UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Miha Jernejčič

Biometrična verifikacija v oblaku

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Peter Peer

Asistent: as. Jernej Bule

Ljubljana, 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01828/2012
Datum: 03.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MIHA JERNEJČIČ**

Naslov: **BIOMETRIČNA VERIFIKACIJA V OBLAKU**
BIOMETRIC VERIFICATION IN THE CLOUD

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Analizirajte koncepte upravljanja z identitetami v oblaku. Nato preglejte obstoječe biometrične rešitve v oblaku. Na podlagi sistema FingerIdent zasnujte storitev za verifikacijo v oblaku. Na tej osnovi implementirajte biometrično verifikacijsko storitev v oblaku in prikažite njeno uporabo na praktičnem primeru.

Mentor:
doc. dr. Peter Peer

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Miha Jernejčič,

z vpisno številko 63030115,

sem avtor diplomskega dela z naslovom:

Biometrična verifikacija v oblaku

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Petra Peera in as. Jerneja Buleta
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 13. 09. 2012 Podpis avtorja:

Zahvala

Zahvaljujem se družini za podporo in potrpežljivost med študijem ter vsem ostalim, ki so pripomogli k nastanku diplomske naloge.

Zahvaljujem se mentorju doc. dr. Petru Peeru in as. Jerneju Buletu za pomoč in vodenje pri izdelavi diplomske naloge.

Mojim najdražjim

Kazalo

Povzetek **1**

Abstract **3**

1 Uvod **5**

2 Oblačni sistemi **7**
2.1 Uvod v oblačne sisteme 7
2.2 Storitveni modeli oblačnih sistemov 8
2.2.1 Programska oprema kot storitev 9
2.2.2 Platforma kot storitev 9
2.2.3 Infrastruktura kot storitev 10
2.3 Varnost v oblaku 10

3 Koncepti upravljanja z identitetami v oblaku **11**
3.1 Upravljanje z identitetami uporabnikov v zaupanja vrednem okolju 11
3.2 Upravljanje z identitetami uporabnikov v javno dostopnem okolju 12
3.3 Upravljanje z identitetami uporabnikov v deljenem okolju 13

4 Pregled obstoječih rešitev biometrije v oblaku **15**
4.1 Obstoječe biometrične rešitve v oblaku 15
4.1.1 Ceelox ID Online 15
4.1.2 TruDonor Online 16
4.1.3 MyBioID 17
4.1.4 BioID Web Services 18
4.1.5 PasswordBank IDaaS 19
4.2 Skupne točke 20

5 Zasnova sistema **23**
5.1 Uporabljena strojna oprema 23
5.2 Sistem Moodle 24
5.2.1 Avtentikacija v sistemu Moodle 25
5.2.2 Integracija bralnika prstnih odtisov v sistem Moodle . . . 26
5.3 Sistem FingerIdent 28

6 Pregled oblačnih platform s .NET podporo **31**
6.1 Amazon Elastic Compute Cloud 32
6.2 AppHarbor 32
6.3 OpenStack 33
6.4 Uhuru 34
6.5 Tier 3 34
6.6 Apprenda 35
6.7 Iron Foundry 36
6.8 Microsoft Windows Azure 36
6.8.1 Varnost v oblačni platformi Windows Azure 41

7 Razvoj biometrične verifikacije v oblaku in prikaz njene uporabe na praktičnem primeru **43**
7.1 Integracija verifikacijskega sistema FingerIdent v oblak 44
7.1.1 Razvoj spletne aplikacije 44
7.1.2 Integracija sistema FingerIdent v spletno aplikacijo . . . 46
7.1.3 Integracija spletne aplikacije v oblačno platformo Windows Azure 48
7.2 Implementacija biometrične verifikacije v sistem Moodle 51
7.2.1 Razvoj zaslonskih mask 52
7.2.2 Razvoj in dopolnitev potrebnih sistemskih knjižnic . . . 55
7.2.3 Razvoj avtentikacijskega vtičnika 56
7.2.4 Pošiljanje podatkov slike prstnega odtisa po omrežju . . 61
7.3 Implementacija varnostnih mehanizmov 61
7.3.1 Prisluškovanje v omrežjih 62
7.3.2 Napad mož v sredini 67

8 Sklepne ugotovitve **69**

Dodatki **71**

A WebApi dokumentacija **71**

B Seznam vseh sprememb v sistemu Moodle **75**
B.1 Spremembe pri razvoju zaslonskih mask 75
B.2 Spremembe pri dopolnitvi potrebnih sistemskih knjižnic 76
B.3 Spremembe pri razvoju avtentikacijskega vtičnika 76

Seznam slik **79**
Literatura **81**

Seznam uporabljenih kratic in simbolov

- **DLL** — dynamic link library
- **API** — application programming interface
- **COM** — common object model
- **OLE** — object linking and embedding
- **ACTIVEX** — type of COM component that can self-register, also known as an "activeX control".
- **MOODLE** — modular object-oriented dynamic learning environment
- **PHP** — hypertext preprocessor
- **SDK** — software development kit
- **CLR** — common language runtime
- **BMP** — bitmap image file
- **HTTP** — hypertext transfer protocol
- **HTTPS** — hypertext transfer protocol secure
- **TLS** — transport layer security
- **SSL** — secure sockets layer
- **IaaS** — infrastructure as a service
- **SaaS** — software as a service

- **PaaS** — platform as a service
- **IDaaS** — identification as a service
- **BaaS** — biometrics as a service
- **BLOB** — binary large object
- **MVC** — model view controller
- **SQL** — structured query language
- **ISO** — international organization for standardization
- **IDM** — identity management
- **IP** — internet protocol

Povzetek

Biometrična identifikacija se nanaša na prepoznavanje osebe na podlagi bioloških podatkov. Temelji na primerjanju binarnega zapisa nekega fiziološkega vzorca s predhodno v podatkovni bazi shranjenimi referenčnimi vzorci. Izmed vseh biometričnih metod je metoda prepoznavanja prstnih odtisov najstarejša in uspešno uporabljena v številnih aplikacijah, saj prstni odtisi zagotavljajo visoko stopnjo unikatnosti.

Cilj te diplomske naloge je implementacija biometrične verifikacije prstnih odtisov v oblačni platformi. Rešitev sestoji iz že obstoječega sistema FingerIdent, ki prepoznava prstne odtise na namiznih aplikacijah. Ta sistem sem integriral v oblačno storitev, ki uporablja najsodobnejše varnostne mehanizme. Za prikaz delovanja sem uporabil sistem za e-učenje Moodle, ki sem ga ustrezno prilagodil ter mu dodal možnost verifikacije s prstnim odtisom. Moj cilj ni bil samo integrirati sistem Fingerident v spletno aplikacijo, ampak tudi poiskati najprimernejšo oblačno platformo za gostovanje oblačne storitve. Končni rezultat je delujoča oblačna storitev FingerIdent, ki omogoča uporabo biometrične verifikacije s prstnim odtisom in prilagojen sistem za e-učenje, ki je s to oblačno storitvijo pridobil možnost avtentikacije z uporabo bralnika prstnih odtisov.

Ključne besede:

biometrija, prstni odtis, oblak, oblačna storitev, verifikacija, FingerIdent, Moodle, e-učenje, Azure

Abstract

Biometrics verification refers to recognition of a person based on biometric data. It is based on matching a binary-encoded pattern of a person's characteristic to the reference patterns previously stored in a database. The fingerprint-based verification is one of the oldest and most widely applied approach among all the biometric techniques, mainly due to highly unique fingerprint patterns.

The goal of this thesis is implementation of a biometric verification in a cloud platform. The solution consists of an existing system FingerIdent, which implements fingerprint recognition for desktop applications. This system was integrated into a cloud service that uses the latest state-of-the-art security mechanisms. The cloud service was demonstrated on e-learning system Moodle, which was modified to accept the cloud-based biometric fingerprint authentication method. The goal of this thesis was not only to integrate the FingerIdent system into cloud application, but also to find the most appropriate cloud platform for hosting. The final result is a working cloud service with FingerIdent, which uses a fingerprint-based recognition and an adapted e-learning system Moodle that gained fingerprint-based authentication.

Key words:

biometry, fingerprint, cloud, cloud service, verification, FingerIdent, Moodle, e-learning, Azure

Poglavje 1

Uvod

Proces verifikacije uporabnikov je zelo pomemben v vseh sistemih. Verifikacija lahko poteka preko običajnih metod, kot so gesla, identifikacijske kartice ali PIN kode. Takšne metode so iz vidika varnosti pomanjkljive, saj se lahko ob vdorih v strežnike, kjer se hrani uporabniške podatke, takšna gesla ali PIN kode ukradejo in tako lahko nepridipravi pridobijo dostop do tujih računov. Metoda identifikacijskih kartic je tako še manj varna, ker lahko kartico preprosto izgubimo ali pa nam jo ukradejo.

Alternativo tem metodam verifikacije predstavljajo biometrične metode. To so metode, ki temeljijo na procesu zbiranja, proučevanja in shranjevanja podatkov o posameznikovih fizičnih lastnostih z namenom identifikacije uporabnika [1]. Najpopularnejše metode biometrije so skeniranje očesne šarenice, prstnih odtisov, prepoznava glasu, obraza itd. Biometrične metode imajo veliko prednosti pred navadnimi metodami in ponujajo zelo visoko stopnjo varnosti v primerjavi z navadnimi metodami.

Za verifikacijo uporabnikov bomo v diplomski nalogi uporabili biometrično metodo skeniranja prstnega odtisa, ki je tudi eden najbolj zanesljivih načinov avtentikacije, saj do sedaj še niso našli dveh ljudi z identičnim prstnim odtisom [2]. Obstoječih sistemov za verifikacijo uporabnikov z razpoznavo prstnih odtisov je veliko, vendar jih je večina v obliki lokalnih rešitev. Veliko današnjih osebnih računalnikov je opremljenih z bralniki prstnih odtisov, ki ponujajo uporabniku alternativni način prijave v operacijski sistem.

V zadnjih nekaj letih se je število biometričnih podatkov izredno povečalo. Sistemi za biometrično verifikacijo so zato zelo obremenjeni, saj morajo biti

sposobni shranjevati v ekstremnih primerih podatke za 500 milijonov in več uporabnikov in izvajati verifikacijo uporabnikov v realnem času [3].

Lokalne rešitve za biometrično verifikacijo so težko kos takim zahtevam in potrebno je iskati alternativne rešitve. Računalništvo v oblaku predstavlja alternativno rešitev s svojimi skorajda neomejenimi podatkovnimi kapacitetami ter zmožnostjo hitre distribucije podatkov ter vzporednega procesiranja.

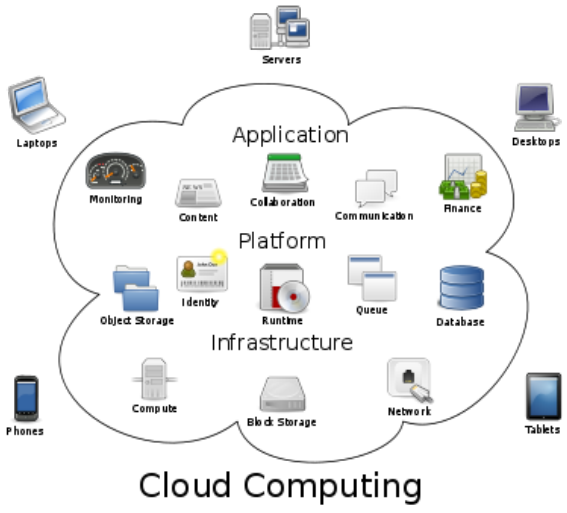
V diplomski nalogi se bomo ukvarjali z integracijo že obstoječega sistema FingerIdent [4], ki uporablja metodo razpoznavanja prstnih odtisov za verifikacijo uporabnikov v oblačni sistem. Kot dokaz koncepta bomo uporabili sistem za e-učenje *Moodle*, ki privzeto omogoča avtentikacijo z uporabniškim imenom in geslom. Opisali bomo postopek implementacije vseh potrebnih sprememb za dopolnitev nove vrste avtentikacije z uporabo bralnika prstnih odtisov in sistem povezali na oblačno storitev.

Poglavje 2

Oblačni sistemi

2.1 Uvod v oblačne sisteme

Računalništvo v oblaku se je že dobro usidralo v naš način razmišljanja o programih. Programe, ki delujejo lokalno na računalniku, vse bolj izpodrivajo novejše aplikacije, ki prebivajo v oblaku. Beseda *oblak* predstavlja metaforo za internet. Na sliki 2.1 je prikazan abstraktni koncept računalništva v oblaku. Oblačne aplikacije so praviloma robustnejše, saj niso več odvisne od fizičnih računalnikov in večinoma tečejo v virtualnih okoljih.



Slika 2.1: Računalništvo v oblaku.

Uporabnik pri oblačnih aplikacijah večinoma nima dostopa do nižjih plasti aplikacije in ga tudi ne zanima, ali teče aplikacija na enem strežniku s podatkovno bazo ali pa ima v ozadju celotni računski center podjetja *Google*, ki ima skorajda nepredstavljivo računsko moč.

Glavne značilnosti oblačnih sistemov:

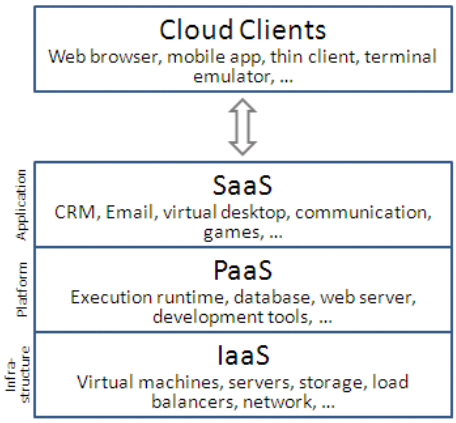
- agilnost,
- nižji stroški vzdrževanja,
- neodvisnost lokacije od naprave,
- sobivanje oblačnih storitev – *multi-tenancy*,
- zanesljivost in
- skalabilnost.

2.2 Storitveni modeli oblačnih sistemov

Oblačne platforme ločimo po vrsti storitve [5] na:

- programska oprema kot storitev,
- platforma kot storitev in
- infrastruktura kot storitev.

Storitve so povezane med seboj, kot je prikazano na sliki 2.2.



Slika 2.2: Pregled storitvenih modelov oblačnih platform.

2.2.1 Programska oprema kot storitev

Programska oprema kot storitev (angl. *software as a service – SaaS*) predstavlja oblačni storitveni model, kjer ponudnik oblačnega sistema namesti vso programsko opremo in vse potrebno za delovanje programske opreme. Uporabniku se ni potrebno ukvarjati z infrastrukturo oblaka in lahko dostopa do oblačnih storitev preko spletnega brskalnika ali namenske programske opreme. Primeri SaaS oblačnih sistemov: *Google Apps*, *Quickbooks Online*, *SalesForce.com*, *Microsoft Office 365*, *ISL Online* in mnogi drugi.

2.2.2 Platforma kot storitev

Platforma kot storitev (angl. *platform as a service – PaaS*) predstavlja oblačni storitveni model, kjer ponudnik zagotavlja oblačno platformo, kar vključuje operacijski sistem in izvajalno okolje za nekatere programske jezike, spletni strežnik in bazo podatkov. Razvijalci programske opreme lahko uporabljajo tako oblačno platformo za poganjanje njihovih aplikacij, ne da bi morali skrbeti za novo strojno opremo ali nameščanje pravilnih programskih paketov. Primeri PaaS oblačnih sistemov: *Amazon Elastic Beanstalk*, *Heroku*, *EngineYard*, *Google App Engine*, *Windows Azure*, *VMWare SpringSource*, *IBM PaaS*, *Uhuru*, *AppHarbor*, *Apprenda*, *Iron Foundry* in mnogi drugi.

2.2.3 Infrastruktura kot storitev

Infrastruktura kot storitev (angl. *infrastructure as a service – IaaS*) je najnižji storitveni model oblačne arhitekture na trgu. Oblačni ponudniki ponujajo računalniško infrastrukturo v obliki fizičnih ali virtualiziranih računalnikov. Sem spadajo tudi strežniki, shramba podatkov, računalniška omrežja in druge storitve. Primeri IaaS oblačnih sistemov: *Amazon CloudFormation*, *Rackspace Cloud*, *Google Compute Engine*, *RightScale*, *OpenStack* in mnogi drugi.

2.3 Varnost v oblaku

Včasih, ko računalniki še niso bili izpostavljeni internetu, je za potrebe varnosti zadostoval omejen dostop do računalnika. Danes se je z uvedbo interneta in oblačnih sistemov vidik varnosti spremenil in izdanih je bilo veliko dokumentov, ki opisujejo varnostne smernice, ki jih je potrebno upoštevati pri izbiri oblačnega ponudnika. Primere takih dokumentov predstavljajo viri [5, 6, 7].

Problem varnosti v oblaku se nanaša predvsem na varnost podatkov v oblaku. Ker imamo pri biometrični verifikaciji uporabnikov v oblaku opraviti z občutljivimi osebnimi podatki, kot so uporabniška gesla in zajeti prstni odtisi, je varnost osebnih podatkov pomemben kriterij pri izbiri ponudnika oblačnih storitev. Podatki so v oblaku shranjeni na strežnikih ponudnika oblačnih storitev in so izven našega fizičnega dosega. Za naročnika oblačnih storitev to pomeni novo vrsto tveganj pri zaščiti podatkov. Pri izbiri ponudnika oblačnih storitev se je potrebno prepričati, da le-ta naših podatkov ne bo zlorabil in da bo poskrbel za ustrezno zaščito pri prenosu podatkov po omrežju. Podatki so na strežnikih običajno skladiščeni skupaj s podatki drugih strank (problem sobivanja – *multi-tenancy*). Ponudnik mora pri skladiščenju poskrbeti za varno izolacijo podatkov. Poskrbeti je potrebno za naše podatke v primeru, če ponudnik preneha poslovati ali spremeni pogoje poslovanja. Ob prenehanju sodelovanja s ponudnikom oblačnih storitev se je potrebno prepričati, da bo ponudnik resnično izbrisal podatke iz svojih strežnikov.

Pred vsemi naštetimi nevarnostmi se lahko zavarujemo z izvedbo analize tveganja hranjenja podatkov v oblaku za potencialnega ponudnika oblačnih storitev.

Poglavje 3

Koncepti upravljanja z identitetami v oblaku

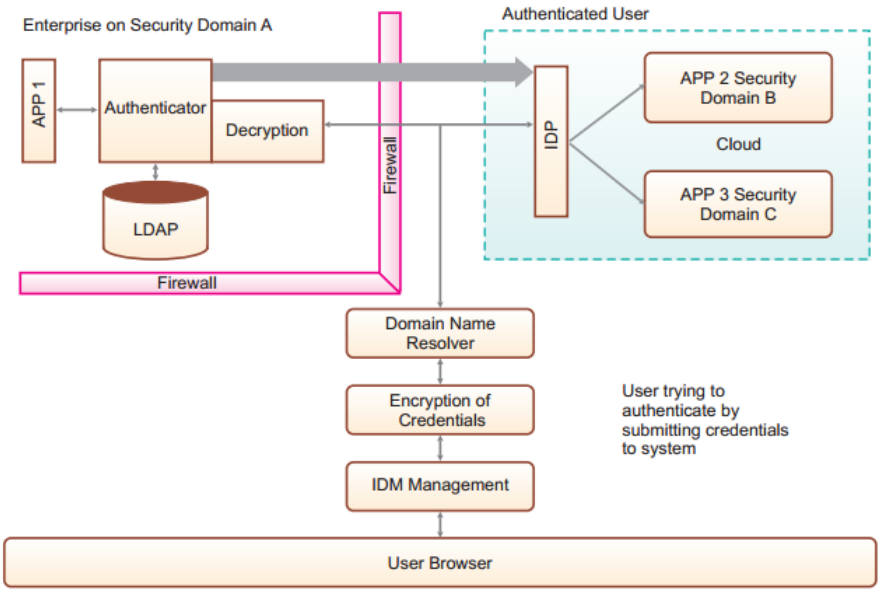
Biometrijo v oblaku lahko v splošnem predstavimo kot koncept upravljanja z identitetami. Pri upravljanju z identitetami v oblaku poznamo tri splošne koncepte, ki so si med seboj sorodni. Vsak od njih je primeren za uporabo v določenem tipu okolja. Pregled konceptov je izpeljan iz članka [8].

3.1 Upravljanje z identitetami uporabnikov v zaupanja vrednem okolju

Koncept upravljanja z identitetami uporabnikov v zaupanja vrednem okolju se uporablja v manjših in privatnih oblačnih sistemih, kjer je potrebno verificirati uporabnika. Zajeti biometrični podatki so preko spletnega brskalnika poslani do vmesnika za upravljanje z identitetami (angl. *identity Management – IDM*), kjer so zakodirani in preko varnega kanala poslani do avtentikacijskega sistema.

Avtentikacijski sistem je zasnovan tako, da ustreza specifikacijam problema. Za avtentikacijo lahko uporablja lastno podatkovno bazo ali pa zunanje avtentikacijske mehanizme, kot so LDAP (angl. *lightweight directory access protocol*), OpenID, OAuth, SSO, OpenSocial, FaceBookConnect in drugi.

