacijo spletne storitve.	Inicializacija je potrebna zgolj enkrat (kadar ustvarimo osnovni kontekstni objekt[1]).

[1]V odjemalski knjižnici AtomID smo uporabili pristop, kjer en primerek razreda predstavlja vstopno točko za vse različne segmente sistema. Kadar ustvarimo ta primerek, istočasno tudi inicializiramo celotno knjižnico.

Lastnost	**Spletne storitve**	**Abstrakcijska plast**
granularnost postopkov	Visoka – kadar postopek ključuje več invokacij, je potrebno vsako izvesti posebej.	Nizka – pogosti postopki in sekvence so lahko združeni v eno metodo.
prevajanje argumentov	Razvijalec mora lastnoročno preverjati veljavnost argumentov, ki jih pošilja spletni storitvi – za vsako invokacijo.	Prevzema ovojna metoda.
hitrost izvajanja	Inicializacija spletnih storitev je dolgotrajna, saj zahteva omrežno komunikacijo s strežnikom. Rešitev je enkratna inicializacija in ponovna uporaba primerka – s predpomnenjem.	Rešuje omenjen problem s pomočjo predpomnenja.

Lastnost	Spletne storitve	Abstrakcijska plast
strošek režije	Minimalen – potrebni so zgolj standardni postopki upravljanja s spletnimi storitvami.	Čeprav vzdrževanje kontekstnega aplikacijskega objekta predstavlja nek strošek režije (tako časovno kakor tudi prostorsko), domnevamo, da je kljub vsemu tako majhen, da v produkcijskih sistemih ne bo zaznaven. Časovni stroški pa so prisotni le ob inicializaciji.
dodatni stroški razvoja	Nizki za razvijalce sitema AtomID, visoki za integratorje.	Visoki za razvijalce sitema AtomID, nizki za integratorje.

Tabela 4.1: Primerjava direktne uporabe spletnih storitev in dodatne plasti abstrakcije.

Osrednji razred odjemalske knjižnice AtomID je `ApplicationManager`. Razvijalci morajo ustvariti zgolj en primerek tega razreda, ki predstavlja osnovni kontekstni objekt. Nato je vsa funkcionalnost sistema AtomID dosegljiva s pomočjo ostalih objektov, ki jih preko instančnih metod ponuja kontekstni objekt.

Primer programa, ki izpiše vse uporabnike sistema AtomID:

Izpis 4.2: Primer uporabe odjemalske knjižnice AtomID.

```
1 ApplicationManager am = new ApplicationManager();
```

```
2 am.getAuthenticationManager().login("Administrator", ↩
     "geslo");
3 List<String> result = am.getUserManager().↩
     getAllUsernames();
4 for (String string : result) {
5     System.out.println(string);
6 }
```

Primer z neposredno uporabo spletnih storitev je neprimerno daljši, zato ga ne navajamo.

Poglavje 5

Izvedba

Izvedba prototipskega sistema z odjemalskimi aplikacijami je zaključni korak tega diplomskega dela. S pomočjo prototipa želimo potrditi izvedljivost zasnovane arhitekture in preveriti delovanje sistema v izbranem realnem okolju.

Ob izvedbi podatkovne plasti smo uporabili sistem za upravljanje z relacijskimi podatkovnimi bazami PostgreSQL. Prototipna relacijska podatkovna shema vsebuje 22 entitet za zagotavljanje vseh opredeljenih zahtev iz prejšnjih poglavij.

Nivo poslovne logike smo izgradili s pomočjo JavaEE 5 aplikacijskega strežnika GlassFish Enterprise Server v2.1.

Odjemalske knjižnice so morda najpomembnejši del prototipskega sistema. Enostavnost integracije v obstoječe sisteme je v veliki meri odvisna od kvalitete izvedbe odjemalskih knjižnic. Žal je bilo v omejenem času raziskave nemogoče preveriti kvaliteto knjižnic z uporabo empirične analize. Osredotočili smo se

na izdelavo prototipskega sistema z odjemalskimi knjižnicami v okviru študije primera aplikacije v zdravstvu[26].

5.1 Prototipska aplikacija v zdravstvu

Tajnost osebnih podatkov je zelo pomemben vidik zagotavljanja zaupnosti v zdravstvu. Prav tako je priporočljivo, da so informacije o bolnikih preprosto dosegljive – vendar ne zgolj osebnim zdravnikom in bolnikom, temveč tudi ostalim, ki so prisotni v postopkih zdravljenja (na primer mobilna nujna pomoč in zdravstveni laboratoriji). Tajnost in preprosta dosegljivost podatkov sta na videz močno nasprotujoči si zahtevi – razumljivo je torej, da se s to težavo trenutno ukvarja mnogo raziskav o računalniških sistemih za upravljanje z osebnimi zdravstvenimi podatki oziroma kartotekami bolnikov[27; 28].

