UNIVERZITET U BEOGRADU

FAKULTET ORGANIZACIONIH NAUKA

ZAVRŠNI (MASTER) RAD

Kriptografske tehnike zaštite u mobilnim aplikacijama

Mentor: Prof. dr Dejan Simić
Student: Vojislav Ristivojević 2016/3079

Beograd, 2018.

Apstrakt

Predmet istraživanja u ovom radu su kriptografske tehnike zaštite u mobilnim aplikacijama. U prvom delu rada biće date opšte smernice o primeni kriptografskih tehnika zaštite podataka, kao i opis specifičnosti mobilnih aplikacija, sa posebnim osvrtom na Android operativni sistem. U drugom delu rada biće obrađeni izazovi i pretnje bezbednosti mobilnih aplikacija, uz navođenje adekvatnih kriptografskih tehnika zaštite kojima se one mogu eliminisati ili umanjiti. Posebna pažnja će biti posvećena zaštiti komunikacionih kanala primenom infrastrukture javnih ključeva. U praktičnom delu rada fokus će biti na mrežnom i aplikativnom sloju, odnosno biće kreirana i opisana Android aplikacija koja koristi dostupne kriptografske biblioteke radi postizanja bezbedne komunikacije sa serverom. Pored navedenog, u radu će biti dat opis testnog sistema i prikaz ostvarenih rezultata i zaključaka.

Ključne reči: kriptografija, Android, mobilne aplikacije, bezbednost

Abstract

The research subject of this paper are cryptographic protection techniques in mobile applications. In the first part of the article, general guidelines will be provided on the application of cryptographic data protection techniques, as well as a description of the specifics of mobile applications, with particular reference to the Android operating system. In the second part of the article, challenges and threats to the security of mobile applications will be addressed, along with the introduction of adequate cryptographic protection techniques that can eliminate or reduce those risks. Special attention will be paid to protecting communication channels using the Public Key Infrastructure. In the practical part of the work, the focus will be on the network and application layer, that is, an Android application that uses the available cryptographic libraries to achieve secure communication with the server will be described and created. In addition to this, the description of the testing system and presentation of the achieved results and conclusions will be given in the paper.

Keywords: cryptography, Android, mobile applications, security

Sadržaj

Lista slika i tabela

1. Uvod

1.1. Predmet istraživanja

Na samom početku je neophodno naglasiti da kriptografija sama po sebi ne može rešiti većinu bezbednosnih problema u savremenim informacionim tehnologijama, niti je ispravna upotreba kriptografskih tehnika zaštite dovoljan uslov da bi se neka aplikacija, odnosno podaci kojima operiše, smatrali bezbednim. Ipak, iako samo jedan od brojnih segmenata metodologije izrade kvalitetnih i bezbednih aplikacija, kriptografija je neizostavan činilac ovog procesa, te je stoga vredan posebne pažnje koju ćemo mu posvetiti u ovom radu.

Predmet istraživanja ovog rada su kriptografske tehnike zaštite u mobilnim aplikacijama, a koje će biti prikazane u kontekstu rasprostranjenih pretnji po bezbednost i privatnost korisničkih podataka. Implementacija adekvatnih kriptografskih mera zaštite u Android aplikacijama je pre svega pitanje izbegavanja uobičajenih grešaka i prečica, odnosno adekvatnog sprovođenja dobrih praksi, te se jasnim ukazivanjem na celishodnost ovakvog pristupa može postići znatno poboljšanje razvojnog procesa u bezbednosnom i funkcionalnom smislu, bez potrebe za iznalaženjem novih rešenja razvojem posebnih algoritama i protokola [18].

Korisnici mobilnih uređaja se susreću sa drugačijom vrstom rizika u odnosu na one prisutne pri upotrebi desktop računara. Osim očigledno uvećanog rizika od krađe ili gubljenja uređaja, ovi događaji, pored materijalne štete, nose mogućnost ugrožavanja privatnosti ili krađe bitnih podataka.

Greške u implementaciji kriptografskog softvera najčešće obezvređuju sve mogućnosti obezbeđenja podataka putem kriptografskih tehnika zaštite [45]. Analizom 269 prijavljenih ranjivosti putem CVE baze u periodu od januara 2011. godine do maja 2014. godine utvrđeno je da samo 17% odlazi na greške u samim kriptografskim bibliotekama, dok je preostalih 83% posledica loše implementacije u specifičnim aplikacijama, pre svega u delu koji se tiče zaštite komunikacionih kanala [29].

Postoje primeri aplikacija (Apple TLS implementacija, GnuTLS) u kojima je samo jedna pogrešno napisana linija koda rezultovala greškom koja je u potpunosti obesmislila implementirani sistem zaštite podataka, čime je bilo pogođeno više miliona korisnika na duži vremenski period. Tehnike za prevenciju ovakvih grešaka podrazumevaju testiranje, statičku analizu, formalnu potvrdu i redizajn aplikacionog interfejsa (API).

S obzirom da su greške u kriptografskim bibliotekama retke, te da je njihovo otklanjanje najčešće van mogućnosti inženjera angažovanih na izradi konkretne aplikacije koja ih koristi, rezultati mnogih studija ukazuju da se najznačajniji efekat povećanja bezbednosti mobilnih aplikacija može postići izbegavanjem i predupređivanjem grešaka i propusta na nivou implementacije kriptografskih tehnika zaštite komunikacionih kanala odnosno protokola.

1.2. Formulacija problema

Bezbednost podataka je višeslojna kategorija koja se previše često tumači samo sa stanovišta prava pristupa i privilegija postojećih korisnika i aplikacija. Sama materijalna priroda medija na kojima su podaci pohranjeni, neophodnost transporta podataka, te njihova nužna dostupnost pojedinim delovima operativnog sistema, čini mogućim brojne druge vektore napada.

Mogućnosti primene kriptografskih tehnika zaštite u mobilnim aplikacijama ograničene su brojnim faktorima (specifičan hardver, nestabilni mrežni uslovi, neophodnost maksimalnog komfora za korisnike itd.) Čak i kada se oni uzmu u obzir, gotove aplikacije često imaju brojne bezbednosne propuste koji ugrožavaju privatnost komunikacije korisnika [13]. Ovi problemi proističu delom iz kompleksnosti problematike, kratkih rokova, ali i ustanovljenih loših praksi u izradi.

Većina savremenih mobilnih aplikacija koristi veliki broj dinamičkih i interaktivnih komponenti te zahteva posebno planiranje i implementaciju bezbednosne komponente sistema. Autentifikacija i autorizacija korisnika sa jedne, uz neporecivu potvrdu identiteta samog sajta korisniku sa druge strane, praćena neophodnošću bezbednog međusobnog prenosa podataka, značajno komplikuju izradu i održavanje same aplikacije kao i konfiguraciju web servera.

Opšteprihvaćena tehnologija zaštite transporta podataka mobilnih aplikacija, kojom se odgovara na najveći broj postavljenih izazova, je SSL/TLS enkripcija (Secure Socket Layer), odnosno njena implementacija u vidu HTTPS protokola. Ovaj protokol je samo još jedan sloj dodat na već postojeći HTTP protokol, ali se njime, upotrebom kriptografije bazirane na algoritmu javnog i tajnog ključa, obezbeđuje privatnost i integritet komunikacije, kao i potvrda identiteta servera.

Najefikasniji i najrasprostranjeniji napad na aplikacije koje koriste ove tehnike zaštite podrazumeva narušeni model poverenja PKI infrastrukture, u smislu nenamenski iskorišćenog RootCA sertifikata

kome sistem implicitno veruje. Neki od ovih slučajeva su primoravanje korisnika na uvoz sertifikata u korporativnim mrežama, preinstalacija sertifikata bez znanja korisnika, navođenje korisnika koji ne raspolažu tehničkim znanjima na instalaciju istog, instalacija od strane zlonamernog softvera, krađa ili zloupotreba regularnih sertifikata, legalnu i nelegalnu prinudu CA tela na saradnju sa državnim organima itd.

Radi ilustracije neophodnosti adekvatne implementacije kriptografskih tehnika zaštite u mobilnim aplikacijama, prikazaćemo ponašanje aplikacije koja nije otporna na ovu vrstu napada. Napad koji ćemo prikazati predstavlja modifikaciju rasprostranjenog *Man In The Middle* koncepta prilagođenu HTTPS protokolu, koja je najzastupljeniji vid zaštite komunikacije Android aplikacija [10]. Iako sam protokol podržava određene tehnike koje mogu smanjiti mogućnost izvođenja ovakvog napada, zbog njihove relativne kompleksnosti kao i problema u kompatibilnosti sa serverskom stranom, one se retko primenjuju.

Napad nije ograničen na Android aplikacije, niti na specifičan backend sistem, već se na sličan način može kompromitovati bilo koja komunikacija putem HTTPS-a. Ipak, zbog preteranog oslanjanja isključivo na ovaj vid zaštite podataka u Android okruženju najpogodnije je prikazati ga baš tu.

Mnoge često korišćene aplikacije kao što su Facebook, Instagram i Twitter tek od nedavno imaju internu proveru validnosti serverskog sertifikata, dok pojedine aplikacije za e-banking i dalje prihvataju sertifikate za koje garantuje bilo koji od sistemski prihvaćenih CA. Takođe, ovaj napad je najefikasniji u kombinaciji ka drugim napadima, kojima se korisnik navodi da sam instalira sertifikat, čime otvara svoju komunikaciju napadaču koji se može nalaziti na bilo kojoj tački na putu do odredišnog servera.

Najefikasniji metod zaštite je implementacija svih adekvatnih mogućnosti HTTPS protokola u samoj aplikaciji, kao i dodavanje dodatnog sloja enkripcije saobraćaja proverenim kriptografskim metodama na aplikativnom nivou, a što ćemo prikazati na praktičnom primeru izradom aplikacije koja koristi ove metode zaštite. Biće kreirana i opisana Android aplikacija koja koristi dostupne kriptografske biblioteke za rad na mrežnom i aplikativnom sloju, kao i mogućnosti postojećih proverenih protokola radi postizanja bezbedne komunikacije sa serverom.

1.3. Svrha i ciljevi istraživanja

Cilj ovog rada je sistematizovan pregled pretnji bezbednosti mobilnih aplikacija i adekvatnih odgovora na njih primenom kriptografskih tehnika zaštite. Pored toga, svrha rada je da se na praktičnom primeru mobilne aplikacije pokaže da se pravilnom primenom postojećih biblioteka, odnosno algoritama, može ostvariti visok stepen zaštite mobilnih aplikacija.

Kriptografija i ostale metode zaštite privatnosti komunikacija dobile su neočekivanu medijsku pažnju u poslednjih nekoliko godina usled ubrzanog tehnološkog napretka, proliferacije multimedijalnih komunikacionih uređaja, ali i određenih društvenopolitičkih okolnosti. Navedena kvantitativna ekspanzija je za posledicu imala da su kvalitet i relevantnost dostupnih podataka o ovim nesvakodnevnim temama značajno opali, zbog čega postoji dodatna motivacija da se na precizan i autoritativan način doprinese pojašnjenju pojedinih aspekata ovih oblasti, a što bi se moglo smatrati društvenim ciljem ovog istraživanja.

