UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA RAČUNARSTVO I INFORMATIKU

Aplikacija za evidentiranje prisustva

ZAVRŠNI RAD

- DRUGI CIKLUS STUDIJA -

Autor:
Malik Koljenović, BA Ing. IT

Mentor:
Vanr. prof. dr Saša Mrdović, dipl.ing.el.

Sarajevo,
februar 2019.

Abstract This thesis addresses the problem of large scale electronic attendance taking in university setting by presenting an Android based attendance taking application, based on immutable and non repudiable location proofs backed by RSA cryptography, utilizing NFC and HCE for ease of use, emulating NFC Forum Tag Type 4 it is also compatible with existing reader infrastructures. It also presents a general overview of the utilized techologies and select implementation details.

Apstrakt Ova teza tretira problem masovnog elektronskog bilježenja prisustva u univerzitetskom okruženju izradom prijedloga aplikacija bazirane na Android platformi korištenjem neizmjenjivih i neporecivih vremensko-lokacijskih dokaza osiguranih korištenjem RSA kriptografije, te NFC i HCE tehnologija u cilju jednostavnosti upotrebe; emulirajući NFC Forum Tag Tip 4 kompatibilna je sa postojećim infrastrukturama čitača. Dat je i opšti pregled korištenih tehnologija i izdvojenih implementacijskih detalja.

MSC Primary 68P25; Secondary 94A60;

Keywords: NFC - near-field communication, HCE - host card emulation, security, Android, attendance, RSA, cryptography, NDEF, NTAG, geolocation, location proofs

Elektrotehnički fakultet, Univerzitet u Sarajevu
Odsjek za računarstvo i informatiku
Vanr. prof. dr Saša Mrdović, dipl.ing.el.
Sarajevo, decembar 2016.

Postavka zadatka završnog rada II ciklusa:
Aplikacija za evidentiranje prisustva

Potrebno je napraviti Android aplikaciju koja omogućava evidentiranje prisustva upotrebom NFC tehnologije. Aplikacija treba da bude jednostavna za upotrebu i zaštićena od zloupotreba.

U okruženju u kom se većina komunikacija odvija elektronski za očekivati je da se i evidentiranje prisustva može raditi na ovaj način. Međutim, postoje otvorena pitanja pogodnosti i sigurnosti elektronskog evidentiranja. NFC može biti osnova za siguran i jednostavan sistem. Kako savremeni mobilni uređaji uglavnom imaju NFC oni mogu biti iskorišteni kao sredstvo prijavljivanja i vođenja evidencije bez potrebe za dodatnim karticama i čitačima.

U radu je potrebno objasniti šta se podrazumjeva pod pojmom evidencija prisustva i koja su otvorena pitanja elektronskog vođenja ove evidencije. Potrebno je objasniti šta je NFC i kako radi. Potrebno je teoretski objasniti kako je moguće napraviti siguran elektronski sistem evidentiranja prisustva zasnovan na NFC koji je lak za upotrebu. U sklopu rada potrebno je napraviti praktičnu izvedbu sistema koji omogućava korisnicima koji imaju mobilne uređaje sa NFC da se pomoću njih registruju i potvrde prisustvo. Ovaj sistem treba biti zaštićen od zloupotreba. Prokomentarisati iskustva stečena tokom praktične realizacije i dati savjete za buduće izvedbe.

Polazna literatura:

[1] V. Coskun, K. Ok, B. Ozdenizci, "Professional NFC Application Development for Android", Wrox, 2013

[2] T. Igoe, D. Coleman, B. Jepson, "Beginning NFC: Near Field Communication with Arduino, Android, and PhoneGap", O'Reilly Media, 2014

[3] J. Annuzzi Jr., L. Darcey, S. Conder, "Introduction to Android Application Development: Android Essentials", 4. izdanje, Addison-Wesley Professional, 2013.

[4] Ross Anderson, "Security Engineering", 2nd edition, Wiley, 2008.

[5] Bill Phillips, Brian Hardy, "Android Programming: The Big Nerd Ranch Guide", Big Nerd Ranch Guides, 2013.

[6] V. Coskun, K. Ok, B. Ozdenizci, "Near Field Communication (NFC): From Theory to Practice", Wiley, 2012

Vanr. prof. dr Saša Mrdović, dipl.ing.el.

Potpisi članova Komisije za ocjenu i odbranu završnog rada II ciklusa

Red. prof. dr Albert Einstein, dipl.el.ing,
predsjednik komisije

Van. prof. dr Alessandro Volta, dipl.el.ing,
mentor i član komisije

Doc. dr Michael Faraday, dipl.el.ing,
član komisije

Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za računarstvo i informatiku

Izjava o autentičnosti radova

Završni rad
II ciklusa studija

Ime i prezime: Malik Koljenović
Naslov rada: Aplikacija za evidentiranje prisustva
Vrsta rada: Završni rad drugog ciklusa studija
Broj stranica: 79

Potvrđujem:

- da sam pročitao dokumente koji se odnose na plagijarizam, kako je to definirano Statutom Univerziteta u Sarajevu, Etičkim kodeksom Univerziteta u Sarajevu i pravilima studiranja koja se odnose na I i II ciklus studija, integrirani studijski program I i II ciklusa i III ciklus studija na Univerzitetu u Sarajevu, kao i uputama o plagijarizmu navedenim na web stranici Univerziteta u Sarajevu;
- da sam svjestan univerzitetskih disciplinskih pravila koja se tiču plagijarizma;
- da je rad koji predajem potpuno moj, samostalni rad, osim u dijelovima gdje je to naznačeno;
- da rad nije predat, u cjelini ili djelimično, za stjecanje zvanja na Univerzitetu u Sarajevu ili nekoj drugoj visokoškolskoj ustanovi;
- da sam jasno naznačio prisustvo citiranog ili parafraziranog materijala i da sam se referirao na sve izvore;
- da sam dosljedno naveo korištene i citirane izvore ili bibliografiju po nekom od preporučenih stilova citiranja, sa navođenjem potpune reference koja obuhvata potpuni bibliografski opis korištenog i citiranog izvora;
- da sam odgovarajuće naznačio svaku pomoć koju sam dobio pored pomoći mentora i akademskih tutora/ica.

Sarajevo, februar 2019.

Potpis:

_____

Malik Koljenović, 984/2015

Sadržaj

Popis slika

Indeks pojmova

ATTN repozitorij potpisanih prisustva spremljen na LAPI.

BUMP približavanje mobilnih uređaja, otvara jednosmjerni komunikacijski kanal u smijeru od slave (S) prema master (M) uređaju.

CERT javni dio korisničkog kriptografskog ključa.

DEVICE korisnički Android uređaj.

HCE *(en. Host card emulation)* softverska arhitektura koja omogućava virtualnu emulaciju elektronskog identiteta.

ISO (*en. International Organization for Standardization*) - Međunarodna organizacija za standardizaciju u Ženevi, Švicarska.

ISO/IEC 14443 Tip A standard fizičkog sloja NFC komunikacijskog protokola.

JSON (*en. Java Simple Object Notation*) - vrlo jednostavna šema serijalizacije objekata u tekstualni oblik.

KEYS jedinstveni set korisničkih RSA ključeva dužine 2048 bita.

LAPI Logit API, Python serverska aplikacija, komponenta LAPP platforme.

LAPP Logit višekomponentna aplikacijska platforma.

M *(en. master)* - Android UI komponenta pokrenuta na uređaju koji bilježi prisustvo.

NDEF vrsta standardizovanog paketa korištena za NFC komunikaciju između uređaja.

NDEFMSG NDEF poruka koja sadrži vremensko-lokacijski dokaz potpisan od strane korisnika.

NFC (*en. near-field communication*) - skup komunikacijskih protokola omogućava uspostavu komunikacijskog kanala između dva uređaja koji se nalaze u neposrednoj blizini.

NFC Forum Tag standardizovani format NFC taga.

S *(en. slave)* - Android komponenta koja se izvršava u pozadini na uređaju čije se prisustvo bilježi.

SPIM *(en. SPacetIME)* - lokacijski dokaz (JSON objekat, struktura podatka).

SSO *(en. Single Sign On)* - politika autentifikacije korištenjem jedinstvenog repozitorija.

UI Android komponente LAPP platforme.

Poglavlje 1

Uvod

Prodor digitalnih računara i komunikacijskih tehnologija u sve sfere ljudskog života i djelovanja, te dramatično povećanje broja korisnika interneta u posljednjoj deceniji nametnulo je mnoštvo novih društvenih i tehničkih izazova. Društveni izazovi najbolje su uočljivi kroz višedecenijsku debatu o privatnosti i vlasništvu nad ličnim podacima, samim time zadiru duboko u diskusiju o ljudskim pravima i identitetu sa jedne i često suprostavljenim komercijalnim interesima sa druge strane. Ukoliko se u tom kontekstu posmatra aktuelna EU uredba o zaštiti podataka[3] (*en. GDPR*) postaje jasno da su digitalna tehnologija i komunikacije postale integralni dio društvene i emocionalno-psihološke realnosti[4], do te mjere da se digitalni tragovi smatraju dijelom nepovredivog identiteta osobe. Iz navedenog je jasno da se radi o institucionalizaciji jedne potpuno nove društveno-tehnološke paradigme unutar pravnih okvira Europske unije.

Sa tehničke strane, dostignuća na poljima kriptografije, teorije mreža i novih komunikacijskih tehnologija, te njihova široka prihvaćenost otvorila su mogućnosti izrade računarskih sistema spremnih da odgovore na novonastale društvene izazove u okviru opisane nove paradigme. Pomenuti računarski sistemi kao dodatno izvršno okruženje imaju društveno-pravnu realnost te se u tim okvirima izvršavaju masovno, dobrovoljno, distribuirano i interaktivno[5] van centralizovanog računarskog izvršnog okruženja u smislu Von Neumannove arhitekture. Opisani sistemi mogu se okarakterisati kao sistemi potpomaganja (*en. assist*), npr. kriptografski računarski sistem u domenu autentifikacije i autorizacija u novoj paradigmi postmatra se kao sistem računarski-potpomognutog povjerenja, ekvivalentno višem nivou apstrakcije.

Registri u kontekstu društvenih institucija su elementarni mehanizam sistema povjerenja, sigurnosne karakteristike takvih institucionalnih registara stoga čine osnov istraživačkog interesa u domenu institucionalne sigurnosi. Napredni elektronski registri izrađeni korištenjem kriptografskih tehnika i savremenih komunikacijskih protokola za prikupljanje i obradu podataka omogućavaju poboljšanje njihovih sigurnosnih osobina, otvarajući nove načine primjene i stvarajući uslove za viši nivo društvenog razvoja i institucionalne efikasnosti, uz to pružaju i adekvatan odgovor na novonastale društvene izazove. Stoga, ukoliko se obezbijede i ispoštuju preduslovi izrade sigurnog sistema[6], evidenciju prisustva u kontekstu naprednog elektronskog registara treba posmatrati i kao vremensko-prostorni dokaz određenog događaja, ovaj rad usmjeren je na izradu jednog takvog sistema računarski-potpomognutog povjerenja u obliku institucionalnog registra elektonske evidencije prisustva.

Poglavlje 2

Postavka problema

Projektni zadatak ovog završnog rada je izrada aplikacije na Android platformi sa pripadajućom udaljenom komponentom, koje u cjelini treba da omoguće evidentiranje prisustva nastavnim aktivnostima na Elektrotehničkom fakultetu u Sarajevu. U skladu sa zadatim funkcionalnim zahtjevima, a iz razloga olakšanog korištenja i praktičnosti upotrebe neophodno je iskoristiti beskontaktne komunikacijske mogućnosti savremenih mobilnih telefona u vidu NFC komunikacijskog protokola.

Također neophodno je osigurati korisnike aplikacije od mogućih zloupotreba korištenjem dostupnih kriptografskih metoda i tehnologija, te stvoriti neophodne uslove za sticanje povjerenja u širi sistem bilježenja prisustva putem neporecivosti i neizmjenjivosti prethodno unesenih podataka. Poželjna mogućnost je jednostavna integracija sa postojećim sistemima, prvenstveno onim autentifikacijskim i autorizacijskim, te planiranje arhitekture za buduća proširenja u vidu omogućavanja integracije sa infrastrukturnim hardverskim čitačima i TAG karticama.

Potrebno je dokumentovati proces izrade i opisati korištene tehnologije, sa posebnim osvrtom na korištene kriptografske metode i tehnologije, te identifikovati otvorena pitanja na polju elektronskih registara prisustva, mogućnosti i izazove koje oni predstavljaju uz rješenja koja navedena aplikacija nudi u datom kontekstu.

Poglavlje 3

Kriptografske osnove rješenja

Enkripcija, tj. skriveno pisanje, izvorni je cilj kriptografije[7], mogućnost tajne komunikacije intrigirala je čovječanstvo tokom poznate historije, te je razvijeno mnoštvo načina da se taj cilj i ostvari, od primitivnih supstitucijskih metoda, pa sve do moderne formalno utemeljene kriptografije[8]. Kako namjena rješenja u okviru ovog rada diktira, primarni fokus ovog poglavlja biti će međutim stavljen na dodatne mogućnosti moderne kriptografije koje su posebno došle do izražaja razvojem kriptografije javnog ključa i to autentifikaciju entiteta, te utvrđivanje autentičnosti i integriteta poruke.

3.1 Hash funkcije

Kriptografske hash funkcije igraju ključnu ulogu u savremenoj kriptografiji, njihova osnovna namjena je preslikavanje domena X u uži kodomen Y. Najčešće se koriste u sklopu utvrđivanja integriteta podataka i autentifikacije poruka. Kao ulaz hash funkcije primaju niz bita a kao izlaz vraćaju rezultirajuće preslikavanje koje nazivamo hash-kod, hash-vrijednost ili jednostavno **hash**[9], koji je i sam niz bita ali u praksi se gotovo uvijek ispisuje u heksadecimalnom zapisu, npr. `B0E2B76996D3A488CEFDA9C4B83ECE1E3E49121A`.

Preciznije hash funkcija h preslikava niz konačne dužine n bita, za domen D i kodomen R tako da $h : D \rightarrow R$ i $|D| > |R|$, ovakvo preslikavanje je *many-to-one* i implicira postojanje *kolizija*, tj. različitih parova ulaza sa identičnim izlazom, ovo je nepoželjna ali i neizbježna osobina hash funkcija te se u praktičnim implementacija pokušava minimizirati njen efekat. Osnovna ideja je da hash vrijednost služi kao kraća reprezentativna slika ili digitalni otisak izvornog objekta i koristi se kao da je jedinstveno identifikabilna sa izvornim objektom.

U okviru ovog rada posebno je zanimljiv slučaj korištenja hash funkcija zajedno sa šemama digitalnog potpisivanja u cilju utvrđivanje integriteta podatkovne strukture, gdje se poruka obično prvo hashuje i rezultirajuća hash vrijednost kao reprezent originalne poruke potpisuje korisničkim privatnim ključem. Prema opisanoj definiciji hash funkcija može uzeti mnoštvo oblika, stoga je za bilo kakvu ozbiljniju diskusiju o hash funkcijam potrebno napraviti klasifikaciju njenih različitih oblika koje u zavisnosti od domena primjene uzimaju različite osobine i nameću dodatne zahtjeve i ograničenja, time dajući jasniju sliku o hash funkcijama uopšte. Dvije osnovne osobine svake hash funkcije su:

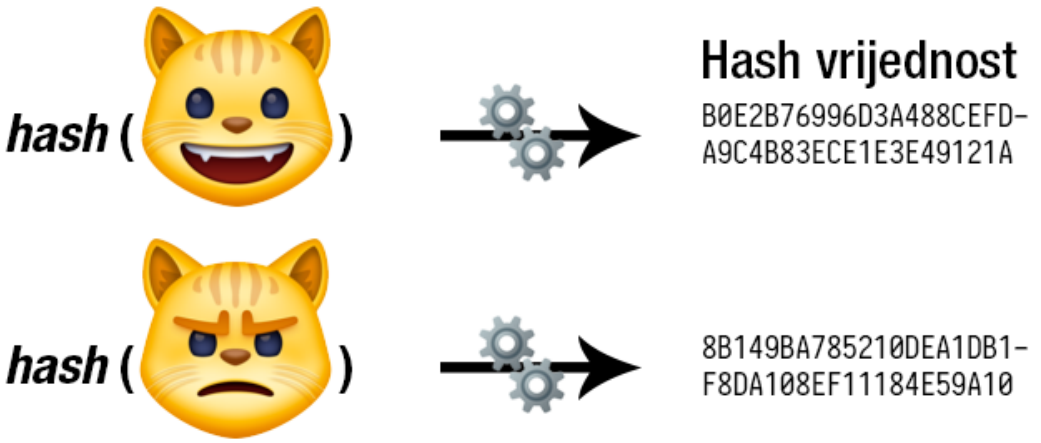
- *kompresija* - funkcija h preslikava ulaz x proizvoljne konačne dužine t niza bita u izlaz $h(x)$ fiksne dužine n bita

- *jednostavnost izračuna* - za datu funkciju h, i ulaz x, $h(x)$ je jednostavna i brza za izračun.

Sve hash funkcije zadovoljavaju dvije pobrojane osobine, ali to svojstvo nije dovoljno za njihovu kriptografsku primjenu za koju je neophodno osigurati dodatne garancije, stoga se nameće podjela na nekriptografske i kriptografske hash funkcije, koje dodatno moraju zadovoljiti i najmanje jedan od dva niženavedena uvjeta:

- *jednosmjernost* - osigurava da je izračunski teško za datu izlaznu hash vrijednost y pronaći izvornu vrijednost parametra x hash funkcije $h(x)$,
- *otpornost na kolizije* - pronalazak dvije iste izlazle hash vrijednosti za dva različita ulazna parametra x izračunski je teško i nepraktično, što obično znači da bi za pronalazak kolizije uzevši najbrže trenutno zamislive računare trebalo više vremena nego je proteklo od postanka univerzuma do danas, što se može smatrati razumnom garancijom, koju je kako se je do sada pokazalo praktično teško ispoštovati.

Uzevši u obzir sve opisane karakteristike jasno je da broj praktičnih implementacija kriptografskih hash funkcija koje pružaju neophodne garancije relativno mali, no postoje funkcije koje pružaju dovoljno dobre garancije za svakodnevnu praktičnu primjenu, od kojih su najpoznatije SHA, MD i BLAKE familije hash funkcija, u okviru ovog rade korištena je SHA-256 hash funkcija.



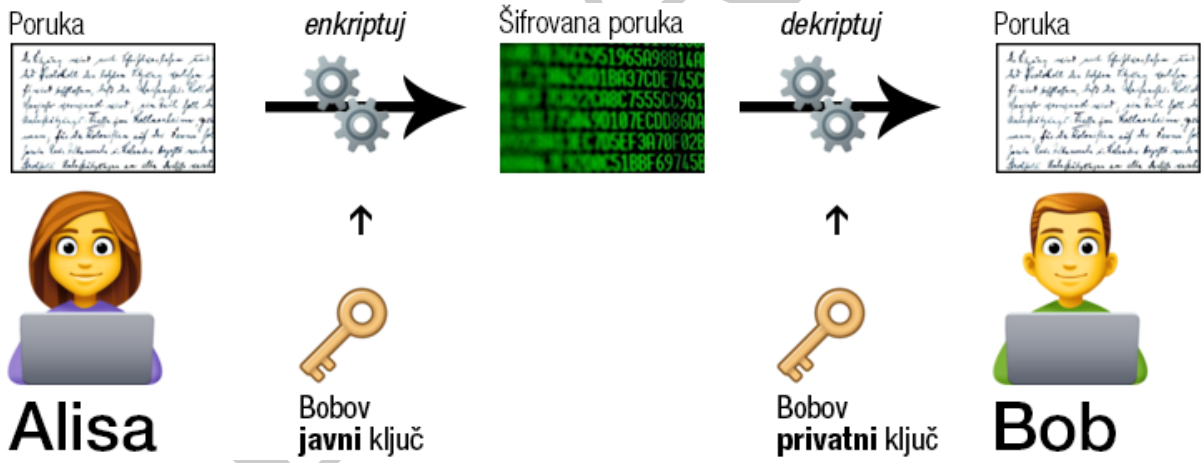
Slika 3.1: Primjena hash funkcije na dva različita ulazna parametra daje **različitu i jedinstvenu** izlaznu hash vrijednost za dati ulazni parametar

Ilustrativno na slici 3.1 dat je prikaz rada SHA-1 hash funkcije za vrijednosti "HAPPYCAT" i "ANGRYCAT", gje je jednostavno uočljiva različita izlazna hash vrijednost, dodatno navedene vrijednosti bi trebale biti jedinstvene za date ulaze, tj. ne bi smjela postojati neka druga vrijednost osim navedene koja bi dala istu hash vrijednost kao rezultat. SHA-1 funkcija uvijek daje rezultat fiksne dužine 40 bajta. Sa sigurnosnog aspekta relevantno je spomenuti još i to da postoje tabele koje sadrže prethodno izračunate hash vrijednosti za mnoge ulazne parametre i različite funkcije, kao i pretraživače po hash vrijednostima što djelimično narušava garanciju jednosmjernosti, takve tabele nazivaju se *rainbow* tabelama.

3.2 Kriptografija javnog ključa

Simetrična kriptografija zahtjeva razmjenu tajnog ključa prije ostvarivanja sigurnog komunikacijskog kanala između dva ili više učesnika, iako vrlo pouzdan metod, često je teško provodiv u praksi, pogotovo u slučajevima ad-hoc i masovne decentralizovane komunikacije nalik internetu, neophodno je da učesnici ili sami izvrše međusobnu razmjenu tajnih ključeva koje će naknadno koristiti ili da jedan autoritativni entitet u kojeg se ima povjerenje to uradi umjesto njih tako što će generisati i distribuirati ključeve svim učesnicima u komunikaciji, očite slabosti ovog modela su da zahtjeva povjerenje i postojanje funkcionalnog sigurnog kanala za razmjenu tajnih ključeva - što često nije slučaj. Asimetrična kriptografija ili kriptografija javnog ključa razvijena je sa ciljem prevazilaženja navedenih nedostataka od strane britanske službe GCHQ početkom sedamdesetih godina XX stoljeća, javnosti je naknadno predstavljena kroz radove Merklea[10], Diffiea i Hellmana[11], a nešto kasnije u obliku danas opštepoznate praktične implementacije RSA kriptosistema Rivesta, Shamira i Adelmana[12], koji dodatno uvodi i pojam elektronskog potipisa, kao i njegove komercijalne namjene.

Kriptosistemi javnog ključa uvode ideju dva različita ali povezana ključa, jedan - javni samo za enkripciju i jedan - privatni samo za dekripciju, a kako nije moguće saznati jedan ključ iz drugog korisnik je slobodan javno objaviti svoj ključ za enkripciju, tako da ukoliko npr. Alisa želi poslati tajnu poruku Bobu, dovoljno je da posjeduje Bobov sada *javni enkripcijski ključ* i da ga iskoristi da njime šifruje poruku. Kada Bob primi takvu poruku iskoristiti će svoj *tajni privatni ključ* i dešifrovati Alisinu poruku, kompletna interakcija i sastavni elementi prikazani su na slici 3.2.



Slika 3.2: Tajna komunikacija unutar javnog kriptosistema

Kriptografija javnog ključa također se naziva i *asimetričnom kriptografijom*, ona ne osigurava samo tajnost komunikacije kao u opisanom slučaju, nego se može koristiti i za potvrdu autentičnosti. Za takav scenario dovoljno je da Bob svojim privatnim ključem *potpiše* željenu poruku ili datotetu i Alisa će biti u mogućnosti da provjeri da je ta poruka autentično kreirana od strane Boba, za tu namjenu potrebno je da Alisa posjeduje Bobov javni ključ od ranije ili da ga dobavi iz nekog povjerljivog izvora jer mora biti sigurna da neko lažno ne podmetne svoj ključ kao Bobov, više o ovoj namjeni asimetričnih kriptosistema biti će dato u zasebnom poglavlju u nastavku, no za sada je bitno imati je na umu. Spomenuti proces potpisivanja sastoji se od par kriptografskih operacija, vrlo sličnih kriptovanju poruke, no kako u ovom slučaju nije potrebno prenijeti kompletan sadržaj nego samo omogućiti verifikaciju, proces je moguće učiniti

efikasnijim tako što će se poruka prije obrade privatnim ključem prethodno provući kroz neku od hash funkcija, koja će ga znatno smanjiti i ubrzati sam proces.