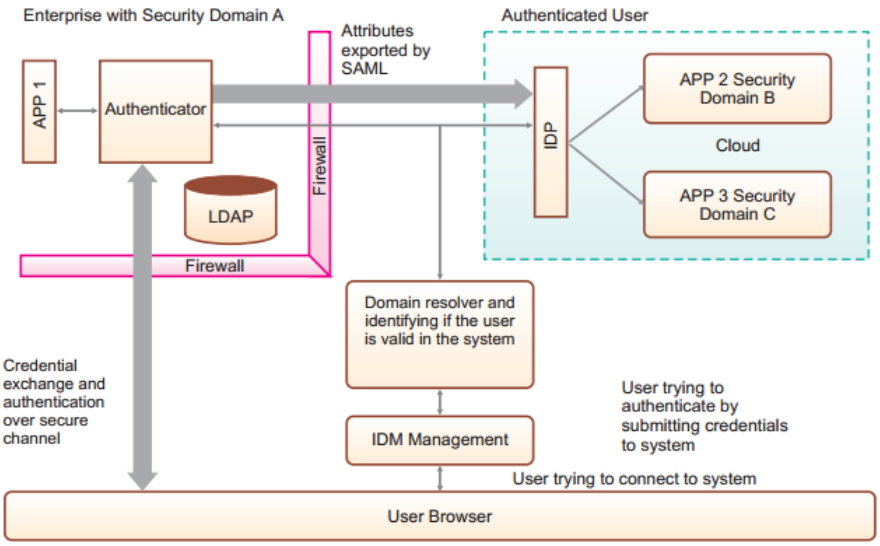
Avtentikacijski sistem avtenticira uporabnika in podatki o uporabniku se nato prenesejo do ponudnika posamezne storitve preko protokola SAML (angl. *security assertion markup language*), ki služi izmenjavi podatkov o avtorizaciji in avtentikaciji. Celoten proces je prikazan na sliki 3.1.



Slika 3.1: Upravljanje z identitetami uporabnikov v zaupanja vrednem okolju.

3.2 Upravljanje z identitetami uporabnikov v javno dostopnem okolju

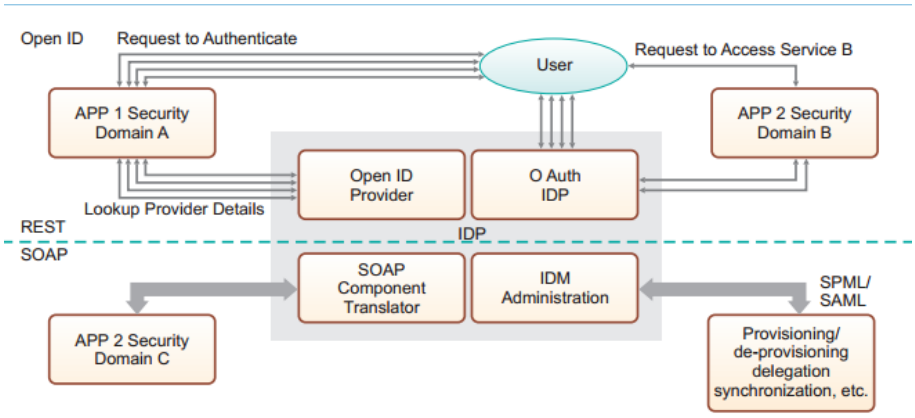
Koncept upravljanja z identitetami uporabnikov v javno dostopnem okolju se uporablja v javnih oblačnih platformah in je zelo podoben tistemu v zaupanja vrednih okoljih. Koncepta za upravljanje se razlikujeta v tem, da se v javnem okolju avtentikacijski uporabniški podatki pošljejo neposredno v avtentikacijski sistem z uporabo *SSL* kriptiranega kanala in ne preko vmesnika za upravljanje z identitetami. Proces upravljanja z identitetami uporabnikov v javno dostopnem okolju je prikazan na sliki 3.2.



Slika 3.2: Upravljanje z identitetami uporabnikov v javnem okolju.

3.3 Upravljanje z identitetami uporabnikov v deljenem okolju

Koncept upravljanja z identitetami uporabnikov v deljenem okolju se uporablja pri povezavi večjega števila oblakov v skupni uporabniški prostor. Bistvo tega pristopa je v uporabi centralnega sistema za upravljanje uporabnikov, v katerega se lahko uporabniki prijavijo z mehanizmi več različnih oblačnih sistemov. Proces upravljanja z identitetami uporabnikov v deljenem okolju je prikazan na sliki 3.3.



Slika 3.3: Upravljanje z identitetami uporabnikov v deljenem okolju.

Poglavje 4

Pregled obstoječih rešitev biometrije v oblaku

4.1 Obstoječe biometrične rešitve v oblaku

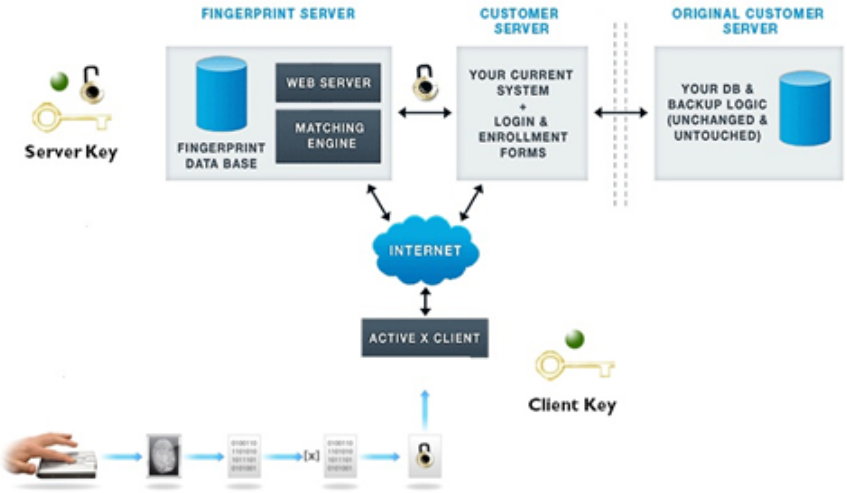
Področje biometrije v oblaku je mlado in se hitro razvija. To je razlog, da je na tržišču trenutno malo ponudnikov komercialnih rešitev. Obstoječi ponudniki biometrične verifikacije v oblaku zato poskušajo ohraniti konkurenčno prednost pred ostalimi s skrivanjem podrobnosti o arhitekturi svojih rešitev.

4.1.1 Ceelox ID Online

Podjetje Ceelox je na tržišču prisotno s produktom *Ceelox ID Online* in za uporabnika predstavlja dodaten sloj varnosti v oblačnih aplikacijah [9]. Ceelox ID Online rešitev je namenjena zagotavljanju najvišje stopnje varnosti z uporabo biometrične metode razpoznavanja prstnih odtisov. V Ceelox ID Online je uporabnikom na voljo uporaba ogrodja OpenID [10], ki je namenjeno upravljanju digitalnih identitet. Uporabnikom je omogočena prijava na različne spletne strani kot so internetne banke, avkcije, investicijska spletna mesta, socialna omrežja z uporabo iste OpenID identitete. V Ceelox ID Online je integrirana podpora za upravljanje digitalnih identitet z uporabo storitve aktivni imenik (angl. *Active Directory*), ki je na voljo za uporabo v *Windows* omrežjih [11].

Ceelox ID Online uporablja pristop, ki je prikazan na sliki 4.1. Strežnik prstnih odtisov generira kodo za preverjanje pristnosti, ki jo nato pošlje odjemalskemu računalniku. Odjemalski računalnik in strežnik za upravljanje prstnih odtisov nato s pomočjo te kode generirata edinstven ključ, ki je različen za

vsako transakcijo posebej, poleg tega pa je tudi časovno občutljiv. Odjemalski računalnik ta ključ uporabi za kodiranje datotek, ki so bile generirane na podlagi biometričnega zajema prstnega odtisa. Strežnik prstnih odtisov nato svoj ključ uporabi za dekodiranje prejetih podatkov in izvede proces verifikacije s primerjavo prejetih biometričnih vzorcev s tistimi v podatkovni bazi. Strežnik obvesti uporabnika na odjemalskem računalniku o uspešnosti avtentikacijskega zahtevka.



Slika 4.1: Arhitektura sistema Ceelox ID Online.

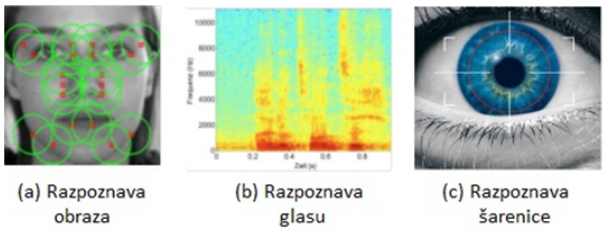
4.1.2 TruDonor Online

TruDonor Online je spletna storitev, ki omogoča identifikacijo darovalcev krvi na podlagi prstnih odtisov. TruDonor Online je SaaS biometrična storitev, ki je nameščena v oblaku in je produkt podjetja BIO-key [12]. Proces poteka tako, da se vsak darovalec ob prvem obisku centra registrira (v sistem se vpišejo podatki o osebi, zajamejo se prstni odtisi), dodeli pa se mu tudi unikatna koda, ki se uporablja za označevanje darovane krvi. Ta proces je prikazan na sliki 4.2. Darovalec se ob vsakem naslednjem obisku lahko identificira samo s pomočjo prstnega odtisa. Kri lahko daruje v kateremkoli centru za odvzem krvi znotraj iste organizacije. To omogoča nadzor nad uporabljeno vrsto bralnikov prstnih odtisov, ki so združljivi z oblačno rešitvijo TruDonor Online.

Slika 4.2: Delovanje sistema TruDonor Online.

4.1.3 MyBioID

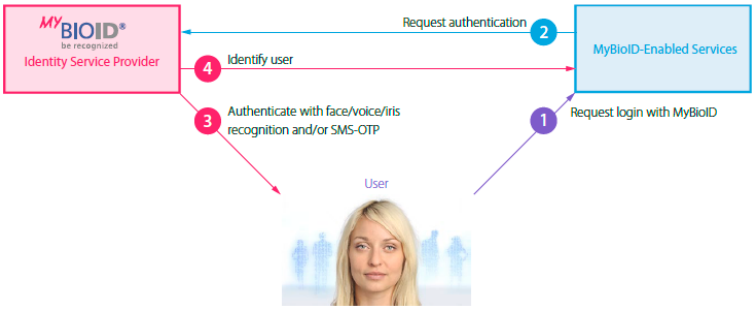
MyBioID je oblačna biometrična rešitev podjetja BioID, ki je na voljo tako za spletne kot tudi za mobilne uporabnike [13]. MyBioID nosi oznako beta, vendar se rešitev že uporablja na produkcijskem okolju. MyBioID omogoča prijavo v spletne storitve s pomočjo razpoznave obraza, šarenice, glasu ali s katerokoli kombinacijo izmed naštetih (slika 4.3).



Slika 4.3: Biometrične tehnike, storitve MyBioID: a) razpoznavanje obraza na podlagi obraznih značilk, b) razpoznava glasu na podlagi unikatnih spektralnih vzorcev, c) razpoznava šarenice na podlagi unikatnih vzorcev.

Ob postopku vpisa (slika 4.4) se sliki obraza, šarenice ali datoteki, ki vsebuje glasovni zapis, doda digitalni podpis in časovni žig, ki dodatno oteži ponarejanje in s tem nepooblaščene vstope v sistem. Algoritmi preverijo trenutne podatke o osebi, ki se želi identificirati, s tistimi v bazi in določijo, kakšna je podobnost med omenjenima. Na podlagi dobljenih rezultatov se sistem odloči, ali je uporabniku dovoljen dostop. Storitev s tehniko razpoznave obraza uporabniku omogoča tudi možnost avtomatske odjave iz sistema, ko uporabnika ni več pred zaslonom računalnika. S tem se uporabnik izogne tveganju, da bi si nekdo drug prisvojil njegovo identiteto. Uporabnikom je v MyBioID na

voljo uporaba ogrodja OpenID za upravljanje digitalnih identitet.



Slika 4.4: Prijavni proces pri oblačni rešitvi MyBioID.

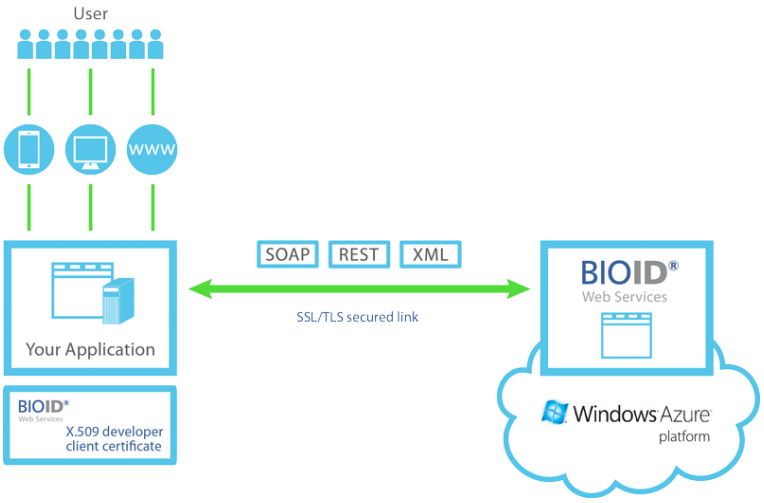
4.1.4 BioID Web Services

Podjetje BioID svojo oblačno rešitev *BioID Web Services* opisuje z besedno zvezo biometrija kot storitev (angl. *"Biometrics as a Service" – BaaS*) [14]. BioID Web Services omogoča prijavo v spletne storitve s pomočjo biometričnih metod razpoznave obraza ali glasu. Biometrična metoda razpoznavanja šarenice pri tej rešitvi še ni na voljo. Storitev razpoznavanja obraza se lahko uporablja tudi za označevanje oseb na slikah.

Za uporabo oblačne storitve je na voljo šest klicev API:

- prijava novega uporabnika (angl. *enroll*),
- verifikacija uporabnika (1:1) (angl. *verify*),
- identifikacija uporabnika (1:n) (angl. *identify*),
- izbris identitete uporabnika (angl. *delete class*),
- preverjanje kvalitete biometričnega zapisa (angl. *quality check*) in
- preverjanje statusa uporabnika (angl. *status*).

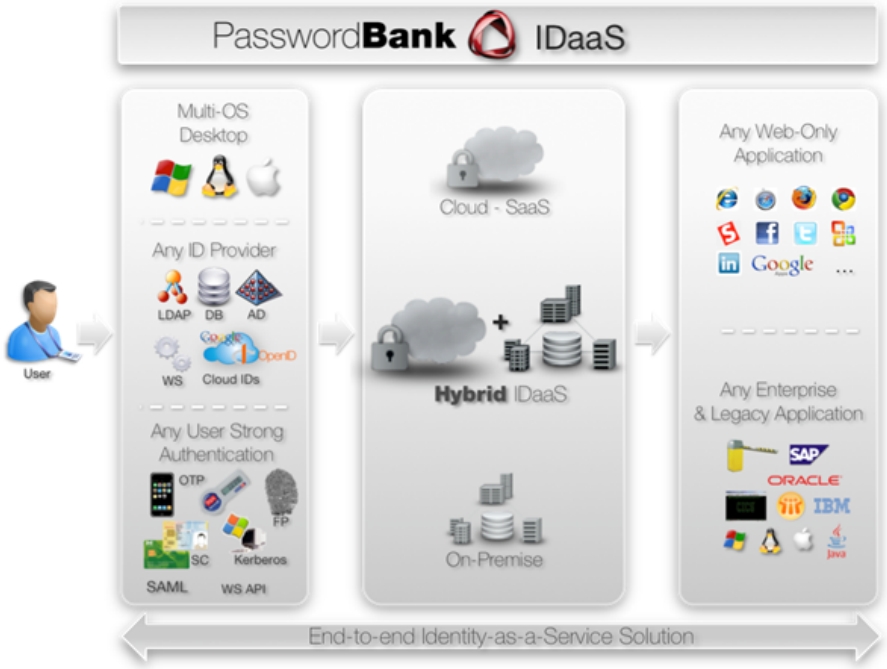
Razvijalcem je dostopna uradna dokumentacija rešitve *BioID Web Services* in API klicev [15]. Z uporabo dokumentacije se lahko to oblačno rešitev integrira v poljubno aplikacijo. Proces uporabe BioID Web Services je prikazan na sliki 4.5.



Slika 4.5: Uporaba oblačne storitve BioID WebServices.

4.1.5 PasswordBank IDaaS

PasswordBank IDaaS (angl. *Identity as a Service*) je oblačna rešitev podjetja PasswordBank za upravljanje z identitetami v oblaku [16]. Spletna storitev, preko katere se uporabnik lahko identificira, je povezana na strežnike v oblaku (slika 4.6). Uporabnik se lahko identificira preko različnih metod, med drugim tudi z metodo razpoznavanja prstnih odtisov, različnih biometričnih kartic, telefona ipd. Zagotovljena je tudi povezljivost z različnimi upravljalci identitet, kot je *OpenID*, in imeniškimi storitvami za upravljanje z uporabniki, kot je *aktivni imenik*. Storitev lahko uporabljamo tako v zasebnem kot tudi v javnem oblačnem sistemu. Uporabnikom je uporaba rešitve *PasswordBank IDaaS* na voljo v obliki spletne aplikacije, vtičnika za spletni brskalnik in kot samostojen program v operacijskih sistemih *Windows*, *Mac OS X* in distribucijah *Linux*. Podrobnega opisa arhitekture ni na voljo.



Slika 4.6: PasswordBank IDaaS.

4.2 Skupne točke

Iz pregleda obstoječih tržnih rešitev na področju biometrije v oblaku je možno izluščiti nekaj skupnih točk med rešitvami. Ker je pri nekaterih opisanih rešitvah znano zelo malo podrobnosti o samem delovanju sistema, ni bilo mogoče poiskati vseh potrebnih informacij za izdelavo obširne primerjave med opisanimi rešitvami.

Vsem zgoraj opisanim rešitvam je skupno, da delujejo na principu *odjemalec–strežnik* [17]. Odjemalec na uporabnikovem računalniku skrbi za zajem biometričnega vzorca ter ga pošlje na strežnik, kjer se izvrši proces verifikacije uporabnika. Za varnost mrežnega prometa med odjemalcem in strežnikom so pri vseh rešitvah poskrbeli z uporabo protokola *https* [18].

Podatek o lokaciji procesiranja zajetih biometričnih vzorcev je bil podan samo pri sistemu *Ceelox ID Online*. Pri tej rešitvi se iz zajetega vzorca, z uporabo namenske activeX komponente, izvaja na uporabnikovi strani proces ekstrak-

cije vektorja značilk. Pri ostalih rešitvah ta podatek ni bil na voljo, vendar predpostavljam, da so ubrali enak pristop, ker optimizira količino mrežnega prometa.

V sistemih, ki uporabljajo metodo biometričnega razpoznavanja prstnih odtisov, je potrebno zagotoviti:

- upravljanje z bralnikom prstnih odtisov in
- podporo različnim tipom bralnikov prstnih odtisov.

Za upravljanje bralnika prstnih odtisov je v *Ceelox ID Online* uporabljena komponenta activeX. Pri oblačni rešitvi *PasswordBank IDaaS* je znano, da je upravljanje z bralnikom prstnih odtisov implementirano v samostojen program, ki deluje na vseh operacijskih sistemih in v spletni vtičnik za uporabo v spletnih aplikacijah. V javno dostopnih specifikacijah sistema *TruDonor Online* ni podatka o načinu upravljanja z bralnikom prstnih odtisov.

Sistema *Ceelox ID Online* in *TruDonor Online* sta združljiva z natanko določenimi bralniki za zajem prstnih odtisov. Uporabnik pri namestitvi sistema *Ceelox ID Online* ročno izbere tip bralnika prstnih odtisov, ki bo uporabljen pri avtentikaciji. Podprti bralniki prstnih odtisov pri tej rešitvi so: *Authentec AES 2501*, *Authentec AES 2510*, *Authentec AES 3500*, *Authentec AES 3400*, *Authentec AES 4000*, *Upek TCRU* in *Upek TCRE*. V specifikacijah sistema *TruDonor Online* ni podanega podatka o tipih bralnika, ki so združljivi s produktom.

Pregled področij oblačnih sistemov, biometrije v oblaku in obstoječih tržnih rešitev nam je pomagal pri boljšem razumevanju problematike *biometrične verifikacije v oblaku* in lažji zasnovi lastne oblačne rešitve.

Poglavje 5

Zasnova sistema

Celoten sistem vsebuje osnovne gradnike, ki jih je bilo potrebno povezati v delujoč sistem. V nadaljevanju so ti osnovni gradniki specificirani. V poglavju 7 je opisan razvoj celotnega sistema.