Usled svog otvorenog karaktera, fleksibilnosti i rasprostranjenosti, Android je zahvalna platforma za eksperimentisanje sa različitim mogućnostima samog sistema kao i sa drugim kompatibilnim tehnologijama, uključujući i kriptografske tehnike zaštite. Takođe, s obzirom na primarnu namenu mobilnih uređaja, komunikaciju, implementacija kriptografije kao načina zaštite privatnosti ima najviše smisla baš u ovakvim uređajima.

Sistematizovana naučna deskripcija najčešćih pretnji po bezbednost podataka u mobilnim aplikacijama i adekvatnih odgovora na iste putem kriptografskih tehnika zaštite predstavljala bi dobru polazišnu osnovu za dalja istraživanja, ali i poslužila kao smernica pri praktičnoj implementaciji ovih tehnika. Eksperimentalna potvrda efikasnosti pojedinih tehnika zaštite, odnosno komparacija ponašanja običnih i ojačanih aplikacija u određenim scenarijima napada, ilustrovala bi polazne navode i ukazala na adekvatnost predloženih metodoloških postupaka u daljem izučavanju ove i srodnih tema.

1.4. Organizacija i metode istraživanja

U prvom delu rada ćemo navesti osnovne kriptografske tehnike zaštite kao i segmente procesa razvoja mobilnih aplikacija u kojima se najčešće primenjuju. Takođe ćemo, koristeći rezultate drugih naučnih istraživanja kao i opise relevantnih primera iz prakse, sastaviti listu najzastupljenijih pretnji po

bezbednost podataka u mobilnim aplikacijama. U te svrhe ćemo koristiti naučne metode analizu sadržaja dokumenata, naučnu deskripciju i komparaciju.

Predloženi teorijski koncept ćemo praktično proveriti korišćenjem metode eksperimenta, odnosno kreiraćemo android aplikaciju koja integriše navedene kriptografske tehnike zaštite, a zatim ćemo, u kontrolisanom testnom okruženju i primenom za to pogodnih alata, proveriti njeno ponašanje u uslovima aktivnog napada na bezbednost podataka. Početna pretpostavka je da kontrolišemo i serversku stranu, jer je razvoj bezbedne aplikacije koja konzumira zadati API sa predefinisanog servera veoma ograničen. S tim u vezi, konfigurisaćemo sve neophodne parametre web servera i kreirati minimalnu backend aplikaciju, a što će takođe biti adekvatno dokumentovano.

Istraživanje ćemo obaviti u više faza. Nakon upoznavanja sa stanjem u oblasti izučavanja, pristupićemo sistematizaciji teorijskih koncepata relevantnih za naš konkretan problem. Po utvrđivanju tehnika koje trebamo da primenimo kako bismo otklonili odabrane probleme, pristupićemo njihovoj implementaciji u konkretnoj aplikaciji. Na kraju ćemo testirati implementaciju i sistematizovati zaključke ujedno određujući pravce daljeg istraživanja i razvoja.

1.5. Opis dela sveta koji će biti izučavan

U ovom radu će biti izučavane kriptografske tehnike zaštite u mobilnim aplikacijama, konkretno onima koje putem mreže, tj. Interneta prenose izvesne podatke koje želimo da zaštitimo od uvida i/ili izmene od strane neovlašćenih lica. Ograničićemo se na Android sistem i mogućnosti enkripcije klijent/server komunikacije aplikacija korišćenjem infrastrukture javnih ključeva PKI putem HTTPS protokola, a takođe ćemo prikazati tehnike kojima se štite podaci u transportu na aplikativnom sloju, kao i podaci aplikacije pohranjeni na samom uređaju.

Posle uvodnog teorijskog dela koji će prikazati relevantne kriptografske koncepte, predstavićemo uporedni prikaz rasprostranjenih pretnji po bezbednost komunikacije i adekvatnih odgovora na iste na Android platformi. Detaljno ćemo opisati proces izrade aplikacije, odnosno primenjene tehnike, a konkretnu implementaciju ćemo testirati za to pogodnim alatima u eksperimentalnom, testnom okruženju.

2. Kriptografske tehnike zaštite

2.1. Vrste kriptografskih tehnika zaštite

2.1.1. Simetrična kriptografija

Najčešća asocijacija kada je reč o kriptografiji, je upravo simetrična kriptografija, odnosno upotreba simetričnih kriptosistema koji pretvaraju nešifrovan tekst (plaintext) u šifrovan (cipertext) i obrnuto pomoću određenog ključa korišćenjem specijalnog postupka (algoritma). Prema definiciji, simetrični kriptosistem se sastoji iz seta mogućih nešifrovanih tekstova, mogućih šifrata, mogućih ključeva kao i dve vrste funkcija za šifrovanje i dešifrovanje (algoritama) koje su međusobno inverzne [41].

Postoje mnogi simetrični kriptosistemi, od kojih su neki relevantni i danas, dok je većina starijih sistema ovog tipa prevaziđena pojavom savremenih računara. Pojedini istorijski primeri ilustruju bitne teorijske koncepte te im se u literaturi ukazuje dužna pažnja bez obzira na tehničku neadekvatnost za praktične primene. U ovom radu ćemo se zadržati na deskripciji osnovnih koncepata simetrične kriptografije, uz dodatna pojašnjenja na primeru AES algoritma.

Najjednostavniji oblik simetričnog kriptosistema su različite varijacije supstitutivnih i transpozicionih šifara. U njima se jedno slovo alfabeta zamenjuje nekim drugim slovom čime nastaje šifrovani tekst, odnosno šifrat. S obzirom da su ovakvi sistemi veoma ranjivi na napade koji koriste informacije o učestanosti pojavljivanja određenih slova, ubrzo su prošireni dodatnim mogućnostima koje pokušavaju da reše ovaj problem. Na primer, razvijeni su algoritmi koji često upotrebljavana slova zamenjuju sa više mogućih nasumičnih kandidata, ili je ovaj princip primenjen na sva slova početnog teksta. Ipak, sve varijacije zaštitnih mehanizama ovog tipa su danas prevaziđene i više se ne koriste u praksi. Izuzetak su tzv. OTP (one time pad) sistemi u kojima je dužina ključa jednaka početnom tekstu i gde se svaki ključ upotrebljava samo jednom. Iako pružaju apsolutnu, matematički dokazivu sigurnost, a sam algoritam se svodi na prostu operaciju sabiranja po modulu, zbog nepraktičnosti preraspodele ključeva i drugih manjkavosti (npr. nepostojanje kontrole integriteta poruke), njihova upotreba je veoma ograničena na specifične sisteme.

Osnovni način klasifikacije savremenih simetričnih kriptosistema je prema veličini jedinice koju taj sistem procesira, a koja može biti jedan ili više bitova, odnonso predstavljati blok bitova [41]. Osim ovoga, bitno je da li je kriptosistem implementiran tako da izlaz, osim od ulaznih parametara, zavisi i od nekog trenutnog stanja sistema. Upravo po ovim kriterijumima razlikujemo stream i block

sisteme. Treba imati u vidu da podela na blok i stream šifre nije stroga, jer je izmenom načina rada moguće pretvoriti pojedine blok sisteme u stream.

Kod blok šifara unutrašnje stanje ne postoji, već izlaz (ciphertext) zavisi samo od početnog teksta (plaintext) i tajnog ključa. Kod stream šifara ono je prisutno, a dodavanjem ove komponente teoretski se postiže naprednije i snažnije šifrovanje. Promena stanja stream sistema se može vršiti sinhrono i asinhrono. To znači da u sinhronom sistemu izlazna šifrovana jedinica ne zavisi od prethodno generisanog izlaza, dok kod asinhronog sistema, zavisi od nekih (ili svih) prethodnih šifrovanih jedinica. Drugi naziv za sinhroni stream sistem je aditivni (sabirajući), a za asinhroni je samosinhronizujući [41]. Njihove komparativne prednosti dolaze do izražaja pri konkretnim implementacijama gde je akcenat na brzini ili paralelizaciji izvršavanja, odnosno na bezbednosti šifrata.

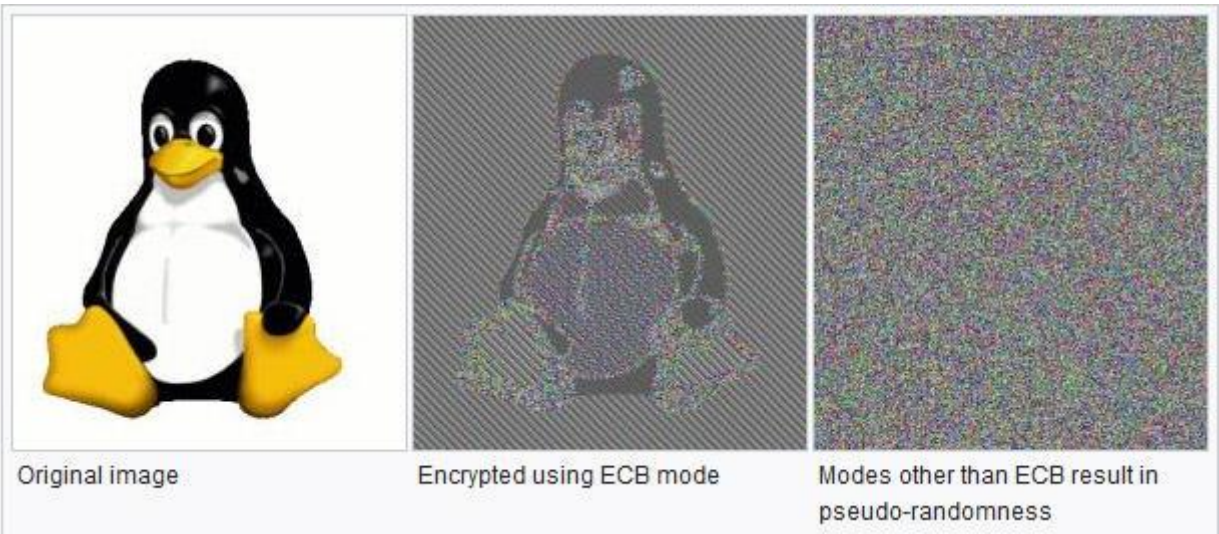
Krajnji cilj simetričnog kriptosistema je da sačuva tajnost početnog teksta. Da bi se to postiglo potrebno je da se pri projektovanju ispoštuju određeni uslovi za svaki od parametara. Simetrični sistemi zavise od tajnog ključa koji mora biti generisan, distribuiran, memorisan i upotrebljavan na bezbedan način, te je iz tih razloga poželjno da bude što kraći. Istovremeno, dužina ključa je, kod većine aktuelnih algoritama, upravo srazmerna sa otpornošću sistema na različite napade, zbog čega je neophodno naći kompromisno rešenje. Slično je i sa kompleksnošću samog algoritma: da bi se postigla efikasnost rada operacije šifrovanja i dešifrovanja ne smeju biti previše složene, dok sa druge strane od njih očekujemo pouzdanost i bezbednost.

Najpopularniji simetrični algoritam je Advanced Encryption Standard – **AES**, koji se može koristiti sa dužinama ključa od 128, 192 ili 256 bita, u različitim modovima rada. Proglašen je standardom 2000. godine, posle temeljnih testiranja i otvorenih konkursa gde su mnogi predloženi algoritmi bili podvrgnuti rigoroznoj kontroli i testiranju. On je zamenio dotadašnji američki (NIST) standardni algoritam Data Encryption Standard – DES, koji je od 1970. godine kada je proglašen, imao veoma široku primenu, ali čije su brojne slabosti došle do izražaja proliferacijom snažnih računarskih sistema i otkrićem novih kriptoanalitičkih tehnika.