Večina ogrodij PHR zagotavlja varnost osebnih podatkov s pomočjo fizičnega varovanja podatkovne shrambe, varnih komunikacijskih kanalov in obljube, da vzdrževalci sistema ne bodo prodajali vsebine tretjim osebam oziroma nepooblaščenim organizacijam[1].

Na podlagi omenjenih zahtev (tajnosti in preproste dosegljivosti podatkov) lahko izluščimo nekaj primerov uporabe, preučimo trenutne pristope ter preverimo uporabnost sistema AtomID kot podatkovnega podsistema ogrodij PHR. Med seboj bomo primerjali trenutne rešitve s pametnimi oziroma zdravstvenimi karticami[29], spletne sisteme (uporabili bomo primer *Google Health*) ter sisteme podprte z ogrodjem AtomID.

[1] Eden novejših sistemov PHR, ki prav tako obljublja, da podatki ne bodo zlorabljeni s strani ponudnika, je *Google Health*.

5.1.1 Dostop do podatkov ob obisku bolnika

Pametne kartice: Pametne kartice so sicer zelo zmogljiva rešitev, vendar so v večini trenutnih sistemov izkoriščene zgolj v namene enofaktorske avtentikacije (bolnik se identificira z nečim, kar ima). Prav tako so podatki v podatkovnih shrambah v večini primerov nešifrirani. Če pa so podatki ob tem scenariju tudi šifrirani, pa postanejo težko dosegljivi[2]. Omejitve tega pristopa so torej omejeno zagotavljanje tajnosti podatkov ali težka dosegljivost podatkov.

Spletni sistemi: Spletni sistemi ponujajo podatkovne shrambe in avtentikacijske postopke, temelječe na uporabniških imenih in geslih. V tem primeru prevzema poslovna logika aplikacijskega nivoja sistemov vso odgovornost omejevanja dostopov do podatkov (na primer s pomočjo uporabniških imen). Tudi v tem primeru podatki niso šifrirani in so vidni nadzornikom sistema. Prednost pa je enostavnost dostopov do podatkov in preprosto naslavljanje.

Ponudniki teh sistemov z obljubo zagotavljajo, da bolnikovih podatkov ne bodo zlorabljali (na primer *Google Health*).

Sistem AtomID: Omenili smo, da so podatki v sistemu AtomID lokalno šifrirani s ključi uporabnikov – kar pomeni, da je za tajnost podatkov poskrbljeno tudi v podatkovni shrambi (vzdrževalci sistema s tem nimajo vpogleda v zasebne vsebine kartotek bolnikov). Sistem AtomID rešuje tudi problem dostopov, kadar bolnik nima s seboj pametne kartice (uporabnika lahko s pomočjo slik v sistemu AtomID identificira kar zdravnik ali pa se bolnik identificira s pomočjo gesla).

[2]Na primer: kadar je uporabnik brez pametne kartice ali kadar zdravnik potrebuje podatke ob bolnikovi odsotnosti.

Naslavljanje poteka tako, da bolnik zaupa osebne podatke zgolj svojemu zdravniku s pomočjo omenjenih E2E pristopov (glej podpoglavje 3.4). Kadar uporabnik nima omrežnega dostopa ali morda celo osebnega računalnika, lahko s pomočjo pametnih kartic to uredi na zdravniških terminalih ali pa zanj to uredijo referenti sistema (to je potrebno storiti le, kadar bolnik zamenja osebnega zdravnika, v vsakem primeru pa je potrebna pametna kartica ali osebni ključek s certifikatom).

5.1.2 Dostop do podatkov v odsotnosti bolnika

Pametne kartice: Kadar so podatki šifrirani z bolnikovim osebnim ključem (ki je vsebovan v pametni kartici), so podatki brez zdravstvene kartice nedosegljivi. Zdravnik potrebuje kartico ali pa morajo biti podatki v sistemu shranjeni v nešifrirani obliki.

Ena izmed možnih rešitev je, da v sistemu hranijo podvojene različice podatkov: ena različica je šifrirana z zdravnikovim ključem, druga različica pa z osebnim ključem bolnika (morda celo fizično shranjena na bolnikovi zdravstveni kartici).