5.1 Uporabljena strojna oprema

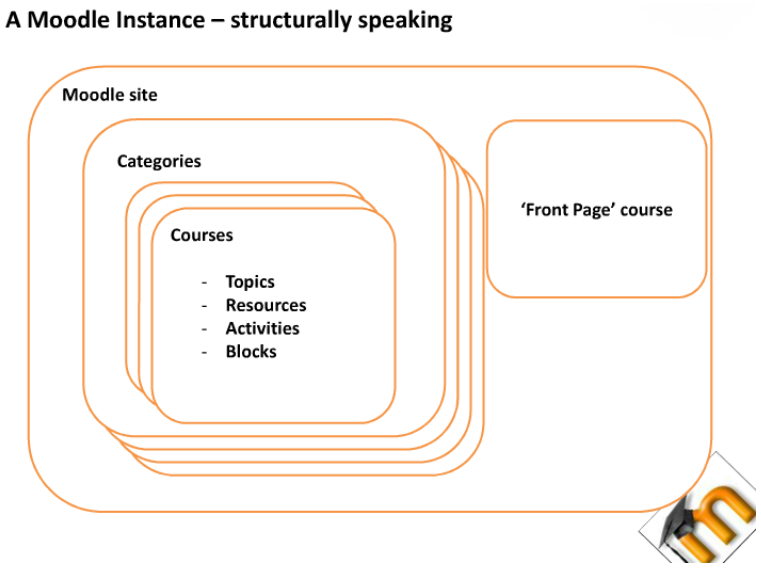
Pri zajemu prstnih odtisov smo uporabili napravo *Secugen Hamster Plus* (slika 5.1), ki nam je bila na voljo iz laboratorija za računalniški vid. Bralnik ima ločljivost slike 260 x 300 točk in 500 DPI (angl. dots per inch). Senzor za zajem prstnih odtisov je zelo dober in zajete slike so bile zelo jasne. Proizvajalec naprave *Secugen* nudi programerjem prenos paketa SDK, v katerem so vključene vse potrebne razvijalske knjižnice za vključitev bralnika v aplikacije, pisane v *.Net*, *C++* ali *Visual Basic* okolju. V paketu SDK so podprte tudi spletne aplikacije z uporabo spletnega vtičnika v obliki komponente *activeX*.

Slika 5.1: Biometrični bralnik Secugen Hamster Plus.

5.2 Sistem Moodle

Moodle je odprtokodni projekt [19] in predstavlja učinkovito integracijo informacijsko–komunikacijskih tehnologij za potrebe izobraževanja. Uporablja se v izobraževalnih ustanovah po svetu in predstavlja odlično orodje za podajanje učnega gradiva in komunikacijo med učitelji in učenci. Možno ga je namestiti na vseh pogosteje uporabljenih operacijskih sistemih. Spisan je v PHP, poganjata ga strežnik Apache [20] in podatkovna baza MySql [21].

Sistem je razvit modularno v sklopih. Dodajanje novih funkcionalnosti poteka z razvojem novih vtičnikov (angl. *plugin*). Tako so tudi naši dodatki v sistemu Moodle v večji meri implementirani z avtentikacijskim vtičnikom.



Slika 5.2: Struktura spletne strani sistema Moodle.

Slika 5.2 ponazarja osnovno strukturo spletne strani sistema Moodle. Ta zgradba je osredotočena na učne predmete, ki predstavljajo spletne strani, kamor lahko učitelji vnašajo učno gradivo. Uporabniki sistema imajo lahko dodeljene različne vloge. Tako so učitelji, učenci ali obiskovalci sistema uporabniki z enakimi pravicami, dokler jim administratorji ne dodelijo primernih vlog. Uporabniški računi so lahko predhodno ustvarjeni v sistemu preko vzdrževalnih skript, lahko se uporablja zunanja avtentikacija (primer: aktivni direktorij – angl. *active directory*) ali pa je uporabnikom dovoljeno ustvarjanje

novih računov.

Pri praktičnem delu diplomskega dela smo uporabili takrat aktualno verzijo sistema Moodle *2.2.2*. Med postopkom nastajanja teoretičnega dela je izšla novejša verzija sistema *2.3.1*, na kateri pa naših sprememb v sistemu Moodle nismo preizkusili.

5.2.1 Avtentikacija v sistemu Moodle

Avtentikacija uporabnikov v sistemu Moodle se izvaja z uporabo avtentikacijskih vtičnikov. Ti omogočajo uporabnikom, da se avtenticirajo tako iz lokalnih kakor tudi iz zunanjih virov. Sistem Moodle ima z razvojem novih avtentikacijskih vtičnikov v teoriji možnost avtenticiranja uporabnikov s katerimkoli od zunanjih avtentikacijskih sistemov.

Ne glede na uporabljeni avtentikacijski vtičnik je postopek avtentikacije vedno enak [22]:

1. postopek avtentikacije se prične, ko uporabnik sproži prijavno povezavo,
2. prikaže se privzeta vstopna stran */login/index.php* ali alternativna vstopna stran, če jo je administrator nastavil,
3. uporabnik vnese svoje uporabniške podatke v zaslonsko masko,
4. izvrši se del skripte */login/index.php*:
   - pridobi se seznam vseh omogočenih avtentikacijskih vtičnikov
   - klic metode *loginpage_hook()*, ki omogoča avtentikacijskim vtičnikom prestrezanje avtentikacijskih zahtevkov
   - preveri se veljavnost uporabniškega imena po merilih sistema Moodle
   - klic metode *authenticate_user_login()* ali *authenticate_user_fp_login()* in rezultat je objekt *$user*

5.2.2 Integracija bralnika prstnih odtisov v sistem Moodle

Za integracijo bralnika prstnih odtisov v sistem Moodle imamo na voljo dva pristopa:

- razvoj namenskega programa za upravljanje z bralnikom prstnih odtisov ali
- integracija bralnika prstnih odtisov v spletno aplikacijo z uporabo vtičnika za spletni brskalnik.

Obema pristopoma je skupno, da se povezujeta na avtentikacijski strežnik v oblaku, ki upravlja z identitetami uporabnikov (angl. *identity provider*). V nadaljevanju bomo podrobneje preučili oba pristopa in prikazali pozitivne in negativne lastnosti obeh pristopov.

Razvoj namenskega programa za upravljanje z bralnikom prstnih odtisov

Bistvo tega pristopa je implementacija samostojnega programa v programskem okolju *.Net*. Osnovna naloga tega programa je upravljanje z bralnikom prstnih odtisov in zagon spletnega bralnika s posebnimi parametri.

Scenarij uporabe:

1. zagon programa,
2. vnos uporabniških podatkov v prijavno masko in zajem prstnega odtisa,
3. program generira avtentikacijski zahtevek in kontaktira avtentikacijski strežnik preko POST zahtevka [23],
4. program prejme odgovor strežnika v obliki potrditve ali zavrnitve in
5. v primeru pozitivnega odgovora program zažene proces spletnega brskalnika s parametrom naslova prilagojene skripte v sistemu Moodle, ki prejme uporabnikov zahtevek in kreira globalni objekt *$user* in ročno sproži postopek avtentikacije v skripti */login/index.php*.

Prednost tega pristopa je v zelo enostavni implementaciji. Delo z bralnikom prstnih odtisov je enostavno v *.Net* okolju, veliko kode pa bi lahko uporabili že iz obstoječega sistema FingerIdent. Slabost je omejitev na platformo Windows

in zelo slaba uporabniška izkušnja pri tem načinu integracije bralnika prstnih odtisov v sistem Moodle. To je bil tudi glavni razlog, zakaj se nisem odločil za ta pristop.

Integracija bralnika prstnih odtisov v spletno aplikacijo z uporabo vtičnika za spletni brskalnik

Pri tem pristopu se upravljanje bralnika prstnih odtisov implementira v namenski vtičnik za spletni brskalnik. Bistvo tega pristopa je, da se logika zajema prstnega odtisa in shranjevanje vrednosti slike izvaja na uporabniški strani v brskalniku z uporabo kombinacije javascript in vtičnika za spletni brskalnik.

Z razvojem vtičnika za spletni brskalnik, ki upravlja z namensko strojno opremo, lahko pričnemo za tem, ko so razviti oziroma zagotovljeni elementi:

- gonilniki za delo s strojno opremo in
- programske knjižnice, ki omogočajo upravljanje z gonilnikom v programskem okolju.

Gonilniki in knjižnice za upravljanje s strojno opremo so razviti specifično za vsak operacijski sistem ločeno. V našem konkretnem primeru uporabe bralnika *Secugen Hamster Plus* proizvajalec nudi gonilnike in podporne knjižnice za razvoj v operacijskih sistemih *Windows* in nekaterih operacijskih sistemih, ki so osnovani na operacijskem sistemu *Unix*.

Platforma *Microsoft Windows*:

1. Gonilniki so na voljo za operacijske sisteme:
   - *Windows 7, Vista, XP, 2000, ME, 98SE, Windows Server 2008 R2, Windows Server 2003*;
2. Programske knjižnice za upravljanje z gonilnikom:
   - *sgfpamx.dll*
   - *sgfplib.dll*.

Platforma *Unix*:

1. Gonilniki so na voljo v sistemih:
   - *RedHat 9 Linux Kernel 2.4.20-8*,

- *Debian Linux Kernel 2.6.18-6-686*,
- *SPARC Solaris 10 Build 118833-33 (SUNBLADE 100)*,
- *SPARC Solaris 10 Thin Client (SUNRAY 150)* in
- *x86 Solaris 10 Build 118833-33 (Compaq DeskPro)*.

2. Programske knjižnice za upravljanje z gonilnikom:
   - *Libusb* in
   - *GIMP* razvojna orodja.

Ko so vsi osnovni gradniki za delo s strojno opremo delujoči, lahko pričnemo z razvojem spletnega vtičnika. Vsak spletni brskalnik ima svoj način implementacije spletnih vtičnikov. Več informacij o razvoju spletnih vtičnikov je opisanih v spletnih virih [24, 25, 26].

Pristop integracije bralnika prstnih odtisov v spletno aplikacijo z uporabo vtičnika za spletni brskalnik prinaša veliko prednosti in je zato boljši od pristopa z razvojem samostojnega programa, ki je opisan v poglavju 5.2.2. Njegova največja prednost je transparenta integracija bralnika prstnih odtisov v spletno aplikacijo in možnost uporabe vtičnika v različnih operacijskih sistemih. Tako lahko zagotovimo delujočo spletno aplikacijo, ne glede na operacijski sistem in spletni brskalnik v uporabi. Slaba stran tega pristopa je v tem, da vsak spletni brskalnik uporablja svoje lastne vtičnike in je zato potrebno implementirati več vtičnikov. Pri vsakem vtičniku je potrebno poskrbeti tudi za morebitne razlike v uporabljenih knjižnicah v oziru na ciljni operacijski sistem, za katerega poteka razvoj.

Po podrobnejšem raziskovanju problematike sem se zaradi vseh zgoraj opisanih prednosti odločil za integracijo bralnika prstnih odtisov v spletno aplikacijo z uporabo spletnega vtičnika za spletni brskalnik.

5.3 Sistem FingerIdent

Sistem FingerIdent predstavlja celovit sistem za biometrično verifikacijo uporabnikov [4]. Nastal je na Fakulteti za računalništvo in informatiko v okviru diplomske naloge z naslovom “Sistem za verifikacijo osebe na podlagi prstnega odtisa” [27]. Sedaj se sistem vzdržuje in razvija naprej kot del projekta *Sistem za razpoznavo prstnih odtisov* [28]. Delujoča verzija sistema je prikazana na

sliki 5.3. Postavljena je v testno okolje na Fakulteti za računalništvo in informatiko in s tem odraža svojo kvalitetno zrelostno stopnjo. Sistem FingerIdent je, podobno kot sistem Moodle, zastavljen zelo modularno in posamezne sklope sistema je enostavno integrirati v druge sisteme.

Slika 5.3: Sistem FingerIdent postavljen v testno okolje.

Poglavje 6

Pregled oblačnih platform s .NET podporo

Sistem FingerIdent je razvit v *.Net* okolju. Izbor oblačnih platform smo zato morali omejiti na tiste, ki nudijo podporo *.Net* okolju.

Najbolj primerne oblačne platforme za integracijo sistema FingerIdent so:

- Amazon Elastic Compute Cloud,
- AppHarbor,
- OpenStack,
- Uhuru,
- Tier 3,
- Apprenda,
- Iron Foundry in
- Microsoft Windows Azure.

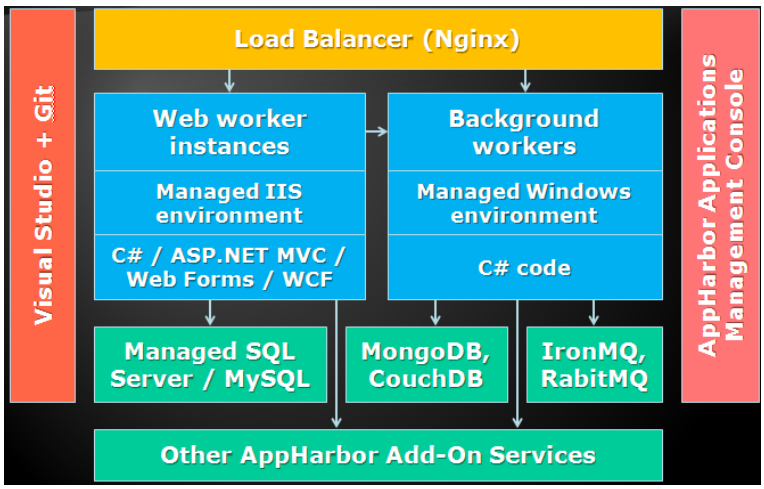
Iz seznama izstopata oblačni platformi Iron Foundry in OpenStack, ki sta predstavnika odprtokodnih oblačnih platform. V nadaljevanju bomo pregledali vse navedene oblačne platfome in končali z obširnejšim opisom uporabljene oblačne platforme Windows Azure.

6.1 Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud [29] je oblačna platforma, ki spada v IaaS arhitekturo. Omogoča nam storitve najemanja virtualnih računalnikov – *instanc*, na katerih tečejo operacijski sistemi *Microsoft Windows Server* ali različne distribucije *Linux* sistema. Za podporo virtualizacije je uporabljena tehnologija Xen [30]. Za podporo shrambam podatkov lahko uporabljamo Microsoft SQL Server ali shrambene storitve, kot so *Amazon Elastic Block Store*. Amazon zagotavlja potrebne vmesnike za upravljanje z virtualnimi sistemi in nadzorovanje celotnega sistema. Za konfiguracijo posameznih virtualnih sistemov pa mora uporabnik sam nastaviti vse potrebno in zagotoviti potrebno programsko opremo.

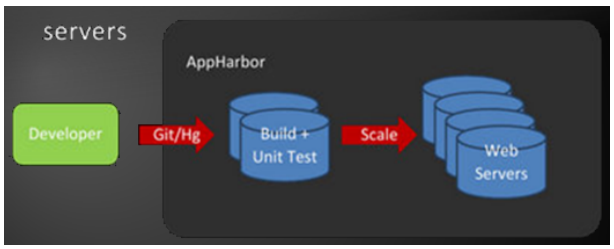
6.2 AppHarbor

AppHarbor [31] je oblačna platforma, ki je relativno mlada. Njeni začetki segajo v leto 2011. Spada v PaaS oblačno arhitekturo in je namenjena gostovanju *.Net* programskih rešitev. AppHarbor teče na infrastrukturi *Amazon Web Services* in uporablja oblačno platformo Amazon Elastic Compute Cloud. Arhitektura oblačne platforme AppHarbor je prikazana na sliki 6.1. Podpora za programsko okolje *.Net* je široka in obsega *asp.net*, *wcf*, *wwf*, *ado.net entity* ogrodje in druge tehnologije.



Slika 6.1: Arhitektura oblačne platforme AppHarbor.

Posebnost platforme AppHarbor je v načinu distribucije aplikacij na oblak (slika 6.2).



Slika 6.2: Proces distribucije *.Net* aplikacije na AppHarbor oblak.

Programer z uporabo orodja za delo z revizijami kode *git* naloži novo verzijo aplikacije na dodeljeni strežnik za prevajanje kode z namenom preverjanja morebitnih napak. Ko se ta proces uspešno zaključi, se poženejo vsi morebitni testi za preverjanje delovanja programske kode (angl. *unit tests*), če so implementirani v kodi. Ko se proces testiranja uspešno zaključi, se prevedena koda naloži na produkcijske strežnike porazdeljeno, glede na nastavitve oblačne storitve. Oblačna storitev na produkciji teče v poddomeni domene *apphb.com*.

Za preizkus oblačne platforme je na voljo preizkusno brezplačno obdobje, cenovna politika pa je odvisna od specifikacij projekta in števila strežniških instanc, na katerih želimo gostiti našo aplikacijo. AppHarbor platforma se zelo hitro razvija in z implementacijo dodatkov nudi podporo različnim programskim okoljem in podatkovnim bazam ter s tem pridobiva na priljubljenosti.

6.3 OpenStack

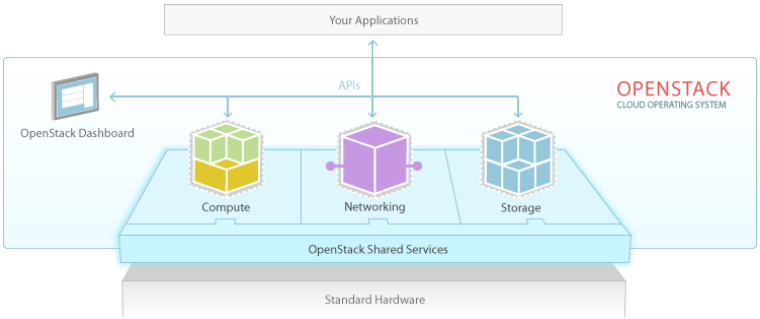
OpenStack [32] je odprtokodna oblačna platforma, ki spada v IaaS arhitekturo. Izvorna koda oblačne platforme je izdana pod *Apache2* licenco.

OpenStack platforma je sestavljena iz treh osnovnih komponent (slika 6.3):

- **računsko izvajalno okolje** (angl. *compute*) – omogoča kreiranje in upravljanje virtualnih okolij. Za programsko opremo v virtualnih okoljih moramo poskrbeti sami. Za shrambo podatkov lahko uporabimo virtualna okolja, na katerih tečejo strežniki SQL;

- **oblačna shramba podatkov** – platforma uporablja distribuirani shrambeni model za shranjevanje objektov in blokov podatkov. Platforma tu sama zagotavlja replikacijo podatkov v večih instancah in sinhronizacijo med njimi. V to komponento se shranjujejo tudi slike virtualnih okolij;
- **mrežna komponenta** – upravljanje z omrežjem in naslovi IP v oblačni platformi.

Za upravljanje oblačne platforme je na voljo nadzorna storitev *OpenStack Dashboard*. Z njeno uporabo lahko upravljamo s celotnim sistemom.



Slika 6.3: Oblačna platforma OpenStack.

OpenStack platforma je združljiva z oblačnima platformama Amazon Elastic Compute Cloud in Amazon S3 z uporabo namenskih API klicev.

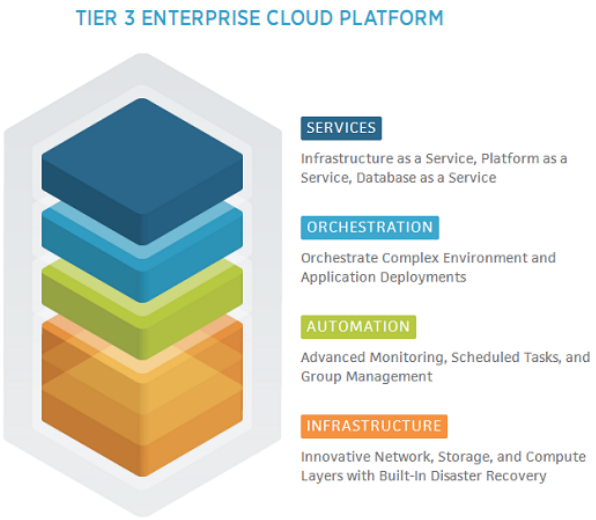
6.4 Uhuru

Podjetje Uhuru [33] je eno izmed novejših ponudnikov oblačnih storitev. Njihova oblačna platforma *Uhuru* je trenutno še v beta stopnji razvoja. Zasnovana je na obstoječi odprtokodni PaaS rešitvi Cloud Foundry [37] in nudi podporo programskim okoljem *PHP*, *Node.js*, *Ruby*, *Java* in *.Net*. Podpira podatkovne baze *MySQL*, *Microsoft SQL Server*, *RabbitMQ*, *MongoDB*, *Postgres* in *Redis*. Oblačna platforma Uhuru je uporabnikom na voljo v okviru brezplačnega beta programa.

6.5 Tier 3

Oblačna platforma Tier 3 je v začetku delovala le kot IaaS storitev, kasneje pa so ji dodali PaaS storitev, imenovano *Web Fabric* [34]. Podjetje Tier 3 na

tržišču deluje od leta 2006 in nudi zelo široko podporo za bolj pogosta programska okolja. Podjetje vzdržuje in razvija tudi odprtokodno PaaS oblačno platformo *Iron Foundry*, ki jo bomo podrobneje spoznali v poglavju 6.7. Plasti oblačne arhitekture Tier 3 so prikazane na sliki 6.4.



Slika 6.4: Plasti oblačne platforme Tier3.

Platforma Tier 3 podpira programska okolja *.Net*, *Node.js*, *PHP*, *Java* in *Python*. Tier 3 ima na voljo vse potrebne vmesnike in vsa orodja za upravljanje z oblačnimi aplikacijami in podatkovnimi shrambami podatkov. Podjetje zagotavlja visoko stopnjo zanesljivosti in predstavlja enega večjih tekmecev na področju ponudnikov oblačnih storitev.

6.6 Apprenda

Apprenda [35] je oblačna platforma tipa PaaS. Podjetje Apprenda predstavlja enega izmed najbolj zrelih ponudnikov oblačnih storitev na tržišču, ki nudi podporo za *.Net* programsko okolje.

Posebnost tega oblačnega ponudnika je, da poleg komercialne verzije oblačnega sistema ponuja tudi brezplačno *express* verzijo, ki si jo lahko uporabnik prenese na svojo obstoječo računalniško infrastrukturo in si vzpostavi privatni oblak.