Različiti načini rada istog algoritma mogu imati drastično drugačije karakteristike u smislu performansi i bezbednosti. Najjednostavniji način rada AES algoritma je elektronski šifarnik (Electronic Code Book - **ECB**). U ovom modu, početni tekst se deli na blokove veličine koju određuje korišćeni algoritam, dok se ostatak do celobrojnog umnoška neophodnog broja blokova popunjava nasumičnim ili posebno odabranim bitovima. Svaki blok se nezavisno šifruje tajnim

ključem, a na isti način se dešifruje na strani primaoca. Neke od prednosti ovog moda su jednostavnost, brzina i izostanak propagacije eventualnih grešaka u jednom bloku na ostale. Ipak, loše strane su daleko brojnije.

U ECB modu, isti početni tekst se mapira na isti šifrat, kada se šifruju jednakim ključem, a što za posledicu može imati uspešnost relativno jednostavne statističke kriptoanalize delova ili celog početnog teksta. Takođe, ne čuva se informacija o broju i redosledu blokova, te je moguće prerasporediti ih ili obrisati, čime se dešifrovana poruka menja, odnosno gubi njen integritet. Usled navedenih mana, ovaj mod se vrlo retko koristi u praksi.



Slika 1 Nedostaci ECB moda [64]

Način rada sa ulančavanjem blokova (Cipherblock chaining – **CBC**) je razvijen radi uklanjanja pojedinih nedostataka ECB-a. U ovom modu, šifrovani blok, osim od početnog teksta i ključa, zavisi i od svih prethodnih blokova, kao i inicijalizacionog vektora (IV), javnog podatka koji se koristi pri šifrovanju početnog bloka. Kao rezultat, čak i prilikom šifrovanja identičnog teksta istim ključem, ukoliko je IV različit, dobićemo drugačiji šifrat.

Osim što produžuje početnu poruku za jedan blok, mnogo bitniji nedostatak ovog moda je što njegovim korišćenjem može doći do propagacije grešaka u celoj poruci [41]. Ipak, on je najrasprostranjeniji i preporučeni način korišćenja AES i sličnih algoritama u praktičnim implementacijama.

Nešto ređe korišćeni modovi su cipher feedback (CFB) i output feedback (OFB). CFB koristi blok način rada kako bi pomoću ključa generisao pseudoslučajni niz bitova koji se nakon toga sabira (po

modulu 2) sa bitovima početnog teksta, dajući na taj način šifrat, praktično u stream modu odnosno bit po bit. Najčešće se koristi za prenos poruka kraćih od ključa, npr. po jednog slova komande. OFB funkcioniše na sličan način, s tim što se niz koji predstavlja ključ generiše nezavisno od prethodnih blokova, čime se izbegava propagacija grešaka i poboljšavaju performanse.

Jedan od savremenijih načina korišćenja AES protokola je **GCM** (Galois/Counter Mode). Ovaj mod rada je veoma popularan usled svoje efikasnosti, odnosno činjenice da je konstruisan na način da se njegove performanse dodatno uvećavaju sa napretkom arhitekture procesora na kojima se izvršava, u smislu korišćenja svih dostupnih mehanizama paralelizacije [17]. Koristi se pri enkripciji komunikacionih kanala visoke brzine odnosno širine protoka, a osim poverljivosti podataka (confidentiality) pruža garanciju autentičnosti i integriteta svakog paketa, s obzirom da implementira metode enkripcije sa autentifikacijom. Za sada postoji samo u varijanti koja radi sa 128 bitnim blokovima podataka.

Simetrična kriptografija i dalje ima veoma značajnu primenu, a pogotovo u specifičnim, uglavnom zatvorenim (npr. vojnim) sistemima, gde se pojedini nedostaci, kao što je neophodnost bezbedne manipulacije ključevima, rešavaju adekvatnim protokolima i pravima pristupa na nivou samih institucija koje ih koriste. U komercijalnim sistemima se najčešće upotrebljava u koordinaciji sa drugim kriptografskim mehanizmima.

2.1.2. Asimetrična kriptografija

Asimetrična kriptografija uglavnom podrazumeva kriptografske sisteme koji sadrže set tajnih parametara, koje učesnici u komunikaciji ne razmenjuju (tajni ključevi) i još jedan skup parametara koji mora biti dostupan svim učesnicima, a može biti i javno objavljen (javni ključevi). Potreban, ali ne i dovoljan uslov da ovakav sistem bude bezbedan je da je praktično neizvodljivo izračunati tajni ključ iz njemu odgovarajućeg javnog. Takođe postoje sistemi koji koriste parove ključeva gde oba ključa moraju biti tajni, kao što je Polig-Helmanov sistem [47], ali je njihova praktična primena veoma ograničena.

Koncept kriptografije zasnovane na postojanju **javnog ključa** izumeli su, 1976. godine, matematičari Vitfild Difi i Martin Helman, dok je do istih saznanja nezavisno došao i Ralf Merkle. Njihov doprinos se ogleda u izvođenju dokaza o mogućnosti postojanja parova ključeva takvih da se iz jednog ne može izvesti drugi, te da se mogu naizmenično koristiti za enkripciju i dekripciju poruka [47]. Potrebno je

napomenuti ograničenja, odnosno da ova teza važi u uslovima korišćenja poznatih matematičkih (kriptoanalitičkih) metoda, u razumnom roku, upotrebom dostupnih računarskih resursa, te da ne postoje dokazi o nekakvoj apsolutnoj sigurnosti ovog koncepta. Na ovim teorijskim temeljima predloženi su mnogi konkretni algoritmi, koji nisu podjednako bezbedni niti praktični za upotrebu, a njihovi problemi se uglavnom svode na prevelike ključeve i preterano uvećanje šifrata u odnosu na početni tekst.

Tek nekoliko algoritama su dovoljno bezbedni i praktični da bi nad njima bili izgrađeni široko upotrebljivi kriptosistemi. i ovde postoje ozbiljna ograničenja, jer su pojedini adekvatni samo za razmenu ključeva, potpisivanje, ili enkripciju male količine podataka. Samo su algoritmi RSA, ElGamal, i Rabin pogodni i za enkripciju i potpisivanje [47]. S obzirom da su sistemi bazirani na paru ključeva, u smislu neophodnih računarskih operacija, odnosno brzine izvršavanja, znatno manje efikasni od simetrične kriptografije, oni se uglavnom koriste za autentifikaciju i razmenu tajnog, sesijskog, ključa nekog simetričnog algoritma [41]. Kombinacijom ovih tehnika nastaju tzv. hibridni kriptosistemi, čija praktična upotreba je veoma rasprostranjena.

Usled svojih specifičnih svojstava, asimetrična kriptografija omogućava bezbednu komunikaciju većih grupa korisnika, pod uslovom da postoji sistem raspodele, odnosno infrastruktura javnih ključeva, a o čemu će biti više reči kasnije. Još jedna značajna prednost ovih sistema je činjenica da se parovi ključeva mogu koristiti više puta, odnosno u praktičnim terminima, više godina, bez bojazni da je na taj način narušena bezbednost komunikacije, a uz pogodnost olakšanog upravljanja razmenom i čuvanjem samih ključeva [36]. Najpopularniji algoritam koji se danas koristi u ove namene je RSA (imenovan prema naučnicima koji su ga osmislili a to su Ron Rivest, Adi Šamir i Leonard Adleman), a najmanja preporučena dužina ključa je 2048 bita.

Digitalni potpis je kriptografski postupak kojim se potvrđuje autentičnost neke digitalne poruke ili dokumenta. Iako postoje mnogi načini na koje se ova funkcionalnost može postići, najpraktičniji je onaj baziran na asimetričnom kriptografskom sistemu u kome postoji javni ključ. Ovaj postupak je obrnut u odnosu na šifrovanje poruke, odnosno, poruka za čiju autentičnost se pruža garancija biva šifrovana tajnim ključem pošiljaoca, te je jedino korišćenjem javno dostupnog ključa istog entiteta moguće izvršiti dešifrovanje, čime se potvrđuje njegovo autorstvo [45].

Usled nepraktičnosti šifrovanja cele poruke, uobičajeno je da šifruje samo rezultat jednosmerne kriptografski jake funkcije sa fiksnom veličinom izlaza, kojoj je početna poruka prosleđena kao ulazni parametar. Ovaj rezultat odnosno hash vrednost se sa još nekim metapodacima o samom dokumentu

šifruje tajnim ključem pošiljaoca i isporučuje uz dokument čija se autentičnost garantuje [45]. Za digitalno potpisivanje se najčešće koristi algoritam RSA, a ostali rasprostranjeni asimetrični algoritmi kao što su DSA i ECDSA se ne mogu koristi na ovakav način.

Eliptične krive su posebna vrsta funkcija koje raspolažu određenim svojstvima pogodnim za upotrebu u konstruisanju i eksploataciji savremenih asimetričnih kriptosistema. Bez ulaženja u matematička pojašnjenja, potrebno je naglasiti da je njihova upotreba rešila mnoge probleme vezane za postojeće asimetrične algoritme, pre svega u smislu dužine ključa, brzine kriptografskih operacija i otpornosti na pojedine kriptoanalitičke tehnike [47]. Takođe, bilo ih je moguće integrisati u postojeće asimetrične algoritme kao što su Difie-Helman i ElGamal, čime su protokoli koji ih implementiraju značajno poboljšani.

Iako zasnovane na strogo dokazivim matematičkim pravilima, koje garantuju bezbednost pod određenim uslovima, postoje sumnje da su korišćeni specifični, široj javnosti nepoznati, parametri pri konstrukciji eliptičnih krivih koje se nalaze u praktičnoj upotrebi, odnosno koje su integrisane u rasprostranjene bezbedne mrežne protokole [45]. Usled navedenog, radi se na proširenju budućih iteracija SSL-TLS protokola, na način da ove funkcije budu izmenjive nekim adekvatno proverenim odnosno konstruisanim na bezbedan način.

2.1.3. Ostale kriptografske tehnike

Jednosmerne (**hash**) funkcije su algoritmi koji pretvaraju ulaz bilo koje dužine u izlaz koji je uvek iste odnosno fiksne, unapred poznate dužine. One se veoma često koriste u matematici i programiranju, pre svega radi brzog i pouzdanog poređenja sadržaja velikih fajlova, ali nisu sve jednako pogodne za upotrebu u kriptografiji [45]. Za te namene neophodno je da raspolažu nekim posebnim svojstvima, kao što je nemogućnost konstruisanja poruke koja odgovara zadatoj izlaznoj vrednosti (u razumnom roku, sa postojećim računarskim resursima), nemogućnost pronalaženja ulaza koji bi davao isti izlaz kao neki već postojeći par (takođe pod navedenim uslovima), nemogućnost pronalaženja kolizija, odnosno ulaza sa identičnim izlazima.

Ponekad se rezultati ovih funkcija nazivaju otisak prsta (fingerprint) ili svedena vrednost (digest), što ukazuje na njihovu osnovnu namenu, potvrdu jedinstvenosti određene poruke ili fajla korišćenjem znatno manje količine podataka. Najčešće korišćeni algoritam za ove namene je SHA, a preporučena varijanta je SHA256.