Seveda v primeru nešifriranih podatkov težav z dostopnostjo podatkov ni. Žal v tem primeru ni možno zagotoviti popolne tajnosti podatkov.

Spletni sistemi: V okviru spletnih sistemov ni težav ob dostopu do podatkov v bolnikovi odsotnosti, saj podatki v sistemu niso šifrirani z osebnimi ključi.

Bolnik mora v uporabniškem vmesniku sistema označiti, da ciljnim institu-

cijam dovoli uporabo svojih zdravstvenih podatkov. Šele nato lahko naslovniki dostopajo do podatkov kadarkoli – brez potrebe po nadaljnjem bolnikovem posredovanju.

Sistem AtomID: Sistem AtomID združuje dobre lastnosti obeh svetov: šifriranje podatkov s pomočjo javnih kriptografskih ključev in z naslavljanjem podatkov za enostavnost dostopov v odsotnosti bolnika.

Čeprav so podatki šifrirani, je možno s pomočjo postopkov, opisanih v podpoglavjih 3.2 in 3.4, doseči enakovredno mero funkcionalnosti spletnih sistemov in ob tem ohraniti tajnost podatkov.

5.1.3 Dostop do podatkov ob nujni pomoči

Ob nujni pomoči je potrebno zagotoviti dostop do podatkov različnim osebam, izvajalcem nujne pomoči – neodvisno od odzivnosti bolnika ter kraja in časa zdravljenja.

V vsakem primeru potrebujejo izvajalci nujne pomoči nekaj informacij o bolniku – šele s pravimi podatki lahko poiščejo kartoteke bolnika.

Pametne kartice: Prednost pametnih kartic je, da lahko že same vsebujejo podatke o bolniku – s tem je odstranjena zahteva po omrežnem dostopu. Ob takšnem načinu uporabe je potrebno zagotoviti, da so podatki na pametni kartici ažurni.

Seveda je nemogoče identificirati nezavestnega bolnika, kadar ta nima s seboj

identifikacijskih dokumentov ali zdravstvene kartice.

Spletni sistemi: Ta pristop zahteva omrežni dostop do sistema PHR – terenski zdravniški posegi so s tem nekoliko omejeni (posebej na lokacijah, kjer je povezljivost z internetnim omrežjem močno omejena). V tem primeru prav tako potrebujemo informacije o uporabniškem računu bolnika na kraju nesreče – žal nezavestni bolniki le stežka posredujejo te informacije, zato v takšnih primerih ni možno uporabiti tega pristopa.

Osnovna značilnost spletnih sistemov je, da morajo uporabniki dovoliti uporabo podatkov izbranim naslovnikom. Ta problem lahko rešimo tako, da ob nujni pomoči privzeto dovolimo dostop do izbranih podatkov, ki jih bodo predvidoma potrebovali izvajalci. Žal je v tem primeru težko določiti, kdo bo izvajalec nujne pomoči.

Seveda spletni sistemi ne izključujejo možnosti uporabe pametnih kartic.

Sistem AtomID: Tudi v tem primeru se lahko poslužujemo uporabe pametnih kartic, ki vsebujejo podatke o bolniku.

Sistem AtomID prav tako omogoča zaupno naslavljanje podatkov nujni pomoči. Izvajalci nujne pomoči imajo skupni ključ, s katerim odjemalski programi bolnika privzeto šifrirajo in naslovijo podatke. S tem so nujni podatki dosegljivi vsem izvajalcem nujne pomoči. Dodatna lastnost, ki jo lahko ob tem uporabimo, je edinstvena sposobnost sistema AtomID za anonimno naslavljanje (glej podpoglavje 3.3). S tem lahko zaupamo podatke široki skupini izvajalcem nujne pomoči, ne da bi izdali svojo identiteto.

5.1.4 Prenos podatkov med več udeleženci

Izvidi kakršnih koli zdravniških pregledov so prav tako del zaupnih osebnih podatkov. Kadar podatki potujejo med bolnikom, zdravniki in ostalimi udeleženci v zdravstvenem postopku, je potrebno zagotoviti, da vsebina ali kopija vsebine ne preideta v last nepooblaščenim osebam.

Osnovna težava v okviru tega primera uporabe je, da je v proces vključenih več naslovnikov (ne več zgolj dva).