Uporabniki si lahko zgradijo hibridni oblačni sistem s sinhronizacijo oblačne platforme Apprenda z javno oblačno platformo Windows Azure.

6.7 Iron Foundry

Iron Foundry [36] je odprtokodna PaaS oblačna platforma. Razvija jo podjetje Tier 3 in predstavlja razvejitev (angl. *fork*) od odprtokodnega projekta Cloud Foundry [37]. Iron Foundry je popolnoma združljiv z oblačno platformo Cloud Foundry, saj razvijalci obeh oblačnih platform tesno sodelujejo skupaj.

Iron Foundry se od oblačne platforme Cloud Foundry razlikuje v tem, da nudi polno podporo za *.Net* okolje in *Microsoft SQL Server* podatkovno bazo.

Razvijalcem je na voljo izvorna koda projekta, ki je zaščitena z licenco *Apache2*. Oblačna platforma Iron Foundry je na voljo za prenos v obliki virtualiziranega okolja. Tako si lahko zgradimo privatni PaaS oblak na svoji infrastrukturi ali pa lahko za gostovanje uporabimo virtualizacijsko platformo *VMWare vSphere* [38] ali IaaS oblačne platforme, kot so *Amazon Elastic Compute Cloud*, *CloudStack*, *OpenStack* in druge.

6.8 Microsoft Windows Azure

Microsoft Windows Azure [39] je oblačna platforma tipa PaaS. Poganja jo operacijski sistem Windows Azure, ki služi kot osnova vsem aplikacijam in zagotavlja storitve, potrebne za razvoj, upravljanje in gostovanje aplikacij.

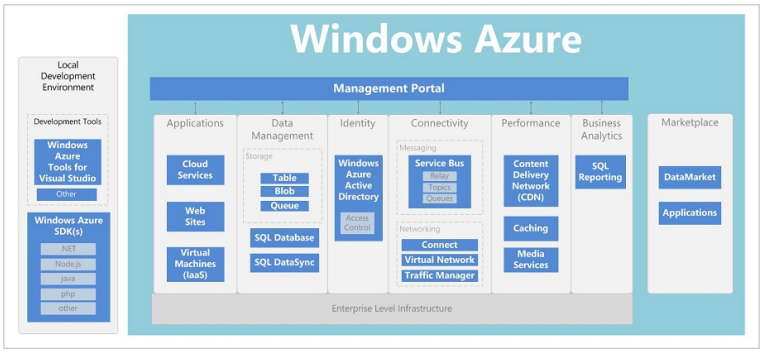
Windows Azure platforma ponuja uporabnikom v osnovi dve funkciji:

- računsko izvajalno okolje v oblaku in
- oblačno shrambo podatkov.

V platformi so ključne komponente (slika 6.5):

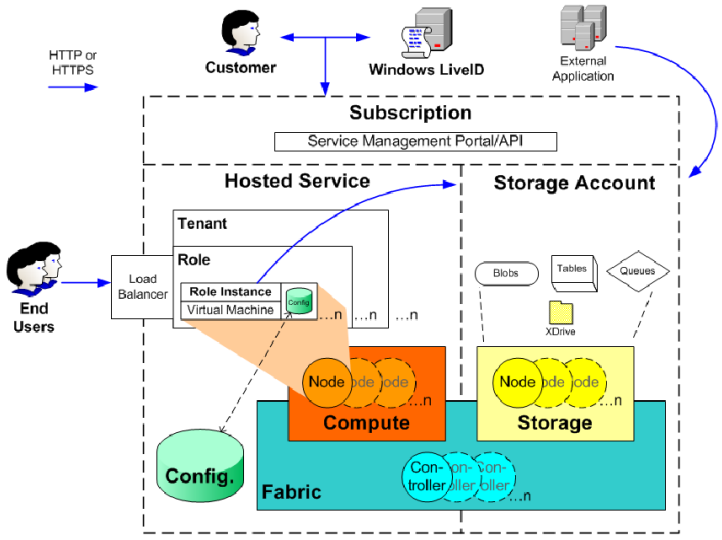
- **AppFabric** – predstavlja jedro distribuiranega sistema Windows Azure,
- **Storage** – shramba podatkov v podatkovna skladišča Blobs, Tables, Queues in SQL Azure,

- **Compute** – predstavlja računsko izvajalno okolje z vlogami: spletna vloga (angl. *web role*), delavska vloga (angl. *worker role*) in virtualizirana vloga (angl. *virtual machine role*).



Slika 6.5: Platforma Windows Azure.

Razmerja med komponentami *AppFabric*, *Storage* in *Compute* so prikazana na sliki 6.6.



Slika 6.6: Prikaz razmerij med ključnimi komponentami v Windows Azure.

Podprta programska okolja v Windows Azure so *.Net*, *Node.js*, *PHP*, *Java* in *Python*.

V Windows Azure lahko gostimo spletne strani in oblačne storitve. Oblačna storitev je postavljena v eno ali več različnih postavitvenih okolij, kot so produkcijsko in testna okolja. Vsaka postavitev oblačne storitve v okolje je sestavljena iz ene ali večih vlog (angl. *role*). Vsaka vloga je sestavljena iz enega ali večjega števila instanc, ki tečejo kot procesi v ločenih virtualnih okoljih.

Oblačna storitev je sestavljena iz naslednjih tipov vlog:

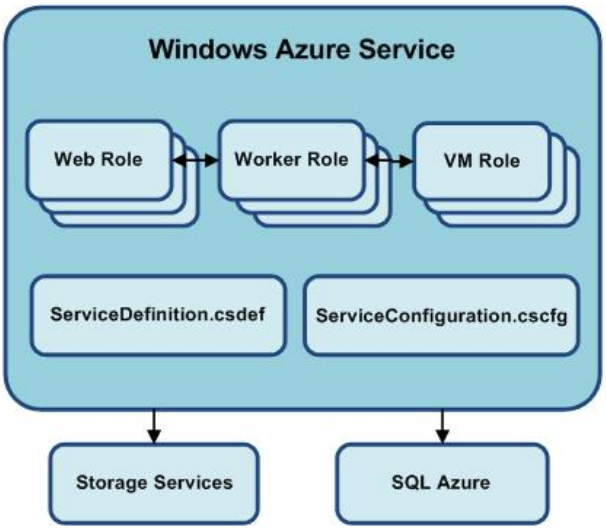
- **spletna vloga** (angl. *web role*) – vzpostavitev IIS strežnika za izvajanje strežniških aplikacij, ki predstavljajo uporabniški vmesnik oblačne storitve,
- **delavska vloga** (angl. *worker role*) – proces namenjen asinhronemu izvrševanju računsko intenzivnih operacij v ozadju in
- **virtualizirana vloga** (angl. *virtual machine role*) – trenutno je še vedno v beta različici in je namenjena ustvarjanju novih virtualnih okolij v Windows Azure. Uporabnik ima kontrolo nad posodobitvami, konfiguracijo in nameščenimi aplikacijami v virtualnem okolju.

Struktura oblačne storitve in prikaz vlog v njej sta prikazana na sliki 6.7. V datotekah *ServiceDefinition.csdef* in *ServiceConfiguration.csdef* so zapisane vse nastavitve oblačne storitve. Pri kreiranju nove oblačne storitve na oblačni platformi se aplikaciji dodeli poddomena domene *cloudapp.net*.

Za shranjevanje podatkov v oblačni platformi Windows Azure imamo na voljo:

- storitve za shranjevanje binarnih tipov podatkov (angl. *Blob services*),
- storitve za shranjevanje strukturiranih podatkov v tabele (angl. *Tables services*),
- storitve za zanesljivo komunikacijo med različnimi vlogami (angl. *Queues services*) in
- shranjevanje podatkov v SQL Azure relacijsko podatkovno bazo.

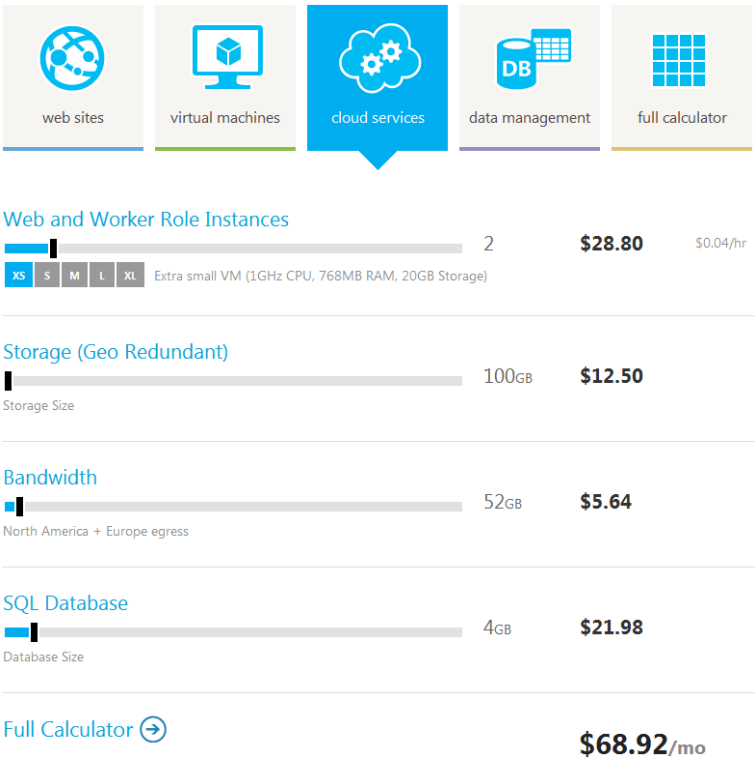
Prednost shranjevanja enostavnejših oblik podatkov v podatkovnih skladiščih (*Blob* in *Tables*), v primerjavi s standardno relacijsko podatkovno bazo SQL, je v ceni skladiščenja. Cena skladiščenja *100 GB* podatkov shranjenih v skladiščih *Blob* in *Tables* je enaka skladiščenju *2 GB* podatkov v relacijski bazi SQL Azure. Oblačne storitve lahko uporabljajo skladiščenje podatkov



Slika 6.7: Struktura oblačne storitve v Windows Azure.

hkrati v SQL Azure podatkovni bazi in v podatkovnih skladiščih. Razvijalci lahko optimizirajo skladiščenje velikih količin podatkov, ki jih zasedajo slike in datoteke v binarnih skladiščih, za uporabniške podatke pa še vedno uporabljajo standardno SQL relacijsko podatkovno bazo in s tem poskušajo izkoristiti prednosti obeh pristopov skladiščenja podatkov.

Za uporabo Windows Azure je potrebno skleniti s podjetjem Microsoft naročniško razmerje. Uporaba je plačljiva in se zaračunava po veljavnem ceniku glede na uporabljena sredstva, kot so število strežniških instanc, velikost podatkovne baze, količina prenesenih podatkov in ostalih kriterijev. V naročnini so vključene vse uporabnikove oblačne storitve in shrambeni računi. Na voljo je tudi brezplačno trimesečno obdobje, v katerem je vključena licenca za poganjanje do 10 spletnih strani in do 20 oblačnih storitev. Okvirni cenik za gostovanje oblačne storitve *FingerIden*t v oblačni platformi Windows Azure je prikazan na sliki 6.8.



Slika 6.8: Cenik za oblačno storitev FingerIdent v Azure.

Med zgoraj opisanimi oblačnimi platformami sem pri izbiri najustreznejše med njimi za integracijo sistema FingerIdent upošteval naslednje kriterije:

- **brezplačna uporaba oblačne platforme** – v ta kriterij sem vključil tudi brezplačna testna obdobja,
- **javno dosegljiva oblačna storitev** – oblačna storitev je morala biti javno dosegljiva za testiranje iz večjega števila zunanjih lokacij in
- **oblačna platforma javnega značaja** – pri javnih oblačnih platformah ponudnik sam skrbi za potrebno mrežno in računalniško infrastrukturo.

Pri pregledu vseh kriterijev se je oblačna platforma Windows Azure izkazala kot najprimernejša. Če bi imel pri razvoju na voljo primerno računalniško infrastrukturo za postavitev privatnega oblačnega sistema in bi s tem lahko

odstranil kriterij uporabe oblačne platforme javnega značaja, bi za integracijo sistema FingerIdent uporabil oblačno platformo Iron Foundry zaradi njenega odprtokodnega porekla. O resnični ustreznosti uporabe oblačne platforme Windows Azure pri delu z zaupnimi podatki se lahko prepričamo z izvedbo analize tveganja. V ta namen je bil opravljen poenostavljen pregled varnostnih mehanizmov.

6.8.1 Varnost v oblačni platformi Windows Azure

Microsoft obljublja visoko stopnjo varnosti pri delu s podatki v oblačni platformi Windows Azure. Pregled varnostnih mehanizmov smo izpeljali iz varnostnih specifikacij Windows Azure platforme [40].

Varnostni mehanizmi, ki so na voljo v oblačni platformi Windows Azure:

- **nadzor nad varnostjo dostopa do podatkov** z omejevanjem dostopa na avtorizirane entitete z uporabo varnostnih ključev in identifikacijskih certifikatov,
- **izolacija podatkov** – reševanje problema sobivanja – multi-tenancy z izolacijo in ščitenjem podatkovnih skladišč pred uporabniškimi podatki ostalih uporabnikov,
- **enkripcija podatkov** znotraj platforme med komponentami in med oblačnim skladiščem podatkov in uporabnikom z uporabo SSL enkripcije za zagotavljanje varnosti pri prenosu podatkov,
- **izbris podatkov iz Windows Azure strežnikov** – platforma zagotavlja resničen izbris podatkov na strežniku pri brisanju uporabniških podatkov iz skladišča,
- **zanesljivost podatkov** – podatki so replicirani v treh ločenih kopijah znotraj podatkovnega centra, da se minimizira vpliv okvarjenih strojnih komponent. Ne glede na trenutno uporabljeno podatkovno skladišče platforma zagotavlja sinhronizacijo med vsemi kopijami. Uporabniki si lahko zagotovijo izdelavo varnostnih kopij podatkov z uporabo ločenih podatkovnih skladišč in implementacijo vloge delavca, ki skrbi za izdelavo varnostnih kopij med podatkovnimi skladišči.

Microsoft zagotavlja varnost tudi pri ostalih dejavnikih zunaj virtualnega sveta, in sicer pri zaposlenih, ki imajo dostop do Windows Azure podatkovnih centrov in pri fizični zaščiti podatkovnih centrov.

Poglavje 7

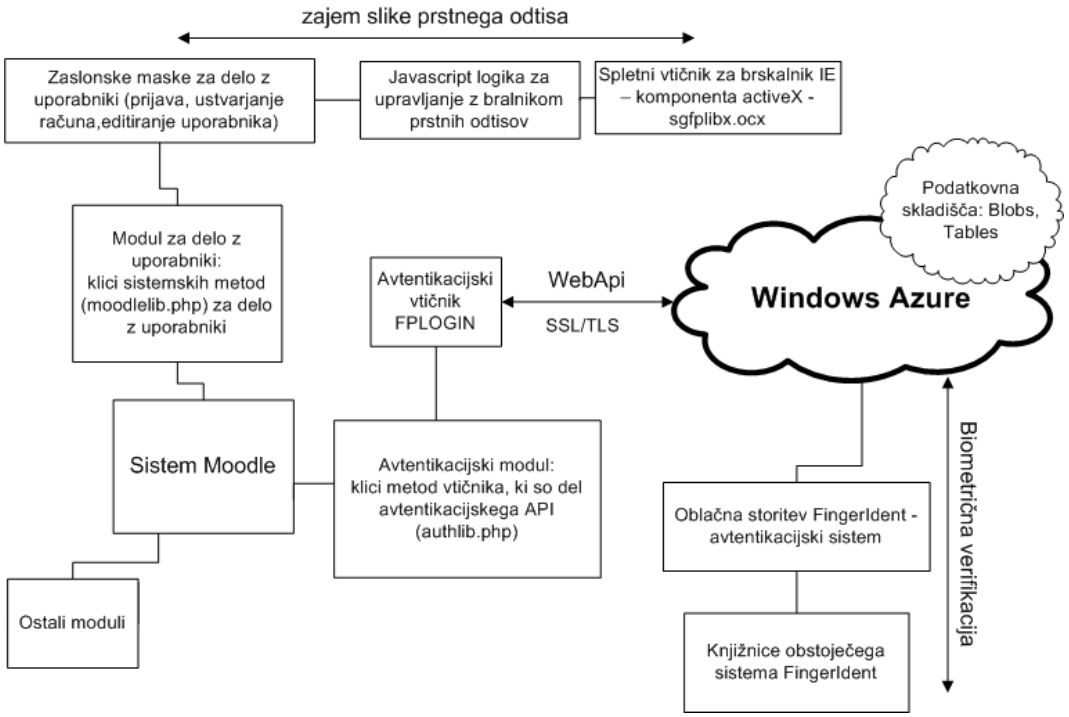
Razvoj biometrične verifikacije v oblaku in prikaz njene uporabe na praktičnem primeru

V okviru praktičnega dela diplomske naloge smo z integracijo obstoječega verifikacijskega sistema FingerIdent v oblak implementirali oblačno storitev. Za prikaz uporabe te oblačne storitve na dejanskem primeru smo uporabili sistem Moodle. Moodle ni omogočal avtentikacije z biometrično metodo razpoznavanja prstnih odtisov in zato smo mu morali dodati manjkajoče funkcionalnosti.

V nadaljevanju tega poglavja bomo podrobno opisali celoten postopek razvoja oblačne storitve in potrebnih prilagoditev v sistemu Moodle za delo z bralnikom prstnih odtisov ter potrebnih korakov pri povezovanju tega sistema na biometrično oblačno storitev *FingerIdent*.

Posebno pozornost smo namenili razvoju varnostnih elementov v sistemu. Poglavje bomo zaključili s podrobnim pregledom vseh implementiranih varnostnih mehanizmov v sistemu in nazornim prikazom, v katerih situacijah so ti mehanizmi uporabni.

Pregled celotnega sistema je prikazan na sliki 7.1. Most med oblačno storitvijo *FingerIdent* in sistemom Moodle predstavlja implementacija avtentikacijskega vtičnika *fplogin*.



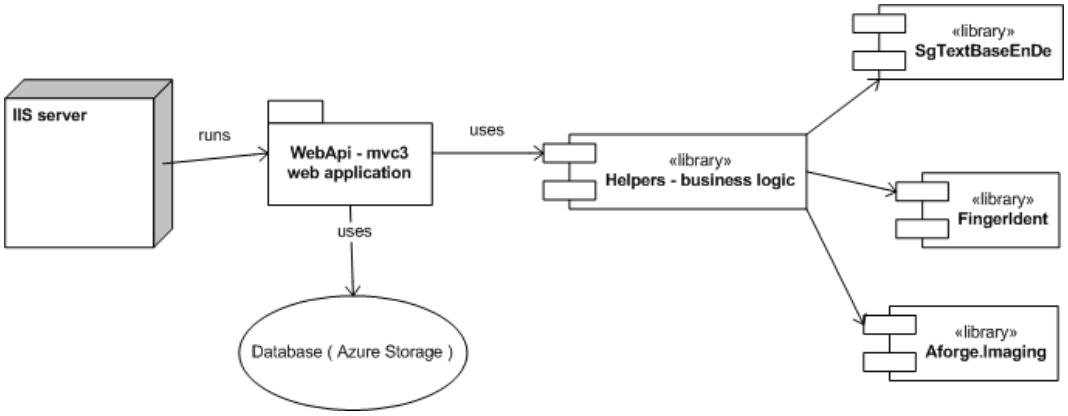
Slika 7.1: Pregled celotnega sistema.

7.1 Integracija verifikacijskega sistema FingerIdent v oblak

7.1.1 Razvoj spletne aplikacije

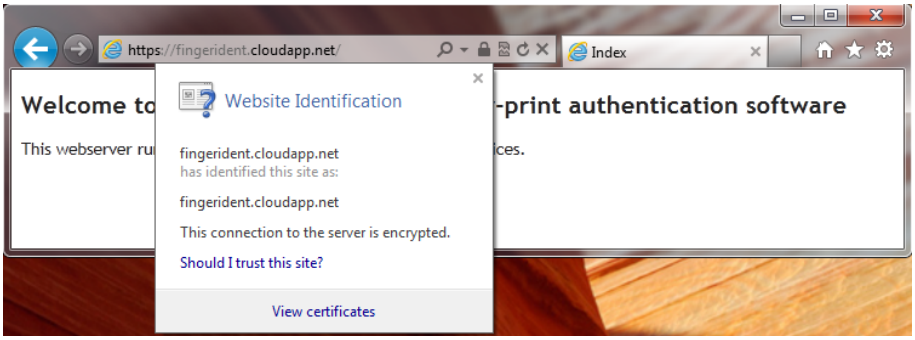
Proces upravljanja z identitami uporabnikov se izvaja na zunanjem avtentikacijskem strežniku, ki teče kot *.Net* spletna aplikacija [41]. To okolje sem si izbral zaradi boljše združljivosti z obstoječimi knjižnicami sistema FingerIdent. Posebnost ogrodja *.Net* je njegovo skupno izvajalno okolje CLR (angl. *common language runtime*), ki omogoča razvoj aplikacij v različnih programskih jezikih v *.Net* okolju. Izvorna koda programov se pretvori v vmesno kodo MSIL (angl. *intermediate language*) in ne v strojno kodo, za kar se uporablja prevajalnik JIT (angl. *just-in-time*). Podobno rešitev poznamo tudi v programskem jeziku Java. Rezultat takega pristopa uporabe vmesne kode je možnost uporabe poljubnega *.Net* programskega jezika in večja varnost pri izvajanju programov zaradi uvedbe varnostnega koncepta peskovnika v programsko okolje *.Net* [42].