3.2.1 Sigurnosni ciljevi

Sigurnosni ciljevi kriptografije javnog ključa pored povjerljivosti, kao primarne funkcije, kako je već i pomenuto, uključuju i mogućnosti za provjeru integriteta i autentičnosti poruke, autentifikacija entiteta, te osiguravanje neporecivosti izvršene akcije[13], ovakav jedinstven i širok skup funkcionalnosti kriptografiji javnog ključa daje fundamentalni značaj u modernim digitalnim komunikacijama i internet poslovanju, stoga su osnovne karakteristike svakog od navedenih ciljeva ukratko opisane u nastavku.

Povjerljivost i privatnost

Povjerljivost je osnovni sigurnosni cilj kriptografije i odnosi se na osobinu da tajni podaci neće biti dostupni neautorizovanim osobama. Povjerljivost omogućava privatnost i međusobno su usko povezane. Privatnost označava sposobnost očuvanja vlastitih podataka tajnim za sve one koji nisu eksplicitno autorizovani za njihov pregled i predstavlja osnovno ljudsko pravo garantovano članom 12 Univerzalne deklaracije o ljudskim pravima[14]:

> *"Niko ne smije biti podvrgnut samovoljnom miješanju u njegovu privatnost, obitelj, dom ili dopisivanje, niti napadima na njegovu čast i ugled. Svako ima pravo na zakonsku zaštitu protiv takvog miješanja ili napada."*

Mnoštvo je primjera važnosti povjerljivosti i privatnosti u savremenoj eri opšteprisutne digitalizacije i internet poslovanja, nažalost narušavanje ovih temeljnih vrijednosti postalo je gotovo normalna i svakodnevna pojava bilo da se radi o krađi osjetljivih podataka i digitalnih dobara od strane malicioznih agenata ili prisluškivanju od strane korporativnih i vladinih agencija koje vrše masovno špijuniranje na globalnom nivout prikupljanjem privatnih podataka putem raznorodnih programa[15].

Autentifikacija entiteta

Autentifikacija se odnosi na proces utvrđivanja stvarnog identiteta, navedeni identitet može se odnositi na osobu, kompaniju ili bilo koji drugi koncept koji se od drugog razlikuje po sebi svojstvenim osobinama. Za isti proces može se približno precizno koristiti i termin identifikacija. Poznavanje identiteta u okviru digitalnog okruženje često je neophodno za ispravno funkcionisanje programskog rješenja i pravilnu raspodjelu autorizacija, kao i za relacije povjerenja između različitih kategorija korisnika. Često se za ovu namjenu koriste korisnička imena i lozinke kao najprostiji vid implementacije, no mnogo su pouzdaniji namjenski izrađeni repozitoriji identiteta u vidu specifičnih rješenja ili infrastrukture javnih ključeva (*PKI - public key infrastructure*), koji omogućavaju mnogo sigurniju autentifikaciju, bolje provođenje dobrih sigurnosnih praksi, širi spektar primjene i bolju integraciju. Aplikacija izrađena u okviru ovog rada sadrži namjenski izrađen pokazni repozitorij identiteta i javnih ključeva u vidu Logit API implementacije.

Integritet i autentičnost poruke

Integritet se odnosi na garanciju da podaci nisu mijenjani nakon što ih je izvorni autor sačinio, mnoštvo je bitnih aktivnosti gdje je upravo ovakva garancija od iznimne važnosti. Koncept in-

tegriteta dodatno se proširuje kroz pojam autentičnosti poruke, gdje se zahtjeva i mogućnost utvrđivanja njenog izvorišta, te njegova autentifikacija. Ove aktivnosti izvode se putem digitalnog potpisivanja i vjerodostojnih repozitorija koji sadrže provjerene identitete entiteta koji se autoriziraju, najčešće u vidu već opisanih PKI.

Kao primjer možemo uzeti svakodnevno poznate korisničke scenarije, svaki računarski program ili nadogradnja kada se distribuira krajnjim korisnicima može biti naknadno izmjenjen u cilju izvršavanja određenih zlonamjernih aktivnosti koje mogu naštetiti korisniku, da bi se ovo izbjeglo većina savremenih operativnih sistema podržava određeni način provjere integriteta softvera prilikom instalacije na korisničkom računaru i autentičnost identiteta njenog izvorišta, ovakve provjere posebno su bitne u sigurnosno osjetljivim okruženjima i djalatnostima gdje bi pokretanje zlonamjernog koda moglo ugroziti živote ili uzrokovati veliku materijalnu štetu, primjeri takvih sistema su računari u zdravstvu i kontrolni sistemi javnih infrastruktura, poput aerodroma, električne ili vodovodne mreže, no nikako ne treba zanemariti i kućne korisnike koji su itekako izloženi raznim vrstama sigurnosnih prijetnji. Za namjene utvrđivanja integriteta i autentifikacije izvorišta programskih rješenja održavaju se repozitoriji sa identitetima softverskih razvojnih kuća i njihovim ključevima.

Dodatno utvrđivanje integriteta poruke i autentifikacija učesnika se koristi u okviru aplikacije za bilježenje prisustva studenata predložene u okviru ovog rada na način da se potpisano vrijeme i lokacija svih studentskih uređaja potpisuje predavačkim ključevima unutar jedinstvene sesije (npr. nastavnog časa) koju po zaključenju nije moguće naknadno mijenjati bez narušavanja integriteta navedene sesije. Identitet svih učesnika i autentičnost njihovih potpisa provjerava se korištenjem namjenskog repozitorija na Logit API.

Neporecivost

Neporecivost je osobina podataka koja sprječava poduzimača određene aktivnosti da istu porekne nakon njenog okončanja, npr. slanje poruke, novčana transakcija ili prisustvo predavanju. Kod primjera prisustva, radi se o nemogućnosti studenta da nakon što digitalno prijavi svoje prisustvo na predavanju unutar predložene Logit NFC aplikacije naknadno to prisustvo porekne jer postoji jedinstveni digitalno potpisan trag koji to dokazuje, za čvrst dokaz neophodno je osigurati i jaku povezanost korisnika i nosioca identiteta, u ovom slučaju mobilnog uređaja, za ovakve namjene posebno su pogodne višefaktore metode identifikacije koje uključuju i biometrijske osobine.

3.3 RSA kriptosistem

Rivest, Shamir i Adelman[12] predložili su kriptosistem koji može osigurati osobine privatnosti i potpisivanja poruka ekvivalentne ili bolje od onih kakve posjeduje papirna pošta. Njihov sistem baziran je na konceptu sistema javnog ključa kakav su ranije predložili Diffie i Hellman[11]. Sveukupna procedura sastoji se od procesa enkripcije E, procesa dekripcije D i poruke M, te za takav sistem vrijedi:

1. dešifrovanje kriptovane forme poruke M daje M,

$$D(E(M)) = M$$

2. i E i D su jednostavne za izračun,
3. javno obznanjujući E korisnik ne otkriva jednostavan način za izračun D. Praktično ovo znači da samo on može dekriptovati poruke kriptovane pomoću E
4. ukoliko je poruka M prvo dešifrovana a onda šifrovana, rezultat je M,

$$E(D(M)) = M$$

Enkripcijska (ili dekripcijska) procedura se sastoji od *opšte metode* i *enkripcijskog ključa*. Opšta metoda, pod kontrolom ključa, šifruje poruku M i rezultira šifrovanom porukom C *(en. ciphertext)*. Svako može koristiti istu opštu metodu, sigurnost date procedure počiva na sigurnosti ključa. Otkrivanje enkripcijskog algoritma tada znači i otkrivanje (*javnog*) ključa.

Kada korisnik otkrije E, on otkriva vrlo neefikasan način izračuna *D(C)* testiranjem svih mogućih poruka M sve dok se ne nađe ona koja zadovoljava $E(M) = C$. Ukoliko je osobina (3) zadovoljena broj takvih poruka nije praktičan za izračun.

Funkcija E ukoliko zadovoljava osobine (1)-(3) predstavlja tzv. *"jednosmjernu funkciju sa stupicom"*, a ukoliko zadovoljava i (4) onda je *"jednosmjerna permutacija sa stupicom"*. Diffie i Hellman[11] uveli su koncepte jednosmjernih funkcija sa stupicom ali nisu dali implementaciju. Ove funkcije nazivaju se jednosmjernim jer ih je jednostavno izračunati u jednom smijeru ali bi trebalo biti vrlo teško u suprotnom, dok su "sa stupicom" jer su njihovi inverzi jednostavni za izračun ukoliko je poznata određena informacija, u slučaju šifrovanja je to privatni ključ. Ovakva funkcija koja također zadovoljava i (4) mora biti permutacija: svaka poruka je ciphertekst neke druge poruke i svaki ciphertekst je dozvoljena poruka. Zadovoljenje osobine (4) neophodno je za implementaciju potpisivanja.

3.3.1 Ključevi i komunikacija

Da bi kreirali vlastite privatne i javne ključeve učesnici moraju svaki za sebe i nasumično odabrati dva velika prosta broja p i q takva da nije vjerovatno da računar može izvršiti cjelobrojnu faktorizaciju $n = p * q$, danas se minimalno preporučuju brojevi slične veličine koji daju proizvod reda 2048 bita kao sigurni do 2030. godine[16]. Proizvod n postaje dijelom javnog ključa, dok se pojedinačni faktori moraju čuvati u tajnosti zbog njihovog korištenja u derivaciji privatnog ključa.

Korisnici također moraju izabrati i cjelobrojnu vrijednost e takvu da,

$$1 < e < \phi(n) = (p-1)(q-1) \wedge \gcd(e,(p-1)(q-1) = 1.$$

Obratite pažnju da je e uvijek neparno jer je $(p-1)(q-1)$ parno. Dalje korisnik računa cjelobrojnu vrijednost d,

$$1 < d < (p-1)(q-1) \wedge de \equiv 1 \bmod (p-1)(q-1).$$

Korisnikov javni ključ sada je par *(n, e)*, dok je njegov privatni ključ d. Broj n naziva se *RSA modulus*, e je *enkripcijski eksponent*, a d - *dekripcijski eksponent*[13].

Ponovno se može poslužiti primjerom Boba i Alise opisanim ranije, sada se može reći da kada je Bob korištenjem opisane procedure generisao neophodne vrijednosti ključeva i poslao Alisi svoj javni ključ, tj. vrijednosti *(n, e)*, koje će ona iskoristiti za šifrovanje proizvoljne poruke *m* tada možemo izraziti kao

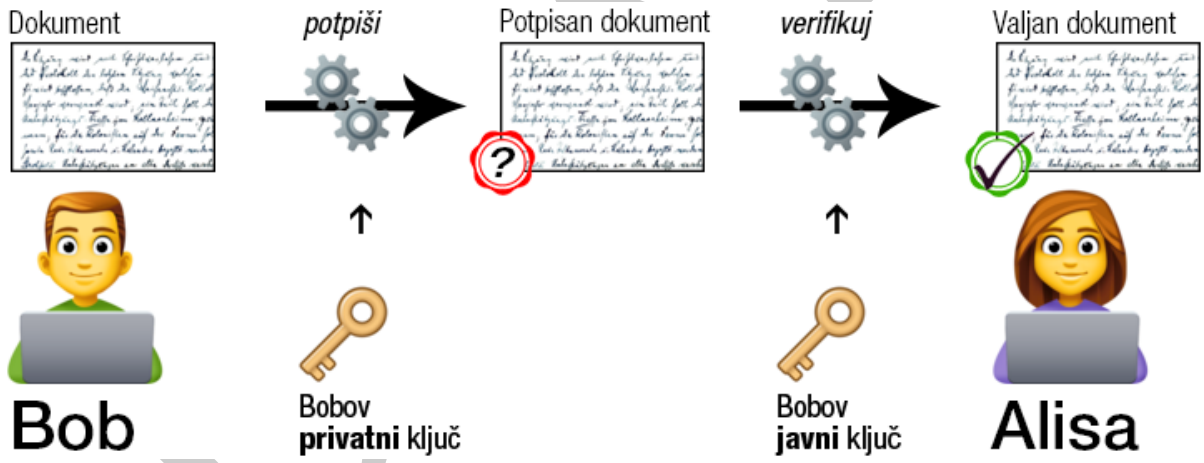
$$c = mopisaneprocedurее \bmod n.$$

Kada Bob primi Alisinu šifrovanu poruku *c* iskoristiti će tajnu vrijednost *d* i dešifrovati poruku

$$m = c^d \bmod n.$$

3.3.2 Digitalni potpis

Digitalni potpis osigurava mehanizam provjere integriteta i u kombinaciji sa adekvantnom infrastrukturom - autentičnost izvorišta poruke. Ukoliko Bob želi potpisati određeni dokument, on korištenjem svog privatnog ključa izračunava jedinstveni niz bita, koji Alisi garantuje da će korištenjem Bobovog prethodno dobavljenog javnog ključa moći verifikovati da navedeni dokument nije izmjenjen u putu do nje, kao i da je Bob originalni potpisnik dokumenta, navedena interakcija prikazana je na slici 3.3, dodatno Bob u budućnosti ne može poreći da je potpisao navedeni dokument, što osigurava još jednu vrlo bitnu funkcionalnost cjelokupnog sigurnosnog sistema. Ovakav sistem predstavlja temelj na kojem je izgrađen siguran internet i digitalna ekonomija te je duboko utkan u osnovne protokole poput TLS, SSH, PGP etc.



Slika 3.3: Verifikacija integriteta i autentičnosti uz garancija neporecivosti u okviru javnog kriptosistema

RSA šema najčešće je korišten algoritam digitalnog potpisivanja. Generacija ključeva funkcionira na istom principu kao i u primjeru šifrovanja poruke opisanom u poglavlj 3.3.1. Svaki digitalni potpis zavisan je od poruke kao i od potpisnika, u protivnom bilo bi moguće da isti potpisnik koristi jedan potpis za više dokumenata, što ovdje nije slučaj. Da bi se uspješno implementiralo digitalno potpisivanje, kriptosistem mora zadovoljavati osobine spomenute jednosmjerne permutacije sa stupicom, budući da će dekripcijski algoritam biti primjenjivan na izvorni - nešifrovan dokument.

Opisani primjer razmjene potpisanog dokumenta ili poruke *M* između Alise i Boba može se unutar RSA kriptosistema preciznije prikazati kao:

$$S = D_B(M),$$

gdje je S digitalno potpisana poruka a D_B Bobova funkcija za dešifrovanje korištenjem privatnog ključa. Kako je već napomenuto primjena funkcije za dešifrovanje na nešifrovanu poruku ima smisla u slučaju da kriptosistem zadovoljava potrebne uvjete. Ukoliko je dodatno potrebno osigurati i tajnost potpisane poruke Bob je može naknadno šifrirati Alisivnim javnim ključem, u tom slučaju Alisa po prijemu prvo vrši dešifrovanje svojim privatnim ključem da bi dobila Bobovu izvornu potpisanu poruku S, korištenjem Bobovog javnog ključa Alisa sada "šifruje" potpisanu poruku

$$M = E_B(S)$$

time dolazeći u posjed uređenog para *(M, S)* sa osobinama sličnim onima koje ima originalni fizički potpisan dokument.

Sama praktična implementacija potpisivanja razlikuje se međutim od opisane procedure u tome da se u praksi ne vrši potpisivanje cjelokupne izvorn poruke, nego se nad njom prvobitno korištenjem hash funkcije otporne na kolizije izračunava jedinstvena hash vrijednost, pa Bob tako umjesto poruke potpisuje dobijenu hash vrijednost svojim privatnim ključem, dok na drugom kraju Alisa ponavlja proces izračunavanja hash vrijednosti koristeći Bobovu izvornu poruku, njegov javni ključ i istu hash funkciju, te dobijeni hash poredi sa Bobovim hashom, u slučaju poklapanja vrijednosti Alisa može biti sigurna u valjanost digitalnog potpisa. Jedina funkcionalna razlika ovakvog pristupa je da se koncept poruke ili dokumenta, odvaja od samog digitalnog potpisa, što olakšava rad i osigurava dodatne sigurnosne i upotrebne prednosti.

Praktična implementacija može se formalno opisati kao primjena hash funkcije h tako da potpis s niza karaktera poruke $m \in \{0,1\}^n$ glasi

$$s = h(m)^d \bmod n.$$

d je u prikazanom slučaju Bobov *dekripcijski eksponent*. Da bi Alisa verifikovala zaprimljeni potpis s koristi Bobov javni ključ *(n, e)* i izračunavanjem hash vrijednost *h(m)* poruke m i vrši provjeru

$$h(m) = s^e \bmod n.$$

Potpis je valjan ako i samo ako važi data jednakost. Za potrebe Logit sistema korištena je SHA-256 hash funkcija u kombinaciji sa RSA potpisom.

3.4 PKI - infrastruktura javnog ključa

Iako kriptografija javnog ključa ne zahtjeva razmjenu tajnih ključeva da bi se ostvarila povjerljiva komunikacij, vrlo je bitan aspek upravljanja ključevima, kako javnim tako i privatnim. Namjena infrastrukture javnog ključa *en. PKI - private key infrastructure* je upravo to, efikasno i sigurno upravljanje javnim i privatnim ključevima tokom njihovog životnog ciklusa.

3.4.1 Životni ciklus ključeva

Životni ciklus počinje od samog generisanja, koje je opisano u poglavlju 3.3.1, nakon čega slijedi upotrebni vijek tokom kojeg se privatni ključevi koriste za potpisivanje ili dešifrovanje primljenih poruka. Korisnici također imaju pristup javnim ključevima drugih korisnika, te ključeve koriste za šifrovanje ili verifikaciju potpisanih dokumenata. U završnoj fazi ključevi izlaze iz upotrebe bilo kroz proces zastarjevanja ili neki drugi događaj. Svaki od navedenih faza u životnom ciklusu para ključeva mora imati određene procedure da bi se osiguralo provođenje dobrih praksi u njihovom upravljanju.

Generisanje i pohrana

Prvi korak je osiguravanje pouzdanog okruženja za generisanje sigurnog para ključeva, najbolje je da tu aktivnost izvode sami korisnici, jer u tom slučaju privatni ključ ne bi trebao biti dostupan nijednoj trećoj strani, no to često nije slučaj, budući da korisničko okruženje može biti kompromitovano ili na neki drugi način neadekvatno za tu namjenu, stoga se mora pribjeći generisanju ključeva u okruženju neke povjerljive treće strane za koju se može biti sigurno da ključeve neće zloupotrijebiti. Pored navedenog i samo skladištenje ključeva, pogotovo privatnih, pruža mnoge sigurnosne izazove. Uzimajući u obzir nabrojano jasno je da navedenim problemima treba pristupiti planski i krajnje ozbiljno jer u protivnom sigurnost kompletnog sistema može biti kompromitovana od samog početka. Logit aplikacija privatne ključeve pohranjuje isključivo unutar Android repozitorija ključeva na korisničkom uređaju, za koji postoje garancije da nije moguće ostvariti pristup korištenjem bilo koje druge aplikacije.

Upotrebni vijek

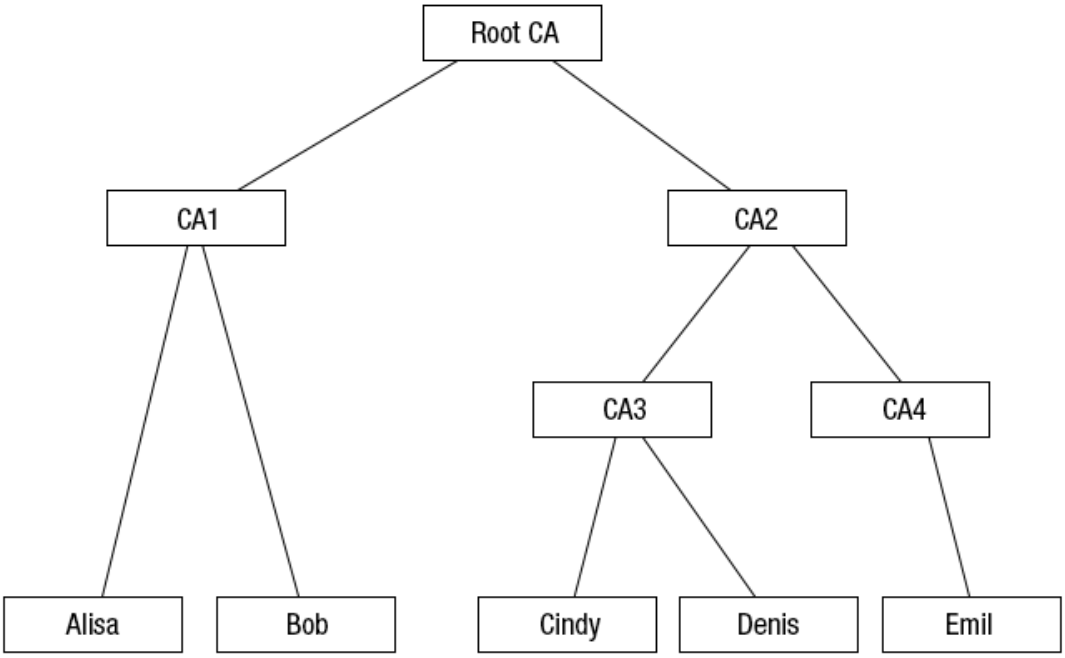
Tokom upotrebne faze životnog vijeka osnovna funkcionalnost je obezbjediti siguran pristup korisničkim javnim ključevima. Pored toga da bi se mogle pružiti adekvatne garancije autentifikacije i potvrde autentičnosti neophodno je utvrditi i provesti jasne procedure koje će osigurati jasnu povezanost entiteta/osoba sa ključevima, u protivnom može doći do krađe identiteta i lažnog predstavljanja. Logit API vršit funkciju repozitorija javnih ključeva za korisnike sistema i povezuje korisnike sa njihovim ključevima, osigurana je osnovna provjera identiteta putem korisničke prijave na fakultetski ZAMGER sistem.

Kraj životnog ciklusa

Dodatno potrebno je utvrditi jasne procedure za kraj životnog vijeka i pohranu starih ključeva, kod Logit API repozitorija ne postoji vremenski ograničen vijek trajanja para ključeva, no pri svakoj novoj instalaciji aplikacije ili zamjeni uređaja, stari par ključeva se arhivira a novogenerisani ključevi se koriste kao sigurnosno relevantni. Pored navedenih funkcionalnosti Logit API se koristi i kao repozitorij potpisanih dokumenata, u ovom slučaju, prisutava predavanjima, no to izlazi iz okvira domena PKI i može se smatrati dodatno funkcionalnošću.

3.4.2 Hijerarhija povjerenja

Za praktičnu upotrebu kriptografije javnog ključa neophodno je da korisnici vjeruju u autentičnost dostupnih javnih ključeva. Stoga su pored prostog direktnog modela povjerenja sa dva učesnika koji razmjenjuju sopstvene javne ključeve uspostavljene raznovrsne hijerarhije povjerljivih učesnika koje omogućavaju formiranje složene mreže koja i sama kao svoju osnovu koristi digitalni potpis i dostupne kriptografske metode.



Slika 3.4: Primjer hijerarhijske infrastrukture javnog ključa

Primjer jedne složene hijerarhije povjerenja prikazan je na slici 3.4, u ovakvom modelu javne ključeve svih nižih učesnika potvrđuje CA *(en. certification authority)*, koji ujedno na adekvatan način vrši i verifikaciju autentičnosti. Većina ovakvih hijerarhijskih PKI arhitektura uređena je u skladu sa standardom X.509. U ovakvom okruženju može da učestvuje više nivoa posrednih CA *(en. intermediate CA)* i obično je uređeno u strukturu drveta sa listovima, gdje se u korjenu nalazi CA *(en. root CA)* kojeg svi posredni CA uzimaju kao pouzdanog i koji sam potpisuje svoj certifikat. Na dnu strukture drveta nalaze se krajnji korisnici kriptosistema. Da bi se ostvarila relacija povjerenja između dva korisnika, nije neophodno da oni dijele isti posredni CA, dovoljan uslov je da se može napraviti veza prema jednom CA u kojeg oba korisnika imaju povjerenje da bi pomenuta relacija bila zadovoljena.