Kodovi za potvrdu autentičnosti poruke (**Message Authentication Codes** – MAC) su kriptografske funkcije koje proširuju prethodno navedeni koncept hash-a dodajući mu funkcionalnost autentifikacije. Da bi se izbegao scenario u kome potencijalni napadač menja i originalnu poruku i hash koji bi trebao da garantuje njen integritet, neophodno je da primalac i pošiljalac raspolažu ključem koji se koristi za kreiranje MAC-a, čime se u ovaj metod potvrde autentičnosti i integriteta poruke uvode osnovni elementi kriptografije [45].

Bilo koja hash funkcija može poslužiti kao osnova za MAC, (odnonso HMAC), dok je sam postupak kojim se ključ kombinuje sa početnom porukom definisan tako da se ova operacija vrši na kriptografski bezbedan način. Primena ovih funkcija je široka, a neizostavan su deo infrastrukture javnih ključeva i protokola koji omogućavaju bezbedan transport podataka kao što su SSL/TLS, VPN i SSH.

Generatori slučajnih brojeva (RNG) su jedan od osnovnih elemenata svakog savremenog kriptosistema. Često je ključ koji se koristi u simetričnim sistemima samo jedan veliki nasumično generisani broj, dok se ovi generatori koriste i u drugim segmentima sistema gde se njihov izlaz podvrgava određenim testovima, odnosno traži zadovoljenje predefinisanih uslova. Problem sa njihovim generisanjem u računarskim sistemima leži u činjenici da su kompjuteri apsolutno determinističke mašine, odnosno iako je sam postupak obrade informacija veoma kompleksan, svaki izlaz je apsolutno zavisan od prethodnog te je stoga predvidiv [45].

Da bi se zaobišlo ovo ograničenje koristi se više mogućih pristupa. Kao najbolje rešenje, digitalizuje se ulaz spoljnog uređaja koji generiše nasumični signal posmatranjem nekog prirodnog fenomena. Npr. optički, zvučni, temperaturni, radio senzori mogu poslužiti u te namene, kao i nešto sofisticiraniji berilijumski moduli ili kvantni generatori koji beleže disipaciju radiološkog izvora odnosno kvantno stanje pojedinačnog fotona. S obzirom da su skupi i komplikovani za široku upotrebu, eksterni uređaji se često zamenjuju minimalnim nasumičnim ulazom, npr. promenom temperature procesora ili vremenskim razmakom unosa korisničkih podataka sa ulaznih uređaja [45]. Ovako dobijeni podaci predstavljaju ulaz posebne funkcije, kriptografskog generatora pseudoslučajnih brojeva (CPRNG), koji s obzirom da poseduju posebna matematička svojstva, proizvode izlaz adekvatan za upotrebu u drugim kriptografskim funkcijama.

Steganografija nije kriptografska tehnika u strogom smislu, to je metodologija kojom se poruke skrivaju unutar drugih poruka odnosno fajlova, na takav način da je samo njihovo prisustvo tajno, odnosno neprimetno [47]. Kao i mnoge druge kriptografske tehnike, razvojem računara i drugih

elektronskih sredstava komunikacije, doživela je značajne promene odnosno prilagođavanja aktuelnom trenutku. Digitalne fotografije i audio zapisi nude velike mogućnosti skrivanja drugih podataka, odnosno poruka unutar njih. Uzimajući u obzir nesavršenosti ljudskih receptora zvuka i slike, moguće je kodirati tajnu poruku izmenom pojedinih bitova (najmanje značajnih) na takav način da se kontejner poruka ne može razlikovati od originalne, dok skrivenu može pročitati samo onaj koji zna gde da je pronađe.

Steganografija ima značajne nedostatke u poređenju sa drugim kriptografskim tehnikama: neophodna je velika količina podataka da bi se sakrio relativno mali broj bitova informacije koju želimo da prikriveno prenesemo, a jednom kada je otkriven, sistem je praktično neupotrebljiv za buduće upotrebe. Sa druge strane, steganografiju je moguće kombinovati sa nekim drugim vrstama enkripcije čime se značajno šire mogućnosti upotrebe, dok je takođe ona veoma pogodan način komunikacije među stranama koje zahtevaju tajnost informacije da je do bilo kakve razmene poruka uopšte došlo [51]. Postoje eksperimentalni pokušaji primene steganografije u internet protokolima, kada se određeni bitovi zaglavlja paketa dinamički menjaju tako da prenose kodiranu poruku, ali oni nisu naišli na značajniju podršku IT zajednice ili industrije.

2.2. Oblasti primene kriptografskih tehnika zaštite

2.2.1. Zaštita statičnih podataka

Bezbednost podataka je višeslojna kategorija koja se previše često tumači samo sa stanovišta prava pristupa i privilegija postojećih korisnika i aplikacija. Sama materijalna priroda medija na kojima su podaci pohranjeni, te njihova nužna dostupnost pojedinim delovima operativnog sistema, čini mogućim brojne druge vektore napada. Takođe, eventualni pristup rezervnim kopijama podataka od strane neovlašćenih lica je veoma teško predupreditи poslovnom tj. bezbednosnom politikom kompanije koja sa tim podacima primarno radi.

Zaštita podataka u mirovanju (data at rest) se na najbolji način postiže kombinovanjem više zaštitinih mehanizama koji ojačavaju jedni druge. Na primer, ako su podešene adekvatne dozvole i prava pristupa na nivou fajl sistema, samo određeni korisnici odnosno aplikacije mogu pristupiti štićenim informacijama. S obzirom da je ipak moguće narušiti integritet većine operativnih sistema, u smislu zaobilaženja ove zaštite, odnosno neovlašćenog pribavljanja viših, administratorskih privilegija, neophodno je dodati dodatne slojeve zaštite, najčešće kriptografskog tipa [49]. Time se obezbeđuje

da čak i ako štićeni podaci dospeju do neovlašćenih lica, oni nisu u upotrebljivom obliku, čime se poštuju preporuke „dubinske“ odnosno slojevite odbrane (Defense in Depth) [1].

Osnovno pravilo da tehnike zaštite moraju biti prilagođene procenjenim (i prihvatljivim) rizicima, naročito dolazi do izražaja pri odabiru i implementaciji načina obezbeđivanja statičnih podataka. Šta više, ukoliko se radi o klijent/server ili mobilnoj aplikaciji, često uopšte nije neophodno čuvati tj. pohranjivati nikakve podatke na korisničkoj strani, te se ceo proces ovog nivoa zaštite može zanemariti, a izrada aplikacije pojednostaviti i ubrzati [49]. Ipak, praksa najčešće nalaže drugačije, te je adekvatan pristup određivanje najmanje moguće količine podataka koju treba štititi, kao i odgovarajućeg metoda kojim se to može postići, pri čemu se ne zanemaruju ostali zahtevi korisnika, pre svega performanse celog sistema.

Šifrovanje **pojedinačnih fajlova** se u zavisnosti od potreba korisnika može izvršiti na više načina, odnosno postoji veliki broj aplikacija koje nude isključivo tu funkcionalnost, ili se ona pruža kao dodatna opcija, ponekad uz doplatu odnosno izmenu licence. Kao tajni element se najčešće koristi lozinka, nešto duža fraza ali laka za pamćenje (passphrase) a u ređim slučajevima se ključ koristi neposredno, kada ga je neophodno čuvati na nekom posebnom uređaju. Neki operativni sistemi pružaju mogućnost enkripcije pojedinačnih fajlova ili direktorijuma, u kojim slučajevima se ključ najčešće izvodi iz lozinke koja se koristi za pristup sistemu. Iako relativno jednostavan za korišćenje, i efikasan u slučajevima kada napadač ima fizički pristup fajlovima (npr. krađa uređaja), ovakav sistem zaštite zahteva određenu disciplinu i znanje samog korisnika koji je u potpunosti odgovoran za čuvanje tajnosti šifre odnosno ključa. Takođe, preporučljivo je da on u svakom trenutku bude svestan eventualnih rezervnih tj. radnih kopija zaštićenih fajlova koje je softver za njihovu obradu ili operativni sistem eventualno napravio. Ukoliko se sistem proširi na više korisnika, npr. slanje šifrovanog fajla putem e-maila, nastaju problemi bezbednog distribuiranja tajnog elementa tj. lozinke, edukacije svih učesnika komunikacije, kao i kompatibilnosti softvera koji se koriste. Takođe, postoje velike varijacije u kvalitetu, odnosno snazi kriptografskih funkcija koje primenjuju pojedini programi, pogotovo oni kojima to nije primarna namena, a često su to aplikacije zatvorenog koda, te detaljna analiza i procena njihovih mogućnosti nije ni moguća.

Nešto naprednije mogućnosti šifrovanja statičnih fajlova nude sofisticirani softverski paketi koji omogućavaju kreiranje tzv. **kontejnera**, odnosno specijalnih fajlova koje oni operativnom sistemu prikazuju kao posebne „virtuelne“ uređaje za pohranjivanje podataka. Na ovaj način omogućava se potpuno transparentan rad sa zaštićenim fajlovima od strane bilo koje aplikacije i delimično rešava problem automatskog čuvanja rezervnih i radnih kopija. Takođe, većina aplikacija ovog tipa je usko

specijalizovana za primenu kriptografskih tehnika zaštite, odnosno koriste proverene algoritme i preduzimaju dodatne mere zaštite, u smislu minimalnog korišćenja virtuelne memorije i drugih delova sistema koji nisu adekvatno zaštićeni. Iako koriste poznate algoritme, postojeći softverski paketi ovog tipa uglavnom nisu međusobno kompatibilni, te uglavnom nije moguće kontejner napravljen u jednoj aplikaciji koristiti u nekoj drugoj istog tipa.

Šifrovanje **diskova i particija** se najčešće vrši putem posebnih aplikacija, ali postoje i hardverski uređaji, odnosno kontroleri, koji mogu obavljati istu funkciju. Ovaj način enkripcije pruža najveći komfor u radu, jer od korisnika zahteva minimalnu ili čak nikakvu dodatnu interakciju. Postoje različiti načini na koje se može implementirati, odnosno može se šifrovati ceo sistemski disk ili particija, neki poseban disk sa osetljivim podacima a ovaj metod se može kombinovati i sa drugim tehnikama, npr. podaci svakog posebnog korisnika mogu biti zaštićeni enkripcijom na nivou direktorijuma. Najveći nedostatak ovog tipa zaštite je pad performansi uređaja za pohranjivanje podataka, koji su ionako najčešće najsporiji deo savremenog računarskog sistema. Kriptografski koprocesori koji postoje u pojedinim procesorima ili se mogu dodati kao nadogradnja u većim sistemima značajno ubrzavaju rad sa velikom količinom podataka, ali istovremeno ograničavaju broj algoritama koji se mogu koristiti. Takođe, u slučaju tipičnih kvarova na uređajima za čuvanje podataka (loši sektori, fizička oštećenja, greške fajl sistema) spašavanje podataka postaje mnogo komplikovanije ukoliko je primenjena enkripcija. Pojedine aplikacije za šifrovanje celih diskova (kao npr. VeraCrypt) nude mogućnost pravljenja nevidljivih (steganografskih) particija, koje se mogu koristiti, odnosno čije prisustvo na sistemu se može nesumnjivo utvrditi, tek po unošenju specijalne lozinke, dok se pre toga ne mogu razlikovati od praznog prostora na šifrovanom disku.