Za implementacijo strežnika smo uporabili ogrodje *MVC3* zaradi enostavne integracije v oblačno platformo Windows Azure [43]. Shemo strežnika prikazuje slika 7.2.



Slika 7.2: Shema avtentikacijskega strežnika.

Slika 7.3 prikazuje vstopno stran strežnika s podrobnostmi SSL certifikata, s katerim spletna stran izkazuje svojo identiteto uporabniku.



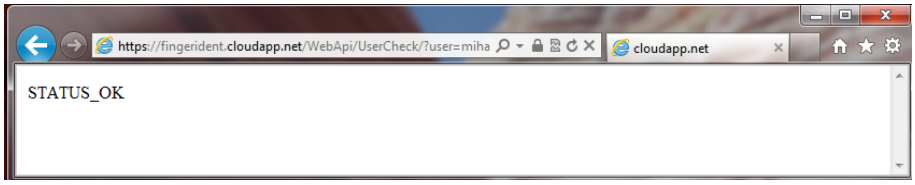
Slika 7.3: Vstopna stran avtentikacijskega strežnika.

Obstoječi sistem Moodle vsebuje že vso potrebno prikazno logiko za delo z uporabniki in zato je ni bilo potrebno dodatno implementirati tudi v spletno aplikacijo.

Zahtevane funkcionalnosti spletne aplikacije:

- avtentikacija obstoječega uporabnika,
- dodajanje novega uporabnika,
- urejanje obstoječega uporabnika,
- brisanje uporabnika in
- preverjanje obstoja uporabnika.

Zgoraj navedene funkcionalnosti smo realizirali z implementacijo spletnega API na strežniku. Na ta način smo zagotovili popolno združljivost strežnika s poljubno aplikacijo, ki ima možnost generiranja https zahtev. API dokumentacija je priložena v dodatku A. Delovanje spletnega API lahko testiramo tudi v spletnem brskalniku, kot je prikazano na sliki 7.4.

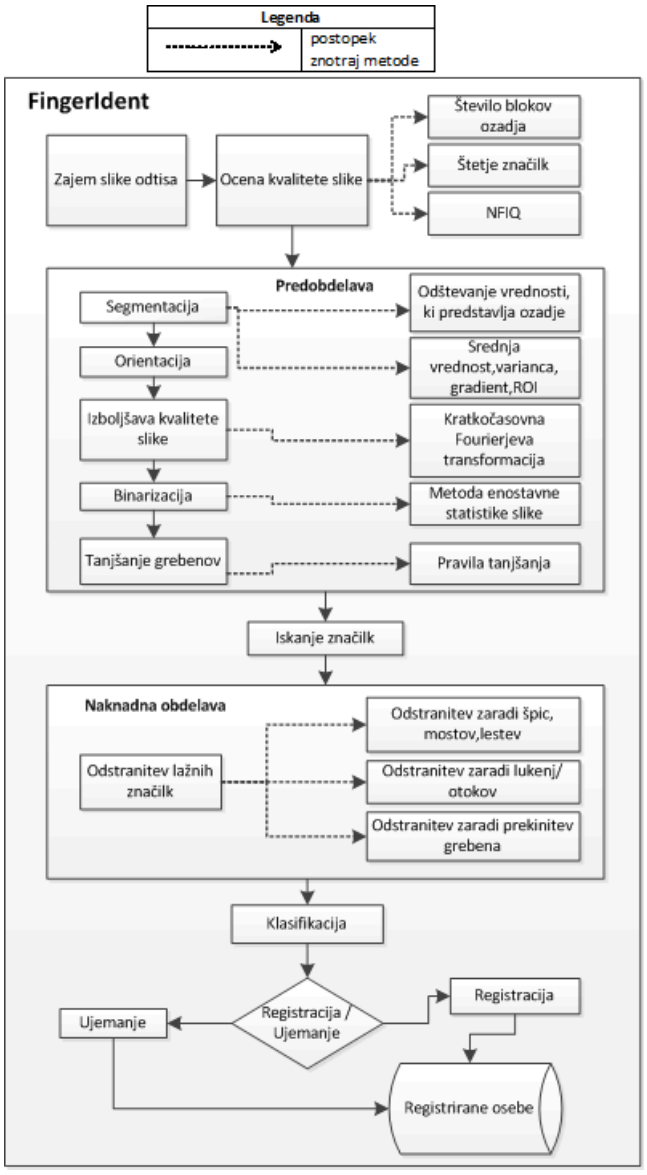


Slika 7.4: Preverjanje delovanja WebApi metode UserCheck.

7.1.2 Integracija sistema FingerIdent v spletno aplikacijo

Pri uporabi sistema FingerIdent smo naleteli na težave pri pretvorbi iz tekstovne predstavitve slike v zahtevani format slike BMP. Glavna težava je bila pri pretvorbi iz tipa *string* v polje *byte[]* zaradi neustreznega kodirnega formata slike, ki ga uporablja Secugen Hamster Plus bralnik. Problem smo rešili s sodelovanjem podporne ekipe podjetja Secugen. Posredovali so nam komponento DLL, spisano v programskem jeziku *C++*, ki vsebuje metodi za ustrezno pretvorbo podatkov in je kompatibilna z njihovim bralnikom. Ko je bil problem pretvorbe tekstovne predstavitve slike v binarno predstavitev rešen, je postal postopek pretvorbe v 8–bitno sliko formata BMP trivialen.

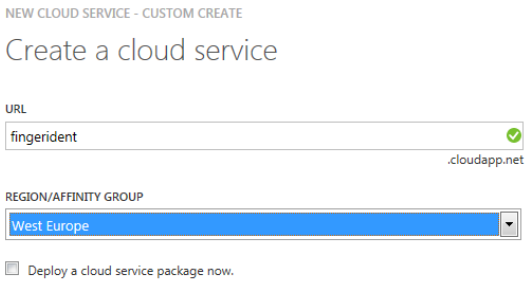
V sistemu FingerIdent je že implementiran modul za delo s podatkovno bazo uporabnikov, ki ni združljiv z načinom shranjevanja podatkov v oblačni platformi Windows Azure. Posledično smo morali implementirati ločeni modul za shranjevanje podatkov v oblačno platformo Windows Azure. Delovanje obstoječega sistema FingerIdent je prikazano na sliki 7.5.



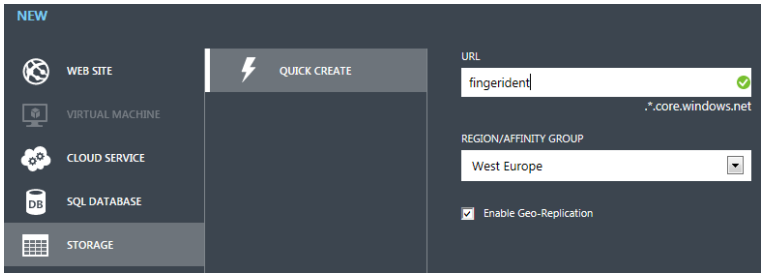
Slika 7.5: Prikaz delovanja sistema FingerIdent.

7.1.3 Integracija spletne aplikacije v oblačno platformo Windows Azure

Za uporabo oblačne platforme Windows Azure lahko uporabljamo simulirano okolje, ki je priloženo razvojnemu paketu SDK, ali pa z registracijo sklenemo naročniško razmerje s podjetjem Microsoft in oblačne storitve prenesemo v testno ali produkcijsko okolje. Integracija spletne aplikacije v oblačno storitev platforme Windows Azure se prične z definiranjem nove oblačne storitve in podatkovnih skladišč, ki bodo na voljo oblačni storitvi *FingerIdent*, kot je prikazano na slikah 7.6 in 7.7.



Slika 7.6: FingerIdent oblačna storitev.



Slika 7.7: FingerIdent podatkovno skladišče.

Na spletu se nahaja nekaj uporabnih vodičev, s katerimi sem si lahko pomagal pri migraciji mvc3 spletne aplikacije v oblačno platformo Windows Azure [44]. Spletni aplikaciji je bilo potrebno dodeliti spletno vlogo v konfiguraciji Azure projekta. To pomeni, da bo platforma Windows Azure spletno aplikacijo pognala na IIS strežniku.

Za povezavo spletne vloge s https protokolom je bilo potrebno uporabiti SSL certifikat, s katerim se spletna aplikacija predstavi brskalniku kot zaupanja vredna.

Pri SSL certifikatu so bile na voljo dve možnosti:

- nakup komercialnega SSL certifikata ali
- generiranje lastnega SSL certifikata.

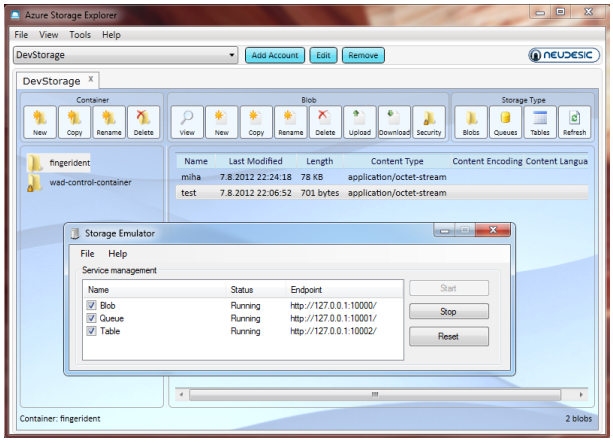
Komercialni certifikati za uporabo v produkcijskih okoljih so dragi. Cene se gibljejo okoli 500 $ in več. Zaradi visokih cen sem se odločil za generiranje lastnega SSL certifikata, ki je za potrebe te diplomske naloge dovolj dober.

Pri generiranju lastnega certifikata moramo sami prevzeti vlogo certifikatne agencije. Ko prevzamemo vlogo certifikatne agencije, lahko sami generiramo in podpisujemo SSL certifikate. Celoten postopek prevzemanja vloge certifikatne agencije in generiranja SSL certifikatov z uporabo odprtokodnega orodja *OpenSSL* [45] je mogoče prebrati v spletnem vodiču [46, 47]. Podoben postopek z uporabo Microsoftovega orodja *makecert* je opisan v vodiču [48]. Certifikat je bilo potrebno nastaviti za uporabo na domeni *fingerident.cloudapp.net*. Uporabljeni postopek generiranja certifikatov ima tudi slabe lastnosti, saj tak certifikat ne velja za zaupanja vrednega in ga je potrebno ročno namestiti pod zaupanja vredne certifikate.

V praksi bi bilo v produkcijskem okolju potrebno uporabiti plačljiv komercialni certifikat enega izmed podjetij, ki imajo vlogo zaupanja vrednih certifikatnih agencij. Primeri takih podjetij so *Verisign*, *Thawte*, *Go Daddy*, *GeoTrust*, *DigiCert*, *Start SSL*, *Instant SSL by Comodo* in mnogi drugi.

Za shranjevanje podatkov v oblačni platformi Windows Azure imamo možnost izbire med SQL Azure relacijsko podatkovno bazo in skladiščenjem podatkov v oblikah skladišč, ki pa še vedno ponujajo zelo dobre performanse in skalabilnost glede na obremenitve. Kljub temu, da SQL Azure ponuja prednosti skladiščenja podatkov v smislu lažje prenosljivosti baze podatkov v druge oblačne sisteme, sem se odločil za uporabo podatkovnih skladišč, saj ponujajo primerno obliko interakcije s sistemom FingerIdent. Za shranjevanje uporabniških podatkov smo uporabili skladišče imenovano *Tables*, ki je zelo podobno standardnim SQL tabelam. *Tables* ima vgrajeno podporo za shranjevanje različnih tipov podatkov, med katere sodijo tudi *integer*, *string* in *byte[]*.

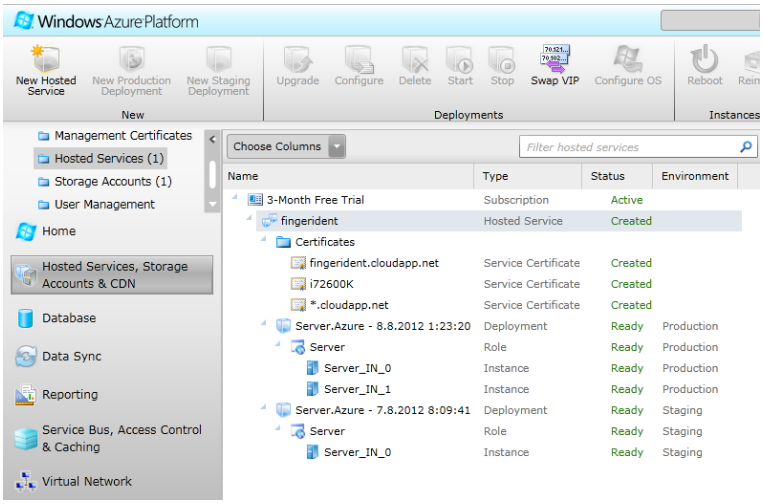
Pri shranjevanju binarnih podatkov v skladišče *Tables* je predpisana zgornja meja podatkov za posamezni vpis na 64 kB. Pri uporabi sistema FingerIdent je veliko pretvorb med poljem vrste *byte[]* in objekti vrste *MyPerson*. Ker so bila ta polja velikosti najmanj 78 kB (zgornja meja lahko znaša tudi 780 kB) ni bilo možno shranjevanje binarnih podatkov v podatkovnem skladišču *Tables*. Zato je bilo potrebno za skladiščenje podatkov *byte[]* uporabiti skladišče podatkov imenovano *Blob–binary large object storage*, ki je optimizirano za shranjevanje binarnih podatkov. Podatkovni skladišči *Blob* in *Tables* je bilo potrebno povezati z oblačno storitvijo v upravljalnem vmesniku oblačne platforme Windows Azure. Binarno skladišče podatkov je prikazano na sliki 7.8.



Slika 7.8: Azure simulator s simuliranim skladiščem podatkov.

Windows Azure omogoča postavitev oblačne storitve v več različnih postavitvenih okolij, kot na primer oblačno storitev, ki teče v fazi produkcije, fazi testiranja ali pa je v fazi mirovanja. To omogoča učinkovito upravljanje z oblačnimi storitvami na produkcijskem in testnem okolju, saj je preklapljanje med njimi zelo enostavno. Preklapljanje med okolji oblačne storitve izvajamo v upravljalni konzoli, ki je prikazana na sliki 7.9.

Za zagotovitev večje zanesljivosti oblačne storitve poskrbimo s tem, da definiramo več instanc spletne vloge, na katerih bo tekla naša storitev. V našem primeru sem zagotovil zanesljivost z dvema. To informacijo Azure intepretira tako, da spletno vlogo oblačne storitve zažene iz dveh virtualnih okolij, ki lahko tečeta na istem ali različnih fizičnih strežnikih. Informacija o dejanskem številu fizičnih računalnikov, ki jih oblačna storitev zaseda, nam ni na voljo. Oblačna platforma Windows Azure zagotavlja, da bo v primeru izpada



Slika 7.9: Azure upravljalno okolje.

virtualnega sistema takoj zagnala novo instanco virtualnega okolja, ki bo nadomestila izpadlo. Število instanc spletne vloge bi lahko bilo tudi večje. V primeru brezplačne naročnine sem izbiral med 20 virtualnimi okolji, ki se porazdelijo med vse oblačne storitve in vsemi postavitvenimi okolji. To pomeni, da smo izkoristili dvanajst ločenih virtualiziranih okolij, če imamo na oblaku tri ločene oblačne storitve, ki tečejo na dveh instancah spletne vloge v testni in produkcijski stopnji razvoja.

Zanesljivost podatkovnih skladišč zagotavlja platforma. Podatkovna skladišča so replicirana trikrat v podatkovnih skladiščih. Za sinhronizacijo med njimi skrbi platforma sama. Dodana je tudi možnost dodatnega podatkovnega skladišča, kamor se prenašajo varnostne kopije podatkov.

7.2 Implementacija biometrične verifikacije v sistem Moodle

Implementacija biometrične verifikacije v sistem Moodle je razdeljena v tri dele:

- razvoj zaslonskih mask,
- razvoj in dopolnitev potrebnih sistemskih knjižnic ter

- razvoj avtentikacijskega vtičnika.

V nadaljevanju bomo opisali vse tri sklope. V dodatku B se nahaja seznam vseh sprememb, narejenih v sistemu Moodle, ki je namenjen boljšemu razumevanju tega poglavja.

7.2.1 Razvoj zaslonskih mask

Razvoj zaslonskih mask poteka z implementacijo prikazne logike v predlogah (angl. *template*) in podporne logike v skriptah php. Predloge so shranjene v obliki html in imajo v imenu pripono *_form*. Taka zasnova ločevanja logike omogoča enostavnejši in hitrejši razvoj zaslonskih mask.

Integracija spletnega vtičnika za upravljanje bralnika prstnih odtisov v zaslonske maske

Integracija bralnika prstnih odtisov v zaslonske maske je implementirana z uporabo spletnega vtičnika, ki so ga razvijalci podjetja *Secugen* priložili razvojnemu paketu SDK. Razviti spletni vtičnik je implementiran v obliki komponente activeX in je namenjen za delo v spletnem brskalniku *Internet Explorer* [49].
Komponento activeX *sgfplibx.ocx* v spletno stran vključimo s kodo:

```
<OBJECT
  ID="objFP"
  CLASSID="CLSID:D547FDD7-82F6-44e8-AFBA-7553ADCEE7C8"
     style="LEFT:0px;WIDTH:149px;TOP:0px;HEIGHT:182px"
     height="182" width="149"
  CODEBASE="sgfplibx.ocx" VIEWASTEXT>
    <PARAM NAME="CodeName" VALUE="1">
</OBJECT>
```

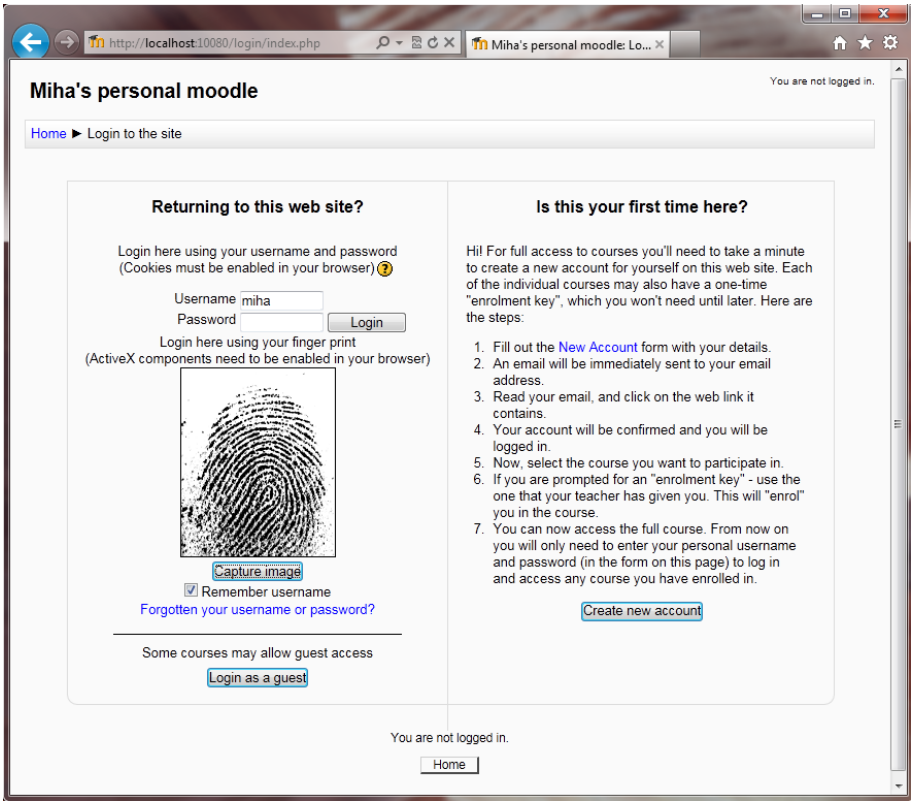
Interakcija z vtičnikom poteka v skriptnem jeziku Javascript ali VBScript. Komponente activeX so uradno podprte samo v brskalniku Internet Explorer, kar predstavlja slabost pri implementaciji, saj se s tem omeji uporaba samo na operacijske sisteme Microsoft Windows. Neuradno obstajajo tudi vtičniki za alternativne brskalnike. Za brskalnik Google Chrome obstaja nekaj vtičnikov [50, 51], vendar kljub temu še vedno ostaja omejitev na delovanje v operacijskih sistemih Windows.

Če bi hoteli omogočiti uporabo bralnika prstnih odtisov tudi na alternativnih operacijskih sistemih, bi to dosegli z razvojem lastnega vtičnika, kot je opisano v poglavju 5.2.2.

Maska za prijavo

Uporabniku je za avtentikacijo (slika 7.10) na voljo več scenarijev uporabe:

- prijava z uporabniškim imenom in geslom,
- prijava z uporabniškim imenom in prstnim odtisom ali
- prijava z uporabniškim imenom, geslom in prstnim odtisom.



Slika 7.10: Zaslonska maska za prijavo.