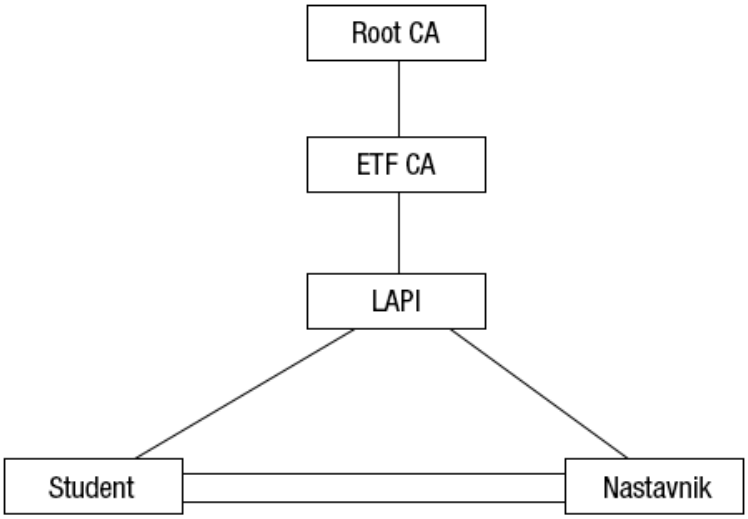
Primjer jedne relacije povjerenja iz priložene hijerarhije može se formirati između korisnika Alisa i Emil, gdje se da jasno utvrditi *certifikacijska putanja* koja za oba korisnika seže do korjenskog Root CA, ilustrativno za Alisu i Emila važi:

$$Alica \rightarrow CA1 \rightarrow RootCA$$

$$Emil \rightarrow CA4 \rightarrow CA2 \rightarrow RootCA$$

dubina certifikacijske putanje također može da igra ulogu u zavisnosti od različitih zahtjeva, u nekim slučajevima duže putanje, poput Emilove se mogu smatrati nepouzdanim za određene namjene, u svakom slučaju poželjno je korištenje što bližih međusovnih putanja.

Logit API posjeduje svoj set ključeva kojim potpisuje sve zaprimljene korisničke sesije prisustva. Da bi se upotpunio model povjerenja neophodno je da se LAPI ceritikat potvrdi od strane institucije koja implementira navedeni sistem, u ovom slučaju Elektrotehničkog fakulteta, koja je dalja poveznica na vanjski korjenski CA i omogućava izgradnju šireg sistema povjerenja gdje više institucija koje koriste isti sistem može ostvariti posredne relacije povjerenja. Važno je istaknuti također da korisnici registracijom bivaju uključeni u LAPI repozitorij, što se može smatrati ekvivalentom certifikata, dodatno korisnici međusobno potvrđuju prisustvo a samim tim ostvaruju vezu međusobnog povjerenja, jer aktivnost prikupljanja potpisa inherentno utvrđuje međusobnu identifikaciju korisnika. Prikaz takvog modela dat je na slici 3.5.



Slika 3.5: Prikaz Logit PKI hijerarhije

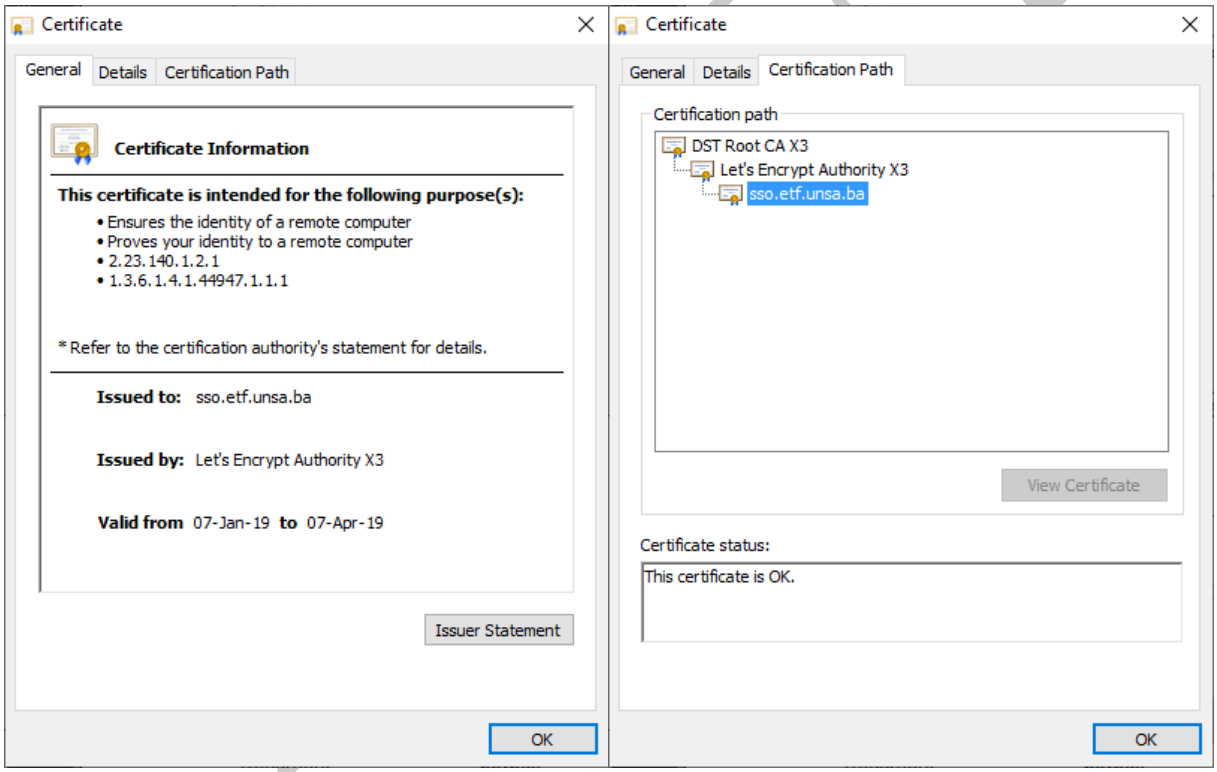
3.4.3 Certifikati

Bitan aspekt PKI je obezbjeđivanje potvrde autentičnosti za javne ključeve, jedan od načina da se to ostvari su svakako certifikati, tj. strukture koje povezuju javne ključeve sa entitetima ili osobama, najčešće su pohranjeni na PKI koji se brinu da osiguravaju što jaču garanciju relacije između entiteta i javnog ključa. U cilju iskoristivosti neophodno je da certifikati obezbjede određeni nivo tehničkih i domenski relevantnih podataka, u većini slučajeva to su:

- ime ili naziv entiteta za čiji javni ključ je vezan
- javni ključ entiteta
- korišteni kriptografski algoritam
- serijski broj
- period važenja
- naziv izdavača certifikata, koji je ujedno i potpisnik
- namjena i ograničenja korištenja javnog ključa

Sam sadržaj certifikata u većini slučajeva je standardiziran, danas najčešće standardom X.509, gdje se i izdati certifikat naziva prema standardu X.509 certifikat, primjer jednog takvog certifikata dat je na slici 3.6a, navedene su osnovne informacije o certifikatu i opšte informacije pobrojane iznad, dodatno na slici 3.6b prikazana je certifikacijska putanja datog certifikata. Pored navedenih informacija certifikat obično sadrži mnoštvo i drugih informacija, budući da standard dopušta proširivanje da bi obuhvatio širok domen primjene. Bitno je još napomenuti da zbog vrlo široke primjene u različitim programskim rješenjima postoji mnoštvo različitih formata zapisivanja i razmjene samih certifikata, te je često neophodno konvertovati certifikate u jedan od odgovarajućih formata.

X.509 certifikati koriste ASN.1 *(en. abstract syntax notation version 1)* kao jezik za izvornu specifikaciju osobina certifikata, pomoću navedenog jezika moguće je izraziti mnoštvo kompleksnih struktura za opis osobina certifikata, jedan od mnoštva načina da se navedena specifikacija zapiše je DER *(en. distinguished encoding rules)*, koja je dalje bazirana na BER pravilima za zapisivanje *(en. basic encoding rules)*, navedene strukture predstavljaju svojevrsnu hijerarhiju deskriptivnih jezika i meta-jezika sličnu relaciji SGML, HTML.



(a) opšte karakteristike (b) lanac povjerenja

Slika 3.6: Osobine certifikata

Logit aplikacija interno koristi namjenski i nestandardni format certifikata, no sve LAPI i korisničke certifikate moguće je pretvoriti u bilo koji od standardnih formata i tako ostvariti interakciju sa ostalim sistemima ukoliko se za to ukaže potreba. Dovoljno je na Logit API strani implementirati novu URI lokaciju koja bi vraćala certifikate u standardizovanom formatu.

Poglavlje 4

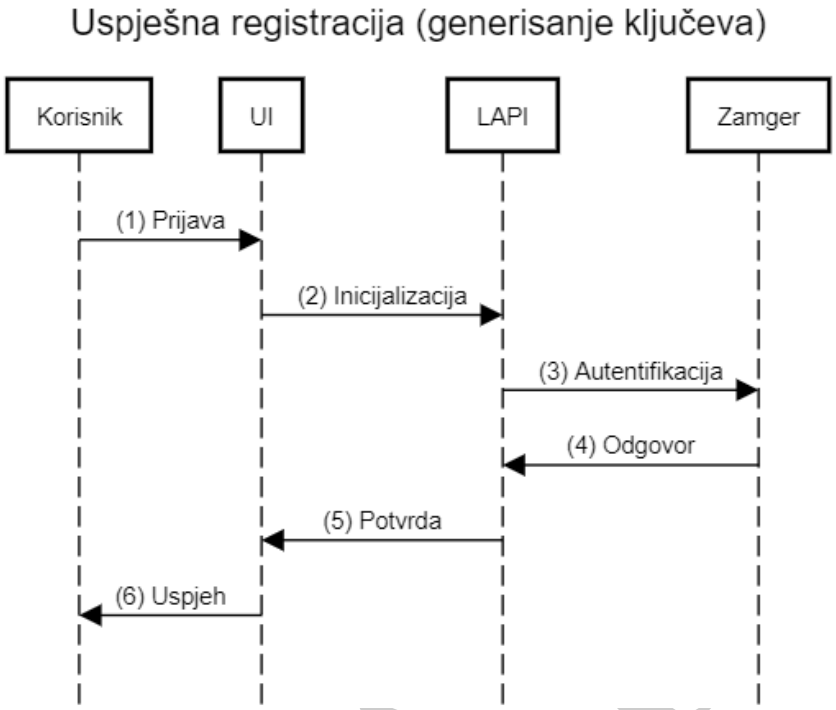
Prijedlog rješenja

U skladu sa datim zahtjevima predložena je izrada aplikacijske platforme pod nazivom Logit (LAPP), opisane u nastavku, sa detaljnim tehničkim detaljima u narednim poglavljima. Uzimajući u obzir data ograničenja, te funkcionalne i nefunkcionalne zahtjeve određeno je da se korisnička aplikacija izradi na Android platformi sa podrškom za Android API nivo počevši od nivoa 19 (4.4 KitKat), to je najniži nivo koji omogućava korištenja naprednih NFC i kriptografskih funkcionalnosti te osigurava dobru pokrivenost potencijalne korisničke baze sa ukupnom adopcijom od preko 90% za navedenu ili višu verziju[17]. Za uspješan rad aplikacije neophodno je da korisnički uređaj podržava i NFC funkcionalnosti, prema prognozama analitičke kuće IHS Technology, do 2020. godine svaki treći uređaj imati će podršku za NFC.[18]

4.1 Logički model rješenja

Priloženi dijagrami interakcije osnovnih funkcionalnosti Logit platforme i pripadajući opis imaju za cilj stvoriti opštu sliku sistema, te tako olakšati praćenje tehničkog modela rješenja datog u nastavku. Tehnički model opisuje dosta detaljniju sliku funkcioniranja sistema i može služiti kao svojevrstan uvod u kod platforme.

Registracija korisnika



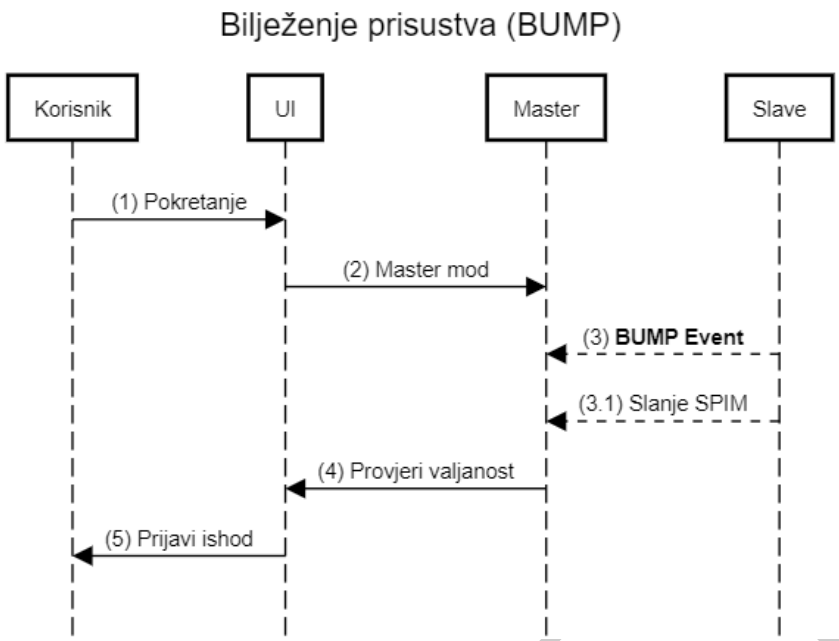
Slika 4.1: Dijagram interakcije - uspješna registracija i generisanje ključeva

Nakon uspješne instalacije aplikacije na korisnički Android uređaj (DEVICE) potrebno je obaviti proces registracije koji se izvršava u dva bitna koraka. Prvi korak sastoji se od unosa već postojećih autentifikacijskih podataka za ZAMGER sistem Elektrotehničkog fakulteta, korisnik se korištenjem datih podataka posredstvom LAPI servisa autentificira na ZAMGER sistemu, bitno je napomenuti da Logit platforma ne sprema korisničku lozinku ZAMGER sistema, navedeni podaci se koriste isključivo za povezivanje postojećeg identiteta i novogenerisanog para korisničkih RSA ključeva (KEYS), što je ujedno i drugi korak u procesu registracije na Logit platformu.

U slučaju uspješnje autentifikacije, korisnika se obavještava o završenoj registraciji te se preusmjerava na glavni ekran za bilježenje prisustva. Generisani javni ključ (CERT) i identifikacioni podaci korisnika spremaju se u LAPI direktorij korisničkih certifikata.

Bilježenje prisustva studenta

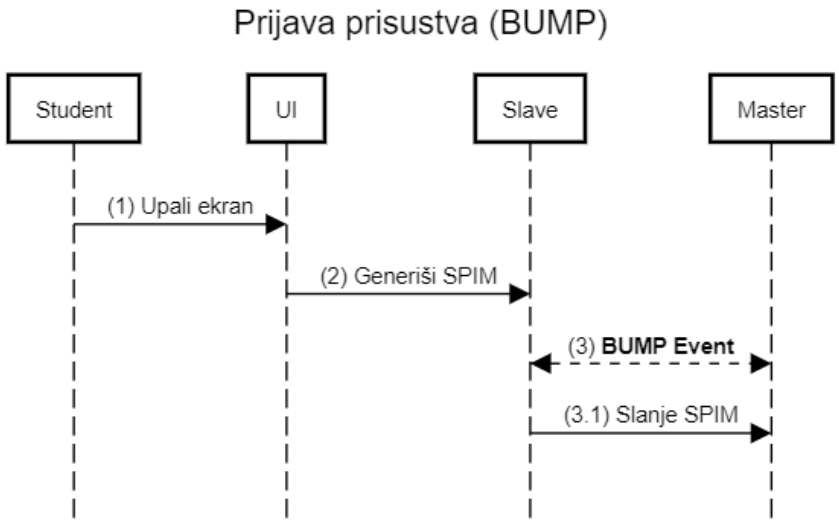
Bilježenje prisustva studenata od strane predmetnog nastavnika izvodi se u Master (M) modu funkcionisanja aplikacije, aplikacija se pri samom pokretanju i nakon uspješno obavljene registracija automatski stavlja u takav mod operacije i u njemu ostaje sve dok je upaljen ekran korisničkog uređaja (DEVICE) i Logit aplikacija (UI) se izvršava u prednjem planu *(en. foreground)*, navedene zahtjeve diktira sama Android platforma.



Slika 4.2: Dijagram interakcije - bilježenje prisustva studenata (Master BUMP)

Ukoliko su ispunjeni prethodno pobrojani zahtjevi, dovoljno je da student sa podešenom Logit aplikacijom na svom uređaju prinese slave (S) uređaj master (M) uređaju i da njegovo prisustvo bude zabilježeno i prikazano na ekranu M uređaja. Samu interakciju (BUMP) inicira studentski S uređaj. Prilikom ovog BUMP događaja dolazi do razmjene kriptografski potpisanih podataka o vremenu i lokaciji (SPIM) sa S na M, gdje M vrši validaciju primljenih podataka poredeći studentsko vrijeme i lokaciju sa vremenom i lokacijom na M uređaju, gdje se prisustvo odbija ukoliko se ustanovi pokušaj lažiranja podataka.

Prijava prisustva



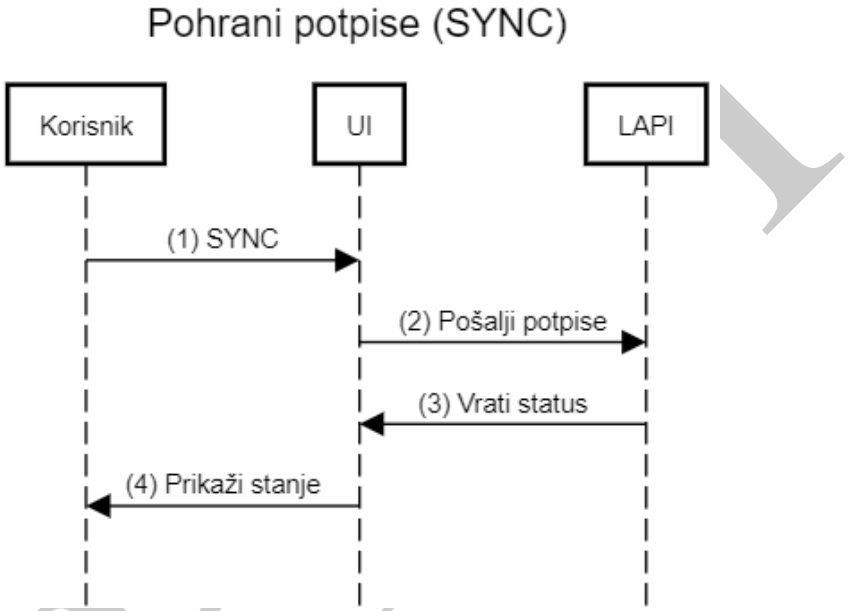
Slika 4.3: Dijagram interakcije - prijava prisustva studenta (Slave BUMP)

Studentski S uređaj i M uređaj nastavnog osoblja podešavaju se na isti način opisan iznad, jedina praktična razlika javlja se prilikom korištenja, gdje je za S uređaj čije se prisustvo bilježi

dovoljno upaliti ekran uređaja da bi se mogla ostvariti BUMP interakcija prislanjanjem S na M. Ovo je moguće jer se NFC HCE emulator Logit aplikacije izvršava u pozadini Android sistema.

Pohranjivanje potpisa sesije na LAPI

Svako bilježenje prisustva unutar Logit Android UI odvija se unutar sesije (SESS) koja se automatski započinje prilikom prvog uspješno zabilježenog prisustva i traje sve dok korisnik ne izvrši pohranu navedene sesije na LAPI servis. Klikom na SYNC dugme prikupljeni podaci šalju se LAPI servisu, provjeravaju se jedinstveni potpisi studenata te potpis ukupne sesije od strane M uređaja, ukoliko se ne pronađu nepravilnosti navedeni podaci se pohranjuju u LAPI repozitorij potpisa, takvi podaci kriptografski su osigurani od naknadne izmjene.



Slika 4.4: Dijagram interakcije - pohranjivanje potpisa na LAPI (SYNC)

Potpisi pohranjeni u LAPI repozitoriju mogu dalje biti korišteni u integrisanim aplikacijskim rješenjima koja zahtijevaju ovakvu vrstu podataka pomoću ponuđenog LAPI REST interfejsa, te se mogu smatrati relevantnim i sigurnim dokazom prisustva.

4.2 Tehnički model rješenja

Uvodi se dodatno pojam lokacijskog dokaza[19] koji u širem smislu u kontekstu podređenog korisnika (en. slave), obuhvata kriptografski potpisan korisnički identitet, korisnički uređaj, vrijeme i GPS lokacijske podatke korisničkog uređaja. Za svrhu osiguranja jedinstvenosti identiteta i vjerodostojnosti potvrde lokacijskih dokaza odabrano je korištenje RSA asimetrične enkripcije, gdje se pri uspješnoj autentifikaciji generiše jedinstveni set ključeva za korisnički uređaj, privatnom dijelu ključa nije moguće pristupiti izvan aplikacije (SEC1), niti je moguće eksportovati ključ (SEC2), a u određenom vremenskom period može postojati samo jedan valjan set ključeva za jednog korisnika jer se raniji ključevi ne uzimaju u obzir ukoliko postoji noviji set (SEC3), sprječavajući tako replikaciju identiteta na više uređaja.

Pored Android komponente aplikacije (UI) izrađena je i serverska aplikacija u programskom jeziku Python (LAPI), čija je namjena posredovanje u komunikaciji sa autentifikacijskim agentom (ZAMGER), te pohranjivanja i održavanje javnih korisničkih kriptografskih ključeva (CERT) i njihovo povezivanje sa autentifikacijskim podacima korisnika, pored toga služi i kao repozitorij za potpisana prisustva (ATTN). Na ovu komponentu se može gledati kao na integrisani namjenski repozitorij korisničkih certifikata i domenski repozitorij neporecivih i neizmjenjivih lokacijskih dokaza (SPIM).

Budući da na Elektrotehničkom fakultetu u Sarajevu postoji SSO (en. Single-Sign On) politika autentifikacije, u serverskoj komponenti (LAPI) je implementiran autentifikacijski posrednik koji prilikom prvog pokretanja aplikacije prijavljuje korisnika koristeći postojeće pristupne podatke, tom prilikom u slučaju uspješne prijave generiše se i jedinstveni set RSA ključeva dužine 2048 bita (KEYS), koji se pohranjuju na korisničkom uređaju (DEVICE), a javni dio, tj. certifikat (CERT) se pohranjuje i u repozitorij ključeva (LAPI) sa poveznicom na korisnički identitet, kasnije se ti certifikati koriste za provjeru valjanosti potpisa lokacijskih dokaza (SPIM).

Da bi se osigurala jednostavnost korištenja aplikacije odabrana je implementacija HCE emulacijskog načina rada NFC komunikatora koji omogućava korisniku da izvrši komunikaciju sa drugim uređajem bez potrebe da pokreće aplikacijski prozor na svom uređaju, dovoljno je da upali ekran svoj uređaja i prinese ga master (M) uređaju koji prikuplja potpise, u ovom slučaju drugoj instanci Logit aplikacije na kojoj je pokrenuta aktivnost za prikupljanje potpisa (LAPP).