Pametne kartice: S pomočjo pametnih kartic je možno prebiranje podatkov, ki so šifrirani z uporabnikovim javnim ključem. Toda v večini primerov sistemi ne šifrirajo podatkov zgolj s ključi naslovnikov – podatki so morda namenjeni zgolj izbranim osebam, vendar so vidni tudi administratorjem in ostalim, ki imajo dostop do sistemske podatkovne shrambe.

Prav tako je težko zagotoviti, da bi identiteta bolnika ostala nerazkrita ostalim udeležencem v procesu zdravljenja (tistim, ki za svoje delo ne potrebujejo osebnih podatkov bolnika – imena, priimka, naslova ipd.).

Spletni sistemi: Podatki s pomočjo omejevanja dostopov potujejo med uporabniškimi računi kakor v spletni pošti. Podatki so nešifrirani in tudi v tem primeru je težko zagotoviti, da identiteta bolnika ni razkrita, kadar to ni potrebno.

Sistem AtomID: Podatki v sistemu AtomID so zmeraj šifrirani s ključi naslovnikov – s tem je zagotovljeno, da je vzdrževalcem sistema onemogočen dostop

do vsebine podatkov.

Da identiteta bolnika ostane prikrita, lahko uporabimo anonimno naslavljanje podatkov (glej podpoglavje 3.3).

5.1.5 Izguba ključev

Če shranjujemo podatke v šifrirani obliki, je vsebina praktično izgubljena natanko tedaj, kadar izgubimo vse šifrirne ključe. Temu se je možno izogniti zgolj z varnostnimi dvojniki ključev ali kopijami podatkov v nešifrirani obliki. Težava teh rešitev je, da podatki ponovno postanejo vidni vzdrževalcem sistema. Smiselno je torej, da so dvojniki ključev ali nešifrirani podatki shranjeni tako, da brez posebnih postopkov podatkov ni možno dešifrirati (na primer s pomočjo dislokacije ključev dvojnikov) oziroma, da nešifrirane kopije podatkov ni možno preprosto povezati z identiteto bolnika (na primer z raztresanjem in šifriranjem fizično ločenih segmentov nešifriranih kopij).

Pametne kartice: Uporaba pametnih kartic ne pomeni nujno, da so v tem primeru podatki v podatkovni shrambi tudi šifrirani. Kadar so podatki šifrirani, lahko izguba kartice pomeni izgubo podatkov. To težavo trenutni sistemi rešujejo na dva načina: z vzdrževanjem kopije podatkov v nešifrirani obliki ali s hranjenjem dvojnikov uporabnikovega ključa.

Spletni sistemi: Podatki v teh sistemih niso šifrirani, zato v tem primeru izguba ključev še ne pomeni izgube podatkov. Kadar uporabnik izgubi dostopno geslo, lahko od ponudnika sistema zahteva dodelitev novega.

Sistem AtomID: Podatki so šifrirani, vendar v večini primerov z več ključi (odvisno od tega, koliko je naslovnikov). Kadar bolnik izgubi ključ in ima dostop do njegovih podatkov še zdravnik (ali kdor koli drug), je podatke še možno dešifrirati in jih ponovno šifrirati z novim ključem uporabnika. V ostalih primerih lahko ukrepamo zgolj s pomočjo zgoraj omenjenih pristopov podvajanja podatkov ali šifrirnih ključev.

Poglavje 6

Sklep

Osnovni rezultat te raziskave je opredelitev in zasnova arhitekture porazdeljenega ogrodja za upravljanje z osebnimi podatki. Bistvena lastnost sistema je, da za omogočanje zaupnega naslavljanja in tajnega prenosa podatkov uporablja varne identifikatorje AtomID ter metode lokalnega odjemalskega šifriranja. Organizacije lahko uporabljajo ta sistem v namene varne shrambe osebnih podatkov ter zaupne izmenjave podatkov (tako med včlanjenimi organizacijami kakor tudi z ostalimi osebami).

Omejitve uporabljenih pristopov smo reševali s pomočjo izsledkov obstoječih raziskav, na primer: težave ob odjemalskem šifriranju podatkov (poglavje 2.1), omejitve upravljanja s šifriranimi podatki in ključi (poglavje 2.2), zahteve ob šifriranju osebnih identifikatorjev (poglavje 2.3) in podobno.

Opredelili smo tudi inovativne tehnologije, na podlagi katerih smo omogočili edinstvene funkcionalnosti sistema AtomID. Tehnologija varnih identifikatorjev (glej poglavje 3.2) ponuja množico izvirnih primerov uporabe, kakor na primer anonimno naslavljanje (opisano v poglavju 3.3), sistem za samodejno ocenjevanje

varnosti (poglavje 3.5) ter fizično naslavljanje podatkov (poglavje 3.6). Omenjena tehnologija zahteva tudi posebne metode ob ustvarjanju naključnih berljivih identifikatorjev – kar smo natančneje raziskali v poglavju 3.2.2.