Jedan od glavnih problema u oblasti bezbednosti **baza podataka** je činjenica da pored administratora i programera, pristup potencijalno osetljivim podacima imaju i osobe koje su odgovorne za kreiranje rezervnih kopija, kao i one koje su prisutne na lokaciji gde se ove kopije čuvaju. Takođe činjenica da se fajlovi koji sačinjavaju aktivnu bazu podataka fizički nalaze na uređajima koji možda nisu adekvatno obezbeđeni često obezvređuje sve protokole i procedure koje su primenjene radi regulisanja prava pristupa podacima od strane korisnika, odnonso aplikacija koje rade sa tom bazom. Transparentna enkripcija podataka (Transparent Data Encryption – TDE) predstavlja napredan metod zaštite osetljivih, odnosno poverljivih podataka i to na takav način da nije neophodna nikakva dodatna izmena u kodu same aplikacije kreirane nad bazom [40]. Uticaj enkripcije na performanse treba da bude minimalan kako po brzini pristupa, tako i po pitanju uvećanja fajlova. Ovim sistemom raspolažu mnogi savremeni sistemi za upravljanje bazama podataka, kao što su Oracle, MySQL i MSSQL i

slični, dok ovaj metod zaštite nije dostupan u jednostavnijim implementacijama koje se najčešće koriste u mobilnim uređajima.

Drugi načini na koje se može postići šifrovanje podataka u bazi su enkripcija unutar same aplikacije, koja vrši šifrovanje pre upisa i dekripciju nakon čitanja, a za pojedine standardizovane načine pristupa postoje i specifični softverski paketi koji se nalaze „između“ aplikacije i baze, te omogućavaju ovu funkcionalnost bez dodatnih intervencija na postojećem softveru [66].

2.2.2. Zaštita podataka u prenosu i komunikacionih kanala

Enkripcija „od tačke do tačke“ (point to point **P2P**) je pojam koji potiče od standarda uspostavljenog u industriji platnih kartica (PCI) kojim se propisuju određena svojstva kojima naplatni sistemi moraju raspolagati da bi njihova upotreba bila dozvoljena. Ipak, osnovni koncept P2P enkripcije je daleko širi i obuhvata mnoge rasprostranjene komunikacione protokole kojima se ostvaruje šifrovana veza između dva entiteta na transportnom sloju.

Shodno navedenom, ove tehnologije, odnosno protokoli se najviše koriste prilikom povezivanja manjeg broja lokacija ili korisnika [22]. Relativno su laki su za implementaciju i održavanje jer ne zahtevaju formiranje kompleksne mrežne infrastrukture i modela poverenja, kao što je slučaj sa PKI. Takođe, krajnje tačke ne moraju biti posebni uređaji već je terminisanje pojedinih P2P protokola moguće izvršiti na već postavljenim ruterima ili serverima, bez značajnijeg uticaja na performanse.

Koriste se pre svega za povezivanje dve mreže, npr. između sedišta i ispostave kompanije ili radnika na izmeštenoj lokaciji, dok skaliranje srazmerno većem broju ovakvih korisnika uglavnom predstavlja poteškoće.

Mnoge implementacije P2P protokola kao podrazumevani način rada koriste enkripciju putem prethodno raspodeljenih ključeva (pre-shared keys), u kojoj ne postoji tzv. savršena buduća bezbednost (perfect forward secrecy - PFS), pa sistem ostavlja mogućnost dešifrovanja svih prethodno poslatih poruka u slučaju kompromitacije ključa. Takođe, pojedini sistemi polaze od pretpostavke da je svaka krajnja tačka komunikacije potpuno bezbedna, te ne primenjuju dodatne mere zaštite ključeva već se oni čuvaju u otvorenom obliku (plaintext). Poznati problem poverenja pri prvoj upotrebi ovde se manifestuje kroz neophodnost kopiranja ključa na udaljenu tačku komunikacije. Iako navedeni nedostaci nisu prisutni kod svih P2P sistema, veoma su rasprostranjeni u gotovim uređajima koji korisnicima garantuju neku vrstu VPN usluge na jednostavan i jeftin način.

Čak je i softverski paket OpenVPN često konfigurisan na ovakav način, čime se radi jednostavnosti implementacije zanemaruju mnoge napredne kriptografske mogućnosti koje pruža.

Za razliku od P2P koncepta, kod enkripcije od kraja do kraja (end to end - **E2E**), podaci se šifruju pre nego što se uopšte ustupe transportnom sloju. Osnovna karakteristika ovakvih sistema je da se privatni, odnosno tajni ključ nikada ne poverava nekom spoljnom autoritetu, pa ni serveru koji se brine o ostalim parametrima komunikacije, već ostaje isključivo na korisničkim uređajima [25].

Ovaj sistem je razvijen iz potrebe da se posebno osetljive vrste poruka, kao što su e-mail, kratke poruke (chat i sms) pa i sami telefonski razgovori obezbede na način da njihova kompromitacija ne bude moguća ni na jednoj tački kroz koju podaci prolaze, uključujući tu i same operatere odnosno pružaoce telefonskih i internet usluga [62]. Šta više, ni u slučajevima kada su na to zakonom obavezani, operateri ne mogu ustupiti poruke svojih korisnika predstavnicima vlasti, jer ukoliko je sistem zaista E2E zaštićen, oni ne raspolažu neophodnim ključevima.

U praksi se međutim javljaju brojna ograničenja predloženog teorijskog koncepta. Najznačajnije je ono koje se tiče grupne komunikacije, kada bi svaka uključena strana morala da raspolaže ključevima svih ostalih korisnika. Pored kompleksnosti izrade robusne softverske arhitekture koja bi podržala ovaj sistem razmene ključeva, javlja se problem višestruke enkripcije iste poruke i linearnog povećanja neophodnog propusnog opsega veze [7]. Postoje veoma sofisticirani algoritmi, kao što je Signal protokol [6], a koji rešavaju većinu izazova veznih za pouzdanu E2E šifrovanu komunikaciju, dok cela koncepcija ostaje delimično pogođena već pomenutim problemom poverenja pri prvoj upotrebi, odnosno inicijalne razmene tajnih ključeva.

Bezbedni pristup komandnom okruženju, poznatiji kao **SSH** (Secure shell) je veoma popularan i moćan softverski paket iz oblasti bezbednosti mreža odnosno komunikacija. Šifrovanje komunikacije kod SSH protokola se obavlja transparentno, odnosno nakon inicijalnog uspostavljanja konekcije, gde je samo prvi put potrebno potvrditi identitet servera, sva ostala interakcija korisnika sa sistemom je neometana [4]. Identitet servera se potvrđuje tako što korisnik na klijentskoj strani prihvati i snimi za buduće provere hash vrednost predstavljenog javnog ključa servera (fingerprint). Ovaj korak može biti izveden bez dodatnih mehanizama zaštite, pri čemu predstavlja poverenje pri prvoj upotrebi, što je ujedno i najranjivija komponenta celog sistema. Alternativno, fingerprint servera se može dostaviti korisniku nekim drugim kanalom komunikacije koji se smatra bezbednim, pa on u tom slučaju može pouzdano da potvrdi identitet servera.

Klijentska strana se takođe može identifikovati serveru svojim javnim ključem, koji se pohranjuje na odgovarajuću lokaciju na serveru, a što se najčešće radi od strane samog korisnika nakon što je već uspostavio autentifikovanu sesiju [31]. I ovde se javlja sličan problem kao i u prethodnom slučaju, s tim što je potencijalnom napadaču dosta teže da ostvari prvobitni pristup korisničkom nalogu.

SSH koristi savremene algoritme i šifre, a u svojoj osnovnoj, besplatnoj varijanti (OpenSSH) je objavljen pod licencom otvorenog koda, te je konstantno podvrgnut detaljnoj analizi stručnjaka iz prakse. Ovaj paket je dostupan na većini arhitektura i operativnih sistema. Njegova klijent-server arhitektura omogućava veliku fleksibilnost primene, odnosno serverska strana osim uobičajenog komandnog okruženja, može klijentu isporučiti konekciju na servis za transport fajlova (SFTP), ili omogućiti naprednije metode preusmeravanja saobraćaja kroz šifrovane tunele.

Virtualne privatne mreže – **VPN** su jedan od prvih metoda obezbeđivanja privatnosti i bezbednosti komunikacija putem Interneta. Nastao je usled rastuće potrebe da se pojedine grupe korisnika izdvoje, odnosno zaštite od nekih drugih korisnika koji bi mogli zloupotrebiti njihove podatke, a da se pri tom ne izgubi komfor i fleksibilnost koji pruža korišćenje zajedničke globalne mreže [8]. VPN omogućava kreiranje zajedničke „lokalne“ mreže među više sistema koji se nalaze na različitim segmentima mreže. Oni mogu biti i na istoj fizičkoj mreži, ali i bilo gde drugde na Internetu, a veza između njih može biti uspostavljena putem različitih medija i protokola.

Kroz ovako uspostavljen „tunel“, kako se ova konekcija najčešće naziva, mogu se uspostaviti svi oblici komunikacije, odnosno protokoli kao i preko zaista lokalne mreže, uz obezbeđenu privatnost komunikacije. Najčešće se koristi kod bankomata, pri povezivanju na nezaštićene i nepoznate bežične mreže, kod komunikacije ispostava i sedišta kompanije, ali i radi zaobilaženja cenzure sadržaja prema geografskoj lokaciji.

Većina VPN implementacija koristi neku vrstu enkripcije i autentifikacije. Jasno je da enkripcija služi da prikrije sadržaj komunikacije od strana kojim nije namenjena, ali je neophodno napomenuti da se autentifikacija sastoji iz dve komponente [8]. Prva utvrđuje da je klijent koji se povezuje zaista ovlašćen da ostvari konekciju, a realizuje se putem sertifikata ili korisničkog imena i lozinke, pri čemu se svakom posebnom korisniku mogu dodeliti specifična pravila konfiguracije kojima će njegova konekcija biti adekvatno podešena. Druga komponenta autentifikacije predstavlja dodatni nivo zaštite komunikacionog kanala. U ovom slučaju, svaki paket koji se šalje biva potpisan sa jedne i proveren sa druge strane i to pre dešifrovanja, kako bi se onemogućilo nenamensko trošenje resursa

(Denial of Service - DoS) i pokušaji presretanja komunikacije od strane drugih članova iste VPN mreže.

Transparentno povezivanje dve mreže je istovremeno i najveća prednost i najveća mana ovog sistema, jer se na obe strane komunikacije zaobilaze uobičajeni načini usmeravanja i filtriranja saobraćaja, te klijenti i serveri mogu biti izloženi saobraćaju koji bi u normalinim okolnostima bio zaustavljen pre nego što do njih dođe. S obzirom da postoje mnogi komercijalni i besplatni pružaoci VPN usluge, korisnici često ne uviđaju da upotrebom ovih servisa polažu apsolutno poverenje u krajnju tačku mreže na koju se povezuju, te da njihova komunikacija u smislu anonimnosti i privatnosti, može zapravo biti ugroženija nego bez korišćenja neproverenih servisa ovog tipa.