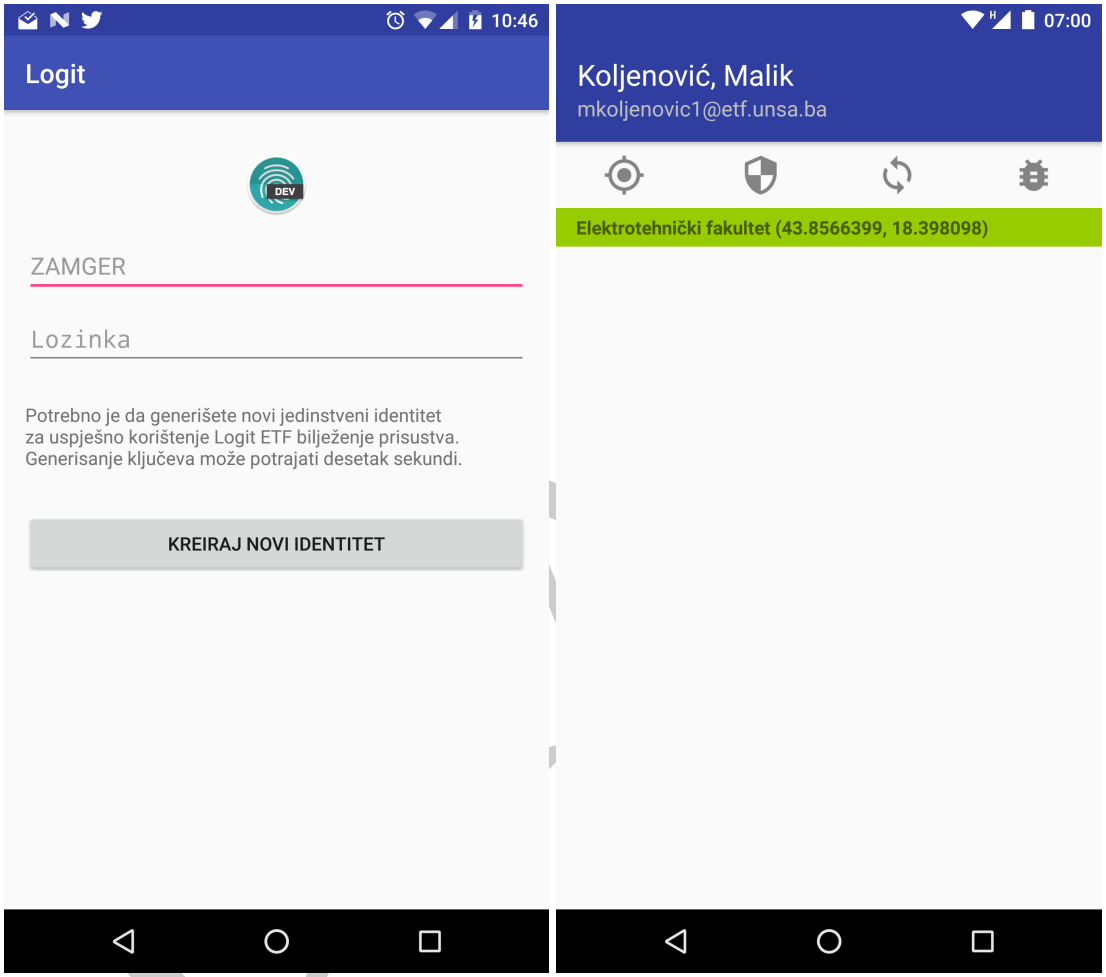
Približavanjem mobilnih uređaja (BUMP) otvara se jednosmjerni komunikacijski kanal u smijeru od slave (S) prema master (M) uređaju korištenjem ISO/IEC 14443 Tip A komunikacijskog protokola pri čemu se emulira NFC Forum Tag tipa 4 i putem NDEF Aplikacije prenosi jedna NDEF poruka (NDEFMSG) koja sadrži vremensko-lokacijski dokaz potpisan od strane korisnika, nadalje u tekstu označen kao SPIM (en. spime)[20].

Po primitku poruke nadređeni uređaj (en. master) koji osluškuje da mu se pridruže podređeni uređaji (en. slave) i ima pokrenutu Logit aplikaciju, tu poruku sprema u lokalni repozitorij potpisa ukoliko ona zadovolja uslove da očitana slave GPS lokacija nije udaljena više od 50 metara od očitane master GPS lokacije (VK1 - validacijski kriterij #1), te da podešena razlika satova master i slave uređaja nije veća od 300 sekundi (VK2), bez da nad SPIM objektom vrši ikakve izmjene, ukoliko SPIM objekat ne zadovoljava date validacijske kriterije odbija se i ispisuje se odgovarajuća poruka na master ekranu. Moguće je naknadno klikom na validacijsko dugme (ACTVAL) u korisničkom interfejsu izvršiti provjeru svih prikupljenih potpisa tokom jedne sesije (SESS), tom prilikom se, ukoliko postoji mrežna veza; svi potpisi pošalju Logit serveru (LAPI) na provjeru i vraća se stanje valjanosti potpisa za sve proslijeđene SPIM objekte.

Ukoliko master (M) želi da pohrani SPIM objekte iz jedne sesije (SESS) na Logit server (LAPI), klikom na sinhronizacijsko dugme u interfejsu (ACTSYNC), on vrši dodatno potpisivanje svakog SPIM objekta svojom komponentom privatnog ključa (MPRK), tako što potpiše hash (SHA256) vrijednost SPIM objekta (AID) sa dodatim svojim jedinstvenim master korisničkim imenom (MUSER) i jedinstvenim identifikatorom sesije (SID) i dodatno generiše SHA256 vrijednosti tih potvrda (CID), nakon čega objedinjuje sve CID vrijednosti i dodatno ih potpisuje svojim MPRK, sve te vrijednosti šalje Logit server (LAPI) na pohranjivanje, ovakvom procedurom se obezbjeđuje neporecivost i neizmjenjivost SPIM i SESS objekata, jer onemogućava

izmjene pojedinačnih SPIM objekata, te brisanje ili dodavanje objekata u finaliziranoj sesiji (SESS) od strane malicioznih aktera bez da naruši integritet SHA256 vrijednosti.

Uzmimajući u obzir bitnost rješenja i visoku vjerovatnoću svakodnevne primjene kod ciljane korisničke grupe, te izazove koje takav slučaj korištenja predstvalja omogućena je i direktna e-mail komunikacija za prijavu grešaka ili slanje prijedloga sa glavnog korisničkog interfejsa (ACTBUG). Kako se radi o slojevitom i kompleksnom softverskom rješenju za više detalja referirati se na izvorni kod priložen u dodatku.



Slika 4.5: Logit UI Android prikaz korisničkog interfejsa

Poglavlje 5

Pregled korištenih tehnologija

5.1 NFC *(en. near-field communication)*

NFC skup komunikacijskih protokola omogućava uspostavu komunikacijskog kanala između dva uređaja koji se nalaze u neposrednoj blizini jedan drugog (1-4 cm) i razmjenu podataka između njih[21]. Komunikacija se odvija na način da MASTER uređaj osluškuje signal na prijemniku i u slučaju detektovanje SLAVE signala uređaja pošiljaoca, propisanog istim standardom zaprima podatke i vrši obradu nad njima, komunikacija se u većini slučajeva odvija jednosmjerno kratkim standardiziranim porukama (NDEF), no moguće je ostvariti i half-duplex komunikaciju između uređaja, kao i razmjenu nestandardnih poruka, u kojem slučaju se sam korisnik mora pobrinuti za implementaciju kompletnog komunikacijskog protokola. Potpuni detalji implementacija dati su u referencama relevantnih standarda u nastavku tehničkog pregleda, odličnu sintezu detalja i implementacije daju Igoe, Coleman i Jepson[22].

5.1.1 NXP NTAG216

U cilju zadovoljenja postavljenih funkcionalnih zahtjeva bilo je neophodno odabrati NFC Tag platformu koja će odrediti relevantne standarde pohrane binarnih podataka na uređajima kao i pripadajuće komunikacijske protokole, također dodatno su postavljeni zahtjevi ekonomičnosti implementacije i kompatibilnosti sa postojećim čitačima. Uzimajući u obzir nabrojane kriterije odabrana je platforma NTAG216 proizvođača NXP Semiconductors[1] bazirana na NFC Forum Tag tipu 2 i ISO/IEC14443 Tip A specifikaciji[23][24].

Mogućnosti navedene platforme dostatne su za ispunjenje navedenih funkcionalnih uslova, a pružaju i neke dodatne sigurnosne mehanizme - poput neizmjenjivog jedinstvenog serijskog broja svakog taga (Tag UID) potpisanog kriptografskim ključem proizvođača, navedena funkcionalnost nije implementirana u predstavljenom rješenju jer se bazira na zaštićenoj NXP tehnologiji i nije kompatibilna sa HCE emulacijom, no umnogome može doprinijeti ukupnoj sigurnosti fizičkih Tag čipova u slučaju produkcijske implementacije rješenja. U nastavku je data proizvođačka lista izdvojenih funkcionalnosti NTAG216:

- 7-bajtni UID programiran od strane proizvođača za svaki tag
- mogućnost jednokratnog programiranja i zaključavanja taga za dalje izmjene
- mogućnost read-only zaključavanja taga
- potpis originalnosti baziran na kriptografiji eliptičnih krivih
- zaštita memorijskih operacija 32-bitnom lozinkom

Page Adr		Byte number within a page				
Dec	Hex	0	1	2	3	Description
0	0h	serial number				Manufacturer data and static lock bytes
1	1h	serial number				
2	2h	serial number	internal	lock bytes	lock bytes	
3	3h	Capability Container (CC)				Capability Container
4	4h	user memory				User memory pages
5	5h					
...	...					
224	E0h					
225	E1h					
226	E2h	dynamic lock bytes			RFUI	Dynamic lock bytes
227	E3h	CFG 0				Configuration pages
228	E4h	CFG 1				
229	E5h	PWD				
230	E6h	PACK		RFUI		

aaa-008089

Slika 5.1: NTAG216 organizacija memorije[1]

5.1.2 NDEF *(en. NFC Data Exchange Format)*

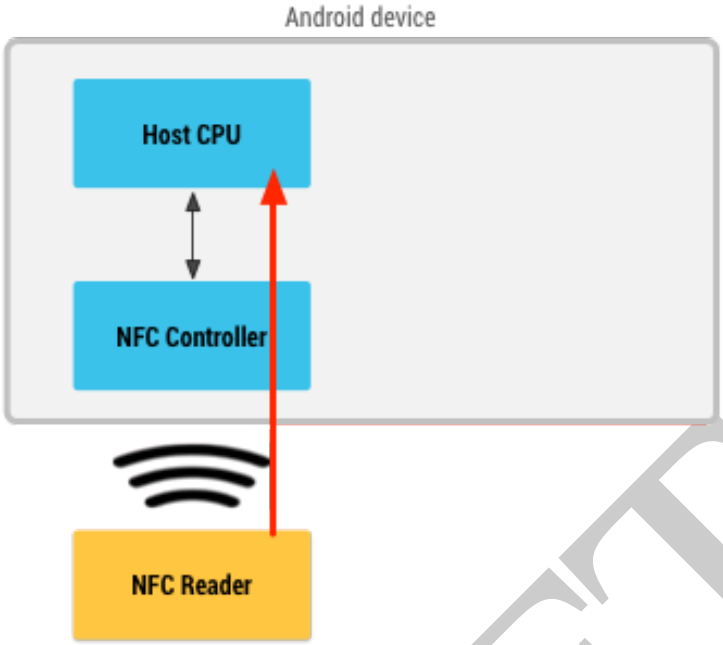
NDEF specifikacija definiše **format enkapsulacije poruke** za razmjenu informacija između dva NFC uređaja. NDEF je lagan binarni format poruke i može se koristiti za enkapsulaciju jednog ili više aplikacijski-definisanih tereta *(en. payload)* raznih vrsta i veličina unutar jedne NDEF poruke. Svaki teret opisan je od strane tipa, dužine i opcionalnog identifikatora. Identifikatori tipa mogu biti URI, MIME media tipovi, ili NFC-specifični tipovi. NDEF je striktno **format** poruke i ne poznaje pojam konekcije ili logičkog kola.[25]

Neki od ciljeva koje NDEF nastoji da ispuni:

- enkapsulacija dokumenata i binarnih objekata, slika etc.
- enkapsulacija podataka nepoznate dužine, npr. stream-a podataka
- agregacija srodnih sadržaja unutar jedne poruke
- kompaktna enkapsulacija malih datagrama

5.1.3 HCE *(en. Host card emulation)*

HCE je metod emuliranja virtuelnog identifikacijskog modula korisnika, u osnovi to je način zaobilaska hardverskih ograničenja (*en. hack, workaround*) koja onemogućavaju direktan pristup SIM (*en. Subscriber Identification Module*) modulu kod mobilnih telefonskih uređaja[26], ovakvo rješenje vuće korijene iz ekonomske realnosti sektora mobilnih komunikacija i kartičnog plaćanja, koja se najpreciznije može okarakterisati kao oligopol, naime Google je nastojao integrisati SIM karticu unutar Android operativnog sistema u vidu eSE (*en. embedded Secure Element*) korištenjem već postojeće SIM kartice operatera a u svrhu razvoja Google Wallet rješenja, no to nije odgovaralo operaterima i odbili su suradnju, nakon toga Google iznalazi alternativne načine rješenja problema poput HCE[27].



Slika 5.2: NFC HCE virtuelnog sigurnog elementa (SE)[2]

HCE na Android OS radi, kako i naziv govori u CE (*en. Card Emulation*) SLAVE modu, gdje se na svaki BUMP sa NFC čitačem odašilju pripremljeni podaci. Podaci koji se pri tome šalju moraju pratiti standard kartice koju žele emulirati i najčešće se vrši prijenos dokumenta ili datagrama unutar jednog ili više NDEF paketa. Logit koristi pristup prijenosa JSON formatiranog SPIM objekta `plain/text` unutar jednog NDEF paketa. Radi se o konceptu sa mnoštvom implementacijskih detalja, te zbog sažetosti za više detalja konsultujte oficijelnu dokumentaciju https://developer.android.com/guide/topics/connectivity/nfc/hce, dok je izvrstan logički prikaz sa primjerima dao Elenkov[28].

5.2 Ostalo

Dodatno koristi se Android arhitektura za dobaljanje geolokacije[29] uz reverzno geokodiranje od strane OpenStreetMap Nominatim projekta[30]. Za potpisivanje i enkripciju korisničkih podataka koristi se RSA[12] kriptografija sa 2048 bit ključem. LAPI je Python flask API i koristi UPnP, kompletan listing koda prikazan je u dodatku.[31]

Poglavlje 6

Izdvojeni detalji implementacije

Detalji izdvojeni u ovom poglavlju ključni su za razumijevanje sigurnosnog modela aplikacije, sa tog aspekta posebno su zanimljiva dva objekta, `Attendance` i `Session`, koji u osnovi predstavljaju proširene kriptografski potpisane SPIM i SESS objekte.

6.1 Podatkovni i kriptografski primitivi

6.1.1 SPIM paket

Spim u širem smislu predstavlja vremensko-lokacijski objekat (*en. SPace-tIMe*), koji korištenjem kriptografske obrade poprima karakteristike lokacijskog dokaza. Izvor za formiranje SPIM objekta sastoji se iz korisničkog imena studenta, geografske širine, geografske dužine i trenutnog vremena na studentovom mobilnom uređaju. Ovako komponovan objekat predstavlja implementaciju lokacijskog dokaza u užem smislu i koristi se dalje kao osnovni podatkovni primitiv za dalju kriptografsku obradu.

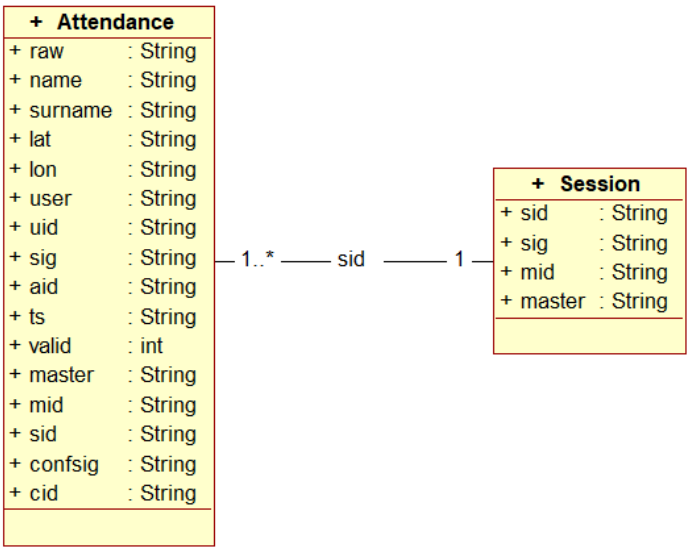
```
type: 'application/octet-stream'
Payload length: ~735 bytes

Payload data:

    {
        'lat':'43.9895212', 'lon':'18.180737', 'ts':'1511308824', 'sig':
        '12947c41451df8ac4586ae9b211decb70087d7db0b1e25202a2b3ec07f267804
        -d526ac641c20e282fe8bd12f7bdc9bcb2a5afcc65ef9b550e80b901c812c9a9f
        -d87bb29828ddcd1bfd1b1d176dd963e336f4b5d5caf9ade5a3b9b60367cceb7d
        -e807dc3dfb80d2b373993725b44e43d0116d72142cb2bf8164c4bcc4a690d949
        -2513a72328181da2c4a0f05e57792322b7b1404ac6c21c8b71596a0cf262e044
        -001151d90d7d9e986d50f2ec930bcaa515c04b645cea27866956f24f285abfd7
        -6e0eb8130f67997ee7a4ded50937f1abe33ebeeb82da86b4c9bfeebf89d3026a
        -423d5c09dd6ec65bd675e2ee6ea873dcc849b5f687c91a825ec010f86f238c5b',
        'uid':'244825e54b1850b2d781bfd7cec37a302d7cf4c9ecac90e8f5e64fa09c-
        66320a','name':'416a64696e', 'surname':'4d756a657a696e6f7669c487',
        'user':'am***'
    }
```

Navedene vrijednosti stringova korisničkog imena studenta, geografske širine, geografske dužine i trenutnog vremena se lančaju u jedan string izloženim redoslijedom i takav string se potpisuje korištenjem RSA kriptografije, tako potpisan paket u obliku JSON objekta (prikazan

u listingu iznad) šalje se na profesorski master uređaj, gdje se dodaju podaci sesije, u vidu jedinstvenog identifikatora sesije (SID), te se potpisano studentsko prisustvo obilježava jedinstvenim heksadecimalnim identifikatorom AID izvedenim iz potpisa prisustva putem SHA256 hash funkcije, navedene vrijednosti, SID i AID se lančaju u jedan binarni string i potpisuju od strane profesora (CONFSIG), naknadno se na iz SHA256 hash vrijednosti CONFSIG profesorskog potpisa formira finalni identifikator potvrde prisustva CID, time se završava kriptografsko osiguravanje valjanosti prisustva u smislu SPIM objekta.



Slika 6.1: Dijagram klasa SPIM i SESS objekata

raw serijalizirana string JSON verzija objekta
name ime studenta
surname prezime studenta
lat geografska širina student
lon geografska dužina student
user ZAMGER korisničko ime studenta
uid User ID - SHA2 hex hash javnog ključa studenta
sig hex potpis SPIM-a (user:lat:lot:ts)
aid Attendance ID - hex SHA2 hash `sig` potpisa
ts vrijeme na studentovom uređaju
valid validation cache
master ZAMGER korisničko ime profesora
mid Master ID - SHA2 hex hash javnog ključa profesora
sid Session ID - identifikator profesorove sesije
confsig Master Conf. profesorov hex potpis (sid:aid)
cid Confirmation ID - SHA2 hex hash confsig-a

Fizički predstavljen opisani SPIM objekat manji je od 1 KB te je pored brzog NFC isl. elektronskog transfera moguće izvršiti prenos alternativnim metodama, kao posebno pogodna čini se QR kod reprezentacija[32] i prenos, koja može biti vrlo korisna u slučaju da nijedan od uređaja ne posjeduje NFC modem. Navedeni modus nije implementiran u aplikaciji i dat je kao sugestija zaobilaženja hardverskih ograničenja, primjer QR oblika ranije datog SPIM objekta prikazan je na slici 6.2.

Dati QR prikaz je čitljiv ali je vidno uočljiva gustina zapisa koja može predstavljati problem u slučaju lošije kvalitete medija prikaza, u tom slučaju, kompletan SPIM paket moguće je značajno smanjiti zamjenom korištenog RSA kriptosistema za kriptosistem baziran na eliptičnim krivim, budući da su ključevi korišteni u tom slučajnu znatno kraći[33], dužina navedenog potpisa bila bi smanjena sa 512 na minimalno 71 bajt[34] navedeni pristup nije prihvaćen u okviru ovog rada zbog povećanja kompleksnosti pokaznog sistema, no praktična implementacija moguća je bez većih programskih izmjena.

(a) oblik korištenjem RSA potpisa

(b) oblik korištenjem ECC potpisa

Slika 6.2: QR oblik SPIM objekta

6.1.2 SESS paket

Dodatno se za SESS objekat prilikom finaliziranja sesije na LAPI serveru vrši prikupljanje svih CID potpisa koji pripadaju datoj sesiji, te se CID vrijednosti ulančane hronološkim redoslijedom potpisuju LAPI ključem koji se nalazi samo na LAPI hardverskom uređaju, stoga je sigurnost LAPI servera od ključne važnosti za sigurnost ukupnog sistema. Ovako potvrđena sesija ne može biti naknadno mijenjana, lažirana ili porečena izvan LAPI izvršnog okruženja.

6.2 Pregled implementacije

6.2.1 MainActivity

Nakon prvobitnog pokretanja aplikacije a za daljnje uspješno korištenje neophodno je izvršiti autentifikaciju korisnika putem nekog već postojećeg korisničkog repozitorija, te generisati pripadajući virtualizirani sigurni element. Navedene aktivnosti izvršavaju se unutar `MainActivity` glavnog početnog prozora Logit aplikacije, prikaz relevantnog dijela koda za generisanje virtualiziranog sigurnog elementa dat je u listingu ispod.

```
1       KeyPairGenerator kpg = KeyPairGenerator.getInstance(
2               "RSA", "AndroidKeyStore");
3       Calendar start = Calendar.getInstance();
4       Calendar end = Calendar.getInstance();
5       end.add(Calendar.YEAR, 1);
6
7       KeyPairGeneratorSpec spec =
8               new KeyPairGeneratorSpec.Builder(this).setAlias("etf_logit_" +
  ↪ ts)
9                       .setKeySize(2048)
10                      .setSubject(new X500Principal("CN=users.etf.ba"))
11                      .setSerialNumber(BigInteger.valueOf(tsLong))
12                                                                          ↩
  ↪ .setStartDate(start.getTime()).setEndDate(end.getTime()).build();
13
14      kpg.initialize(spec);
15
16      KeyPair kp = kpg.generateKeyPair();
17
18      KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
19      ks.load(null);
```

Sigurnosni element generisan kao u primjeru iznad dalje se pohranjuje na korisničkom uređaju, gdje privatni dio nikada ne napušta uređaj i dostupan je isključivo Logit aplikaciji putem Android KeyStore providera. Javni dio se koristi kao dio identifikatora korisnika, te se dodatno pohranjuje i na Logit API korisnički repozitorij za potrebe identifikacije i verifikacije potpisa lokacijskog paketa.