Logika maske je implementirana v skripti */login/index.php*. Pri implementaciji logike sem omogočil vse tri scenarije za avtentikacijo uporabnika. Opravi se

analiza vnešenih uporabniških podatkov in izvede se primeren scenarij. Omejitev avtentikacije na posamezen scenarij ali poljubno kombinacijo scenarijev je mogoče realizirati na nivoju avtentikacijskega vtičnika in jo bomo opisali v poglavju 7.2.3. Če uporabnik ne vnese potrebnih podatkov za uspešno avtentikacijo, je o tem obveščen. Prav tako se uporabniku izpiše obvestilo ob neuspešnem avtentikacijskem poizkusu.

Vsi avtentikacijski dostopi potekajo na zunanji avtentikacijski strežnik, kjer se opravlja proces verifikacije uporabnika. Strežnik vrne rezultat *AUTH_OK* ali *AUTH_FAILED*.

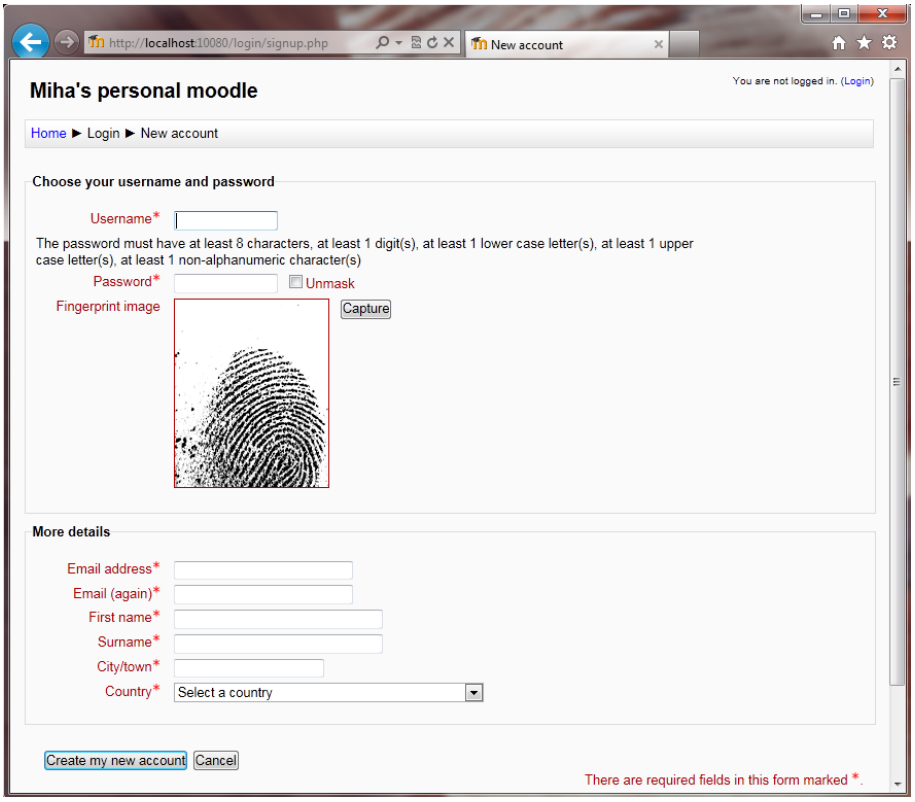
Maska za ustvarjanje uporabniškega računa

Masko za ustvarjanje uporabniškega računa smo dopolnili z možnostjo dodajanja prstnega odtisa uporabniškemu računu (slika 7.11). Možnost uporabe bralnika prstnih odtisov je dodana le kot dodatna možnost in ne kot obvezen parameter, za kar obstajata dva razloga:

- najpomembnejši razlog je, da nima vsak uporabnik pri sebi bralnika prstnih odtisov. Če bi bil prstni odtis obvezen parameter, bi s tem onemogočili uporabo sistema za takšne uporabnike. Uporabnikom, ki nimajo bralnika prstnih odtisov, je za avtentikacijo namenjen scenarij z uporabniškim imenom in geslom;

- spletne maske in podporna logika so enake, ne glede na vrsto uporabljenega vtičnika za opravljanje prijave novih uporabnikov v sistemu Moodle. Administrator sistema Moodle lahko prijavi različne vtičnike za opravljanje vloge ustvarjanja novih računov, kot je na primer vtičnik, ki omogoča prijavo z uporabo e-pošte (angl. *email-based self-registration plugin*). Logika, razvita za delo s parametrom zajetega prstnega odtisa, pa je na voljo samo v avtentikacijskem vtičniku *fplogin*. Ostali vtičniki bodo zahtevo za prijavo uporabnika z uporabo bralnika prstnih odtisov zavrnili in zato je boljša možnost, da ostane prstni odtis opcijski parameter.

Proces kreiranja novega uporabniškega računa pomeni ustvarjanje novega sinhroniziranega uporabniškega računa na avtentikacijskem strežniku in v sistemu Moodle. V sistem Moodle se ne shranjuje prstnih odtisov in gesel uporabniških računov. Ta dva tipa informacij se shranjujeta in preverjata samo na zunanjem strežniku.



Slika 7.11: Zaslonska maska za ustvarjanje novega računa.

Odgovor strežnika je *STATUS_OK* v primeru uspešnega ali *STATUS_FAILED* v primeru neuspešnega postopka analize prstnega odtisa.

Ostale maske

Maski za urejanje uporabniškega računa in ob izgubi uporabniškega gesla sta enaki originalnim. Vse funkcionalnosti pri teh maskah so implementirane v avtentikacijskem vtičniku *fplogin*, ki skrbi za sinhronizacijo uporabniških podatkov in vso potrebno podporno logiko.

7.2.2 Razvoj in dopolnitev potrebnih sistemskih knjižnic

Pomembne funkcije za delo z uporabniki se nahajajo v skriptah:

- `/lib/moodlelib.php` in

- `/lib/authlib.php`.

V skripti */lib/authlib.php* je implementiran osnovni razred *auth_plugin_base*, ki služi kot osnovni razred vsem avtentikacijskim vtičnikom. Z razredom *auth_plugin_base* je v sistemu Moodle implementiran avtentikacijski API.

Razredu sem dodal novi metodi *user_fp_login* in *user_signup_fp*. S tem smo razširili avtentikacijski API s potrebnimi funkcionalnostmi za uporabo novega tipa avtentikacije z uporabo bralnika prstnih odtisov.

V skripti */lib/moodlelib.php* so implementirane vse metode za delo z uporabniki. Dodal sem metodo *authenticate_user_fp_login*, ki omogoča prijavo uporabnika v sistem z uporabo bralnika prstnih odtisov. Ta metoda, v primerjavi z obstoječo *authenticate_user_login*, opravlja klice metod vtičnika, ki so namenjene delu z novo vrsto avtentikacije.

Po implementaciji vseh sprememb v sistemskih knjižnicah se lahko lotimo implementacije avtentikacijskega vtičnika. Z implementacijo lastnega vtičnika pridobimo možnost zunanje avtentikacije z oblačno storitvijo *FingerIdent*.

7.2.3 Razvoj avtentikacijskega vtičnika

Za razvoj lastnega avtentikacijskega vtičnika za sistem Moodle je potrebno preučiti uradno dokumentacijo [22].

Osnovna struktura komponent v avtentikacijskem vtičniku:

- **auth.php** – glavni razred avtentikacijskega vtičnika, ki obvezno razširja osnovni razred *auth_plugin_base(authlib.php)*. S tem se definira sklop funkcionalnosti, ki morajo biti na voljo v vtičniku za uporabo v sistemu Moodle;

- **version.php** – v tej komponenti se definira trenutno verzijo vtičnika in minimalno verzijo sistema Moodle, ki je združljiva z vtičnikom. Te nastavitve so zelo pomembne in se upoštevajo pri namestitvi vtičnika v sistem Moodle. Opcijsko se lahko doda še informacije o zrelostni stopnji vtičnika in informacije o avtorju;

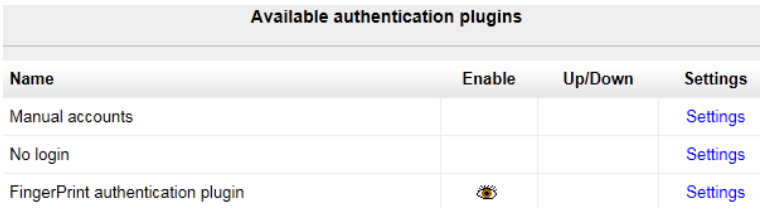
- **config.html** – spletna stran, ki je namenjena specifičnim nastavitvam vtičnika, ki so na voljo administratorju;

- **lang/language_code/auth_name.php** – definirani tekstovni prevodi, ki so uporabljeni v avtentikacijskem vtičniku za uporabljeni jezik, specificirani z ISO kodo – *language_code*.

Avtentikacijski vtičnik mora v glavnem razredu (*auth.php*) iz osnovnega razreda reimplementirati vsaj metodo *user_login(authlib.php)*, v primeru našega vtičnika pa še metodo *user_fp_login(authlib.php)*. Brez reimplementacije obveznih metod je avtentikacijski vtičnik neuporaben za delo.

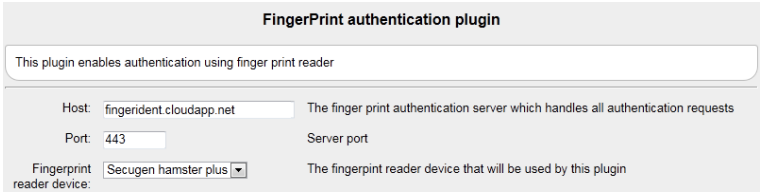
Ostale metode, ki jih je potrebno reimplementirati za implementacijo polno delujočega vtičnika, ki omogoča prijavo, urejanje, poizvedovanje, brisanje in vse ostale pomembne stvari, so opisane v uradni dokumentaciji.

Implementirani vtičnik moramo v avtentikacijskih nastavitvah omogočiti in ga s tem vključiti v proces avtentikacije. Prikaz omogočenega vtičnika prikazuje slika 7.12.



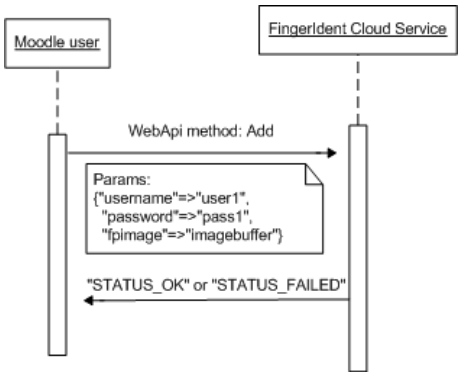
Slika 7.12: Nastavitve avtentikacijskih vtičnikov.

Vtičnik moramo nato nastaviti, da se bo povezoval na pravilen spletni naslov in vrata strežnika (slika 7.13). Vse nastavitve vtičnika se vnašajo na spletno stran *config.html*, ki je del vtičnika in služi za vnos pomembnih parametrov za delovanje vtičnika. Te nastavitve so na voljo administratorju sistema Moodle. V nastavitvah vtičnika bi lahko omejili možne scenarije za avtentikacijo z izbiro posameznih možnosti. Avtentikacijski vtičnik bi v tem primeru zavrnil tip avtentikacije, ki bo v nasprotju z administratorjevimi nastavitvami in bo obvestil uporabnika z obvestilom o vrsti napake.



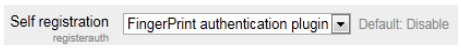
Slika 7.13: Nastavitve avtentikacijskega vtičnika *fplogin*.

Vtičnik *fplogin* komunicira z zunanjim strežnikom s pomočjo php knjižnice cURL [52]. Z njo ustvarja https dogodke in prejema rezultate. Komunikacija poteka z uporabo spletnega API, ki je dosegljiv na avtentikacijskem strežniku. Primer uporabe spletnega API za dodajanje novega uporabnika je viden na sliki 7.14.



Slika 7.14: WebApi metoda za ustvarjanje novega uporabnika.

Administrator lahko določi v sistemu Moodle samo en vtičnik, ki opravlja vlogo registracije novih uporabnikov. Vtičnik se registrira v globalnih nastavitvah avtentikacijskih vtičnikov, kot je prikazano na sliki 7.15.



Slika 7.15: Registracija vtičnika za opravljanje vloge kreiranja uporabnikov.

Moodle ima v svoji interni tabeli uporabnikov polje *auth*, ki predstavlja referenco na uporabnikov dodeljeni vtičnik. Pri uspelem procesu prve prijave no-

vega uporabnika v sistem Moodle se v tabelo uporabnikov poleg uporabniških podatkov zapiše tudi ime vtičnika, ki je izvršil prijavo (slika 7.16).

id	auth	confirmed	policyagreed	deleted	suspended	mnethostid	username	password
1	manual	1	0	0	0	1	guest	c0e2fec665f745eb0576c734c20e62b2
2	manual	1	0	0	0	1	miha_jernejcic	5685fc94f62f0ccffced4c3b31542a3d
3	fplogin	1	0	0	0	1	miha	not cached

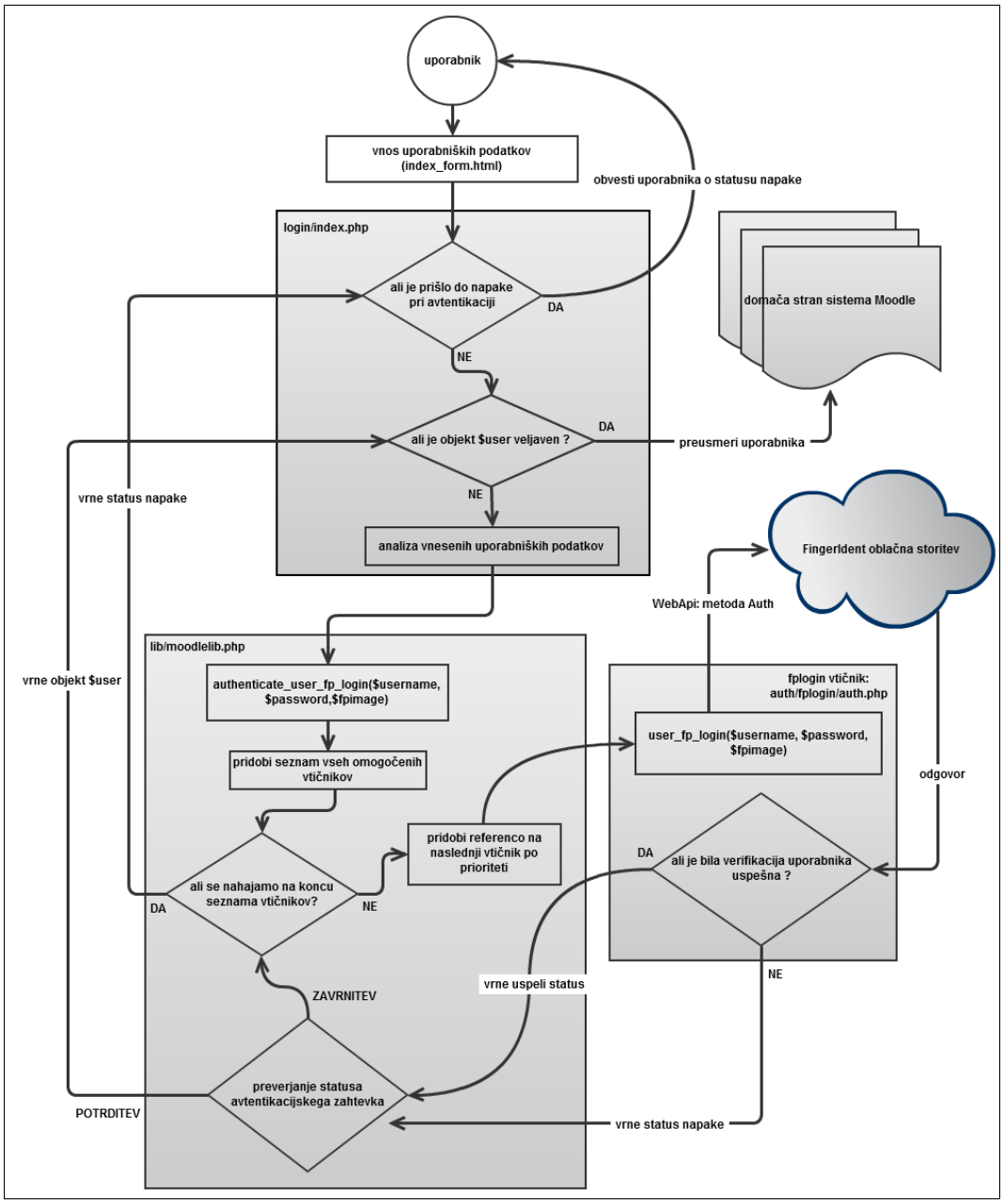
Slika 7.16: Tabela moodle.mdl_user.

Za verificiranje uporabnika, ki je že obstoječ v sistemu Moodle in ima v tabeli *moodle.mdl_user* zapisan odgovorni vtičnik v polju *auth*, se v procesu avtentikacije pridobi referenco na dodeljeni vtičnik in vse operacije se izvajajo na tem vtičniku.

Pri novih uporabnikih, ki so v sistemu Moodle še neznani in zato še nimajo dodeljenega vtičnika, obstajajo pa že na oblačnem avtentikacijskem strežniku, je proces avtentikacije drugačen. Pridobi se seznam vseh vtičnikov, ki so omogočeni v nastavitvah sistema. V nastavitvah sistema se jim lahko določi prioriteto in to vpliva na vrstni red v seznamu. Vtičnik, ki uspe prvi uspešno validirati uporabnika, se zapiše kot odgovorni vtičnik za tega uporabnika v tabelo podatkov in se s tem registrira za vse nadaljnje avtentikacijske zahteve tega uporabnika.

V primeru, ko uporabnik ne obstaja v oblačnem avtentikacijskem strežniku, bo prijava pri poizkusu avtentikacije z avtentikacijskim vtičnikom *fplogin*, neuspešna. Neuspešnemu avtentikacijskemu zahtevku bodo sledili novi avtentikacijski zahtevki na preostalih omogočenih avtentikacijskih vtičnikih, ki so nameščeni v sistemu Moodle. Če bo kateri izmed ostalih avtentikacijskih vtičnikov uspešen, se bo zapisal kot odgovorni vtičnik za uspešno verificiranega uporabnika. Če bodo vsi avtentikacijski zahtevki opravljeni na vseh avtentikacijskih vtičnikih v sistemu Moodle neuspešni, bo uporabnik dobil obvestilo o neuspešni prijavi v sistem.

Na sliki 7.17 je prikazan celoten avtentikacijski proces z uporabniškim imenom, geslom in prstnim odtisom, v iteracijskem koraku z uporabo vtičnika *fplogin*. Če bo verifikacija uporabnika z uporabo vtičnika *fplogin* neuspešna, bo avtentikacijski modul v naslednjem iteracijskem koraku izvršil klic metode *user_fp_login($username,$password,$fpimage)* na vtičniku, ki sledi po prioriteti v seznamu.



Slika 7.17: Avtentikacijski proces z uporabo vtičnika *fplogin*.

7.2.4 Pošiljanje podatkov slike prstnega odtisa po omrežju

Zajeta slika prstnega odtisa je z uporabo bralnika Hamster Plus velika 78 kB. To ne predstavlja hujšega bremena za današnje pasovne širine omrežja, je pa dovolj, da smo začeli razmišljati o optimizaciji količine poslanih podatkov o sliki.

Analiziranje slike lahko poteka:

- na avtentikacijskem strežniku ali
- v brskalniku na uporabniški strani z uporabo spletnega vtičnika.

Pri pristopu **analize slike na avtentikacijskem strežniku** se po omrežju pošilja tekstovna predstavitev slike, ki se jo obdela na strežniku. Prednost tega pristopa je zelo enostavna implementacija. Sistem FingerIdent se uporablja samo na strežniku, kjer ima programer veliko več manevrskega prostora kot pri razvoju spletnega vtičnika v obliki komponente activeX. Velikost poslanih podatkov po omrežju je v tem primeru nekompresiranih 78 kB.

Pri pristopu **analize slike v brskalniku na uporabniški strani z uporabo spletnega vtičnika** v obliki komponente activeX bralnik zajame prstni odtis in ustvari se nova instanca activeX objekta. Pokliče se metoda za pridobitev vektorja značilk iz slike, ta vrednost se shrani in podatek se pošlje na strežnik. Velikost vektorja značilk iz slike v trenutni implementaciji sistema FingerIdent znaša 2158 B. Prednost tega pristopa je v tem, da dosežemo neodvisnost od ločljivosti bralnika prstnih odtisov. Velikost poslanih podatkov po omrežju je dovolj majhna, da ne predstavlja težav nobeni vrsti širokopasovne povezave. Slabost tega pristopa je v tem, da so komponente activeX na voljo samo v brskalniku Internet Explorer. Težavo predstavlja tudi težavni razvoj komponent activeX, ki operirajo v *.Net* in *C++* okolju. V trenutni implementaciji sistema FingerIdent je objekt, ki enkapsulira vektor značilk implementiran tako, da ohranja referenco na originalno sliko (ali več teh, če jih uporabnik doda). Velikost takega objekta je tako večja od pošiljanja samo tekstovne predstavitve ene slike po omrežju in predstavlja najmanjšo skupno velikost 80158 B.

7.3 Implementacija varnostnih mehanizmov

Varnost je zelo pomemben vidik implementacije in povezovanja sistemov. Pri implementaciji oblačne storitve smo imeli opravka z varnostjo na nivoju spletne

aplikacije in varnostjo pri delu z uporabniškimi podatki v oblačni platformi Windows Azure.