Prvobitna verzija **SSL/TLS** protokola je kreirana u kompaniji Netscape, a ubrzo je postala široko prihvaćena od strane većine ostalih Internet pretraživača i servera. Svoju popularnost duguje prevashodno upotrebi u aplikacijama odnosno sajtovima za elektronsku trgovinu i bankarske usluge. Dovoljan nivo bezbednosti za ove namene SSL postiže unapređivanjem TCP protokola, odnosno implementacijom poverljivosti, integriteta i autentifikacije klijenta i servera koji obavljaju komunikaciju [28]. Najčešće se koristi u kombinaciji sa HTTP protokolom, ali je teoretski moguće iskoristiti njegove mogućnosti za ojačanje komunikacije bilo koje aplikacije koja koristi TCP. Iako je, strogo posmatran, SSL aplikativni protokol, čijim klasama i bibliotekama se pristupa pomoću odgovarajućeg interfejsa (Application Programmer Interface - API) on se u razvoju aplikacije najčešće tretira kao deo transportnog sloja.

Unapređenje u vidu kodifikacije RFC standarda je, osim novog naziva Transport Layer Security – TLS, donelo jasne specifikacije kojim sve protokolima mora raspolagati svaka implementacija, njihovu slojevitu povezanost i koje funkcije vrše [51]. Na najnižem nivou se nalazi SSL Record protokol koji omogućava rad višim protokolima koji zapravo implementiraju funkcionalnosti neophodne za uspostavljanje i održavanje bezbedne konekcije. To su protokol za uspostavljanje inicijalne konekcije (Handshake), za promenu metoda šifrovanja (Change Cipher Spec) i protokol za obaveštavanje (Alert).

Dva osnovna koncepta primenjena u SSL/TLS protokolu su sesija (session) i konekcija (connection). Konekcija je transportni kanal, u smislu kako ga definiše OSI model, odnosno ona omogućava pružanje određene usluge od jedne ka drugoj strani i obratno. Konekcije su tranzijentne i povezane su uvek samo sa jednom sesijom. Sesija je veza između klijenta i servera koja se kreira nakon uspešnog izvršenja handshake protokola. Sesija definiše set kriptografskih parametara koji će se

koristiti u komunikaciji klijenta i servera, a koji se mogu upotrebljavati u više odvojenih konekcija [51]. To je ujedno i njihova osnovna funkcija, jer je dogovor oko parametara i uspostavljanje bezbedne veze, računski i vremenski „najskuplji“ deo kreiranja svake konekcije. Uobičajeni slučaj je da se između dva entiteta koji komuniciraju putem SSL/TLS protokola kreira jedna sesija koja sadrži jednu ili više konekcija, dok se stvaranje više sesija između istih učesnika u komunikaciji retko praktikuje.

Neki sofisticirani načini bezbedne komunikacije razvijeni su i od strane pojedinaca i grupa koje imaju specifične potrebe i manjak poverenja u postojeće metode zaštite ili su im oni nedostupni iz raznih razloga. Najpoznatiji protokol ove vrste je **Tor** (The Onion Router). Tor štiti podatke tako što svaku konekciju preusmerava putem mreže relejnih servera koje održavaju pojedinci širom sveta, na dobrovoljnoj bazi. On onemogućava da eventualni napadač sazna koje sajtove posećuje korisnik, njegovu fizičku lokaciju, dok istovremeno omogućava pristup sajtovima kojima je pristup inače blokiran nekom bezbednosnom ili korporativnom kontrolnom merom.

Postoji u vidu softverskog paketa lakog za korišćenje na svim platformama, koji uz adekvatnu konfiguraciju može poslužiti i za bezbedno preusmeravanje saobraćaja bilo koje mrežne aplikacije, ne samo Internet pretraživača. Komunikacija ovim protokolom je veoma brza, raspolaže naprednim algoritmima za kontrolu integriteta i dostupnosti linkova, a sa kriptografske strane garantuje bezbednost prethodnih komunikacija u slučaju kompromitacije ključa (perfect forward secrecy) [11]. Njegova značajna prednost je što ne zahteva posebnu konfiguraciju sistema, minimalno opterećuje protok kontrolnim paketima (overhead) te pruža veoma dobar odnos anonimnosti, upotrebljivosti i efikasnosti.

U funkcionalnom smislu, Tor pri svakoj konekciji formira mrežu od više članova koji prosleđuju podatke dodajući po jedan sloj enkripcije, a gde svaki pojedinačni član ima podatke samo o prethodnom i sledećem čvorištu, ali ne i o celoj formiranoj mreži. Očigledna slaba tačka ove koncepcije su takozvane „izlazne tačke“ (exit nodes) koje se na odredištu pojavljuju kao konkretna ip adresa sa koje je došla komunikacija. U pravnom smislu, moguće je tretirati korisnike koji su omogućili izlazni saobraćaj kao saučesnike ili čak izvršioce eventualnog napada ili nedozvoljene radnje, dok u nešto složenijem, ali ne i sasvim nemogućem scenariju, strana koja želi „deanonimizaciju“ korisnika Tora može nadgledati sve izlazne tačke (njihova lista je javna) te pokušati korelaciju sa odlaznim saobraćajem ciljanog korisnika.

Projekat „nevidljivog Interneta“ (The Invisible Internet Project - **I2P**) nastao je kao pokušaj odgovora na neke od manjkavosti Tora i sličnih predloženih sistema. I2P omogućava pružanje i korišćenje anonimnih Internet servisa putem posebne skrivene mreže „unutar“ interneta („Darknet“). Usled svoje decentralizovane prirode, I2P pruža veći nivo zaštite od pojedinih sofisticiranih napada, a koncipiran je tako da svaki korisnik može podešavati parametre koji kontrolišu nivo bezbednosti, anonimnosti, količinu i brzinu protoka svake komunikacije.

I2P se sastoji od grupe virtualnih rutera, odnosno programa koji omogućavaju aplikacijama da komuniciraju putem mreže koju oni kreiraju. Osnovna karakteristika protokola je da nikada ne dolazi do direktne komunikacije dve tačke, već se ona ostvaruje kroz više posrednika, pri čemu je svaki učesnik mreže istovremeno i posrednik komunikacije za ostale. U komunikaciji se koristi veliki broj kriptografskih algoritama i protokola, uključujući SSL-TLS, simetričnu kriptografiju i razne jednosmerne (hash) funkcije, koji obezbeđuju pouzdanu end-to-end enkripciju. Podaci o svim mrežnim tačkama nalaze se u posebnoj bazi, odnosno distribuiranoj hash tabeli (DHT) koja se zove netDB, a koju osim svih korisnika poseduje i nekoliko kontrolnih servera [14]. Implementirana je na specifičan način te ova minimalna centralizacija ne utiče negativno na bezbednost ili anonimnost korisnika.

2.2.3. Autentifikacija

Autentifikacija u najširem smislu znači mogućnost potvrde da je određeni entitet zaista taj koji tvrdi da jeste, odnosno da nije došlo do manipulacije podacima o identitetu od strane nekog trećeg, neovlašćenog lica. Predstavlja jedan od najvažnijih segmenata bezbednosti informacionog sistema, a neophodna je u slučajevima kontrole pristupa, potvrde identiteta entiteta, potvrde identiteta pošiljaoca poruke, potvrde integriteta poruke, potvrde neporecivosti poruke i dokazivanja vlasništva ključeva.

Do sredine sedamdesetih godina XX veka, smatralo se da je autentifikacija nerazdvojna od tajnosti poruke, ali je razvojem hash funkcija i digitalnih potpisa zasnovanih na modelu asimetrične kriptografije sa javnim ključem zaključeno da ih treba posmatrati odvojeno iz teorijskih i praktičnih razloga [36]. Npr. nekada je neophodno istovremeno postići ciljeve transparentnosti komunikacije i neporecive potvrde identiteta sagovornika, što je moguće samo korišćenjem pojedinih delova asimetričnog kriptosistema, koji garantuju identitet i integritet, ali bez enkripcije samih poruka koje se razmenjuju.

Identifikacija, odnosno autentifikacija entiteta je postupak kojim se, nakon pružanja određenih dokaza, potvrđuje identitet jedne ili obe strane u trenutku ostvarene komunikacije. Najčešće se implementira odvojeno od ostalih faza komunikacije, odnosno dokaz o identitetu je jedini podatak koji se razmenjuje u toj fazi. U konkretnim protokolima se može implementirati na različite načine, a u zavisnosti od slučajeva korišćenja koje treba pokriti.

Npr. prilikom upotrebe bankomata, očitavaju se određeni podaci sa kartice dok korisnik unosi PIN kod. Tek nakon obrade ovih podataka, ukoliko je njihova provera potvrdna, korisniku se nudi mogućnost korišćenja nekog servisa koji uređaj pruža [36]. U ovom slučaju vrši se jedino autentifikacija odnosno identifikacija korisnika, dok server ne nudi podatke koji bi potvrdili njegov identitet.

Potvrda **porekla podataka** (data origin) odnonso poruke garantuje primaocu, pružanjem određenih dokaza, da je poruka zaista potekla od entiteta koji tvrdi da ju je poslao. Usled načina na koji se ova funkcionalnost implementira, najčešće je neodvojiva od garancije integriteta poruke, što je uglavnom poželjna propratna pojava. Primer upotrebe ove tehnike je potpisivanje elektronske pošte, odnosno provera validnosti potpisa i integriteta poruke od strane primaoca, korišćenjem prethodno poznatog (eventualno javno objavljenog) ključa za koji pouzdano zna da pripada pošiljaocu.

Proces autentifikacije je neophodan i u kriptosistemima zasnovanim na modelu javnog ključa. Osnovni problem koji se javlja u ovim sistemima je nemogućnost garancije identiteta sagovornika bez pribegavanja nekoj vrsti međusobne ili jednosmerne autentifikacije [36]. Celokupna infrastruktura javnih ključeva (PKI), o kojoj će više reči biti kasnije, je stvorena upravo da bi omogućila garanciju identiteta, odnosno potvrdila da prikazani ključ zaista pripada entitetu koji to tvrdi.

Autentifikacija se može vršiti na osnovu tri glavne kategorije:

1) nešto poznato, npr. lozinka, PIN kod, tajni ili privatni ključ (iz kojeg se izvodi informacija na osnovu koje se vrši autentifikacija)
2) nešto što je u posedu, npr. predmet kao što je magnetna ili čip kartica, uređaj - generator jednokratne lozinke, razne vrste periferijskih uređaja kao što su USB ili ranije RS232 tokeni.
3) nešto inherentno čoveku koji se autentifikuje, npr. fizička karakteristika ili nevoljna aktivnost kao što su rukopis, otisak prsta, uzorak glasa, izgled rožnjače, geometrija šake, kao i drugi biometrijski parametri.

Jedna od osnovnih oblasti primene autentifikacionih protokola je omogućavanje prava pristupa određenom resursu. To može biti računarski nalog, pristup bankomatu, komunikacioni kanal, specifična aplikacija ili fizički pristup određenoj zoni. Može se implementirati na različite načine.

Lozinka predstavlja najjednostavniji način utvrđivanja identiteta korisnika. Ona je uglavnom reč dužine 6-10 znakova koju korisnik može relativno lako upamtiti. S obzirom da je u pitanju tajni podatak koji znaju obe strane u komunikaciji, npr. korisnik i server, može se reći da se radi o vrsti simetričnog kriptosistema sa tajnim ključem, u kojem samo jedna strana (u ovom slučaju korisnik) ima obavezu dokazivanja svog identiteta odnosno autentičnosti, a što postiže pružanjem adekvatnog dokaza odnosno lozinke.