6.2.2 LogitAPDUService

Ekstenzija Androidovog native interfejsa `HostApduService` koji za instaliranu aplikaciju sa `android.permission.BIND_NFC_SERVICE` permisijom vrši pokretanje HCE emulatora prilikom svakog starta operativnog sistema, emulator se u ovom slučaju ponaša kao generički NFC NTAG sa korisnički programiranom memorijom u obliku NDEF poruke koja prenosi jedinstveni potpisan studentski lokacijski dokaz. Navedena servisna komponenta aplikacije aktivna je svaki put dok je i ekran uređaja aktivan ili dok korisnik sam ne zaustavi pripadajući servis. Navedene funkcionalnosti postižu se uključivanjem dijela koda datog u nastavku unutar `<application>` direktive manifest fajla.

```
1       <service
2           android:name=".LogitApduService"
```

```
        android:permission="android.permission.BIND_NFC_SERVICE">
        <intent-filter>
            <action android:name="android.nfc.cardemulation.action.HOST_APDU_SERVICE" />

            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>

        <meta-data
            android:name="android.nfc.cardemulation.host_apdu_service"
            android:resource="@xml/apduservice" />
    </service>
```

Nadalje `LogitAPDUService` sadrži logiku za ispravno konstruisanje i formatiranje NDEF paketa[35] lokacijskog dokaza i njegovo potpisivanje, te ostalu neophodnu kriptografsku obradu. U nastavku će biti dat prikaz kompletnog servisa sa komentarima relevantnih dijelova.

```
    final static int APDU_INS = 1;
    final static int APDU_P1 = 2;
    final static int APDU_P2 = 3;
    final static int APDU_SELECT_LC = 4;
    final static int APDU_READ_LE = 4;
    final static int FILEID_CC = 0xe103;
    final static int FILEID_NDEF = 0xe104;
    final static byte INS_SELECT = (byte) 0xa4;
    final static byte INS_READ = (byte) 0xb0;
    final static byte INS_UPDATE = (byte) 0xd6;
    final static byte P1_SELECT_BY_NAME = (byte) 0x04;
    final static byte P1_SELECT_BY_ID = (byte) 0x00;
    final static int DATA_OFFSET = 5;

    final static byte[] DATA_SELECT_NDEF = {(byte) 0xd2, (byte) 0x76, (byte) 0x00, (byte) 0x00, (byte) 0x85, (byte) 0x01, (byte) 0x01};
    final static byte[] RET_COMPLETE = {(byte) 0x90, (byte) 0x00};
    final static byte[] RET_NONDEF = {(byte) 0x6a, (byte) 0x82};
    final static byte[] FILE_CC = {
            (byte) 0x00, (byte) 0x0f,       // CCLEN - CC container size
            (byte) 0x20,                    // Mapping version
            (byte) 0x04, (byte) 0xff,       // MLe - max. read size
            (byte) 0x08, (byte) 0xff,       // MLc - max. update size

            // TLV Block (NDEF File Control)
            (byte) 0x04,                    // Tag - Block type
            (byte) 0x06,                    // Length
            (byte) 0xe1, (byte) 0x04,       // File identifier
            (byte) 0x04, (byte) 0xff,       // Max. NDEF file size
            (byte) 0x00,                    // R permission
            (byte) 0x00,                    // W permission
    };
```

Deklariše konstantne vrijednosti brojnih dijelova neophodnih za kontrukciju standardne NDEF poruke, između ostalog capability fajl koji predstavlja svojevrsno zaglavlje NDEF paketa.

Metoda generateSignature instancira lokacijske servise i vrši konstukciju paketa lokacijskog dokaza kao i kriptografskih primitiva neophodnih za potpisivanje istog. Za konstrukciju lokacijskog dokaza neophodno je pribaviti trenutno vrijeme uređaja, to je prikazano u narednom dijelu koda.

```
1     Long tsLong = System.currentTimeMillis() / 1000;
2     String ts = tsLong.toString();
3     byte[] signature;
```

Nakon toga vrši se instanciranje i učitavanja Android KeyStore objekta koji sadrži korisnički par ključeva neophodnih za potpisivanje lokacijskog paketa.

```
1     KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
2     ks.load(null);
3     KeyStore.ProtectionParameter pp = new
↪KeyStore.PasswordProtection(null);
```

Nadalje iz `KeyStore` objekta učitava se najažurniji virtualizirani sigurnosni element.

```
1     Enumeration<String> aliases = ks.aliases();
2     String alias = aliases.nextElement();
3     Entry entry = ks.getEntry(alias, pp);
```

Zbog potrebe za što kompaktnijim prenosom podataka za sve vrijednosti gdje je to moguće generišu se hash preslikavanja koja se kasnije koriste za verifikaciju i pohranjivanje. Za potrebe generisanja hash vrijednosti instancirase SHA-256 `MessageDigest` objekat.

```
1     MessageDigest md = MessageDigest.getInstance("SHA-256");
```

Iz virtualiziranog sigurnog elementa dalje učitavamo korisnički certifikat sa javnim ključem, te za potrebe verifikacije identiteta heksadecimalnu reprezentaciju njegove hash vrijednosti pripremamo za uključenje u paket lokacijskog dokaza.

```
1     Certificate c = ks.getCertificate(alias);
2     byte [] pubKey = c.getPublicKey().getEncoded();
3     md.update(pubKey, 0, pubKey.length);
4     byte [] pubKeyHash = md.digest();
5     String pubKeyHashString = LogitApplication.toHext(pubKeyHash);
```

Zatim iz virtualiziranog sigurnog elementa korištenjem korisničkog privatnog ključa pripremamo objekat koji vrši potpisivanje lokacijskog paketa.

```
1     Signature s = Signature.getInstance("SHA256withRSA");
2     s.initSign(((PrivateKeyEntry) entry).getPrivateKey());
3     SharedPreferences userData = getSharedPreferences("UserData", 0);
```

Dio lokacijskog paketa koji je pokriven korisničkim digitalnim potpisom sadrži korisničko ime, lokacijske parametre geografske dužine i širine, te vrijeme uređaja u trenutku kreiranja digitalnog potpisa.

```
    String sigPkg = userData.getString("user", "unknown") +
            ":" + location.getLatitude() +
            ":" + location.getLongitude() +
            ":" + ts;
```

Takav paket se potpisuje i bilježi se njegova SHA-256 preslikana vrijednost za potrebe verifikacije.

```
    s.update(sigPkg.getBytes("UTF-8"));
    signature = s.sign();
```

Osnovni potpisani lokacijski paket se proširuje vrijednostima generisanih SHA-256 preslikavanja i osnovnim korisničkim podacima, te se prosljeđuje metodi createMessage koja formira standardizovan NDEF paket, takav spreman NDEF paket stavlja se na raspolaganje komponenti za prenos putem NFC protokola.

```
    msg = "{\"lat\":\"" + location.getLatitude() +
            "\", \"lon\":\"" + location.getLongitude() +
            "\", \"ts\":\"" + ts +
            "\", \"sig\":\"" + LogitApplication.toHext(signature) +
            "\", \"uid\":\"" + pubKeyHashString +
            "\", \"name\":\"" +
↪LogitApplication.toHext(userData.getString("name",
↪"unknown").getBytes("UTF-8")) +
            "\", \"surname\":\"" +
↪LogitApplication.toHext(userData.getString("surname",
↪"unknown").getBytes("UTF-8")) +
            "\", \"user\":\"" + userData.getString("user", "unknown") +
↪"\"}";

    NdefMessage ndef = createMessage(msg.getBytes("UTF-8"));
    byte[] ndefarray = ndef.toByteArray();

    mNdefFile = new byte[ndefarray.length + 2];

    mNdefFile[0] = (byte) ((ndefarray.length & 0xff00) >> 8);
    mNdefFile[1] = (byte) (ndefarray.length & 0x00ff);

    System.arraycopy(ndefarray, 0, mNdefFile, 2, ndefarray.length);

    logitApp.setMessage(mNdefFile);
```

6.2.3 processCommandApdu

Budući da se u prikazanom slučaju vrši emulacija pasivnog NTAG uređaja, APDU servis se izvršava u slave modu i zaprima instukcije od strane master uređaja, neophodno je implementirati

parser petlju i logiku za komunikaciju sa master uređajem, unutar koje se vrši prepoznavanje zadatih instrukcija i priprema adekvatan odgovor, navedena logika implementirana je unutar `processCommandApdu` metode. Kompletna logika emulacije tag uređaja svodi se na prosljeđivanje adekvatnog zaglavlja taga i READ FROM TO binarnog protokola za čitanje memorije koju šalje master, stoga glavninu navedene metode čini jedna switch petlja koja u skladu sa zadatom komandom vraća zaglavlje ili raspon bita emuliranog taga, u ovom slučaju sadržaj koji se emulira je proširena lokacijski paket iznad. Detalje opisane metode možete pogledati u prilogu koda u dodatku.

6.2.4 AttendanceActivity

Za ostvarivanje pune funkcionalnosti aplikacije neophodna je bila implementacija master moda za prikupljanje i obradu NDEF paketa, u ovom slučaju korisničkih potpisa, enkapsuliranih u obliku potpisanog lokacijskog paketa. `AttendanceActivity` vrši navedenu funkcionalnost te dodatno vrši provjeru valjanosti potpisa i podataka korisničkih lokacijskih paketa. Verifikacija korisničkih lokacijskih paketa vrši se tako što se korisnički slave podaci o vremenu i lokaciji porede sa podacima o vremenu i lokaciji master uređaja, time se osigurava zaštita od napada lažiranja podataka, te bi za takvu vrstu prevare bila neophodna koluzija dva aktera suprostavljenih interesa, što znatno umanjuje vjerovatnoću takvih napada. Moguće je podesiti vrijednosti dozvoljenih odstupanja verifikacijskih parametara izmjenom koda za tu namjenu datog u nastavku.

```
1      final long timediff = System.currentTimeMillis() / 1000 -
  Long.parseLong(tmpAttn.getTs());
2      final Location userLocation = new Location("MOCK");
3      userLocation.setLatitude(Double.valueOf(tmpAttn.getLat()));
4      userLocation.setLongitude(Double.valueOf(tmpAttn.getLon()));
5      if (Build.VERSION.SDK_INT >= 23
6              && ContextCompat.checkSelfPermission(that,
  android.Manifest.permission.ACCESS_FINE_LOCATION ) ==
  PackageManager.PERMISSION_GRANTED
7              && ContextCompat.checkSelfPermission(that,
  android.Manifest.permission.ACCESS_COARSE_LOCATION) ==
  PackageManager.PERMISSION_GRANTED
8              || Build.VERSION.SDK_INT < 23) {
9          FusedLocationProviderClient mFusedLocatiionClient =
  LocationServices.getFusedLocationProviderClient(that);
10         mFusedLocatiionClient.getLastLocation().addOnSuccessListener(new
  OnSuccessListener<Location>() {
11             @Override
12             public void onSuccess(Location location) {
13                 Float locdiff = userLocation.distanceTo(location);
14                 if (Math.abs(timediff) < 300) {
15                     if (locdiff < 100) {
16                         // Lokacija i vrijeme SLAVE uredaja su VALIDNI
17                     } else {
18                         Toast.makeText(that, "Greška: lokacije udaljene " +
  locdiff.intValue() + " metara.", Toast.LENGTH_LONG).show();
19                     }
20                 } else {
21                     Toast.makeText(that, "Greška: vrijeme nije tačno ili je
  TAG zastario.", Toast.LENGTH_LONG).show();
22                 }
```

Da bi se izbjegla obaveza reimplementiranja NDEF protokola za podatke primljene putem NFC podatkovnog interfejsa Android nudi predefinisani intent filter za direktnu manipulaciju NDEF poruka, te je za njegovo korištenje potrebno dodati ispod prikazani kod unutar manifest fajla Android aplikacije. Korištenjem ovog filtera programer kao rezultat uspješnog NFC prenosa dobija standardizovanu NDEF poruku spremnu za obradu. Ovaj interfejs se koristi unutar `AttendanceActivity`.

```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED" />

    <category android:name="android.intent.category.DEFAULT" />

    <data android:mimeType="application/octet-stream" />
</intent-filter>
```

Ostatak koda unutar `AttendanceActivity` klase koristi se za prikaz i obradu elemenata korisničkog interfejsa master moda za prikupljanje korisničkih potpisa.

Poglavlje 7

Korisničko uputstvo

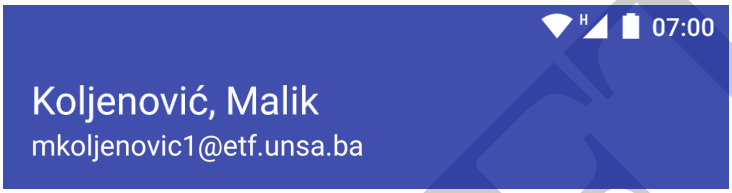
1. **Uspostavite internet konekciju** prema uputama vašeg instruktora.
   (a) potrebno je da na mreži bude dostupna Logit serverska aplikacija i certifikacijski repozitorij za uspješnu prijavu i korištenje, dostupnost možete provjeriti posjetom na https://logit.mine.nu:5000
2. **Uključite lokacijske usluge** vašeg Android mobilnog uređaja.
   (a) detaljno uputstvo možete pronaći na https://support.google.com/accounts/answer/3467281?hl=hr
3. **Omogućite NFC komunikaciju** na vašem Android mobilnom uređaju i **isključite Android Beam** uslugu za optimalan rad aplikacije.
   (a) `Settings > NFC > Enable`
   (b) `Settings > NFC > Android Beam > Disable`
   (c) više detalja za navedene postavke pročitajte na https://support.google.com/nexus/answer/2781895?hl=hr
4. Ukoliko niste, **omogućite sigurnosnu funkcionalnost zaključavanja vašeg ekrana**
   (a) Android OS nudi usluge integrisane sigurnosti mobilnih uređaja, te je Keystore funkcionalnost sigurnog pohranjivanja privatnih ključeva usko vezana za ostale sigurnostne postavke, stoga omogućite zaključavanje ekrana slijedeći uputstvo na https://support.google.com/nexus/answer/2819522?hl=hr
5. Prihvatite poziv za alpha testiranje posjetom na https://play.google.com/apps/testing/ba.unsa.etf.logit i nastavite na Play Store te **instalirajte aplikaciju**
   (a) ukoliko vaš mobilni uređaj nije izlistan kao podržan obratite se vašem instruktoru i biti će vam izdat jedinstveni NFC Certifikat, koji ćete koristiti za bilježenje prisustva
6. **Pokrenite Logit aplikaciju**
7. **Unesite vaše ZAMGER korisničke podatke**, ovi podatci koriste se jednokratno za provjeru valjanosti identiteta prije generisanja vašeg para ključeva, vaša lozinka ne ostaje pohranjena na Logit sistem i prenosi se https kanalom prema ZAMGER servisu

8. Aplikacija je spremna za korištenje i ne mora biti pokrenuta u prednjem planu za prijavu prisustva, **za optimalne rezultate** dovoljno je da upalite ekran vašeg Android uređaja na "lock screen" i prislonite na Android uređaj instruktora.

9. **Ukoliko želite koristiti aplikaciju u instruktorskom modu** i prikupljati prisustvo, dovoljno je da pokrenete Logit aplikaciju u prednjem planu te prislonite vaš uređaju studentskom uređaju u skladu sa korakom 8.

7.1 Instruktorski način rada

Pokretanjem glavnog prozora Logit aplikacije ulazite u mod za prikupljanje studentskih potpisa prisustva. Ovaj zaslon podijeljen je na četiri komponente, opisi kako slijedi u nastavku.

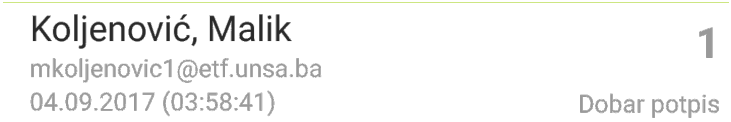


Slika 7.1: Zaglavlje aplikacije prikazuje aktivnog korisnika

Slika 7.2: Glavni izbornik, opisi funkcionalnosti u nastavku

Dugme 1 služi za ručno osvježavanje trenutne lokacije
Dugme 2 koristite za provjeru valjanosti ključeva korištenih pri potpisu
Dugme 3 sinhronizacija trenutne sesije na Logit server, svi potpisi se pohranjuju u jednu sesijsku cjelinu i brišu sa mobilnog uređaja (kreira se nova sesija)
Dugme 4 otvara e-mail klijent po izboru korisnika u cilju lakše prijave grešaka

Slika 7.3: Trenutno zabilježena lokacija korisničkog uređaja



Slika 7.4: Ordinalno numerisan spisak prisutnih studenata

Poglavlje 8

Sigurnosna analiza

Sa aspekta teorije igara[36] u okruženju dva igrača unutar kooperativne igre[37] pronalazak optimalne strategije je uvijek prost u slučaju da međusobna saradnja maksimizira profit oba igrača, stoga ukoliko se prikupljanje obostrano korisnih dokaza u vidu prisustva predavanju svede u navedene okvire tada se radi o kolaborativnoj igri studenta i predavača te ne bi trebao da se javlja problem međusobnog povjerenja, dovoljan uvjet za sigurnost takvog sistema je ispravna tehnička implementacija kriptografskih rješenja koja sprječavaju napade jednostranim falsifikovanjem dokaza. Aplikacija izrađena u okviru ovog rada zadovoljava preduvjete kolaborativne igre **dva igrača**, budući da je prisustvo studenata dokaz prisustva profesora i obratno, njihov odnos je kriptografski osiguran strukturom nalik na hijerarhiju uređenog hash skupa, takva struktura osigurava osobine neizmjenjivosti i neporecivosti.

Sa tehničkog aspekta jedan vektor napada predstavlja lažiranje lokacije ili vremena na korisničkim uređajima, stim što je preduvjet za uspješnost takvog napada koluzija predavača i studenta da priskrbe korist na štetu treće strane, tj. institucije korisnice sistema i neučesnika u koluziji. Moguće je otežati izvodivost i osigurati detekciju ovog napada programskim mjerama zabrane i bilježenja korištenja ručno podešene (*en. mock*) lokacije uređaja i provjerom vremenskih podataka potpisa na LAPI strani koja osigurava pouzdane vremenske podatke. Dodatno treba napomenuti da cijena opisanog napada raste proporcionalno broju studenata učesnika jer svi studenti koji žele priskrbiti neostvarenu korist moraju učestvovati u koluziji, također svi neučesnici imaju štetu (npr. predavanje nije održano u predviđenom terminu zbog odsustva predavača, naknadno falsifikovan dokaz o održavanju u koluziji sa dva studenta, svi ostali studenti gube bodove za prisustvo), stoga i direktu korist od razotkrivanja napada.

Sigurnost izloženog rješenja polazi i počiva na pretpostavci povjerljivosti i jedinstvenosti **korisničkog privatnog ključa** koji nikada ne napušta okruženje emuliranog sigurnosnog elementa korisničkog Android uređaja, ukoliko se naruši data pretpostavka ne postoje garancije sigurnosti sistema. Postoji jedan-na-jedan asocijacija između korisnika i sigurnosnog uređaja, gdje se zbog prirode problema i niskog nivoa prijetnje kao dovoljan uvjet asocijacije uzima posjedovanje uređaja, no takav uvjet ne može se smatrati dovoljnim za čvrst dokaz identiteta te se za namjene gdje je to neophodno preporučuje implementacija dodatnog faktora biometrijske identifikacije unutar pouzdanog okruženja (ne na korisničkom uređaju).

Poglavlje 9

Zaključak

Aplikacija izrađena u okviru ovog rada zadovoljava zahtjeve navedene u postavci zadatka i pripadajućem opisu. Korištene su savremene kriptografske metode za implementaciju sigurnosno osjetljivih funkcionalnosti i osigurano je stabilno okruženje za neometano funkcionisanje aplikacije, dodatno je prema zahtjevima uspješno realiziran NFC komunikacijski interfejs između studentskih i instruktorskih mobilnih uređaja. U cilju lakšeg skaliranja težilo se je što više koristiti standardizovane tehnologije, posebno kada je u pitanju NFC, gdje je dodatno implementirana emulacija NTAG vrste taga kao NDEF medija, time je omogućeno da se sistem u budućnosti prilagodi stacionarnim NFC čitačima i korištenju samostalnih NFC tagova.

Pokušana je pilot primjena sistema u saradnji sa nastavnim osobljem na predmetu "Tehnologije sigurnosti" u školskoj godini 2017/18. kojom prilikom je sačinjen spisak studenata i izvršene pripreme sistema. Navedena pilot okončan je neuspješno zbog sigurnosnih problema u integraciji sa postojećim sistemima, nedostatka resursa i nepostojanja pokusnog sistema pogodnog za projekte u ranoj fazi testiranja, u cilju povećanja inovativnosti i razvoja novih usluga preporučuje se izrada paralelnih pokusnih sistema odvojenih od produkcijskog u okviru Elektrotehničkog fakulteta u Sarajevu.

Tokom pripreme pilot primjene identifikovano je da značajan broj studenata ne posjeduje NFC omogućene mobilne uređaje, te su za njihove potrebe izrađene NTAG216 NFC token naljepnice, no primjena navedenih tokena uvjetovana je dodatnim istraživanjem i doradom Logit sistema za rad sa NTAG216 da bi osigurao isti ili viši nivo sigurnosnih garancija od onog koje pruža Android izvršno okruženje. Kao dodatna smjernica u istraživanju dat je prijedlog korištenja QR kodova za namjenu supstitucije u slučajevima nepostojanja tehničkih predispozicija za upotrebu sistema na strani korisnika, navedena tehnologija može dati dobre rezultate u praktičnoj primjeni i zavređuje dalji istraživački tretman.

Krajnja težnja Logit rješenja je obuhvatanje cjelokupnog sistema autentifikacije i modeliranje relacija povjerenja u materijalnopravnom okruženju, kroz izradu proširive bazne platforme koja može obuhvatiti digitalizaciju mnoštva svakodnevnih administrativnih zadataka jedne institucije, sa tim ciljem daljnje istraživačke napore zavređuje usmjeriti ka razvoju stabilne PKI infrastrukture, kao i digitalizaciji vjerodostojnih institucionalnih registara poput registra ispita sa ciljem digitalizacije studentskog indeksa i srodnih dokumenata.

Prilozi

Prilog A

Izvorni kod

Pripremljeno za štampanje uz pomoć LaTeX-a i dopunskog paketa `minted`, Python interpretera i Pygments sintaksičkog parsera. Puni LaTeX izvorni kod dostupan na https://github.com/koljenovic/logit-tex.