Varnost pri delu z uporabniškimi podatki je bila zagotovljena že v zasnovi oblačne platforme Windows Azure. V to smo se prepričali pri izvedbi analize varnosti v poglavju 6.8.1. Dostop do podatkovnega skladišča je zaščiten s primarnim in sekundarnim ključem ter imenom podatkovnega skladišča, ki služi kot uporabniško ime. Prenos podatkov poteka preko https protokola in je pri prenosu kriptiran.

Varnost pri delu s spletno aplikacijo smo morali zagotoviti sami. Uporabniška gesla smo z uporabo kriptirne enosmerne zgoščevalne funkcije md5 [53, 54] spremenili v izvlečke. Za varnost je pri prenosu podatkov po omrežju uporabljen protokol https. Varnost pri uporabi metod spletnega API smo zagotovili z omejevanjem dostopa na avtorizirane aplikacije. Aplikacija se mora pri uporabi spletnega API identificirati z 40 mestnim geslom. Če oblačna storitev geslo prepozna kot veljavno, bo dovolila izvajanje metod, v nasprotnem primeru pa bo vrnila status *UNAUTHORIZED_ACCESS*.

Osnovo protokola https predstavlja protokol http, ki je ovit v varnostni protokol SSL/TLS [55]. Ta mu doda varnostne mehanizme, ki ščitijo prenos podatkov po omrežju pred napadi.

Napadi v omrežjih, pred katerimi smo zaščiteni z uporabo https protokola:

- prisluškovanje v omrežjih,
- napad mož v sredini.

7.3.1 Prisluškovanje v omrežjih

Varnost pred prisluškovanjem v omrežjih je zelo pomembna v brezžičnih omrežjih, kjer lahko nepooblaščeni uporabnik s pomočjo programov, namenjenim prisluškovanjem mrežnih paketov, odkriva občutljive informacije in jih tudi spreminja. V ta namen https protokol omogoča reverzibilno enkripcijo podatkov s pomočjo javnih in privatnih ključev. Vse v https sporočilu je kriptirano, kar vključuje zahtevano skripto strežnika, zaglavje (angl. *header*), parametre zahteve in piškotke. Iz prisluškovanja se da še vedno ugotoviti domeno in vrata strežnika kljub uporabi https protokola.

Primer iz uporabe avtentikacijskega strežnika:

- WebApi klic na oblačni storitvi:
  - *https://fingerident.cloudapp.net:443/WebApi/Auth/?user="username"*
- vidni podatki pri prisluškovanju:
  - *https://fingerident.cloudapp.net:443*
- kriptirani podatki:
  - skripta: */WebApi/Auth/*
  - parameter: *user="username"*

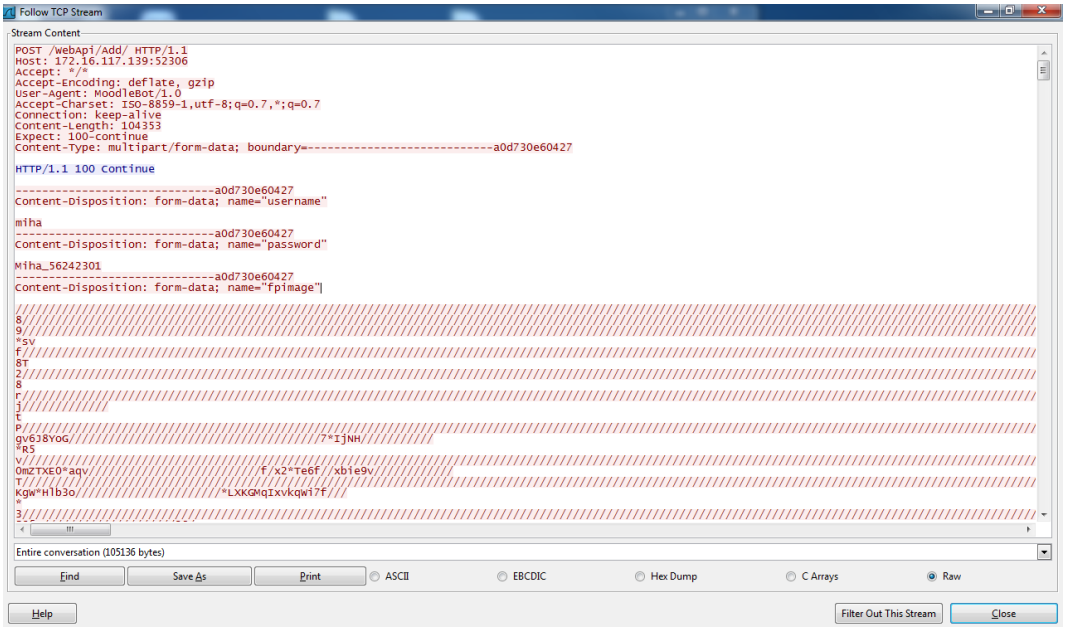
Opravili smo analizo poslanih mrežnih podatkov z uporabo protokolov http in https, s prisluškovanjem v nadziranem lokalnem omrežju z uporabo programa WireShark [56].

Analizirali smo scenarija:

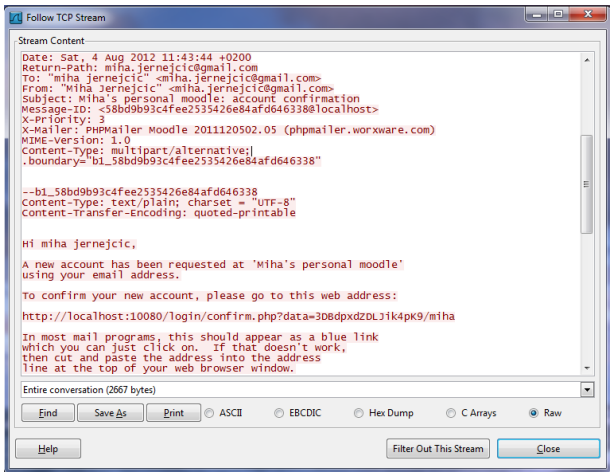
- ustvarjanje novega uporabniškega računa in
- prijava z obstoječim uporabniškim računom.

Ustvarjanje novega uporabniškega računa

Ustvarjanje novega računa poteka v dveh korakih. Ko uporabnik izpolni zaslonsko masko, se podatki posredujejo na avtentikacijski strežnik, ki odgovori s statusom operacije. Ob uspelem statusu se na uporabnikov poštni račun pošlje potrditveno poštno sporočilo. Z uporabo protokola http vidimo, da sta oba koraka izpostavljena prisluškovanju, saj so podatki prosto dostopni vsem, ki se nahajajo v istem omrežju. Koraka sta prikazana na slikah 7.18 in 7.19.



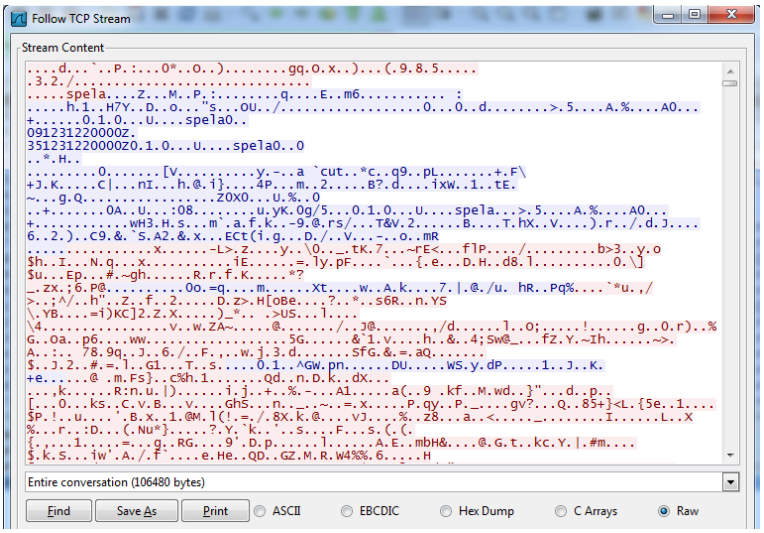
Slika 7.18: Mrežni promet pri ustvarjanju uporabnika s protokolom http.



Slika 7.19: Mrežni promet pri pošiljanju potrditvenega sporočila s protokolom http.

Pri uporabi prokola https je mrežni promet veliko bolj zaščiten in je prikazan na sliki 7.20. Vidno je ime računalnika “spela”, ki v tem primeru predstavlja

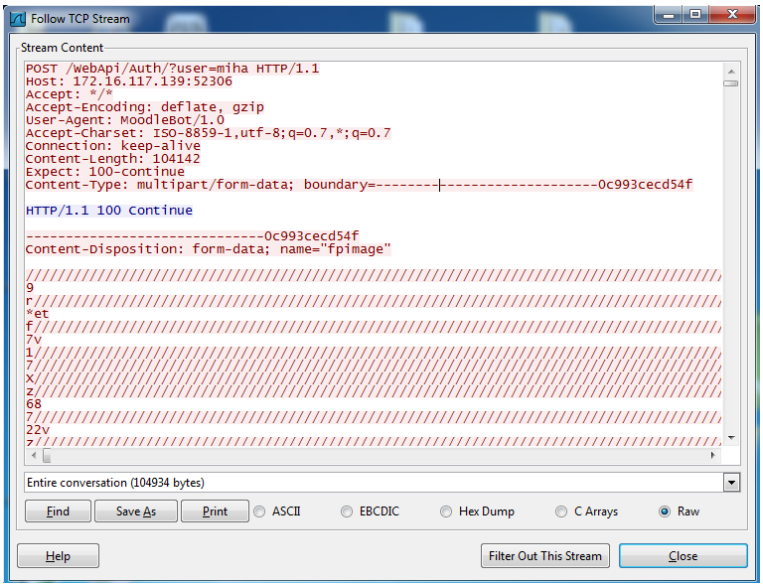
mrežni naslov *https://172.16.117.139:443*.



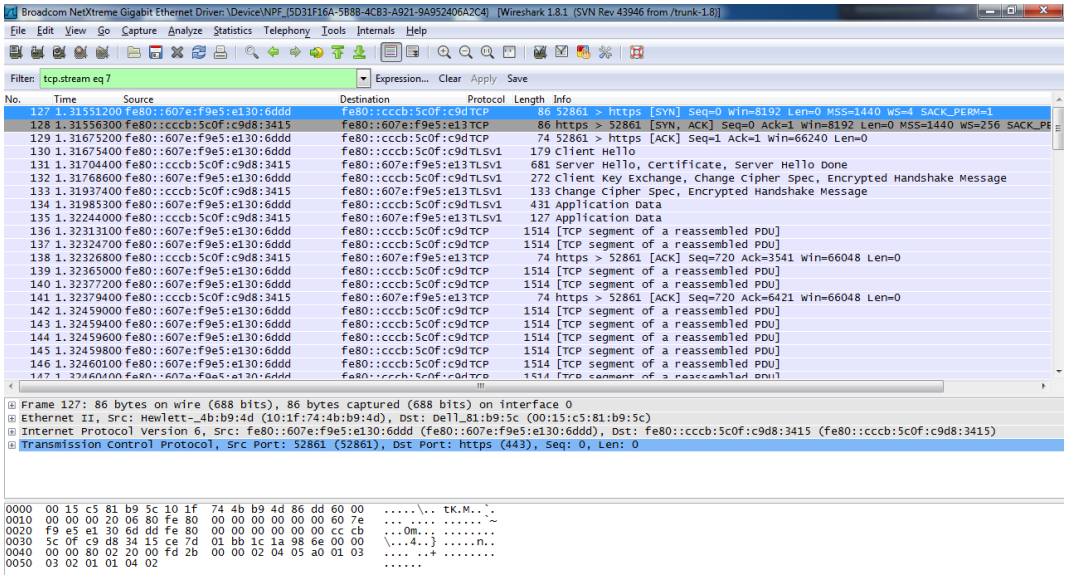
Slika 7.20: Mrežni promet pri ustvarjanju uporabnika s protokolom https.

Prijava uporabnika z obstoječim uporabniškim računom

V procesu prijave uporabnika se pošlje avtentikacijski zahtevek na avtentikacijski strežnik. Pri uporabi http protokola so vstopni podatki vidni in predstavljajo varnostno tveganje (slika 7.21). Z uporabo https protokola so ti podatki zaščiteni. Na sliki 7.22 je razvidno SSL rokovanje (angl. *handshake*) pri avtentikacijskem zahtevku.



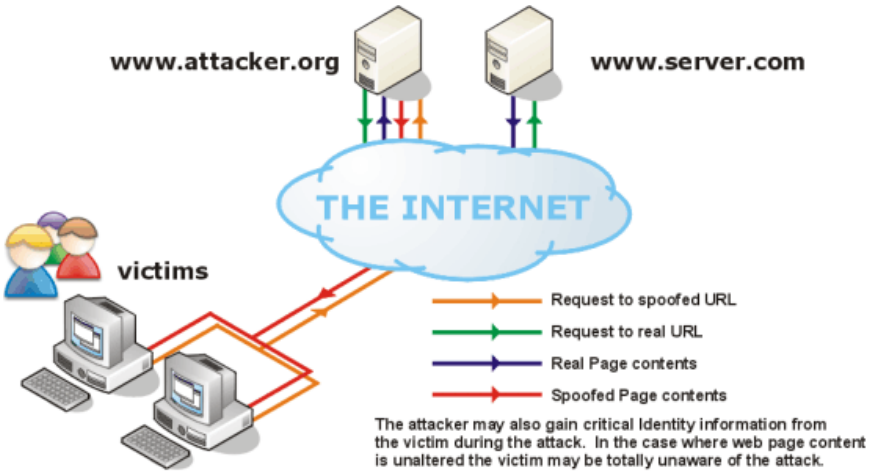
Slika 7.21: Mrežni promet pri prijavi uporabnika s protokolom http.



Slika 7.22: Potek komunikacije pri prijavi uporabnika s protokolom https.

7.3.2 Napad mož v sredini

Napad mož v sredini (angl. *man in the middle*) je primer aktivnega prisluškovanja, ko se napadalec vrine med dva uporabnika in prestreza mrežne pakete, ki si jih pošiljata. Te pakete lahko modificira ali jih nadomesti z lažnimi in tako nadzira celoten proces komunikacije med njima. Uporabnika se ne zavedata, da je komunikacija med njima lažna. Prikaz takega napada je viden na sliki 7.23.



Slika 7.23: Shema napada mož v sredini.

Za zaščito pred napadom mož v sredini se uporablja protokol https, ki je osnovan na kriptografiji z uporabo javnega in privatnega ključa. Javni ključ se uporablja za enkripcijo podatkov, privatni ključ pa se uporablja pri dekripciji podatkov in je prisoten samo na spletnem strežniku. SSL certifikat je v osnovi kombinacija javnega ključa in oznake, ki označuje identiteto spletnega strežnika. Ko se spletni brskalnik poveže na spletni strežnik z uporabo https protokola, bo ta strežnik odgovoril s svojim SSL certifikatom. Spletni brskalnik bo nato preveril veljavnost certifikata:

- **informacija o identiteti spletnega strežnika** se mora ujemati z imenom zahtevanega spletnega strežnika in
- **certifikat mora biti podpisan s strani zaupanja vredne certifikatne agencije** – v našem primeru smo sami prevzeli vlogo certifikatne agencije in podpisali SSL certifikat, kot je opisano v poglavju 7.1.3.

Ko se proces verifikacije veljavnosti SSL certifikata uspešno zaključi, spletni brskalnik iz certifikata pridobi javni ključ, ki ga nato uporablja za kriptiranje podatkov pri nadaljnjem poteku komunikacije s spletnim strežnikom. Če se preverjanje veljavnosti SSL certifikata neuspešno zaključi, se uporabniku v spletnem brskalniku prikaže varnostno opozorilo o dostopu do potencialno nevarnega območja.

Če napadalec pridobi SSL certifikat, ki je v uporabi na spletnem strežniku, to ne predstavlja varnostnega tveganja za ostale uporabnike pri komunikaciji s spletnim strežnikom. Varnost komunikacije s spletnim strežnikom bi bila ogrožena in izpostavljena napadu mož v sredini samo v primeru, če bi napadalec pridobil dostop do privatnega ključa, ki je v uporabi na spletnem strežniku [55].

Pri biometrični oblačni storitvi *FingerIdent* smo se pred to vrsto napada zaščitili z generiranjem privatnega ključa in SSL certifikata, ki vsebuje oznako o identiteti spletnega strežnika in javni ključ za kriptiranje podatkov.

Nazoren prikaz izvedbe napada mož v sredini je opisan v članku [57].

Poglavje 8

Sklepne ugotovitve

V diplomskem delu smo opisali razvoj in postopek integracije obstoječega sistema FingerIdent v oblačno platformo Windows Azure. Primer uporabe te storitve smo prikazali z integracijo bralnika prstnih odtisov in implementacijo sprememb v spletni projekt Moodle ter povezovanjem tega sistema na oblačno storitev.

S sistemom FingerIdent sem bil zelo zadovoljen. Deloval je dobro in izkazal se je za zanesljivega. V zadnji iteraciji implementacije je sistem dozorel ter postal dobra osnova implementirani oblačni storitvi.

V oblačno storitev smo implementirali celovit sistem, ki omogoča delo z uporabniki na vseh nivojih. Spletni API na avtentikacijskem strežniku je zasnovan čimbolj univerzalno, tako da je primeren za nadaljnjo uporabo tudi v drugih projektih. Programerji, ki bodo želeli uporabljati avtentikacijski strežnik kot identitetnega ponudnika, bodo lahko s https zahtevami in dokumentacijo spletnega API implementirali vso potrebno logiko za integracijo FingerIdent oblačne storitve v svoj obstoječi sistem ne glede na uporabljeno platformo. Tako bodo programski klici lahko prišli iz njihove najljubše iOS ali Android aplikacije ali pa iz resnega projekta, napisanega v Javi ali v čem drugem.

Zagotovili smo osnovne koncepte varnosti na nivoju prenosa podatkov po omrežju in shranjevanja gesel v varnejši obliki in s tem pripravili dobro osnovo za nadaljnji razvoj tega projekta.

Oblačna storitev je bila stestirana na lokalnem simulatorju in v produkcijskem okolju. SSL certifikat je bil zaradi ekonomskih razlogov neprilagojen resni pro-

dukcijski rabi. V pravem produkcijskem okolju bi potrebovali zaupanja vreden komercialni certifikat in plačljiv dostop do Windows Azure produkcijskega okolja, kjer bi lahko zagotovili boljšo zanesljivost oblačne storitve.

Prostora za izboljšave je veliko. Na uporabniški strani bi lahko optimizirali pošiljanje podatkov o sliki po omrežju z razvojem spletnega vtičnika in izvlečkom vektorja značilk iz slike. Z implementacijo tega spletnega vtičnika bi tudi rešili problem lokalnega preverjanja kvalitete zajete slike.

Prav tako bi bilo potrebno zagotoviti izboljšano logiko za uporabo večjega števila bralnikov prstnih odtisov in ne samo določene naprave Secugen Hamster Plus, ki sem jo imel na voljo pri implementaciji. Pri pregledu sorodnih rešitev biometrije v oblaku sem ugotovil, da nihče od ponudnikov obstoječih biometričnih rešitev na tržišču ni rešil tega problema. Mislim, da bi bilo to mogoče učinkovito rešiti v primeru, da bi se vsi obstoječi proizvajalci bralnikov prstnih odtisov dogovorili o standardnem razvoju gonilnikov za vse naprave. S tako standardizacijo razvoja gonilnikov bi se lahko implementiralo standardne programske knjižnice za upravljanje z bralniki prstnih odtisov.

Zelo dobrodošla izboljšava bi bila implementacija sinhronizacije uporabniškega prostora z aktivnim imenikom in v teoriji bi to lahko v okviru večjega projekta postal centraliziran avtentikacijski sistem za celotno Univerzo v Ljubljani. Vsi študentje bi se lahko v sistem Moodle prijavljali bodisi z uporabniškim imenom in geslom ali pa kar s prstnim odtisom, če bi to želeli. Tu bi prišla prav izboljšana logika za upravljanje z različnimi vrstami bralnikov prstnih odtisov. Potrebno bi bilo tudi implementirati uporabniški vmesnik za delo z uporabniki na spletu. Ta vmesnik bi lahko predstavljal nadgradnjo obstoječe oblačne storitve z implementacijo potrebnih elementov v mvc3 spletni aplikaciji ali pa bi bil lahko implementiran kot ločen sistem, bodisi v obliki samostojnega programa ali spletne aplikacije, ki bi v ozadju uporabljala oblačno storitev FingerIdent.