Zaupna zasnova sistema AtomID prav tako omejuje določene funkcionalnosti, ki so sicer zelo pomembne v mnogih primerih uporabe. Iskanje po vsebini podatkov je morda najpomembnejši takšen aspekt – ključna težava je namreč, da je iskanje po vsebini šifriranih podatkov na strežniški strani močno oteženo. Kljub temu smo v poglavju 4.1.2 preverili nekaj rešitev, ki zadovoljivo rešujejo tudi ta problem.

Ob koncu smo s pomočjo študije primera preverili tudi delovanje prototipskega sistema v zdravstvu. V poglavju 5 smo opredelili več primerov uporabe in primerjali rešitev AtomID z ostalimi obstoječimi razredi sistemov. Opazovali smo predvsem zmogljivost ob zagotavljanju zasebnosti in tajnosti osebnih podatkov. Ugotovili smo, da je bilo anonimno naslavljanje podatkov še posebej koristno v določenih primerih uporabe.

Kljub vsemu ostaja še nekaj odprtih vprašanj. V prihodnosti želimo preveriti, kako lahko preprečimo izgubo podatkov v primeru izgubljenih šifrirnih ključev. Prav tako želimo raziskati možnosti dodatnih izboljšav iskalnih algoritmov in algoritmov ustvarjanja naključnih berljivih identifikatorjev. Pričakujemo tudi, da bo z empirično analizo možno v prihodnje oceniti še hitrost delovanja sistema v različnih realnih scenarijih.

Domnevamo, da bo zaupnost osebnih podatkov predstavljala veliko preizkušnjo ob zasnovi varnih računalniških sistemov tudi v prihodnje. Pričakujemo torej, da bodo izsledki te raziskave še posebej koristni v nadaljnjih projektih.

Dodatek A

Algoritmi in izračuni

A.1 Izračun izhodnih kombinacij algoritma 3.3

Prešteti želimo število različnih možnih kombinacij identifikatorjev ustvarjenih s pomočjo algoritma 3.3.

Pregledali bomo ločene primere, kadar v identifikatorju nastopa skupina z 0, 1, 2 ali 3 števkami. Omeniti je potrebno, da je prva črka identifikatorja rezerzvirana za označbo razreda, kar pomeni, da opazujemo nize sedmih znakov.

Brez števk: Kadar v identifikatorju ni števk, bodo nastopali zgolj izmenjujoči se soglasniki in samoglasniki. Imamo dve možnosti: prva črka je samoglasnik ali soglasnik.

$$N_0 = 20^4 \cdot 6^3 + 20^3 \cdot 6^4 = 44.928.000 \tag{A.1}$$

Ena števka: V tem primeru imamo štiri možnosti, kjer se edina števka lahko pojavi na prvem in zadnjem mestu (A.2), na drugem in predzadnjem (A.3), na tretjem in petem (A.4) ter na sredini (A.5).

$$
\begin{aligned}
N_1 &= 2 \cdot 10 \cdot (2 \cdot 20^3 \cdot 6^3) + && \text{(A.2)}\\
&+ 2 \cdot 10 \cdot 26 \cdot (20^3 \cdot 6^2 + 20^2 \cdot 6^3) + && \text{(A.3)}\\
&+ 2 \cdot 10 \cdot (2 \cdot 20^1 \cdot 6^1) \cdot (2 \cdot 20^2 \cdot 6^2) + && \text{(A.4)}\\
&+ 10 \cdot (20^2 \cdot 6^1 + 20^1 \cdot 6^2)^2 && \text{(A.5)}\\
&= 499.392.000 && \text{(A.6)}
\end{aligned}
$$

Dve števki: V tem primeru imamo tri možnosti, kjer se števki lahko pojavita na indeksih (1, 2) ali (6, 7) (A.7), na (2, 3) ali (5, 6) (A.8) ter na (3, 4) ali (4, 5) (A.9).