A.1 LAPI izvorni kod

Repo: https://github.com/koljenovic/logit-node/

```
.
├── Logit
│   ├── Logit
│   │   ├── __init__.py
│   │   ├── logit.db
│   │   └── static
│   └── logit.wsgi
├── README.md
└── README.md~
```

A.1.1 `__init__.py`

```
from __future__ import print_function
from bs4 import BeautifulSoup
from flask import Flask, request
from asn1crypto.x509 import Certificate
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Hash import SHA256
import os, sys, sqlite3, requests, json, binascii

app = Flask(__name__)
db_name = 'logit.db'
dbcon = None

def init_db(dbcon):
    dbcon.execute("CREATE TABLE Users (id INTEGER PRIMARY KEY autoincrement NOT NULL, uid TEXT, user TEXT, name TEXT, surname TEXT, cert TEXT, time TIMESTAMP DEFAULT current_timestamp NOT NULL)")
```

```
16      dbcon.execute("CREATE TABLE Attendances (id INTEGER PRIMARY KEY
    autoincrement NOT NULL, sid TEXT, mid TEXT, uid TEXT, lat TEXT, lon TEXT,
    ts TEXT, sig TEXT, aid TEXT, confsig TEXT, cid TEXT, time TIMESTAMP DEFAULT
    current_timestamp NOT NULL)")
17      dbcon.execute("CREATE TABLE Sessions (id INTEGER PRIMARY KEY
    autoincrement NOT NULL, sid TEXT, sig TEXT, master TEXT, time TIMESTAMP
    DEFAULT current_timestamp NOT NULL)")
18      dbcon.commit()
19
20  if not os.path.isfile(db_name):
21      dbcon = sqlite3.connect(db_name)
22      init_db(dbcon)
23  else:
24      dbcon = sqlite3.connect(db_name)
25
26  @app.route("/")
27  def main():
28      return 'Work'
29
30  @app.route("/auth/", methods=['POST'])
31  def auth():
32      user = request.form['user']
33      passw = request.form['pass']
34      cert = request.form['cert']
35      uid = request.form['uid']
36      s = requests.Session()
37      r = s.post('https://zamger.etf.unsa.ba/index.php',
    data={'loginforma':1, 'login': user, 'pass': passw})
38      r = s.get('https://zamger.etf.unsa.ba/index.php?sta=common/profil')
39      soup = BeautifulSoup(r.text, 'html.parser')
40      nameTag = soup.find('input', attrs={"name": "ime"})
41      surnameTag = soup.find('input', attrs={"name": "prezime"})
42      name = nameTag['value'].encode('utf8')
43      surname = surnameTag['value'].encode('utf8')
44      dbcon.execute("INSERT INTO Users (uid, user, name, surname, cert)
    VALUES (?, ?, ?, ?, ?)", (uid, user, buffer(name), buffer(surname), cert))
45      dbcon.commit()
46      return json.dumps({ 'name': name, 'surname': surname})
47
48  @app.route("/validate/", methods=['POST'])
49  def validate():
50      data = request.data
51      attns = json.loads(data)
52      # print(attns, file=sys.stderr)
53      c = dbcon.cursor()
54
55      result = []
56
57      for attn_string in attns:
58          attn = json.loads(attn_string)
59          c.execute("SELECT max(id) id FROM Users WHERE user=? GROUP BY
    user", (attn['user'],))
60          certId = c.fetchone()
61          c.execute("SELECT * FROM Users WHERE id = ?", (certId[0],))
62          user = c.fetchone()
63          cert = Certificate.load(binascii.unhexlify(user[5]))
64          n = cert.public_key.native['public_key']['modulus']
65          e = cert.public_key.native['public_key']['public_exponent']
```

```
66          package = attn['user'] + ':' + attn['lat'] + ':' + attn['lon'] +
    ':' + attn['ts']
67          digest = SHA256.new()
68          digest.update(package)
69
70          public_key = RSA.construct((n, e))
71
72          verifier = PKCS1_v1_5.new(public_key)
73          verified = verifier.verify(digest, binascii.unhexlify(attn['sig']))
74
75          attn['valid'] = 1 if verified else -1
76          attn['raw'] = json.dumps(attn)
77          attn['name'] = binascii.unhexlify(attn['name'])
78          attn['surname'] = binascii.unhexlify(attn['surname'])
79          result.append(attn)
80          # print(verified, file=sys.stderr)
81
82      return json.dumps(result)
83
84  @app.route("/sync/", methods=['POST'])
85  def sync():
86      data = request.data
87      session = json.loads(data)
88      c = dbcon.cursor()
89      c.execute("SELECT * FROM Users WHERE uid = ?", (session['mid'],))
90      master = c.fetchone()
91      cert = Certificate.load(binascii.unhexlify(master[5]))
92      n = cert.public_key.native['public_key']['modulus']
93      e = cert.public_key.native['public_key']['public_exponent']
94
95      hash_package = ''.join(sorted([attn['cid'] for attn in
    session['attns']]))
96      digest = SHA256.new()
97      digest.update(hash_package)
98
99      public_key = RSA.construct((n, e))
100
101     verifier = PKCS1_v1_5.new(public_key)
102     verified = verifier.verify(digest, binascii.unhexlify(session['sig']))
103
104     if verified:
105         dbcon.execute("INSERT INTO Sessions (sid, sig, master) VALUES (?,
    ?, ?)", (session['sid'], session['sig'], session['master']))
106         for attn in session['attns']:
107             db_tuple = (attn['sig'], attn['mid'], attn['uid'], attn['lat'],
    attn['lon'], attn['ts'], attn['sig'], attn['aid'], attn['confsig'],
    attn['cid'])
108             dbcon.execute("INSERT INTO Attendances (sid, mid, uid, lat,
    lon, ts, sig, aid, confsig, cid) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)",
    db_tuple)
109         dbcon.commit()
110         return "", 201 # Created
111     else:
112         return "", 401 # Unauthorized
113
114 if __name__ == "__main__":
115         app.run(host='0.0.0.0', port=5000)
```

A.2 Android izvorni kod

Repo: https://github.com/koljenovic/logit/tree/master/android/app/src/main

```
.
├── AndroidManifest.xml
├── ic_launcher-web.png
├── java
│   └── ba
│       └── unsa
│           └── etf
│               └── logit
│                   ├── api
│                   │   └── LogitService.java
│                   ├── AttendanceActivity.java
│                   ├── AttendanceAdapter.java
│                   ├── LogitApduService.java
│                   ├── LogitApplication.java
│                   ├── MainActivity.java
│                   └── model
│                       ├── Attendance.java
│                       ├── Place.java
│                       ├── Session.java
│                       └── User.java
└── res
```

A.2.1 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ba.unsa.etf.logit">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.NFC" />
    <uses-permission android:name="android.permission.INTERNET"/>

    <uses-feature
        android:name="android.hardware.nfc.hce"
        android:required="true" />
    <uses-feature android:name="android.hardware.location.gps" />

    <application
        android:name=".LogitApplication"
        android:allowBackup="false"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <service
            android:name=".LogitApduService"
            android:permission="android.permission.BIND_NFC_SERVICE">
            <intent-filter>
                <action android:name="android.nfc.cardemulation.action.HOST_APDU_SERVICE" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>

            <meta-data
                android:name="android.nfc.cardemulation.host_apdu_service"
                android:resource="@xml/apduservice" />
        </service>

        <!--<intent-filter>-->
            <!--<action android:name="android.nfc.action.NDEF_DISCOVERED" />-->
            <!--<category android:name="android.intent.category.DEFAULT" />-->
            <!--<data android:mimeType="application/octet-stream" />-->
        <!--</intent-filter>-->

        <activity android:name=".MainActivity"
            android:noHistory="true"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name=".AttendanceActivity"
            android:theme="@style/AppTheme.NoActionBar"
```

```
51              android:screenOrientation="portrait">
52              <intent-filter>
53                  <action android:name="android.nfc.action.NDEF_DISCOVERED"
    />
54              </intent-filter>
55          </activity>
56      </application>
57
58  </manifest>
```

A.2.2 model/Attendance.java

```
1   package ba.unsa.etf.logit.model;
2
3   import org.json.JSONObject;
4
5   import java.text.DateFormat;
6   import java.text.SimpleDateFormat;
7   import java.util.Date;
8
9   import ba.unsa.etf.logit.LogitApplication;
10
11  public class Attendance {
12      // Raw JSON version of this object
13      public String raw;
14      // Attendee name
15      public String name;
16      public String surname;
17      // Attendee latitude
18      public String lat;
19      // Attendee longitude
20      public String lon;
21      // Attendee username - zamger
22      public String user;
23      // Attendee User ID - hex hash of public certificate key
24      public String uid;
25      // Hex Signature String of attendee package data (lat:lot:ts)
26      public String sig;
27      // Attendance ID - hex hash of signature
28      public String aid;
29      // Attendance TimeStamp from attendees device
30      public String ts;
31      // Is the signature valid check performed remotely by Logit Service on
    demand
32      public short valid;
33      // Master username - zamger
34      public String master;
35      // Master ID - hex hash of masters public certificate key
36      public String mid;
37      // Attendance Session ID
38      public String sid;
39      // Master Confirmation Signature - hex signature string of (sid:aid)
40      public String confsig;
41      // Confirmation ID - hex hash of confsig
42      public String cid;
43
```

```
44      public String getMid() {
45          return mid;
46      }
47
48      public void setMid(String mid) {
49          this.mid = mid;
50      }
51
52      public String getSid() {
53          return sid;
54      }
55
56      public void setSid(String sid) {
57          this.sid = sid;
58      }
59
60      public String getConfsig() {
61          return confsig;
62      }
63
64      public void setConfsig(String confsig) {
65          this.confsig = confsig;
66      }
67
68      public String getCid() {
69          return cid;
70      }
71
72      public void setCid(String cid) {
73          this.cid = cid;
74      }
75
76      public String getMaster() {
77          return master;
78      }
79
80      public void setMaster(String master) {
81          this.master = master;
82      }
83
84      public String getAid() {
85          return aid;
86      }
87
88      public void setAid(String aid) {
89          this.aid = aid;
90      }
91
92      public boolean isValidBasic() {
93          if (this.getRaw() != null &&
94                  this.getUser() != null &&
95                  this.getUid() != null &&
96                  this.getLat() != null &&
97                  this.getLon() != null &&
98                  this.getTs() != null &&
99                  this.getSig() != null) {
100             return true;
101         } else {
```

```
102                 return false;
103             }
104         }
105
106         public String getRaw() {
107             return raw;
108         }
109
110         public void setRaw(String raw) {
111             this.raw = raw;
112         }
113
114         public String getSurname() {
115             return surname;
116         }
117
118         public void setSurname(String surname) {
119             this.surname = surname;
120         }
121
122         public String getLat() {
123             return lat;
124         }
125
126         public void setLat(String lat) {
127             this.lat = lat;
128         }
129
130         public String getLon() {
131             return lon;
132         }
133
134         public void setLon(String lon) {
135             this.lon = lon;
136         }
137
138         public String getUser() {
139             return user;
140         }
141
142         public void setUser(String user) {
143             this.user = user;
144         }
145
146         public String getUid() {
147             return uid;
148         }
149
150         public void setUid(String uid) {
151             this.uid = uid;
152         }
153
154         public String getSig() {
155             return sig;
156         }
157
158         public void setSig(String sig) {
159             this.sig = sig;
```

```
160         }
161
162         public String getTs() {
163             return ts;
164         }
165
166         public void setTs(String ts) {
167             this.ts = ts;
168         }
169
170         public Date getDate() {
171             return new Date(Long.parseLong(this.ts) * 1000L);
172         }
173
174         public String getDateString() {
175             DateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy
    (HH:mm:ss)");
176             return dateFormat.format(this.getDate());
177         }
178
179         public Attendance(String raw) {
180
181             try {
182                 this.raw = raw;
183                 JSONObject jResult = new JSONObject(raw);
184
185                 this.name = new
    String(LogitApplication.fromHext(jResult.getString("name")), "UTF-8");
186                 this.surname = new
    String(LogitApplication.fromHext(jResult.getString("surname")), "UTF-8");
187                 this.lat = jResult.getString("lat");
188                 this.lon = jResult.getString("lon");
189                 this.user = jResult.getString("user");
190                 this.uid = jResult.getString("uid");
191                 this.sig = jResult.getString("sig");
192                 this.ts = jResult.getString("ts");
193                 this.valid = jResult.has("valid") ?
    (short)jResult.getInt("valid") : 0;
194             } catch (Exception e) {
195                 e.printStackTrace();
196             }
197         }
198
199         public short getValid() {
200             return valid;
201         }
202
203         public void setValid(short valid) {
204             this.valid = valid;
205         }
206
207         public String getMail() {
208             return this.user + "@etf.unsa.ba";
209         }
210
211         public String getName() {
212             return name;
213         }
```

```
214
215     public String getFullName() {
216         return surname + ", " + name;
217     }
218
219     public void setName(String name) {
220         this.name = name;
221     }
222
223     public String getSigPkg() {
224         return this.getUser() + ":" + this.getLat() + ":" + this.getLon() +
    ":" + this.getTs();
225     }
226
227     public String getConfSigPkg() {
228         return this.getMaster() + ":" + this.getSid() + ":" +
    this.getAid();
229     }
230 }
```

A.2.3 model/Place.java

```
1  package ba.unsa.etf.logit.model;
2
3  import java.util.List;
4  import com.google.gson.annotations.Expose;
5  import com.google.gson.annotations.SerializedName;
6
7  public class Place {
8
9      @SerializedName("place_id")
10     @Expose
11     private String placeId;
12     @SerializedName("licence")
13     @Expose
14     private String licence;
15     @SerializedName("osm_type")
16     @Expose
17     private String osmType;
18     @SerializedName("osm_id")
19     @Expose
20     private String osmId;
21     @SerializedName("lat")
22     @Expose
23     private String lat;
24     @SerializedName("lon")
25     @Expose
26     private String lon;
27     @SerializedName("display_name")
28     @Expose
29     private String displayName;
30     @SerializedName("boundingbox")
31     @Expose
32     private List<String> boundingbox = null;
33
34     public String getPlaceId() {
```

```
35          return placeId;
36      }
37
38      public void setPlaceId(String placeId) {
39          this.placeId = placeId;
40      }
41
42      public String getLicence() {
43          return licence;
44      }
45
46      public void setLicence(String licence) {
47          this.licence = licence;
48      }
49
50      public String getOsmType() {
51          return osmType;
52      }
53
54      public void setOsmType(String osmType) {
55          this.osmType = osmType;
56      }
57
58      public String getOsmId() {
59          return osmId;
60      }
61
62      public void setOsmId(String osmId) {
63          this.osmId = osmId;
64      }
65
66      public String getLat() {
67          return lat;
68      }
69
70      public void setLat(String lat) {
71          this.lat = lat;
72      }
73
74      public String getLon() {
75          return lon;
76      }
77
78      public void setLon(String lon) {
79          this.lon = lon;
80      }
81
82      public String getDisplayName() {
83          return displayName;
84      }
85
86      public void setDisplayName(String displayName) {
87          this.displayName = displayName;
88      }
89
90      public List<String> getBoundingbox() {
91          return boundingbox;
92      }
```

```
93
94      public void setBoundingbox(List<String> boundingbox) {
95          this.boundingbox = boundingbox;
96      }
97
98  }
```

A.2.4 model/Session.java

```
1   package ba.unsa.etf.logit.model;
2
3   import java.util.List;
4
5   public class Session {
6       public String sid;
7       public String sig;
8       public String mid;
9       public String master;
10      public List<Attendance> attns;
11
12      public String getMid() {
13          return mid;
14      }
15
16      public void setMid(String mid) {
17          this.mid = mid;
18      }
19
20      public String getMaster() {
21          return master;
22      }
23
24      public void setMaster(String master) {
25          this.master = master;
26      }
27
28      public String getSid() {
29          return sid;
30      }
31
32      public void setSid(String sid) {
33          this.sid = sid;
34      }
35
36      public String getSig() {
37          return sig;
38      }
39
40      public void setSig(String sig) {
41          this.sig = sig;
42      }
43
44      public List<Attendance> getAttns() {
45          return attns;
46      }
47
```

```
48      public void setAttns(List<Attendance> attns) {
49          this.attns = attns;
50      }
51  }
```

A.2.5 model/User.java

```
1   package ba.unsa.etf.logit.model;
2
3   public class User {
4       public String name;
5       public String surname;
6
7       public User(String name, String surname) {
8           this.name = name;
9           this.surname = surname;
10      }
11
12      public String getName() {
13          return name;
14      }
15
16      public void setName(String name) {
17          this.name = name;
18      }
19
20      public String getSurname() {
21          return surname;
22      }
23
24      public void setSurname(String surname) {
25          this.surname = surname;
26      }
27  }
```

A.2.6 api/LogitService.java

```
1   package ba.unsa.etf.logit.api;
2
3   import java.util.ArrayList;
4   import java.util.List;
5
6   import ba.unsa.etf.logit.model.Attendance;
7   import ba.unsa.etf.logit.model.Place;
8   import ba.unsa.etf.logit.model.Session;
9   import ba.unsa.etf.logit.model.User;
10  import okhttp3.OkHttpClient;
11  import okhttp3.ResponseBody;
12  import retrofit2.Call;
13  import retrofit2.http.Body;
14  import retrofit2.http.Field;
15  import retrofit2.http.FormUrlEncoded;
16  import retrofit2.http.GET;
17  import retrofit2.http.Headers;
```

```
18 import retrofit2.http.POST;
19 import retrofit2.http.Path;
20 import retrofit2.http.Query;
21
22 public interface LogitService {
23     @FormUrlEncoded
24     @POST("auth/")
25     Call<User> auth(@Field("user") String user, @Field("pass") String pass,
   @Field("cert") String cert, @Field("uid") String uid);
26
27     @POST("validate/")
28     Call<List<Attendance>> validate(@Body List<String> attns);
29
30     @POST("sync/")
31     Call<ResponseBody> sync(@Body Session session);
32
33     @Headers({
34             "User-Agent: ETF Logit v1.0b /SAPERE AVDE/",
35             "Referrer: http://etf.unsa.ba/"
36     })
37     @GET("reverse")
38     Call<Place> getAddress(@Query("email") String email, @Query("format")
   String format, @Query("lat") double lat, @Query("lon") double lon,
   @Query("zoom") int zoom, @Query("addressdetails") int addressdetails);
39 }
```

A.2.7 AttendanceActivity.java

```
1  package ba.unsa.etf.logit;
2
3  import android.content.SharedPreferences;
4  import android.content.pm.PackageManager;
5  import android.location.Location;
6  import android.nfc.NdefMessage;
7  import android.nfc.NdefRecord;
8  import android.nfc.Tag;
9  import android.nfc.tech.Ndef;
10 import android.os.AsyncTask;
11 import android.os.Build;
12 import android.os.Bundle;
13 import android.support.v4.content.ContextCompat;
14 import android.support.v7.app.AppCompatActivity;
15 import android.support.v7.widget.Toolbar;
16 import android.util.Log;
17 import android.app.Activity;
18 import android.app.PendingIntent;
19 import android.content.Intent;
20 import android.content.IntentFilter;
21 import android.content.IntentFilter.MalformedMimeTypeException;
22 import android.nfc.NfcAdapter;
23 import android.view.View;
24 import android.widget.ArrayAdapter;
25 import android.widget.ListView;
26 import android.widget.ProgressBar;
27 import android.widget.TextView;
28 import android.widget.Toast;
```

```
29
30 import com.google.android.gms.common.api.GoogleApiClient;
31 import com.google.android.gms.location.FusedLocationProviderClient;
32 import com.google.android.gms.location.LocationListener;
33 import com.google.android.gms.location.LocationRequest;
34 import com.google.android.gms.location.LocationServices;
35 import com.google.android.gms.tasks.OnSuccessListener;
36 import com.google.gson.Gson;
37
38 import org.json.JSONObject;
39 import org.w3c.dom.Text;
40
41 import java.security.KeyStore;
42 import java.security.MessageDigest;
43 import java.security.Signature;
44 import java.security.cert.Certificate;
45 import java.util.ArrayList;
46 import java.util.Arrays;
47 import java.util.Collections;
48 import java.util.Date;
49 import java.util.Enumeration;
50 import java.util.HashSet;
51 import java.util.List;
52 import java.util.Set;
53
54 import ba.unsa.etf.logit.api.LogitService;
55 import ba.unsa.etf.logit.model.Attendance;
56 import ba.unsa.etf.logit.model.Place;
57 import ba.unsa.etf.logit.model.Session;
58 import ba.unsa.etf.logit.model.User;
59 import okhttp3.OkHttpClient;
60 import okhttp3.ResponseBody;
61 import retrofit2.Call;
62 import retrofit2.Callback;
63 import retrofit2.Response;
64 import retrofit2.Retrofit;
65 import retrofit2.converter.gson.GsonConverterFactory;
66
67
68 public class AttendanceActivity extends AppCompatActivity implements
    GoogleApiClient.ConnectionCallbacks {
69
70     public static final String MIME = "application/octet-stream";
71     public static final String TAG = "Logit";
72     private AttendanceActivity that = this;
73
74     private ListView listview;
75     private List<Attendance> attns = new ArrayList<Attendance>();
76
77     private GoogleApiClient mGoogleApiClient;
78     private NfcAdapter mNfcAdapter;
79
80     @Override
81     protected void onCreate(Bundle savedInstanceState) {
82         super.onCreate(savedInstanceState);
83         setContentView(R.layout.activity_attendance);
84
85         Toolbar topToolbar = (Toolbar) findViewById(R.id.top_toolbar);
```

```
86          setSupportActionBar(topToolbar);
87          getSupportActionBar().setDisplayShowTitleEnabled(false);
88          SharedPreferences userData = getSharedPreferences("UserData", 0);
89
90          mGoogleApiClient = new GoogleApiClient.Builder(this)
91                  .addConnectionCallbacks(this).addApi(LocationServices.API)
92                  .build();
93          mGoogleApiClient.connect();
94
95          refreshLocation();
96
97          topToolbar.setTitle(userData.getString("surname", "Unknown") + ", "
    + userData.getString("name", "Unknown"));
98          topToolbar.setSubtitle(userData.getString("user", "unknown") +
    "@etf.unsa.ba");
99
100         listview = (ListView) findViewById(R.id.prisutni);
101
102         // Session ID control sequence
103         if(!userData.contains("sid")) {
104             SharedPreferences.Editor editor = userData.edit();
105             try {
106                 MessageDigest md = MessageDigest.getInstance("SHA-256");
107                 byte [] sidPayload = (userData.getString("user", "unknown")
    + System.currentTimeMillis()).getBytes();
108                 md.update(sidPayload, 0, sidPayload.length);
109                 editor.putString("sid",
    LogitApplication.toHext(md.digest()));
110                 editor.apply();
111             } catch (Exception e) {
112                 Toast.makeText(that, "Greška: neispravan CRYPT zahtjev.",
    Toast.LENGTH_LONG).show();
113             }
114         }
115
116         if(!userData.contains("attns")) {
117             SharedPreferences.Editor editor = userData.edit();
118             editor.putStringSet("attns", Collections.synchronizedSet(new
    HashSet<String>()));
119             editor.apply();
120         } else {
121             HashSet<String> attnSet = (HashSet<String>)
    userData.getStringSet("attns", Collections.synchronizedSet(new
    HashSet<String>()));
122             for (String s : attnSet) {
123                 attns.add(0, new Attendance(s));
124             }
125             if (!attns.isEmpty()) {
126                 Attendance[] attnsArray = (new Attendance[attns.size()]);
127                 attns.toArray(attnsArray);
128
129                 final ArrayAdapter adapter = new AttendanceAdapter(that,
    attnsArray);
130                 listview.setAdapter(adapter);
131             }
132         }
133
134         mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
```

```
135
136         if (mNfcAdapter == null) {
137             // Stop here, we need NFC
138             Toast.makeText(this, "This device doesn't support NFC.",
    Toast.LENGTH_LONG).show();
139             finish();
140             return;
141
142         }
143
144         if (!mNfcAdapter.isEnabled()) {
145             // @TODO
146         } else {
147
148         }
149     }
150
151     @Override
152     protected void onResume() {
153         super.onResume();
154
155         setupForegroundDispatch(this, mNfcAdapter);
156     }
157
158     @Override
159     protected void onPause() {
160         stopForegroundDispatch(this, mNfcAdapter);
161
162         super.onPause();
163     }
164
165     @Override
166     protected void onNewIntent(Intent intent) {
167         handleIntent(intent);
168     }
169
170     private void handleIntent(Intent intent) {
171         String action = intent.getAction();
172         if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {
173
174             String type = intent.getType();
175             if (MIME.equals(type)) {
176
177                 Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
178                 new NdefReaderTask().execute(tag);
179
180             } else {
181                 Log.d(TAG, "Wrong mime type: " + type);
182             }
183         } else if (NfcAdapter.ACTION_TECH_DISCOVERED.equals(action)) {
184
185             // In case we would still use the Tech Discovered Intent
186             Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
187             String[] techList = tag.getTechList();
188             String searchedTech = Ndef.class.getName();
189
190             for (String tech : techList) {
191                 if (searchedTech.equals(tech)) {
```

```
192                     new NdefReaderTask().execute(tag);
193                     break;
194                 }
195             }
196         }
197     }
198 
199     private class NdefReaderTask extends AsyncTask<Tag, Void, String> {
200 
201         @Override
202         protected String doInBackground(Tag... params) {
203             Tag tag = params[0];
204 
205             Ndef ndef = Ndef.get(tag);
206             if (ndef == null) {
207                 // NDEF is not supported by this Tag.
208                 return null;
209             }
210 
211             NdefMessage ndefMessage = ndef.getCachedNdefMessage();
212 
213             NdefRecord[] records = ndefMessage.getRecords();
214             for (NdefRecord ndefRecord : records) {
215                 if (ndefRecord.getTnf() == NdefRecord.TNF_MIME_MEDIA) {
216                     return readText(ndefRecord);
217                 }
218             }
219 
220             return null;
221         }
222 
223         private String readHext(NdefRecord record) {
224             byte[] data = record.getPayload();
225             return LogitApplication.toHext(data);
226         }
227 
228         private String readText(NdefRecord record) {
229             byte[] data = record.getPayload();
230             String ret;
231             try {
232                 ret = new String(data, "UTF-8");
233             } catch (Exception e) {
234                 ret = "Error";
235                 e.printStackTrace();
236             }
237             return ret;
238         }
239 
240         @Override
241         protected void onPostExecute(String result) {
242             if (result != null) {
243                 try {
244                     final Attendance tmpAttn = new Attendance(result);
245                     if (tmpAttn.isValidBasic()) {
246                         for (Attendance a : attns) {
247                             if (tmpAttn.getUid().equals(a.getUid()) ||
↳tmpAttn.getUser().equals(a.getUser())) {
248 //                              if (tmpAttn.getUid().equals(a.getUid())) {
```