Lični identifikacioni broj (Personal identification number – **PIN**) je takođe jedna vrsta lozinke, a koristi se najčešće u kombinaciji sa nekim drugim faktorom autentifikacije, npr. magnetnom ili čip karticom. Da bi se u ovakvom sistemu dokazala autentičnost neophodno je priložiti oba dokaza, posedovani i poznati, čime se umanjuje pretnja od zloupotrebe pojedinačnog faktora [36]. Budući da je PIN, radi lakšeg pamćenja, uglavnom broj od 4 do 8 cifara, te da je teško onemogućiti njegovo pogađanje putem isprobavanja svih kombinacija tj. grube sile (brute force), najčešće se implementira uz neki dodatni zaštitni mehanizam koji ograničava broj pokušaja unosa.

S obzirom da su prilagođene lakom pamćenju, ni lozinka ni PIN ne raspolažu dovoljnom količinom entropije da bi same po sebi pružile adekvatnu bezbednost u kiptografskom smislu, usled čega su uvedene tehnike **dvoslojne autentifikacije**. Jedna od ovih tehnika koristi PIN kao identifikator korisnika ka određenom posredniku, npr. čip kartici, koja onda nakon potvrde verodostojnosti unetog podatka, generiše drugu, kriptografski jaku informaciju koja se prosleđuje sistemu kome korisnik želi da pristupi.

Druga rasprostranjena tehnika pretvara lozinku u ključ nekog proverenog algoritma (npr. AES256), korišćenjem posebne funkcije (Key derivation function – KDF). Ove funkcije, za razliku od većine ostalih koje se upotrebljavaju u kriptografiji, su ciljano komplikovane i spore, kako bi se otežalo pogađanje lozinke isprobavanjem svih mogućih kombinacija (brute force). Ključevi koji se generišu direktno iz lozinke se prvenstveno koriste za autentifikaciju i razmenu drugih ključeva, dok se njihova direktna upotreba za enkripciju korisničkih podataka ne preporučuje [36].

Autentifikacija odgovorom na izazov (**Challenge-response**) je kriptografski protokol u kome se kao dokaz identiteta pruža određena informacija bazirana na tajnom podatku koji poseduju oba entiteta, s tim da se sam taj tajni podatak nikada ne razmenjuje u komunikaciji. Ova funkcionalnost se može postići na više načina, npr. izazov može biti neki jedinstveni slučajni broj koji šalje server, a odgovor je šifrat tog broja od strane klijenta pomoću tajnog podatka (ključa) kojim obe strane raspolažu [36]. Vrlo je rasprostranjeno proširivanje ovog metoda korišćenjem infrastrukture javnih ključeva, kada se umesto zajednički poznatog tajnog podatka koriste javni ili tajni ključevi entiteta, a za koje adekvatni autoriteti garantuju ispravnost.

Autentifikacija sa enkripcijom povezanih podataka (Authenticated encryption with associated data - **AEAD**) je kombinacija šifrovanja i provere integriteta i autentičnosti poruke. Može se reći da ova tehnika sadrži svojstva i blok i stream šifara, uz neke specifičnosti. Ne koristi popunjavanje blokova podataka do neke predefinisane dužine (padding) ni inicijalizacione vektore (IV), ali upotrebljava poseban parametar koji mora biti jedinstven u svakoj komunikaciji (Number used once – nonce) [45]. Njegove specifične mogućnosti dolaze do izražaja u sofisticiranim slučajevima korišćenja kao što su mrežni protokoli, gde je neophodno da različiti delovi svakog paketa informacija budu tretirani na poseban način: zaglavlje mora biti čitljivo, autentično i zadržati integritet, dok sami podaci moraju očuvati integritet, autentičnost i poverljivost. Ovaj metod je veoma rasprostranjen unutar SSL-TLS protokola usled svoje brzine, pouzdanosti, relativno lake implementacije i otpornosti na pojedine kriptoanalitičke tehnike koje su pretile prethodno korišćenim standardima.

2.3. Preduslovi i ograničenja pojedinih tehnika zaštite

2.3.1. Zavisnost od spoljnog autoriteta

Osim za proveru autentičnosti servera, infrastruktura javnih ključeva (PKI) je od nedavno prihvaćeni model potvrde pouzdanosti aplikacija koju pri njihovoj instalaciji vrši operativni sistem, pre svega Windows). S obzirom da ova konkretnu implementacija ima brojne nedostatke, ona može poslužiti kao primer za loše posledice koje u PKI modelu proizvodi preterana zavisnost od spoljnog autoriteta. Implicitno poverenje u ovaj model prihvataju i brojni antivirusni softverski paketi, koji često ni ne vrše skeniranje tih aplikacija, ukoliko potvrde autentičnost digitalnog potpisa odnosno sertifikata koji je uz njih priložen [24]. Greške u implementaciji ovih kontrolnih mehanizama, ali i kompromitacija samog sistema poverenja na različite načine, predstavljaju značajan vektor napada koji se često koristi radi instalacije neželjenog softvera ili preuzimanja kontrole nad sistemima.

Kao što smo napomenuli, osnova problema leži u preteranom oslanjanju na digitalni potpis kao sredstvo potvrde apsolutne ispravnosti neke aplikacije. Iako Windows sistem upozorava i u podrazumevanoj konfiguraciji odbija da izvrši ili instalira nepotpisane aplikacije, verifikacija potpisanih je najčešće nepotpuna ili se može zaobići različitim mehanizmima. Jedan od najefikasnijih načina na koje se postiže potpisivanje zlonamernih aplikacija validnim sertifikatima je krađa ili zloupotreba tajnih ključeva sertifikacionih tela kojima sistem implicitno veruje.

Ovo je moguće jer ceo sistem poverenja zavisi od centralnog autoriteta (CA) koji proizvođačima softvera izdaje sertifikate koji potvrđuju njihov identitet, dok sa druge strane, korisnik jedino i proverava da li je softver koji želi da pokrene potpisan od strane navodnog proizvođača, koristeći za to sistemski preinstaliranu listu validnih centralnih autoriteta. Takođe, postoje tendencije proširenja inicijalnog poverenja ukazanog jednoj aplikaciji, na način da ona naknadno može vršiti izmene (npr. skidati unapređene verzije ili dodatke) bez dodatne provere verodostojnosti tih novih komponenti [24].

Iako se verovalo da je kompromitacija modela poverenja zasnovanog na centralnom autoritetu PKI više teorijski koncept, u praksi dostupan samo veoma motivisanim i moćnim napadačima, više primera je potvrdilo da to nije sasvim tačno. Crv „Stuxnet“ jeste bio potpisan korišćenjem validnih ključeva jedne tajvanske kompanije, ali se ispostavilo da je do njih mogao doći i bilo koji spretniji napadač, bez posebnih resursa i podrške. Virus „Flame“ je takođe imao validan potpis, ali je za njegovo kreiranje bila iskorišćena ranjivost (collision attack) MD5 algoritma koje je neko regularno CA telo i dalje koristilo u svojim sertifikatima [24]. Ipak, mnogi drugi primeri govore o trendovima zloupotreba koje je moguće detektovati.

Istraživanja [24] su pokazala da je prostim kopiranjem validnog potpisa neke aplikacije u drugu (zlonamernu) moguće navesti značajan broj antivirusnih programa da odustanu od skeniranja i prihvate aplikaciju kao validnu (ne uočivši pritom da se hash vrednost samog izvršnog fajla ne poklapa sa onom u potpisu). Takođe je utvrđeno da preko 30% zlonamernih aplikacija prilaže bilo kakav validan potpis da bi izazvalo upravo ovakvo ponašanje antivirusa.

S obzirom da se potpisivanje aplikacija često vrši od strane samih programera koji ih razvijaju, korišćenjem prethodno izdatih validnih posrednih (intermidiary) sertifikata, nije retka praksa da sofisticirani napadači ciljaju upravo njihove mašine, odnosno pokušavaju da ubace maliciozni kod pre postupka potpisivanja i distribucije legitimne aplikacije. Takođe, sofisticirani napadači mogu

registrovati softversku kompaniju i ukoliko ispune određene uslove, legalno dobiti pravo da potpisuju svoj softver, što će kasnije zloupotrebiti [27].

Ipak, najveću grupu zloupotreba PKI modela poverenja čine one koje podrazumevaju neku vrstu kompromitacije CA autoriteta, odnosno najveći broj zlonamernih aplikacija biva potpisan regularnim sertifikatima. Potrebno je napomenuti da je baza sertifikata od poverenja (trusted computing base – TCB) koja se koristi za validaciju potpisa aplikacija značajno veća od one koja se uobičajeno koristi za proveru serverskih sertifikata pri HTTPS komunikaciji, jer obuhvata brojne proizvođače softvera, operativnih sistema, sistemskih biblioteka, a često postoje i mehanizmi kojima se ona relativno lako može proširivati. U ovakvoj varijanti PKI infrastrukture, do posebnog izražaja dolaze ranije prepoznate slabosti ovog modela, kao što je mogućnost da bilo koji autoritet garantuje za bilo kog proizvođača, odnosno da postoje brojne tačke čijom kompromitacijom može biti narušen ceo sistem.

Konkretna istraživanja [27] su pokazala da se potpisivanje zlonamerne aplikacije validnim sertifikatom najčešće postiže na jedan od tri načina: krađom CA sertifikata, njihovom prodajom ili uslužnim potpisivanjem aplikacije. S obzirom da istraživanja ovog tipa prate brojne teškoće tehničke i pravne prirode, istraživači uglavnom iznose opšte, zbirne ocene i retko imenuju CA tela koja figuriraju kao izvori zlonamerno upotrebljenih sertifikata. Kao zaključak se navodi da je većina analiziranog zlonamernog softvera potpisana malim brojem visokokvalitetnih sertifikata (koji bivaju automatski validirani od strane odbrambenih alata u Windows-u), a da njima raspolaže mali broj sofisticiranih napadača koji usluge potpisivanja zlonamernih aplikacija prodaje na crnom tržištu.

2.3.2. Poverenje pri prvoj upotrebi – TOFU

Jedna od često upotrebljavanih metoda autentifikacije je takozvano „poverenje pri prvoj upotrebi“ (Trust on first use – Tofu). Iako se ne može govoriti o kodifikovanom protokolu, njegove varijante su veoma rasprostranjene, a ponekad i dovoljno efikasne tj. bezbedne, te adekvatno zamenjuju pouzdane kriptografske tehnike potvrde identiteta. Takođe, ovde je reč o specifičnoj implementaciji koncepta koji je više puta implicitno ili eksplicitno pominjan, a koji podrazumeva da je bar jedna, odnosno početna komunikacija između dve strane dovoljno bezbedna, te da je moguće ovo poverenje „proširiti“ i na ostatak te komunikacije, kao i na sve buduće komunikacije ta dva entiteta. Ovaj metod se ponekad naziva i upravljanje kontinuitetom validnosti ključeva (key continuity management) [45].