$$
\begin{aligned}
N_2 &= 2 \cdot 100 \cdot (20^3 \cdot 6^2 + 20^2 \cdot 6^3) + && \text{(A.7)}\\
&+ 2 \cdot 100 \cdot 26 \cdot (2 \cdot 20^2 \cdot 6^2) + && \text{(A.8)}\\
&+ 2 \cdot 100 \cdot (2 \cdot 20^1 \cdot 6^1) \cdot (20^2 \cdot 6^1 + 20^1 \cdot 6^2) + && \text{(A.9)}\\
&= 374.400.000 && \text{(A.10)}
\end{aligned}
$$

Tri števke: V tem primeru imamo ponovno tri možnosti, kjer se števke lahko pojavito na indeksih (1, 2, 3) ali (5 6, 7) (A.11), na (2, 3, 4) ali (4, 5, 6) (A.12) ter na sredini (A.13).

$$
\begin{aligned}
N_3 &= 2 \cdot 1000 \cdot (2 \cdot 20^2 \cdot 6^2) + && \text{(A.11)}\\
&+ 2 \cdot 1000 \cdot 26 \cdot (20^2 \cdot 6^1 + 20^1 \cdot 6^2) + && \text{(A.12)}\\
&+ 1000 \cdot (2 \cdot 20^1 \cdot 6^1)^2 + && \text{(A.13)}\\
&= 277.440.000 && \text{(A.14)}
\end{aligned}
$$

Celotno število različnih kombinacij identifikatorjev:

$$N = N_0 + N_1 + N_2 + N_3 = 1.196.160.000. \tag{A.15}$$

A.2 Primerjava obteženega in neobteženega algoritma

V namene primerjave smo uporabili algoritem A.1. Ob neobteženi različici smo ustvarili še obteženo in s pomočjo empirične analize določili uteži. Osnovna ideja postopka določanja uteži je bila v tem, da smo želeli doseči enakomerno porazdelitev kolizij med posameznimi skupinami.

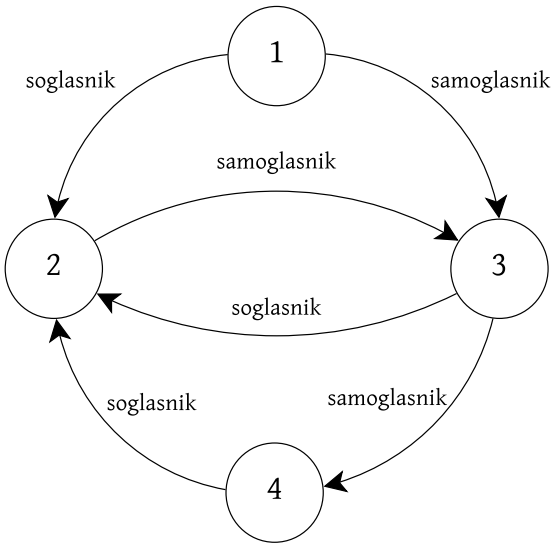
Izkaže se, da za algoritem A.1 ustrezajo uteži prikazane v tabeli A.1.

Velikost skupine števk	**Utež**
0	0,232
1	0,282
2	0,282
3	0,204

Tabela A.1: Uteži prilagojene za algoritem A.1.

Ob zaporednem ustvarjanju 1.000.000 identifikatorjev, se je obtežen algoritem obnesel 6.4 % bolje kakor neobtežen algoritem (ob ustvarjanju identifikatorjev s prvim algoritmom je prišlo v povprečju do 502 koliziji, ob drugem pa v povprečju do 536,3 kolizij).

A.3 Končna različica algoritma za ustvarjanje identifikatorjev



Slika A.1: Nedeterministični končni avtomat za ustvarjanje berljivega niza črk v identifikatorjih.