```
249                                 Toast.makeText(that, "Student potpisan.",
    Toast.LENGTH_LONG).show();
250                                 return;
251                             }
252                         }
253                         final long timediff = System.currentTimeMillis() /
    1000 - Long.parseLong(tmpAttn.getTs());
254                         final Location userLocation = new Location("MOCK");
255
    userLocation.setLatitude(Double.valueOf(tmpAttn.getLat()));
256
    userLocation.setLongitude(Double.valueOf(tmpAttn.getLon()));
257                         if (Build.VERSION.SDK_INT >= 23
258                                 && ContextCompat.checkSelfPermission(that,
    android.Manifest.permission.ACCESS_FINE_LOCATION ) ==
    PackageManager.PERMISSION_GRANTED
259                                 && ContextCompat.checkSelfPermission(that,
    android.Manifest.permission.ACCESS_COARSE_LOCATION) ==
    PackageManager.PERMISSION_GRANTED
260                                 || Build.VERSION.SDK_INT < 23) {
261                             FusedLocationProviderClient
    mFusedLocatiionClient =
    LocationServices.getFusedLocationProviderClient(that);
262
    mFusedLocatiionClient.getLastLocation().addOnSuccessListener(new
    OnSuccessListener<Location>() {
263                                 @Override
264                                 public void onSuccess(Location location) {
265                                     Float locdiff =
    userLocation.distanceTo(location);
266  //                                   if (Math.abs(timediff) < 300) {
267                                     if (true) {
268  //                                       if (locdiff < 100) {
269                                         if (true) {
270                                             SharedPreferences userData =
    getSharedPreferences("UserData", 0);
271                                             SharedPreferences.Editor editor
    = userData.edit();
272                                             HashSet<String> attnSet = new
    HashSet<String>((HashSet<String>) userData.getStringSet("attns",
    Collections.synchronizedSet(new HashSet<String>())));
273                                             attnSet.add(tmpAttn.getRaw());
274                                             editor.putStringSet("attns",
    attnSet);
275                                             editor.apply();
276
277                                             attns.add(0, tmpAttn);
278
279                                             Attendance[] attnsArray = (new
    Attendance[attns.size()]);
280                                             attns.toArray(attnsArray);
281
282                                             final ArrayAdapter adapter =
    new AttendanceAdapter(that, attnsArray);
283                                             listview.setAdapter(adapter);
284                                         } else {
```

```
285                                                 Toast.makeText(that, "Greška: lokacije udaljene " + locdiff.intValue() + " metara.", Toast.LENGTH_LONG).show();
286                                             }
287                                         } else {
288                                             Toast.makeText(that, "Greška: vrijeme nije tačno ili je TAG zastario.", Toast.LENGTH_LONG).show();
289                                         }
290                                     }
291                                 });
292                             } else {
293                                 Toast.makeText(that, "Greška: lokacija nije dostupna.", Toast.LENGTH_LONG).show();
294                                 return;
295                             }
296                         } else {
297                             Toast.makeText(that, "Greška: TAG nije valjan.", Toast.LENGTH_LONG).show();
298                         }
299                     } catch (Exception e) {
300                         e.printStackTrace();
301                     }
302                 }
303             }
304         }
305 
306         public static void setupForegroundDispatch(final Activity activity, NfcAdapter adapter) {
307             final Intent intent = new Intent(activity.getApplicationContext(), activity.getClass());
308             intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
309 
310             final PendingIntent pendingIntent = PendingIntent.getActivity(activity.getApplicationContext(), 0, intent, 0);
311 
312             IntentFilter[] filters = new IntentFilter[1];
313             String[][] techList = new String[][]{};
314 
315             // Notice that this is the same filter as in the manifest
316             filters[0] = new IntentFilter();
317             filters[0].addAction(NfcAdapter.ACTION_NDEF_DISCOVERED);
318             filters[0].addCategory(Intent.CATEGORY_DEFAULT);
319 
320             try {
321                 filters[0].addDataType(MIME);
322             } catch (MalformedMimeTypeException e) {
323                 throw new RuntimeException("Check your mime type.");
324             }
325 
326             adapter.enableForegroundDispatch(activity, pendingIntent, filters, techList);
327         }
328 
329         public static void stopForegroundDispatch(final Activity activity, NfcAdapter adapter) {
330             adapter.disableForegroundDispatch(activity);
331         }
332 
```

```
333         @Override
334         public void onConnected(Bundle connectionHint) {
335             LocationRequest mLocationRequest = new LocationRequest();
336             mLocationRequest.setFastestInterval(10000);
337             mLocationRequest.setNumUpdates(3);
338             mLocationRequest.setSmallestDisplacement(1);
339 mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
340 
341             if (Build.VERSION.SDK_INT >= 23
342                     && ContextCompat.checkSelfPermission(this,
    android.Manifest.permission.ACCESS_FINE_LOCATION ) ==
    PackageManager.PERMISSION_GRANTED
343                     && ContextCompat.checkSelfPermission(this,
    android.Manifest.permission.ACCESS_COARSE_LOCATION) ==
    PackageManager.PERMISSION_GRANTED
344                     || Build.VERSION.SDK_INT < 23) {
345 
    LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
    mLocationRequest, new LocationListener() {
346                     @Override
347                     public void onLocationChanged(Location location) {
348                         Log.d("LOCATION",
    Double.toString(location.getLatitude()));
349                     }
350                 });
351             }
352         }
353 
354         @Override
355         public void onConnectionSuspended(int i) {
356 
357         }
358 
359         public void onSyncButton(View v) {
360             if (attns.size() > 0) {
361                 final ProgressBar validateProgress = (ProgressBar)
    findViewById(R.id.validate_progress);
362                 validateProgress.setVisibility(View.VISIBLE);
363                 final SharedPreferences userData =
    getSharedPreferences("UserData", 0);
364                 final SharedPreferences.Editor editor = userData.edit();
365 
366                 try {
367                     final MessageDigest md =
    MessageDigest.getInstance("SHA-256");
368                     byte[] confSig;
369 
370                     KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
371                     ks.load(null);
372                     KeyStore.ProtectionParameter pp = new
    KeyStore.PasswordProtection(null);
373 
374                     // Get the most recent master secure entry element
375                     Enumeration<String> aliases = ks.aliases();
376                     String alias = aliases.nextElement();
377                     KeyStore.Entry entry = ks.getEntry(alias, pp);
378 
```

```
379                     // Read in the master certificate
380                     Certificate c = ks.getCertificate(alias);
381
382                     // Instantiate a signature object and obtain the private
    key
383                     Signature s = Signature.getInstance("SHA256withRSA");
384                     s.initSign(((KeyStore.PrivateKeyEntry)
    entry).getPrivateKey());
385
386                     // Generate a hash for each attendance signature to be used
    as unique ID
387                     ArrayList<String> cidHashes = new ArrayList(attns.size());
388                     for (int i = 0; i < attns.size(); i++) {
389                         attns.get(i).setMaster(userData.getString("user",
    "unknown"));
390                         attns.get(i).setMid(userData.getString("uid",
    "unknown"));
391                         md.update(attns.get(i).getSig().getBytes());
392
    attns.get(i).setAid(LogitApplication.toHext(md.digest()));
393                         attns.get(i).setSid(userData.getString("sid",
    "unknown"));
394
395                         // Sign the confsig logit confirmation package
396                         s.update(attns.get(i).getConfSigPkg().getBytes());
397                         confSig = s.sign();
398
399
    attns.get(i).setConfsig(LogitApplication.toHext(confSig));
400                         md.update(confSig);
401
    attns.get(i).setCid(LogitApplication.toHext(md.digest()));
402                         cidHashes.add(attns.get(i).getCid());
403                     }
404                     Collections.sort(cidHashes);
405
406                     StringBuilder builder = new StringBuilder();
407                     for (String cidHash : cidHashes) {
408                         builder.append(cidHash);
409                     }
410                     String hashPackage = builder.toString();
411                     s.update(hashPackage.getBytes());
412                     byte [] rootSignature = s.sign();
413                     String rootHash = LogitApplication.toHext(rootSignature);
414
415                     Session session = new Session();
416                     session.setSid(userData.getString("sid", "unknown"));
417                     session.setSig(LogitApplication.toHext(rootSignature));
418                     session.setMid(userData.getString("uid", "unknown"));
419                     session.setMaster(userData.getString("user", "unknown"));
420                     session.setAttns(attns);
421
422                     Retrofit retrofit = new Retrofit.Builder()
423                             .baseUrl(LogitApplication.SERVICE_URL)
424                             .addConverterFactory(GsonConverterFactory.create())
425                             .build();
426                     final LogitService service =
    retrofit.create(LogitService.class);
```

```
427
428                     Call<ResponseBody> sync = service.sync(session);
429                     sync.enqueue(new Callback<ResponseBody>() {
430                         @Override
431                         public void onResponse(Call<ResponseBody> call,
    Response<ResponseBody> response) {
432                             if (response.code() == 201) {
433                                 // Reset the session ID and clear the previous
    attendances list
434                                 byte[] sidPayload = (userData.getString("user",
    "unknown") + System.currentTimeMillis()).getBytes();
435                                 md.update(sidPayload, 0, sidPayload.length);
436                                 editor.putString("sid",
    LogitApplication.toHext(md.digest()));
437                                 editor.putStringSet("attns",
    Collections.synchronizedSet(new HashSet<String>()));
438                                 editor.apply();
439                                 attns.clear();
440                                 listview.setAdapter(null);
441                                 Toast.makeText(that, "Podaci uspješno
    pohranjeni.", Toast.LENGTH_LONG).show();
442                             } else if (response.code() == 401) {
443                                 byte[] sidPayload = (userData.getString("user",
    "unknown") + System.currentTimeMillis()).getBytes();
444                                 md.update(sidPayload, 0, sidPayload.length);
445                                 editor.putString("sid",
    LogitApplication.toHext(md.digest()));
446                                 editor.putStringSet("attns",
    Collections.synchronizedSet(new HashSet<String>()));
447                                 editor.apply();
448                                 attns.clear();
449                                 listview.setAdapter(null);
450                                 Toast.makeText(that, "Greška: loš potpis
    sesije.", Toast.LENGTH_LONG).show();
451                             } else {
452                                 Toast.makeText(that, "Greška: neuspješan Logit
    zahtjev.", Toast.LENGTH_LONG).show();
453                             }
454                             validateProgress.setVisibility(View.INVISIBLE);
455                         }
456
457                         @Override
458                         public void onFailure(Call<ResponseBody> call,
    Throwable t) {
459                             Toast.makeText(that, "Greška: Logit servis
    nedostupan.", Toast.LENGTH_LONG).show();
460                             validateProgress.setVisibility(View.INVISIBLE);
461                         }
462                     });
463                 } catch (Exception e) {
464                     e.printStackTrace();
465                     Toast.makeText(that, "Greška: neispravan CRYPT zahtjev.",
    Toast.LENGTH_LONG).show();
466                 }
467             }
468         }
469
470         protected void refreshLocation() {
```

```
471         final ProgressBar validateProgress = (ProgressBar)
    findViewById(R.id.validate_progress);
472         validateProgress.setVisibility(View.VISIBLE);
473
474         FusedLocationProviderClient mFusedLocationClient =
    LocationServices.getFusedLocationProviderClient(this);
475         Retrofit retrofit = new Retrofit.Builder()
476                 .baseUrl("https://nominatim.openstreetmap.org/")
477                 .addConverterFactory(GsonConverterFactory.create())
478                 .build();
479         final LogitService service = retrofit.create(LogitService.class);
480
481         if (Build.VERSION.SDK_INT >= 23
482                 && ContextCompat.checkSelfPermission(this,
    android.Manifest.permission.ACCESS_FINE_LOCATION ) ==
    PackageManager.PERMISSION_GRANTED
483                 && ContextCompat.checkSelfPermission(this,
    android.Manifest.permission.ACCESS_COARSE_LOCATION) ==
    PackageManager.PERMISSION_GRANTED
484                 || Build.VERSION.SDK_INT < 23) {
485
486             mFusedLocationClient.getLastLocation().addOnSuccessListener(new
    OnSuccessListener<Location>() {
487                 @Override
488                 public void onSuccess(final Location location) {
489                     if (location != null) {
490                         Call<Place> validate =
    service.getAddress("mkoljenovic1@etf.unsa.ba", "json",
    location.getLatitude(), location.getLongitude(), 18, 0);
491                         validate.enqueue(new Callback<Place>() {
492                             @Override
493                             public void onResponse(Call<Place> call,
    Response<Place> response) {
494                                 if (response.code() == 200) {
495                                     Place p =  response.body();
496                                     String [] address =
    p.getDisplayName().split(", ");
497                                     TextView geoText = (TextView)
    findViewById(R.id.geoText);
498                                     if (address.length > 0) {
499                                         geoText.setText(address[0] + " (" +
    location.getLatitude() + ", " + location.getLongitude() + ")");
500                                     }
501                                     if (location.getTime() -
    System.currentTimeMillis() > 600000) {
502
    geoText.setBackgroundResource(android.R.color.holo_orange_light);
503                                     }
504                                 } else {
505                                     Toast.makeText(that, "Greška:
    neispravan OSM zahtjev.", Toast.LENGTH_LONG).show();
506                                 }
507
    validateProgress.setVisibility(View.INVISIBLE);
508                             }
509
510                             @Override
```

```
511                                 public void onFailure(Call<Place> call,
    Throwable t) {
512                                     Toast.makeText(that, "Greška: OSM servis
    nedostupan.", Toast.LENGTH_LONG).show();
513
    validateProgress.setVisibility(View.INVISIBLE);
514                                 }
515                             });
516                         } else {
517                             Toast.makeText(that, "Greška: lokacija nije
    dostupna.", Toast.LENGTH_LONG).show();
518                             validateProgress.setVisibility(View.INVISIBLE);
519                         }
520                     }
521                 });
522             }
523         }
524
525         public void onValidateButton(View v) {
526             final ProgressBar validateProgress = (ProgressBar)
    findViewById(R.id.validate_progress);
527             validateProgress.setVisibility(View.VISIBLE);
528             Retrofit retrofit = new Retrofit.Builder()
529                     .baseUrl(LogitApplication.SERVICE_URL)
530                     .addConverterFactory(GsonConverterFactory.create())
531                     .build();
532             LogitService service = retrofit.create(LogitService.class);
533             ArrayList<String> attnsRaw = new ArrayList<String>(attns.size());
534             for (Attendance attn : attns) {
535                 attnsRaw.add(attn.getRaw());
536             }
537             Call<List<Attendance>> validate = service.validate(attnsRaw);
538             validate.enqueue(new Callback<List<Attendance>>() {
539                 @Override
540                 public void onResponse(Call<List<Attendance>> call,
    Response<List<Attendance>> response) {
541                     if (response.code() == 200) {
542                         attns.clear();
543                         for(Attendance a : response.body()) {
544                             attns.add(a);
545                         }
546                         Attendance[] attnsArray = (new
    Attendance[attns.size()]);
547                         attns.toArray(attnsArray);
548
549                         final ArrayAdapter adapter = new
    AttendanceAdapter(that, attnsArray);
550                         listview.setAdapter(adapter);
551                     } else {
552                         Toast.makeText(that, "Greška: neispravan Logit
    zahtjev.", Toast.LENGTH_LONG).show();
553                     }
554                     validateProgress.setVisibility(View.INVISIBLE);
555                 }
556
557                 @Override
558                 public void onFailure(Call<List<Attendance>> call, Throwable t)
    {
```

```
559                     Log.d("validate", "error");
560                     Toast.makeText(that, "Greška: Logit servis nedosupan.",
    Toast.LENGTH_LONG).show();
561                     validateProgress.setVisibility(View.INVISIBLE);
562                 }
563             });
564         }
565
566         public void onBugButton(View v) {
567             final Intent emailIntent = new
    Intent(android.content.Intent.ACTION_SEND);
568
569             getSupportActionBar().setDisplayShowTitleEnabled(false);
570             SharedPreferences userData = getSharedPreferences("UserData", 0);
571
572             emailIntent.setType("plain/text");
573             emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL, new
    String[]{"mkoljenovic1@etf.unsa.ba"});
574             emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "Logit
    bug @" + userData.getString("user", "unknows") + ":" +
    userData.getString("uid", "unknown"));
575
576             this.startActivity(Intent.createChooser(emailIntent, "Prijavite
    grešku putem e-maila ..."));
577         }
578
579         public void onGeoButton(View v) {
580             refreshLocation();
581         }
582     }
```

A.2.8 AttendanceAdapter.java

```
1   package ba.unsa.etf.logit;
2
3   import android.content.Context;
4   import android.support.annotation.NonNull;
5   import android.support.annotation.Nullable;
6   import android.view.LayoutInflater;
7   import android.view.View;
8   import android.view.ViewGroup;
9   import android.widget.ArrayAdapter;
10  import android.widget.TextView;
11
12  import java.text.DateFormat;
13  import java.text.SimpleDateFormat;
14  import java.util.Date;
15
16  import ba.unsa.etf.logit.model.Attendance;
17
18  /**
19   * Created by koljenovic on 5/30/17.
20   */
21
22  public class AttendanceAdapter extends ArrayAdapter<Attendance> {
23      private final Context context;
```

```
24      private final Attendance[] values;
25
26      public AttendanceAdapter(Context context, Attendance[] values) {
27          super(context, R.layout.prisutni_row, values);
28
29          this.context = context;
30          this.values = values;
31      }
32
33      @NonNull
34      @Override
35      public View getView(int position, @Nullable View convertView, @NonNull
    ↪ViewGroup parent) {
36          LayoutInflater inflater = (LayoutInflater) context
37                  .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
38          View rowView = inflater.inflate(R.layout.prisutni_row, parent,
    ↪false);
39
40          TextView pName = (TextView) rowView.findViewById(R.id.pName);
41          TextView pMail = (TextView) rowView.findViewById(R.id.pMail);
42          TextView pDate = (TextView) rowView.findViewById(R.id.pDate);
43          TextView pSeq = (TextView) rowView.findViewById(R.id.pSeq);
44          TextView pMisc  = (TextView) rowView.findViewById(R.id.pMisc);
45
46          pName.setText(this.values[position].getFullName());
47          pMail.setText(this.values[position].getMail());
48
49          pDate.setText(this.values[position].getDateString());
50
51          pSeq.setText(Integer.toString(this.values.length - position));
52          short valid = this.values[position].getValid();
53          String msg = valid > 0 ? "Dobar potpis" : (valid < 0 ? "Loš potpis"
    ↪: "");
54          pMisc.setText(msg);
55          return rowView;
56      }
57  }
```

A.2.9 LogitApduService.java

```
1   package ba.unsa.etf.logit;
2
3   import java.io.UnsupportedEncodingException;
4   import java.security.KeyStore;
5   import java.security.KeyStore.Entry;
6   import java.security.KeyStore.PrivateKeyEntry;
7   import java.security.MessageDigest;
8   import java.security.PublicKey;
9   import java.security.Signature;
10  import java.security.cert.Certificate;
11  import java.security.interfaces.RSAPublicKey;
12  import java.util.Enumeration;
13
14  import android.content.SharedPreferences;
15  import android.content.pm.PackageManager;
16  import android.location.Location;
```

```
17 import android.nfc.NdefMessage;
18 import android.nfc.NdefRecord;
19 import android.nfc.cardemulation.HostApduService;
20 import android.os.Build;
21 import android.os.Bundle;
22 import android.support.v4.content.ContextCompat;
23 import android.util.Log;
24
25 import com.google.android.gms.location.FusedLocationProviderClient;
26 import com.google.android.gms.location.LocationServices;
27 import com.google.android.gms.tasks.OnSuccessListener;
28
29
30 public class LogitApduService extends HostApduService {
31
32     final static int APDU_INS = 1;
33     final static int APDU_P1 = 2;
34     final static int APDU_P2 = 3;
35     final static int APDU_SELECT_LC = 4;
36     final static int APDU_READ_LE = 4;
37     final static int FILEID_CC = 0xe103;
38     final static int FILEID_NDEF = 0xe104;
39     final static byte INS_SELECT = (byte) 0xa4;
40     final static byte INS_READ = (byte) 0xb0;
41     final static byte INS_UPDATE = (byte) 0xd6;
42     final static byte P1_SELECT_BY_NAME = (byte) 0x04;
43     final static byte P1_SELECT_BY_ID = (byte) 0x00;
44     final static int DATA_OFFSET = 5;
45
46     final static byte[] DATA_SELECT_NDEF = {(byte) 0xd2, (byte) 0x76,
↪ (byte) 0x00, (byte) 0x00, (byte) 0x85, (byte) 0x01, (byte) 0x01};
47     final static byte[] RET_COMPLETE = {(byte) 0x90, (byte) 0x00};
48     final static byte[] RET_NONDEF = {(byte) 0x6a, (byte) 0x82};
49     final static byte[] FILE_CC = {
50             (byte) 0x00, (byte) 0x0f,       // CCLEN - CC container size
51             (byte) 0x20,                    // Mapping version
52             (byte) 0x04, (byte) 0xff,       // MLe - max. read size
53             (byte) 0x08, (byte) 0xff,       // MLc - max. update size
54
55             // TLV Block (NDEF File Control)
56             (byte) 0x04,                    // Tag - Block type
57             (byte) 0x06,                    // Length
58             (byte) 0xe1, (byte) 0x04,       // File identifier
59             (byte) 0x04, (byte) 0xff,       // Max. NDEF file size
60             (byte) 0x00,                    // R permission
61             (byte) 0x00,                    // W permission
62     };
63     private final static String TAG = "LogitApduService";
64     private final static String ALL = "AllLogitApduService";
65     private CardSelect mCardSelect = CardSelect.SELECT_NONE;
66     private boolean mSelectNdef = false;
67     private byte[] mNdefFile = null;
68     private LogitApplication logitApp;
69     private FusedLocationProviderClient mFusedLocationClient;
70     protected String msg;
71
72     public LogitApduService() {
73         super();
```

```
74      }
75
76      private void generateSignature() {
77          try {
78              mFusedLocationClient =
    LocationServices.getFusedLocationProviderClient(this);
79
80              // App has to check if the user has granted an explicit
    permission to use the location
81              // This only applies to Android API level 23 and up
82              if (Build.VERSION.SDK_INT >= 23
83                      && ContextCompat.checkSelfPermission(this,
    android.Manifest.permission.ACCESS_FINE_LOCATION ) ==
    PackageManager.PERMISSION_GRANTED
84                      && ContextCompat.checkSelfPermission(this,
    android.Manifest.permission.ACCESS_COARSE_LOCATION) ==
    PackageManager.PERMISSION_GRANTED
85                      || Build.VERSION.SDK_INT < 23) {
86
87
    mFusedLocationClient.getLastLocation().addOnSuccessListener(new
    OnSuccessListener<Location>() {
88                      @Override
89                      public void onSuccess(Location location) {
90                          if (location != null) {
91                              try {
92                                  Long tsLong = System.currentTimeMillis() /
    1000;
93                                  String ts = tsLong.toString();
94                                  byte[] signature;
95
96                                  // Instantiate and load a Android KeyStore
    object
97                                  KeyStore ks =
    KeyStore.getInstance("AndroidKeyStore");
98                                  ks.load(null);
99                                  KeyStore.ProtectionParameter pp = new
    KeyStore.PasswordProtection(null);
100
101                                 // Get the most recent user secure entry
    element
102                                 Enumeration<String> aliases = ks.aliases();
103                                 String alias = aliases.nextElement();
104                                 Entry entry = ks.getEntry(alias, pp);
105
106                                 // Instantiate a digest object for hashing
107                                 MessageDigest md =
    MessageDigest.getInstance("SHA-256");
108
109                                 // Read in the user certificate
110                                 Certificate c = ks.getCertificate(alias);
111
112                                 // Generate public key hash as user
    identifier
113                                 byte [] pubKey =
    c.getPublicKey().getEncoded();
114                                 md.update(pubKey, 0, pubKey.length);
115                                 byte [] pubKeyHash = md.digest();
```