Dodatek A

WebApi dokumentacija

```
1  ////////////////////////////////////
2  //responses
3  ////////////////////////////////////
4  private const string AUTH_OK_RESULT = "AUTH_OK";
5  private const string AUTH_FAILED_RESULT = "AUTH_FAILED";
6
7  private const string STATUS_OK_RESULT = "STATUS_OK";
8  private const string STATUS_FAILED_RESULT = "
       STATUS_FAILED";
9
10 private const string FORBIDDEN_ACCESS = "
       UNAUTHORIZED_ACCESS";
11
12 ////////////////////////////////////
13 //actions
14 ////////////////////////////////////
15 /// <summary>
16 /// method: Auth
17 /// Authenticate user with username, password or
       fingerprint image
18 /// </summary>
19 /// <param type=string name="user" required=true/>
20 /// <param type=string name="password" required=false/>
21 /// <param type=string name="fpimage" required=false/>
22 /// <param type=string name="appId" required=true/>
23 /// <returns type=string>AUTH_OK_RESULT or
       AUTH_FAILED_RESULT or UNAUTHORIZED_ACCESS</returns>
```

```
24 //Primer uporabe:
25 //GET: https://fingerident.cloudapp.net:443/WebApi/Auth/?
      user="username"
26 //POST: password data or fingerprint image ( or both ),
      authorized appId
27
28 /// <summary>
29 /// method: UserCheck
30 /// Checks if user exists in the database
31 /// </summary>
32 /// <param type=string name="user" required=true/>
33 /// <param type=string name="appId" required=true/>
34 /// <returns type=string>STATUS_OK_RESULT or
      STATUS_FAILED_RESULT or UNAUTHORIZED_ACCESS</returns>
35 //Primer uporabe:
36 //GET: https://fingerident.cloudapp.net:443/WebApi/
      UserCheck/?user="username"
37 //POST: authorized appId
38
39 /// <summary>
40 /// method: Edit
41 /// Updates user with new password
42 /// </summary>
43 /// <param type=string name="user" required=true/>
44 /// <param type=string name="newpassword" required=true/>
45 /// <param type=string name="appId" required=true/>
46 /// <returns type=string>STATUS_OK_RESULT or
      STATUS_FAILED_RESULT or UNAUTHORIZED_ACCESS</returns>
47 //Primer uporabe:
48 //GET: https://fingerident.cloudapp.net:443/WebApi/Edit/?
      user="username"
49 //POST: newpassword data, authorized appId
50
51 /// <summary>
52 /// method: Add
53 /// Adds new user
54 /// </summary>
55 /// <param type=string name="username" required=true/>
56 /// <param type=string name="password" required=true/>
57 /// <param type=string name="fpimage" required=false/>
58 /// <param type=string name="appId" required=true/>
```

```
59  /// <returns type=string>STATUS_OK_RESULT or
        STATUS_FAILED_RESULT or UNAUTHORIZED_ACCESS</returns>
60  //Primer uporabe:
61  //GET: https://fingerident.cloudapp.net:443/WebApi/Add/
62  //POST: username data, password data, fingerprint data,
        authorized appId
63
64  /// <summary>
65  /// method: Delete
66  /// deletes user if he exists in the database
67  /// </summary>
68  /// <param type=string name="user"/>
69  /// <param type=string name="appId" required=true/>
70  /// <returns type=string>STATUS_OK_RESULT or
        STATUS_FAILED_RESULT or UNAUTHORIZED_ACCESS</returns>
71  //Primer uporabe:
72  //GET: https://fingerident.cloudapp.net:443/WebApi/Delete
        /?user="username"
73  //POST: authorized appId
```

Dodatek B

Seznam vseh sprememb v sistemu Moodle

Seznam vseh sprememb v sistemu Moodle, s katerimi smo omogočili nov tip avtentikacije.

B.1 Spremembe pri razvoju zaslonskih mask

- **lang/en/moodle.php** – dopolnjena datoteka
  - dodani potrebni prevodi v angleškem jeziku za delo z bralnikom prstnih odtisov v zaslonskih maskah
  - dodana dodatna sporočila za bolj informativno obveščanje o napakah
- **login/index.php** – dopolnjena datoteka
  - dodana logika za natančnejše obveščanje o napakah pri vnosu parametrov
  - dodana logika pri avtentikaciji uporabnika za klicanje novih metod avtentikacijskega vtičnika z uporabo parametra prstnega odtisa
- **login/index_form.html** – dopolnjena datoteka
  - dodana javascript logika implementirana za delo z bralnikom prstnih odtisov
  - dodani zaslonski elementi za delo z bralnikom prstnih odtisov

- **login/signup.php** – dopolnjena datoteka
  - dodana logika pri ustvarjanju uporabnika za klicanje novih metod avtentikacijskega vtičnika z uporabo parametra prstnega odtisa
  - vključitev javascript logike (*signup_js.html*) za delo z bralnikom prstnih odtisov
- **login/signup_form.php** – dopolnjena datoteka
  - dodani zaslonski elementi za delo z bralnikom prstnih odtisov
- **login/signup_js.html** – dodana datoteka
  - implementacija javascript logike za delo z bralnikom prstnih odtisov, uporabljena v zaslonski maski *signup_form.html*

B.2 Spremembe pri dopolnitvi potrebnih sistemskih knjižnic

- **lib/authlib.php** – dopolnjena datoteka
  - dopolnitev avtentikacijskega API:
    - dodana metoda *user_fp_login($username, $password, $fpimage)*
    - dodana metoda *user_signup_fp($user, $fpimage, $notify=true)*
- **lib/moodlelib.php** – dopolnjena datoteka
  - dodana metoda *authenticate_user_fp_login($username, $password, $fpimage)*

B.3 Spremembe pri razvoju avtentikacijskega vtičnika

- **auth/fplogin/auth.php** – dodana datoteka
  - implementacija glavnega razreda avtentikacijskega vtičnika *auth_plugin_fplogin)*, ki razširja osnovni razred *auth_plugin_base)*
- **auth/fplogin/config.html** – dodana datoteka

  - nastavitve avtentikacijskega vtičnika

- **auth/fplogin/version.php** – dodana datoteka
  - definicija verzije avtentikacijskega vtičnika
  - definicija minimalne verzije platforme Moodle, s katero je avtentikacijski vtičnik združljiv
  - definicija zrelostne stopnje vtičnika
- **auth/fplogin/lang/en/auth_fplogin.php** – dodana datoteka
  - dodani prevodi v angleškem jeziku, specifični za avtentikacijski vtičnik

Slike

2.1 Računalništvo v oblaku. 7
2.2 Pregled storitvenih modelov oblačnih platform. 9

3.1 Upravljanje z identitetami uporabnikov v zaupanja vrednem okolju. 12
3.2 Upravljanje z identitetami uporabnikov v javnem okolju. 13
3.3 Upravljanje z identitetami uporabnikov v deljenem okolju. . . . 14

4.1 Arhitektura sistema Ceelox ID Online. 16
4.2 Delovanje sistema TruDonor Online. 17
4.3 Biometrične tehnike, storitve MyBioID: a) razpoznavanje obraza na podlagi obraznih značilk, b) razpoznava glasu na podlagi unikatnih spektralnih vzorcev, c) razpoznava šarenice na podlagi unikatnih vzorcev. 17
4.4 Prijavni proces pri oblačni rešitvi MyBioID. 18
4.5 Uporaba oblačne storitve BioID WebServices. 19
4.6 PasswordBank IDaaS. 20

5.1 Biometrični bralnik Secugen Hamster Plus. 23
5.2 Struktura spletne strani sistema Moodle. 24
5.3 Sistem FingerIdent postavljen v testno okolje. 29

6.1 Arhitektura oblačne platforme AppHarbor. 32
6.2 Proces distribucije *.Net* aplikacije na AppHarbor oblak. 33
6.3 Oblačna platforma OpenStack. 34
6.4 Plasti oblačne platforme Tier3. 35
6.5 Platforma Windows Azure. 37
6.6 Prikaz razmerij med ključnimi komponentami v Windows Azure. 37
6.7 Struktura oblačne storitve v Windows Azure. 39
6.8 Cenik za oblačno storitev FingerIdent v Azure. 40

7.1 Pregled celotnega sistema. 44
7.2 Shema avtentikacijskega strežnika. 45
7.3 Vstopna stran avtentikacijskega strežnika. 45
7.4 Preverjanje delovanja WebApi metode UserCheck. 46
7.5 Prikaz delovanja sistema FingerIdent. 47
7.6 FingerIdent oblačna storitev. 48
7.7 FingerIdent podatkovno skladišče. 48
7.8 Azure simulator s simuliranim skladiščem podatkov. 50
7.9 Azure upravljalno okolje. 51
7.10 Zaslonska maska za prijavo. 53
7.11 Zaslonska maska za ustvarjanje novega računa. 55
7.12 Nastavitve avtentikacijskih vtičnikov. 57
7.13 Nastavitve avtentikacijskega vtičnika *fplogin*. 58
7.14 WebApi metoda za ustvarjanje novega uporabnika. 58
7.15 Registracija vtičnika za opravljanje vloge kreiranja uporabnikov. 58
7.16 Tabela moodle.mdl_user. 59
7.17 Avtentikacijski proces z uporabo vtičnika *fplogin*. 60
7.18 Mrežni promet pri ustvarjanju uporabnika s protokolom http. . 64
7.19 Mrežni promet pri pošiljanju potrditvenega sporočila s protokolom http. 64
7.20 Mrežni promet pri ustvarjanju uporabnika s protokolom https. . 65
7.21 Mrežni promet pri prijavi uporabnika s protokolom http. 66
7.22 Potek komunikacije pri prijavi uporabnika s protokolom https. . 66
7.23 Shema napada mož v sredini. 67

Literatura

[1] A. K. Jain, A. Ross, S. Pankanti, "Biometrics: A Tool for Information Security", *IEEE Transactions on Information Forensics and Security*, Vol. 1, No. 2, str. 125–143, junij 2006. Dostopno na: http://biometrics.cse.msu.edu/Publications/GeneralBiometrics/JainRossPankanti_BiometricsInfoSec_TIFS06.pdf (zadnji obisk: 10. 9. 2012)

[2] A. K. Jain, J. Feng, K. Nandakumar, "Fingerprint Matching", *IEEE Computer Society*, Vol. 43, No. 2, str. 36–44, februar 2010. Dostopno na: http://biometrics.cse.msu.edu/Publications/Fingerprint/-JainFpMatching_IEEEComp10.pdf (zadnji obisk: 20. 8. 2012).

[3] Cloud Computing for Biometrics. Dostopno na: http://www.mitre.org/work/areas/research/2011iebriefings/05MSR155-BB.pdf (zadnji obisk: 28. 8. 2012).

[4] M. Tovšak, J. Bule, P. Peer, "Nadgradnja sistema za verifikacijo na podlagi prstnega odtisa", *Zbornik dvajsete mednarodne Elektrotehniške in računalniške konference ERK 2011*, zv. B, str. 135–138, Portorož, Slovenija, september 2011.

[5] W. Jansen, T. Grance, "Guidelines on Security and Privacy in Public Cloud Computing", tehnično poročilo 800-144, National Institute of Standards and Technology (NIST), ZDA, 2011. Dostopno na: http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf (zadnji obisk: 20. 8. 2012).

[6] Informacijski pooblaščenec & Cloud Security Alliance Slovenia Chapter, Slovenski inštitut za revizijo, Slovenski odsek ISACA, Zavod e-Oblak, EuroCloud Slovenia, "Varstvo osebnih podatkov & računalništvo v oblaku", tehnično poročilo, Slovenija, 2012. Dostopno na:

http://www.gzdbk.si/media/pdf/novice/2012/Smernice_varstvo_-osebnih_podatkov_in_racunalnistvo_v_oblaku.pdf (zadnji obisk: 20. 8. 2012).

[7] D. Catteddu, G. Hogben, "Cloud Computing Risk Assessment", tehnično poročilo, European Network and Information Security Agency (ENISA), 2009. Dostopno na: https://www.enisa.europa.eu/activities/risk-management/files/-deliverables/cloud-computing-risk-assessment/at_download/fullReport (zadnji obisk: 20. 8. 2012).

[8] A. Gopalakrishnan, "Cloud Computing Identity Management", *SETLabs Briefings*, Vol. 7, No. 7, str. 45–54, 2009. Dostopno na: http://www.infosys.com/infosys-labs/publications/setlabs-briefings/-Documents/cloud-computing-identity-management.pdf (zadnji obisk: 28. 8. 2012).

[9] Ceelox ID Online. Dostopno na: http://www.ceelox.com/ceeloxidonline.html (zadnji obisk: 29. 8. 2012).

[10] D. Recordon, D. Reed, "OpenID 2.0: a platform for user-centric identity management", *Proceedings of the second ACM workshop on Digital identity management*, str. 11–16, Alexandria, VA, ZDA, oktober 2006. Dostopno na: http://wise.ajou.ac.kr:8080/pdfs/pdffile24836.pdf (zadnji obisk: 29. 8. 2012).

[11] "Active Directory Technical Specification", tehnično poročilo, Microsoft, 2012. Dostopno na: http://download.microsoft.com/download/a/e/6/ae6e4142-aa58-45c6-8dcf-a657e5900cd3/%5BMS-ADTS%5D.pdf (zadnji obisk: 29. 8. 2012).

[12] BIO-Key TruDonor, Biometric Donor ID Service. Dostopno na: http://www.bio-key.com/fingerprintbiometrics/Current/-TruDonorOL.pdf (zadnji obisk: 29. 8. 2012).

[13] MyBioID Product Brochure. Dostopno na: https://www.mybioid.com/fileadmin/user_upload/Documents/Product--Brochure-MyBioID-201111.pdf (zadnji obisk: 30. 8. 2012).

[14] BioID Web Services. Dostopno na: http://www.bioid.com/assets/files/Procuct%20Information/Product--Brief-BWS-BioID-Web-Services-201202.pdf (zadnji obisk: 30. 8. 2012).

[15] BioID Web Services API Documentation. Dostopno na: https://playground.bioid.com/BioIDCloudService/API (zadnji obisk: 30. 8. 2012).

[16] PasswordBank IDaaS. Dostopno na: http://www.passwordbank.com/passwordbank-private-cloud (zadnji obisk: 29. 8. 2012).

[17] T. Vidmar, "Mehanizem odjemalec–strežnik", *Informacijsko–komunikacijski sistem*, str. 116–124, Ljubljana, Slovenija: Pasadena, 2002.

[18] E. Rescorla, "HTTP Over TLS", tehnično poročilo 2818, Internet Engineering Task Force (IETF), ZDA, 2000. Dostopno na: http://tools.ietf.org/html/rfc2818 (zadnji obisk: 20. 8. 2012).

[19] Moodle. Dostopno na: http://moodle.org (zadnji obisk: 20. 8. 2012).

[20] Apache. Dostopno na: http://www.apache.org (zadnji obisk: 20. 8. 2012).

[21] MySql. Dostopno na: http://www.mysql.com (zadnji obisk: 20. 8. 2012).

[22] Moodle – development of authentication plugins. Dostopno na: http://docs.moodle.org/dev/Authentication_plugins (zadnji obisk: 20.8.2012).

[23] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", tehnično poročilo 2616, Internet Engineering Task Force (IETF), ZDA, 1999. Dostopno na: http://www.ietf.org/rfc/rfc2616 (zadnji obisk: 20. 8. 2012)

[24] Developer Hub :: Add-ons for Firefox. Dostopno na: https://addons.mozilla.org/en-us/developers (zadnji obisk: 25. 8. 2012).

[25] Developer's guide – Google Chrome Extension. Dostopno na: http://developer.chrome.com/extensions/devguide.html (zadnji obisk: 25. 8. 2012).

[26] Building ActiveX Controls for Internet Explorer. Dostopno na: http://msdn.microsoft.com/en-us/library/aa751970%28v=vs.85%29.aspx (zadnji obisk: 25. 8. 2012).

[27] U. Klopčič, "Sistem za verifikacijo osebe na podlagi prstnega odtisa", diplomska naloga, Fakulteta za računalništvo in informatiko Univerze v Ljubljani, Slovenija, 2009. Dostopno na: http://eprints.fri.uni-lj.si/936/1/Klopcic_U._UN.pdf (zadnji obisk: 20. 8. 2012).

[28] Sistem za razpoznavo prstih odtisov. Dostopno na: http://www.fri.uni-lj.si/si/raziskave/projekti/12528/project.html (zadnji obisk: 20. 8. 2012).

[29] Amazon EC2. Dostopno na: http://aws.amazon.com/ec2/ (zadnji obisk: 20. 8. 2012).

[30] Xen.org. Dostopno na: http://www.xen.org (zadnji obisk: 20. 8. 2012).

[31] AppHarbor. Dostopno na: https://appharbor.com (zadnji obisk: 20. 8. 2012).

[32] OpenStack Open Source Cloud Computing Software. Dostopno na: http://www.openstack.org (zadnji obisk: 26. 8. 2012).

[33] Uhuru AppCloud. Dostopno na: http://www.uhurucloud.com (zadnji obisk: 20. 8. 2012).

[34] Tier 3 PaaS. Dostopno na: http://www.tier3.com/services/fabric (zadnji obisk: 20. 8. 2012).

[35] Apprenda. Dostopno na: http://apprenda.com (zadnji obisk: 20. 8. 2012).

[36] Iron Foundry. Dostopno na: http://ironfoundry.org (zadnji obisk: 26. 8. 2012).

[37] Cloud Foundry Open Source. Dostopno na: http://cloudfoundry.org (zadnji obisk: 26. 8. 2012).

[38] VMWare vSphere. Dostopno na: http://www.vmware.com/files/pdf/products/vsphere/VMware-vSphere-Standard-DataSheet-DS-EN.pdf (zadnji obisk: 26. 8. 2012).

[39] D. Chappel, "Introducing the Azure Services Platform", tehnično poročilo, Microsoft, ZDA, 2009. Dostopno na: http://www.davidchappell.com/Azure_Services_Platform_v1.1–Chappell.pdf (zadnji obisk: 12. 9. 2012).

[40] C. Kaufman, R. Venkatapathy, "Windows Azure Security Overview", tehnično poročilo, Microsoft, ZDA, 2010. Dostopno na: http://download.microsoft.com/download/6/0/2/6028B1AE-4AEE--46CE-9187-641DA97FC1EE/Windows%20Azure%20Security-%20Overview%20v1.01.pdf (zadnji obisk: 20. 8. 2012).

[41] .Net – MSDN dokumentacija. Dostopno na: http://msdn.microsoft.com/en-us/vstudio/aa496123.aspx (zadnji obisk: 20. 8. 2012).

[42] J. Ansel, P. Marchenko, Ú. Erlingsson, E. Taylor, B. Chen, D. L. Schuff, D. Sehr, C. L. Biffle, B. Yee, "Language-independent sandboxing of just-in-time compilation and self-modifying code", *Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation*, str. 355–366, San Jose, ZDA, junij 2011. Dostopno na: http://static.googleusercontent.com/external_content/untrusted_dlcp/-research.google.com/en//pubs/archive/37204.pdf (zadnji obisk: 12. 9. 2012).

[43] ASP.NET MVC3. Dostopno na: http://www.asp.net/mvc/mvc3 (zadnji obisk: 20. 8. 2012).

[44] Running ASP.NET MVC3 Applications on Azure - .NET Web Development and Tools Blog. Dostopno na: http://blogs.msdn.com/b/webdevtools/archive/2011/08/11/running--asp-net-mvc-3-applications-on-azure.aspx (zadnji obisk: 20. 8. 2012).

[45] OpenSSL: The Open Source tookit for SSL/TLS. Dostopno na: http://www.openssl.org (zadnji obisk: 25. 8. 2012).

[46] Becoming a X.509 Certificate Authority. Dostopno na: http://www.davidpashley.com/articles/cert-authority.html (zadnji obisk: 25. 8. 2012).

[47] Setting up OpenSSL to Create Certificates. Dostopno na: http://www.flatmtn.com/article/setting-openssl-create-certificates (zadnji obisk: 25. 8. 2012).

[48] Working with SSL at Development Time. Dostopno na: http://www.hanselman.com/blog/WorkingWithSSLAtDevelopment-TimeIsEasierWithIISExpress.aspx (zadnji obisk: 25. 8. 2012).

[49] ActiveX – MSDN dokumentacija. Dostopno na: http://msdn.microsoft.com/en-us/library/aa751968%28v=vs.85%29 (zadnji obisk: 20. 8. 2012).

[50] Chrome vtičnik – ActiveX for Chrome. Dostopno na: https://chrome.google.com/webstore/detail/-lgllffgicojgllpmdbemgglaponefajn (zadnji obisk: 20. 8. 2012).

[51] Chrome vtičnik – IE Tab. Dostopno na: https://chrome.google.com/webstore/detail/-hehijbfgiekmjfkfjpbkbammjbdenadd (zadnji obisk: 20. 8. 2012).

[52] cURL – php library. Dostopno na: http://php.net/manual/en/book.curl.php (zadnji obisk: 20. 8. 2012).

[53] R. Rivest(MIT), "The MD5 Message-Digest Algorithm", tehnično poročilo 1321, Internet Engineering Task Force (IETF), ZDA, 1992. Dostopno na: http://tools.ietf.org/html/rfc1321 (zadnji obisk: 12. 9. 2012).

[54] S. Turner(IECA), L. Chen(NIST), "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", tehnično poročilo 6151, Internet Engineering Task Force (IETF), ZDA, 2011. Dostopno na: http://tools.ietf.org/html/rfc6151 (zadnji obisk: 12. 9. 2012).

[55] C. M. Chernick, C. Edington III, M. J. Fanto, R. Rosenthal, "Guidelines for the Selection and Use of Transport Layer Security (TLS) Implementations", tehnično poročilo 800-52, National Institute of Standards and Technology (NIST), ZDA, 2005. Dostopno na: http://csrc.nist.gov/publications/nistpubs/800-52/SP800-52.pdf (zadnji obisk: 10. 9. 2012)

[56] Wireshark – Go deep. Dostopno na: http://www.wireshark.org (zadnji obisk: 20. 8. 2012).

[57] Č. Gorup, B. Kaluža, A. Tavčar, "Eavesdropping on VoIP network", *Zbornik šestnajste mednarodne Elektrotehniške in računalniške konference*

ERK 2007, zv. A, str. 69–72, Portorož, Slovenija, september 2007. Dostopno na:
http://dis.ijs.si/bostjan/papers/erk2007-voip.pdf (zadnji obisk: 20. 8. 2012).