Izpis A.1: Končna različica algoritma za ustvarjanje berljivih identifikatorjev.

```
1  /**
2   * Sestavi berljiv identifikator AtomID.
3   * @param razred crka razreda AtomID identifikatorja.
4   * @param dolzina dolzina celotnega AtomID identifikatorja.
5   * @param maxStStevk najvecje stevilo stevk v gruci stevk (v
6   * identifikatorju se lahko pojavi najvec ena skupina stevk).
7   * @return berljiv identifikator AtomID.
8   */
9  public static String ustvariAtomId(char razred, int dolzina, ↩
       int maxStStevk) {
10     Random rnd = new Random();
11     StringBuilder sb = new StringBuilder(dolzina);
12     sb.append(razred);
13     // Stevilo stevk ne sme presegati dolzino celotnega ↩
          identifikatorja
14     maxStStevk = Math.min(dolzina - 1, maxStStevk);
15     // Koliko stevk naj bo v tem identifikatorju?
16     int stStevk = rnd.nextInt(maxStStevk + 1);
17     // Na katerm mestu naj se pojavi skupina stevk?
18     int idxStevk = rnd.nextInt(dolzina - maxStStevk) + 1;
19     int stanje = ustvariNizCrk(idxStevk - 1, 1, rnd, sb);
20     for (int i = 0; i < stStevk; i++) {
21         sb.append((char) (rnd.nextInt(10) + '0'));
22     }
23     stanje = ustvariNizCrk(dolzina - idxStevk - stStevk, (↩
          stStevk == 0) ? 1 : stanje, rnd, sb);
24     return sb.toString();
25 }
26
```

```
27 /**
28  * Ustvari niz crk (soglasniki in samoglasniki si sledijo
29  * izmenicno, samoglasniki se lahko pojavijo v parih).
30  * @param dolzina dolzina ustvarjenega niza.
31  * @param stanje zacetno stanje koncnega avtomata.
32  * @param rnd generator nakljucnih stevil s pomocjo katerega
33  * izbiramo crke.
34  * @param sb objekt, kamor vstavljamo crke niza.
35  * @return koncno stanje avtomata.
36  * @throws java.lang.RuntimeException izjema se sprozi, kadar
37  * koncni avtomat preide v nedovoljeno stanje.
38  */
39 private static int ustvariNizCrk(int dolzina, int stanje, ↩
       Random rnd, StringBuilder sb) throws RuntimeException {
40     // Stanje doloca, iz katerega nabora lahko izberemo ↩
          naslednji znak (glej diagram koncnega avtomata).
41     int nakljucnoSt;
42     for (int i = 0; i < dolzina; i++) {
43         switch (stanje) {
44             case 1:
45                 // V stanju 1 lahko vstavimo tako soglasnik ↩
                      kakor samoglasnik
46                 nakljucnoSt = rnd.nextInt(26);
47                 if (nakljucnoSt < 6) {
48                     // Vstavili bomo samoglasnik in presli v ↩
                          stanje 3
49                     sb.append(samoglasniki[nakljucnoSt]);
50                     stanje = 3;
51                 } else {
52                     // Vstavili bomo soglasnik in presli v ↩
                          stanje 2
53                     sb.append(soglasniki[nakljucnoSt - 6]);
54                     stanje = 2;
55                 }
56                 break;
57             case 2:
58                 // V stanju 2 lahko vstavimo zgolj samoglasnik ↩
                      in preidemo v stanje 3
59                 sb.append(samoglasniki[rnd.nextInt(6)]);
60                 stanje = 3;
61                 break;
62             case 3:
63                 // V stanju 3 lahko vstavimo tako soglasnik ↩
                      kakor samoglasnik
64                 nakljucnoSt = rnd.nextInt(26);
65                 if (nakljucnoSt < 6) {
66                     // Vstavili bomo samoglasnik in presli v ↩
                          stanje 4
67                     sb.append(samoglasniki[nakljucnoSt]);
68                     stanje = 4;
69                 } else {
70                     // Vstavili bomo soglasnik in presli v ↩
                          stanje 2
71                     sb.append(soglasniki[nakljucnoSt - 6]);
```

```
72                      stanje = 2;
73                  }
74                  break;
75              case 4:
76                  // V stanju 4 lahko vstavimo zgolj soglasnik in↩
                        preidemo v stranje 2
77                  sb.append(soglasniki[rnd.nextInt(20)]);
78                  stanje = 2;
79                  break;
80              default:
81                  throw new RuntimeException("Neveljavno stanje."↩
                        );
82          }
83      }
84      return stanje;
85  }
```

Literatura

[1] W. Stalling, *Cryptography and network security, principles and practice*, 4. izdaja, Prentice Hall, 2006, str. 18–19.

[2] J. Verschuren, R. Govaerts, J. Vandewalle, ISO-OSI Security Architecture, *Lecture Notes In Computer Science; Computer Security and Industrial Cryptography - State of the Art and Evolution, ESAT Course*, 741, (1993), str. 179–192.

[3] Direktiva evropskega parlamenta in Sveta 95/46/ES, *31995L0046*, Uradni list L 281 (23.11.1995); str. 0031 - 0050.

[4] *Zakon o varstvu osebnih podatkov (ZVOP-1)*, Uradni list RS, št. 86/2004, 2004, št. 86.

[5] A. Baraani, J. Pieprzyk, R. Safavi, *Security In Databases, A Survey Study*, 1996.