```
116                                 String pubKeyHashString =
    LogitApplication.toHext(pubKeyHash);
117
118                                 // Instantiate a signature object and
    obtain the private key
119                                 Signature s =
    Signature.getInstance("SHA256withRSA");
120                                 s.initSign(((PrivateKeyEntry)
    entry).getPrivateKey());
121
122                                 SharedPreferences userData =
    getSharedPreferences("UserData", 0);
123
124                                 // Prepare the logit data package to be
    signed
125                                 String sigPkg = userData.getString("user",
    "unknown") +
126                                         ":" + location.getLatitude() +
127                                         ":" + location.getLongitude() +
128                                         ":" + ts;
129
130                                 // Sign the logit data package
131                                 s.update(sigPkg.getBytes("UTF-8"));
132                                 signature = s.sign();
133
134                                 // Generate a tx package to be sent to
    master
135                                 msg = "{\"lat\":\"" +
    location.getLatitude() +
136                                         "\", \"lon\":\"" +
    location.getLongitude() +
137                                         "\", \"ts\":\"" + ts +
138                                         "\", \"sig\":\"" +
    LogitApplication.toHext(signature) +
139                                         "\", \"uid\":\"" + pubKeyHashString
    +
140                                         "\", \"name\":\"" +
    LogitApplication.toHext(userData.getString("name",
    "unknown").getBytes("UTF-8")) +
141                                         "\", \"surname\":\"" +
    LogitApplication.toHext(userData.getString("surname",
    "unknown").getBytes("UTF-8")) +
142                                         "\", \"user\":\"" +
    userData.getString("user", "unknown") + "\"}";
143
144                                 // Create a NDEF message from the tx
    package
145                                 NdefMessage ndef =
    createMessage(msg.getBytes("UTF-8"));
146                                 byte[] ndefarray = ndef.toByteArray();
147
148                                 // Prepare a NDEF file for HCE Tag
    emulation
149                                 mNdefFile = new byte[ndefarray.length + 2];
150
151                                 // Append length bytes as per NDEF NDFILE
    specification
```

```
152                                     mNdefFile[0] = (byte) ((ndefarray.length &
    0xff00) >> 8);
153                                     mNdefFile[1] = (byte) (ndefarray.length &
    0x00ff);
154
155                                     // Copy the NDEF message into the NDEF file
156                                     System.arraycopy(ndefarray, 0, mNdefFile,
    2, ndefarray.length);
157
158                                     logitApp.setMessage(mNdefFile);
159                                 } catch (Exception e) {
160                                     e.printStackTrace();
161                                 }
162                             }
163                         }
164                     });
165                 }
166             } catch (Exception e) {
167                 e.printStackTrace();
168             }
169         }
170
171         @Override
172         public void onDeactivated(int reason) {
173             Log.d(TAG, "onDeactivated");
174             mCardSelect = CardSelect.SELECT_NONE;
175             mSelectNdef = false;
176         }
177
178         protected byte [] prepareRetData(byte [] commandApdu) {
179             return prepareRetData(commandApdu, null);
180         }
181
182         protected byte [] prepareRetData(byte [] commandApdu, byte [] src) {
183             if (src == null) {
184                 Log.d(TAG, "return complete");
185                 return RET_COMPLETE;
186             }
187
188             int offset = ((commandApdu[APDU_P1] & 0xff) << 8) |
    (commandApdu[APDU_P2] & 0xff);
189             Log.d(TAG, "offset: " + Integer.toString(offset));
190             int Le = commandApdu[APDU_READ_LE] & 0xff;
191             byte [] retData = new byte[Le + RET_COMPLETE.length];
192
193             // Copy payload data into R-APDU
194             System.arraycopy(src, offset, retData, 0, Le);
195             // Add terminator to R-APDU
196             System.arraycopy(RET_COMPLETE, 0, retData, Le,
    RET_COMPLETE.length);
197
198             Log.d(TAG, "*****************************");
199             for (byte ch : retData) {
200                 Log.d(TAG, Integer.toHexString(ch & 0xff));
201             }
202             Log.d(TAG, "*****************************");
203
204             return retData;
```

```
205     }
206
207     @Override
208     public byte[] processCommandApdu(byte[] commandApdu, Bundle extras) {
209         for (int i = 0; i < commandApdu.length; i++) {
210             Log.d(ALL, Integer.toHexString(commandApdu[i] & 0xff));
211         }
212
213         byte [] retData = RET_NONDEF;
214
215         switch (commandApdu[APDU_INS]) {
216             case INS_SELECT:
217
218                 switch (commandApdu[APDU_P1]) {
219                     case P1_SELECT_BY_NAME:
220                         Log.d(TAG, "select : name");
221                         // 1. NDEF Tag Application Select
222                         if (memCmp(commandApdu, DATA_OFFSET,
    DATA_SELECT_NDEF, 0, commandApdu[APDU_SELECT_LC])) {
223                             //select NDEF application
224                             Log.d(TAG, "select NDEF application");
225                             mSelectNdef = true;
226                             retData = prepareRetData(commandApdu);
227                         } else {
228                             Log.e(TAG, "select: fail");
229                         }
230                         break;
231
232                     case P1_SELECT_BY_ID:
233                         Log.d(TAG, "select : id");
234                         if (mSelectNdef) {
235                             int file_id = 0;
236                             for (int loop = 0; loop <
    commandApdu[APDU_SELECT_LC]; loop++) {
237                                 file_id <<= 8;
238                                 file_id |= commandApdu[DATA_OFFSET + loop]
    & 0xff;
239                             }
240                             switch (file_id) {
241                                 case FILEID_CC:
242                                     Log.d(TAG, "select CC file");
243                                     mCardSelect = CardSelect.SELECT_CCFILE;
244                                     retData = prepareRetData(commandApdu);
245                                     break;
246
247                                 case FILEID_NDEF:
248                                     Log.d(TAG, "select NDEF file");
249                                     mCardSelect =
    CardSelect.SELECT_NDEFFILE;
250                                     retData = prepareRetData(commandApdu);
251                                     break;
252
253                                 default:
254                                     Log.e(TAG, "select: unknown file id : "
    + file_id);
255                                     break;
256                             }
257                         } else {
```

```
258                                 Log.e(TAG, "select: not select NDEF app");
259                             }
260                             break;
261
262                         default:
263                             Log.e(TAG, "select: unknown p1 : " +
↳commandApdu[APDU_P1]);
264                             break;
265                     }
266                     break;
267
268                 case INS_READ:
269                     Log.d(TAG, "read");
270                     if (mSelectNdef) {
271                         byte[] src = null;
272                         switch (mCardSelect) {
273                             case SELECT_CCFILE:
274                                 Log.d(TAG, "read cc file");
275                                 retData = prepareRetData(commandApdu, FILE_CC);
276                                 break;
277
278                             case SELECT_NDEFFILE:
279                                 Log.d(TAG, "read ndef file");
280                                 retData = prepareRetData(commandApdu,
↳logitApp.getMessage());
281                                 break;
282                         }
283                     } else {
284                         Log.e(TAG, "read: not select NDEF app");
285                     }
286                     break;
287
288                 case INS_UPDATE:
289                     Log.d(TAG, "UPDATE not implemented");
290
291                 default:
292                     Log.e(TAG, "unknown INS : " + commandApdu[APDU_INS]);
293                     break;
294             }
295
296             if (retData == RET_NONDEF) {
297                 Log.d(TAG, "ret notdef");
298             }
299
300             return retData;
301         }
302
303         private boolean memCmp(final byte[] p1, int offset1, final byte[] p2,
↳int offset2, int cmpLen) {
304             final int len = p1.length;
305             if ((len < offset1 + cmpLen) || (p2.length < offset2 + cmpLen)) {
306                 Log.d(TAG, "memCmp fail : " + offset1 + " : " + offset2 + " ("
↳+ cmpLen + ")");
307                 Log.d(TAG, "memCmp fail : " + len + " : " + p2.length);
308                 return false;
309             }
310
311             boolean ret = true;
```

```
312             for (int loop = 0; loop < cmpLen; loop++) {
313                 if (p1[offset1 + loop] != p2[offset2 + loop]) {
314                     Log.d(TAG, "unmatch");
315                     ret = false;
316                     break;
317                 }
318             }
319 
320             return ret;
321         }
322 
323 //https://github.com/bs-nfc/WriteRTDUri/blob/master/src/jp/co/brilliantservice/android/w
324         private NdefMessage createUriMessage(int index, String uriBody) {
325             try {
326                 byte[] uriBodyBytes = uriBody.getBytes("UTF-8");
327                 byte[] payload = new byte[1 + uriBody.length()];
328                 payload[0] = (byte) index;
329                 System.arraycopy(uriBodyBytes, 0, payload, 1,
uriBodyBytes.length);
330                 return new NdefMessage(new NdefRecord[]{
331                         new NdefRecord(NdefRecord.TNF_WELL_KNOWN,
NdefRecord.RTD_TEXT, new byte[0], payload)
332                 });
333             } catch (UnsupportedEncodingException e) {
334                 throw new RuntimeException(e);
335             }
336         }
337 
338         private NdefMessage createMessage(byte [] body) {
339             try {
340                 NdefRecord r0 =
NdefRecord.createMime("application/octet-stream", body);
341                 return new NdefMessage(r0);
342             } catch (Exception e) {
343                 e.printStackTrace();
344                 throw new RuntimeException(e);
345             }
346         }
347 
348         @Override
349         public void onCreate() {
350             super.onCreate();
351             logitApp = ((LogitApplication) this.getApplication());
352             generateSignature();
353         }
354 
355         enum CardSelect {
356             SELECT_NONE,
357             SELECT_CCFILE,
358             SELECT_NDEFFILE,
359         }
360     }
```

A.2.10 LogitApplication.java

```
package ba.unsa.etf.logit;

import android.app.Application;

public class LogitApplication extends Application {
    private byte[] message;
    public static final String SERVICE_URL = "https://logit.mine.nu:5000";

    public static String toHext(byte [] data) {
        StringBuilder buf = new StringBuilder();
        for (byte b : data) {
            int halfbyte = (b >>> 4) & 0x0F;
            int two_halfs = 0;
            do {
                buf.append((0 <= halfbyte) && (halfbyte <= 9) ? (char) ('0' + halfbyte) : (char) ('a' + (halfbyte - 10)));
                halfbyte = b & 0x0F;
            } while (two_halfs++ < 1);
        }
        return buf.toString();
    }

    public static byte[] fromHext(String sData) {
        int len = sData.length();
        byte[] data = new byte[len / 2];
        for (int i = 0; i < len; i += 2) {
            data[i / 2] = (byte) ((Character.digit(sData.charAt(i), 16) << 4)
                    + Character.digit(sData.charAt(i + 1), 16));
        }
        return data;
    }

    public void setMessage(byte[] message) {
        this.message = new byte[message.length];
        System.arraycopy(message, 0, this.message, 0, message.length);
    }

    public byte[] getMessage() {
        return this.message;
    }
}
```

A.2.11 MainActivity.java

```
package ba.unsa.etf.logit;

import android.Manifest;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.os.Build;
import android.security.KeyPairGeneratorSpec;
import android.support.v4.app.ActivityCompat;
```

```
10 import android.support.v4.content.ContextCompat;
11 import android.support.v7.app.AppCompatActivity;
12 import android.os.Bundle;
13 import android.util.Log;
14 import android.view.View;
15 import android.widget.ArrayAdapter;
16 import android.widget.EditText;
17 import android.widget.ListView;
18 import android.widget.ProgressBar;
19 import android.widget.Toast;
20
21 import java.math.BigInteger;
22 import java.security.KeyPair;
23 import java.security.KeyPairGenerator;
24 import java.security.KeyStore;
25 import java.security.MessageDigest;
26 import java.security.cert.Certificate;
27 import java.util.Calendar;
28 import java.util.Collections;
29 import java.util.Enumeration;
30 import java.util.List;
31
32 import javax.security.auth.x500.X500Principal;
33
34 import ba.unsa.etf.logit.api.LogitService;
35 import ba.unsa.etf.logit.model.User;
36 import retrofit2.Call;
37 import retrofit2.Callback;
38 import retrofit2.Response;
39 import retrofit2.Retrofit;
40 import retrofit2.converter.gson.GsonConverterFactory;
41
42 public class MainActivity extends AppCompatActivity {
43
44     @Override
45     protected void onCreate(Bundle savedInstanceState) {
46         super.onCreate(savedInstanceState);
47         setContentView(R.layout.activity_main);
48         try {
49             if (Build.VERSION.SDK_INT >= 23 &&
50                     ContextCompat.checkSelfPermission(this,
    android.Manifest.permission.ACCESS_FINE_LOCATION ) !=
    PackageManager.PERMISSION_GRANTED ) {
51                 ActivityCompat.requestPermissions(this, new String[] {
    Manifest.permission.ACCESS_FINE_LOCATION }, 1337);
52             }
53         } catch (Exception e) {
54             e.printStackTrace();
55         }
56
57         SharedPreferences userData = getSharedPreferences("UserData", 0);
58
59         if(userData.contains("uid")) {
60             Intent attnIntent = new Intent(this, AttendanceActivity.class);
61             startActivity(attnIntent);
62         }
63     }
64
```

```
65      public void onNewKeyButton(View v) {
66          final ProgressBar progressBar = (ProgressBar)
    findViewById(R.id.progressBar);
67          progressBar.setVisibility(View.VISIBLE);
68
69          Long tsLong = System.currentTimeMillis() / 1000;
70          String ts = tsLong.toString();
71          String certDer = null;
72          String pubKeyHashString = null;
73          final MainActivity that = this;
74
75          try {
76              KeyPairGenerator kpg = KeyPairGenerator.getInstance(
77                      "RSA", "AndroidKeyStore");
78              Calendar start = Calendar.getInstance();
79              Calendar end = Calendar.getInstance();
80              end.add(Calendar.YEAR, 1);
81
82              KeyPairGeneratorSpec spec =
83                      new
    KeyPairGeneratorSpec.Builder(this).setAlias("etf_logit_" + ts)
84                              .setKeySize(2048)
85                              .setSubject(new
    X500Principal("CN=users.etf.ba"))
86                              .setSerialNumber(BigInteger.valueOf(tsLong))
87
    .setStartDate(start.getTime()).setEndDate(end.getTime()).build();
88
89              kpg.initialize(spec);
90
91              // Ref: Android Security Internals: An In-Depth Guide to
    Android's Security Architecture By Nikolay Elenkov
92
93              KeyPair kp = kpg.generateKeyPair();
94
95              KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
96              ks.load(null);
97
98  //              List<String> aliasesList = Collections.list(aliases);
99  //              ListView existingCredList = (ListView)
    findViewById(R.id.existingCredList);
100 //              existingCredList.setAdapter(new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, aliasesList));
101
102             // Get the most recent user secure entry element
103             Enumeration<String> aliases = ks.aliases();
104             String alias = aliases.nextElement();
105             KeyStore.ProtectionParameter pp = new
    KeyStore.PasswordProtection(null);
106             KeyStore.Entry entry = ks.getEntry(alias, pp);
107
108             // Instantiate a digest object for hashing
109             MessageDigest md = MessageDigest.getInstance("SHA-256");
110
111             // Read in the user certificate
112             Certificate c = ks.getCertificate(alias);
113
114             // Generate public key hash as user identifier
```

```
115             byte [] pubKey = c.getPublicKey().getEncoded();
116             md.update(pubKey, 0, pubKey.length);
117             byte [] pubKeyHash = md.digest();
118             pubKeyHashString = LogitApplication.toHext(pubKeyHash);
119
120             certDer = LogitApplication.toHext(c.getEncoded());
121             Log.d("DER", certDer);
122         } catch (Exception e) {
123             Log.d("logit", Log.getStackTraceString(e));
124         }
125
126         Retrofit retrofit = new Retrofit.Builder()
127                 .baseUrl(LogitApplication.SERVICE_URL)
128                 .addConverterFactory(GsonConverterFactory.create())
129                 .build();
130
131         final EditText usernameBox = (EditText) findViewById(R.id.usernameBox);
132         final String usernameValue = usernameBox.getText().toString();
133         EditText passwordBox = (EditText) findViewById(R.id.passwordBox);
134         String passwordValue = passwordBox.getText().toString();
135         final String uid = pubKeyHashString;
136
137         LogitService service = retrofit.create(LogitService.class);
138         Call<User> auth = service.auth(usernameValue, passwordValue, certDer, uid);
139         auth.enqueue(new Callback<User>() {
140             @Override
141             public void onResponse(Call<User> call, Response<User> response) {
142                 if (response.code() == 200) {
143                     User user = response.body();
144
145                     SharedPreferences userData = getSharedPreferences("UserData", 0);
146                     SharedPreferences.Editor editor = userData.edit();
147                     editor.putString("uid", uid);
148                     editor.putString("user", usernameValue);
149                     editor.putString("name", user.getName());
150                     editor.putString("surname", user.getSurname());
151                     editor.apply();
152
153                     Intent attnIntent = new Intent(that, AttendanceActivity.class);
154                     startActivity(attnIntent);
155                 } else {
156                     try {
157                         KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
158                         ks.load(null);
159                         Enumeration<String> aliases = ks.aliases();
160                         List<String> aliasesList = Collections.list(aliases);
161                         for (String a : aliasesList) {
162                             ks.deleteEntry(a);
163                         }
164                         Toast.makeText(that, "Došlo je do greške, pokušajte ponovo.", Toast.LENGTH_LONG).show();
```

```
165                         } catch (Exception e) {
166                             e.printStackTrace();
167                             progressBar.setVisibility(View.INVISIBLE);
168                         }
169                     }
170                     progressBar.setVisibility(View.INVISIBLE);
171                 }
172 
173                 @Override
174                 public void onFailure(Call<User> call, Throwable t) {
175                     Log.d("RESP", "err");
176                     Toast.makeText(that, "Došlo je do greške, pokušajte ponovo.", Toast.LENGTH_LONG).show();
177                 }
178             });
179         }
180 
181 }
```

Literatura

[1] NXP Semiconductors, "NTAG213/215/216 NFC Forum Type 2 Tag compliant IC with 144/504/888 bytes user memory", NXP Semiconductors, Tech. Rep. October, 2011, dostupno na: http://www.jp.nxp.com/documents/short{_}data{_}sheet/NTAG203{_}SDS.pdf

[2] Google, - https://developer.android.com/guide/topics/connectivity/nfc/hce, dostupno na: https://developer.android.com/guide/topics/connectivity/nfc/hce Pristupano: 2018-09-14. 2018.

[3] EU, "Uredba (EU) 2016/679 Europskog parlamenta i Vijeća od 27. travnja 2016. o zaštiti pojedinaca u vezi s obradom osobnih podataka i o slobodnom kretanju takvih podataka te o stavljanju izvan snage Direktive 95/46/EZ (Opća uredba o zaštiti podataka)", str. 88, dostupno na: http://data.europa.eu/eli/reg/2016/679/oj 2016.

[4] Searle, J. R., Willis, S. *et al.*, The construction of social reality. Simon and Schuster, 1995.

[5] Cahill, V., Gray, E., Seigneur, J. M., Jensen, C. D., Chen, Y., Shand, B., Dimmock, N., Twigg, A., Bacon, J., English, C., Wagealla, W., Terzis, S., Nixon, P., Di Marzo Serugendo, G., Bryce, C., Carbone, M., Krukow, K., Nielsen, M., "Using trust for secure collaboration in uncertain environments", IEEE Pervasive Computing, Vol. 2, No. 3, 2003, str. 52–61.

[6] ISO/IEC, "27001: 2013", International Organization for Standardization, Standard, 2013.

[7] Ferguson, N., Schneier, B., Kohno, T., Cryptography engineering: design principles and practical applications. John Wiley & Sons, 2011.

[8] Singh, S., The code book: the science of secrecy from ancient Egypt to quantum cryptography. Anchor, 2000.

[9] Katz, J., Menezes, A. J., Van Oorschot, P. C., Vanstone, S. A., Handbook of applied cryptography. CRC press, 1996.

[10] Merkle, R. C., "Secure communications over insecure channels", Communications of the ACM, Vol. 21, No. 4, 1978, str. 294–299.

[11] Diffie, W., Hellman, M., "New directions in cryptography", IEEE transactions on Information Theory, Vol. 22, No. 6, 1976, str. 644–654.

[12] Rivest, R. L., Shamir, A., Adleman, L., "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, Vol. 21, No. 2, 1978, str. 120–126.

[13] Buchmann, J. A., Karatsiolis, E., Wiesmaier, A., Introduction to public key infrastructures. Springer Science & Business Media, 2013.

[14] Assembly, U. G., "Universal declaration of human rights", UN General Assembly, 1948.

[15] , - https://wikileaks.org/, dostupno na: https://wikileaks.org/ Pristupano: 2019-02-08. 2016.

[16] Kaliski, B., "Twirl and rsa key size", 2003.

[17] "Android statistics dashboard", - https://developer.android.com/about/dashboards/, pristupano: 2017-09-01.

[18] "Nfc world - adoption statistics", - https://nfcworld.com/2014/02/12/327790/two-three-phones-come-nfc-2018/, pristupano: 2017-09-11. 2014.

[19] Khan, R., Zawoad, S., Haque, M. M., Hasan, R., "Who, when, and where? location proof assertion for mobile devices", in IFIP Annual Conference on Data and Applications Security and Privacy. Springer, 2014, str. 146-162.

[20] Sterling, B., Wild, L., Shaping things. The MIT Press, 2005.

[21] NFC Forum, "NFC Digital Protocol Technical Specification", NFC Forum, Standard, 2010, dostupno na: http://www.nfc-forum.org/specs/

[22] Igoe, T., Coleman, D., Jepson, B., Beginning NFC. O'Reilly, 2014.

[23] NFC Forum, "Type 2 Tag Operation Specification", NFC Forum, Standard, 2011, dostupno na: http://apps4android.org/nfc-specifications/NFCForum-TS-Type-4-Tag{_}2.0.pdf

[24] ISO/IEC, "ISO/IEC 14443-1:2018(en) 1-4 - Cards and security devices for personal identification", International Organization for Standardization, Geneva, CH, Standard, 2018.

[25] NFC Forum, "NFC Data Exchange Format (NDEF)", NFC Forum, Standard, 2006.

[26] Elenkov, N., "Accessing the embedded secure element in android 4.x", - https://nelenkov.blogspot.com/2012/08/accessing-embedded-secure-element-in.html, dostupno na: https://nelenkov.blogspot.com/2012/08/accessing-embedded-secure-element-in.html Pristupano: 2018-09-14. 2012.

[27] C. Paya, - https://randomoracle.wordpress.com/2014/08/12/fakeid-android-nfc-stack-and-google-wallet-part-i/, dostupno na: https://randomoracle.wordpress.com/2014/08/12/fakeid-android-nfc-stack-and-google-wallet-part-i/ Pristupano: 2018-09-14. 2014.

[28] Elenkov, N., Android Security Internals. no starch press, 2015.

[29] Google, - https://developer.android.com/guide/topics/location/strategies, dostupno na: https://developer.android.com/guide/topics/location/strategies Pristupano: 2018-09-14. 2018.

[30] OpenStreetMaps, - https://wiki.openstreetmap.org/wiki/Nominatim, dostupno na: https://wiki.openstreetmap.org/wiki/Nominatim Pristupano: 2018-09-14. 2018.

[31] Knuth, D. E., “Literate programming”, The Computer Journal, Vol. 27, No. 2, 1984, str. 97–111.

[32] Soon, T. J., “Qr code”, Synthesis Journal, Vol. 2008, 2008, str. 59–78.

[33] Maletsky, K., “RSA vs ECC Comparison for Embedded Systems”, Atmel Corporation, Tech. Rep., 2015.

[34] Cheneau, T., Laurent, M., Shen, S., Vanderveen, M., “Ecc public key and signature support in cryptographically generated addresses (cga) and in the secure neighbor discovery (send)”, 2009.

[35] Youssef Ojeil, - https://e2e.ti.com/support/wireless-connectivity/other-wireless/f/667/p/486317/1778299?pi320995=3, dostupno na: https://e2e.ti.com/support/wireless-connectivity/other-wireless/f/667/p/486317/1778299?pi320995=3 Pristupano: 2018-11-14. 2016.

[36] Davis, M. D., Game theory: A nontechnical introduction. Courier Corporation, 2012.

[37] Nash, J., “Two-person cooperative games”, Econometrica: Journal of the Econometric Society, 1953, str. 128–140.