[6] L. Bouganim, P. Pucheral, Chip-secured data access, confidential data on untrusted servers, *Proceedings of the 28th international conference on Very Large Data Bases*, (2002), str. 131–142.

[7] H. Hacigümüş, B. Iyer, C. Li, S. Mehrotra, Executing SQL over encrypted data in the database-service-provider model, *Proceedings*

of the 2002 ACM SIGMOD international conference on Management of data, (2002), str. 216–227.

[8] N. R. Wagner, P. S. Putter, M. R. Cain, Encrypted database design, Specialized approaches, *IEEE Symposium on Security and Privacy*, (1986), str. 148–153.

[9] D. E. Denning, Cryptographic Checksums for Multilevel Database Security, *IEEE Symposium on Security and Privacy*, (1984), str. 52–61.

[10] K. A. Omar, D. L. Wells, Modified Architecture for the Sub-Keys Model, *IEEE Symposium on Security and Privacy*, (1983), str. 79.

[11] L. Seitz, J.-M. Pierson, L. Brunie, Key Management for Encrypted Data Storage in Distributed Systems, *Proceedings of the Second IEEE International Security in Storage Workshop*, (2003), LIRIS, INSA de Lyon, France, str. 20–20.

[12] E. Damiani, S. C. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, Key Management for Multi-User Encrypted Databases, *Workshop On Storage Security And Survivability*, *Proceedings of the ACM workshop on Storage security and survivability*, (2005), str. 74–83.

[13] H. Hacigümüş, B. Iyer, S. Mehrotra, Ensuring the integrity of encrypted databases in the database-as-a-service model, *Data and applications security XVII: status and prospects*, (2004), str. 61–74.

[14] E. Meux, Encrypting personal identifiers, *Health Services Research*, 29(2), (junij 1994), str. 247–256.

[15] D. Recordon, D. Reed, OpenID 2.0, a platform for user-centric identity management, *Conference on Computer and Communications Security, Proceedings of the second ACM workshop on Digital identity management*, (2006), str. 11–16.

[16] E. Tsyrklevich, *Single Sign-On for the Internet, A Security Story*, BlackHat USA, Las Vegas, 2007.

[17] M. Kokol, I. Pernek, D. Pezdiček, M. Urbas, *Storitev AtomID™, Poslovni načrt*, mimeIT d.o.o., Maribor, 2009.

[18] J. Yan, A. Blackwell, R. Anderson, A. Grant, Password memorability and security, empirical results, *Security and Privacy, IEEE*, 2(5), (2004), str. 25–31.

[19] M. D. Leonhard, *A Comparison of Three Random Password Generators*, College of Engineering, University of Illinois at Chicago, ZDA, (2006).

[20] C. Kuo, S. Romanosky, L. F. Cranor, Human Selection of Mnemonic Phrase-based Passwords, *Proceedings of the second symposium on Usable privacy and security*, (2006), str. 67–78.

[21] T. R. Peltier, *Information security risk analysis*, CRC Press, (2005).

[22] S. Northcutt, J. Novak, *Network Intrusion Detection*, 3. izd., New Riders Publishing, 2002.

[23] L. Portnoy, E. Eskin, S. Stolfo, Intrusion Detection with Unlabeled Data Using Clustering, *Proceedings of ACM CSS Workshop on Data Mining Applied to Security*, (2001), str. 5–8.

[24] W. Lee , S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop , J. Zhang, Real Time Data Mining-based Intrusion Detection, *DARPA Information Survivability Conference & Exposition II*, 2, (2001), str. 89–100.

[25] R. L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, 21, (1978), 2, str. 120-126.

[26] W. D. Pace, E. W. Staton, G. S. Higgins, D. S. Main, D. R. West, D. M. Harris, Database Design to Ensure Anonymous Study of Medical Errors, A Report from the ASIPS collaborative, *Journal of the American Medical Informatics Association*, 10, (2003), 6, str. 531–540.

[27] C. Pagliari, D. Detmer, P. Singleton, Potential of electronic personal health records, *BMJ*, 335, (2007), str. 330–333.

[28] L. Seitz, J.-M. Pierson, L. Brunie, *Encrypted storage of medical data on a grid*, Institut National des Sciences Appliquées, Lyon, 2005.

[29] P. C. Tang, J. S. Ash, D. W. Bates, J. M. Overhage, D. Z. Sands, Personal Health Records, Definitions, Benefits, and Strategies for Overcoming Barriers to Adoption, *Journal of American Medical Informatics Association*, 13, (2006), 2, str. 121–126.