S obzirom da je orijentisan ka krajnjim korisnicima i veoma jednostavan za upotrebu, često se koristi kao alternativa složenijim protokolima pri korišćenju SSH konekcija i povezivanju na HTTPS koji poseduju samopotpisani (self-signed) sertifikat [58]. Prihvatanje potvrde da je server zaista taj za koga se izdaje, vrši se pri prvoj konekciji, kada se, u slučaju SSH protokola korisniku prikazuje hash vrednost (fingerprint) ključa, te se isti snima u lokalnu memoriju klijenta, dok se kod HTTPS-a od korisnika traži dozvola da se prihvati izuzetak (exception), odnosno prihvati priloženi sertifikat kao garanciju autentičnosti servera za tu i sve buduće konekcije.

Ova dva primera prikazuju da iako doprinosi uvećanju bezbednosti (za buduće konekcije), ovaj model poverenja može lako biti zloupotrebljen pri prvoj upotrebi, čime se kompromituju i sve buduće komunikacije, u smislu bezbednosti ili dostupnosti.

Takođe, ovi izuzeci se kreiraju pojedinačno za svaki sertifikat, što znači da će se upozorenje pojaviti svaki put kada se promeni sertifikat servera na koji smo prethodno odobrili povezivanje. U slučaju SSH protokola, ovo upozorenje veoma detaljno opisuje mogućnost malicioznog scenarija koji je mogao dovesti do izmene hash vrednosti ključa, te zahteva složen postupak korisnika, koji mora obrisati prethodno prihvaćene vrednosti. Naravno, očekuje se da korisnik bude prethodno obavešten o izmenama na serveru, odnosno da bude u mogućnosti da drugim, bezbednim, kanalom komunikacije sazna nove parametre pristupa. Ovo je prihvatljivo ponašanje za protokol koji je generalno namenjen naprednijim korisnicima, ali podrazumevano ponašanje Internet pretraživača je prikazivanje poruke o izuzetku koja se ne razlikuje od inicijalne [45]. U tom slučaju, većina korisnika ne bi ni primetila ukoliko bi sertifikat bio zamenjen usled aktivnog napada na konekciju, što je ujedno i najjači argument protiv korišćenja samopotpisanih sertifikata na javno dostupnim sajtovima koji razmenjuju osetljive privatne podatke sa korisnicima.

S obzirom da implicitno poverenje u slučaju prve konekcije, odnosno prihvatanje mogućnosti da se sertifikat zaista promenio ipak predstavljaju ozbiljne potencijalne vektore napada na ove protokole, razvijeni su sistemi za ublažavanje tih manjkavosti, a koji se sastoje od nezavisnih servera koji beleže informacije o aktuelnim ključevima za pojedine domene (notary servers). Jedan od predloženih sistema, koji je u domenu otvorenog koda (open source) je „Perspectives“, a njegova glavna karakteristika je da krajnjeg korisnika ne opterećuje dodatnim procedurama neophodnim za procenu bezbednosti konekcije koju uspostavlja, već koristi poseban algoritam na osnovu kojeg odlučuje da li će prikazati upozorenje o eventualnom napadu [58].

Ovaj softverski sistem se sastoji od grupe servera koji sa različitih lokacija na Internetu kontinuirano prikupljaju informacije (hash vrednosti) o važećim sertifikatima domena zaštićenih HTTPS-om, te održavaju bazu ovih podataka. Druga komponenta je dodatak za Internet pretraživač koji prilikom svake HTTPS konekcije kontaktira ove servere i poredi priloženi sertifikat sa onima koje su serveri ranije zabeležili. Ukoliko je sertifikat drugačiji nego što se očekuje, ako je ta promena nastala nedavno ili u tom trenutku, a pogotovo ako se samo sa jedne ili malog broja lokacija na Internetu vidi drugačija hash vrednost, prikazuje se upozorenje o mogućem napadu.

Opšta karakteristika svakog pristupa konceptu poverenja pri prvoj upotrebi je nužnost postojanja kompromisa između bezbednosti i dostupnosti usluge. Kada korisnik primi ključ po prvi put, ili dođe do njegove promene, suočen je sa izborom da li da ga prihvati, čime potencijalno ugrožava svoju bezbednost, ili da ga odbije, čime sebi uskraćuje usluge servera. U tom smislu, sistem kao što je „Perspectives“ pruža dodatnu informaciju o validnosti ključa koja pomaže korisniku (ili aplikaciji) da donese ispravnu odluku.

Autentifikacija nepoznatih, udaljenih servera je problem koji se može rešavati na više načina. Osim infrastrukture javnih ključeva, gde je briga o poverenju delegirana spoljnim autoritetima kojima se implicitno veruje, poverenje pri prvoj upotrebi, bilo ono potkrepljeno dodatnim informacijama ili ne predstavlja delimično pouzdanu alternativu u određenim situacijama u kojima nije potrebno ili moguće uspostavljanje mreže PKI [58]. Treća rasprostranjena varijanta uspostavljanja odnosa poverenja su protokoli kao što je „**Kerberos**“, u kome je svakom učesniku u komunikaciji prethodno, na bezbedan način, dostavljen deljeni tajni podatak (shared secret), pomoću kojega će biti uspostavljen siguran komunikacioni kanal.

Kerberos protokol je razvijen na univerzitetu MIT, sa ciljem da odgovori na potrebe manjih organizacija kojima i implementacija infrastrukture javnih ključeva predstavljala preveliko finansijsko i tehničko opterećenje, a koje imaju potrebe za bezbednom i pouzdanom internom komunikacijom [53]. Iako bi sistem baziran na paru javnog i tajnog ključa mogao da se implementira i bez centralnog autoriteta, jasno je da ga ne bi bilo moguće skalirati na racionalan način čim bi broj korisnika premašio nekoliko desetina, jer bi svaki učesnik u komunikaciji morao raspolagati javnim ključevima svih ostalih.

Ovo je centralni problem koji rešava Kerberos protokol, jer on uvodi centralizovani mehanizam treće strane od poverenja (Trusted Third Party – TTP, odnosno KDC – key distribution center) koji omogućava da se sva razmena ključeva odvija isključivo sa njim, bez potrebe za interakcijom

korisnika u fazi uspostavljanja poverenja [53]. U suštini, može se reći da TTP preuzima ulogu CA u PKI ali je sam protokol značajno pojednostavljen i prilagođen manjem broju korisnika. Naravno, treba imati u vidu da je TTP server ključna tačka za bezbednost celog sistema, te da je neophodno sprovesti naročite mere radi sprečavanja njegove eventualne kompromitacije.

Kerberos se pre svega koristi za autentifikaciju korisnika, pri čemu se kreira sesijski ključ, kojim se naknadno obezbeđuje poverljivost i integritet komunikacije. U principu, nema ograničenja koji će se simetrični algoritam koristiti za samu enkripciju, ali je u najvećem broju konkretnih implementacija ovog protokola to i dalje DES (Data Encryption Standard) [53]. U funkcionalnom smislu, Kerberos se zasniva na različitim vrstama tiketa (tickets) koji korisnicima omogućavaju pristup različitim resursima sistema. Najvažniji je inicijalni tiket koji služi za dobijanje svih ostalih TGT (ticket-granting ticket). On se dodeljuje prilikom uspešnog prijavljivanja na sistem i koristi se prilikom svih ostalih komunikacija umesto ponovnog unošenja korisničkog imena i lozinke. Na ovaj način se postiže transparentno korišćenje zaštićenih resursa, podrazumevajući implicitno inicijalno poverenje u KDC sistem koji izdaje tikete.

2.3.3. Teškoće pri tehničkoj implementaciji konkretnih tehnika

Bez obzira na eventualnu matematičku dokazivost celishodnosti i bezbednosti neke kriptografske tehnike zaštite, konkretna implementacija u korisničko okruženje predstavlja poseban izazov za programere. Poznata obrnuta srazmera između bezbednosti i komfora u radu dopunjava se objektivnim problemima koji proističu iz veoma kompleksne prirode kriptografskih algoritama i protokola koji ih implementiraju. Takođe, ne može se zanemariti ni realan kontekst proizvodnje savremenih aplikacija, u kome zahtevi tržišta za brzinom izrade i implementacijom novih funkcionalnosti najčešće bivaju sprovedeni na račun razvoja adekvatnih bezbednosnih mehanizama.

Još jedan faktor koji ne treba imati u vidu je raširena percepcija kriptografskih tehnika kao kontroverzne problematike, za koju su konstanto zainteresovani značajni društvenopolitički akteri. Zakonska rešenja koja ograničavaju, otežavaju ili zabranjuju pristup ovim tehnologijama nisu novost, a u savremeno doba se proširuju zahtevima za implementaciju konkretnih tehničkih rešenja koja ih suštinski obesmišljavaju (npr. zahtevi za ugradnjom „back door“ opcija ili master ključeva u algoritme i konkretne uređaje). Takođe, nedavna nekontrolisana proliferacija zlonamernih aplikacija razvijenih od strane pojedinih bezbednosnih službi omogućila je brz razvoj sofisticiranih napada koji su zaobišli mnoge ispravno implementirane kriptografske tehnike i uništili ili oštetili brojne informacione sisteme.

Napadi na pojedine tehnike, kao što je presecanje HTTPS protokola putem nenamenski iskorišćenog CA sertifikata, su istovremeno i osnova biznis modela pojedinih proizvođača legitimne telekomunikacione opreme, te oni veoma uspešno lobiraju protiv usvajanja savremenih rešenja koja bi onemogućila ove postupke. Slično, prodavci SSL sertifikata su tek nedavno dobili adekvatnu besplatnu konkurenciju (Let‘s Encrypt) ali i dalje odbijaju veliki broj potencijalnih korisnika izuzetno visokim cenama svoje usluge.

Lozinke i PIN kodovi su najrasprostranjeniji mehanizam autentifikacije, koji kada je ispravno implementiran, pruža dovoljan nivo zaštite za većinu situacija. Ipak, pri projektovanju aplikacije neophodno je imati u vidu celokupan proces kreiranja, upotrebe i čuvanja lozinki jer se u svakoj od ovih faza može napraviti puno grešaka koje omogućavaju različite napade. Kod ovog metoda autentifikacije možda najviše dolazi do izražaja obrnuta srazmera bezbednosti i komfora upotrebe, jer većina korisnika ne želi da pamti dugačke lozinke ili PIN kodove. Iz tog razloga, dobre prakse nalažu upotrebu jakih algoritama za enkripciju šifara, koji su otporni na savremene napade grubom silom (bruteforce) putem distribuiranih računarskih sistema ili grafičkih koprocesora (GPU). Takođe, kada god je moguće pribegava se dvofaktorskoj autentifikaciji, koja za korisnika predstavlja minimalno opterećenje, jer su oba koraka uglavnom jednostavna, a ukoliko je izvedena korektno, značajno povećava otpornost sistema.

Ljudski faktor je prisutan i mnogim drugim segmentima upotrebe sistema, te je neophodno isti imati u vidu, odnosno graditi sistem tako da se preduprede najverovatnije korisničke greške koje izazivaju bezbednosne posledice. Teško je pobrojati sve probleme ove vrste, kao i napraviti jasnu granicu koje slučajeve treba prihvatiti i raditi na iznalaženju adekvatnih kompromisa, a šta predstavlja jasnu nameru korisnika da upotrebljava sistem na neadekvatan način izlažući svoje podatke riziku. Npr. može se reći da su preventivni mehanizmi kao što je Google Play ili iStore koji nastoje da se pozicioniraju kao jedini legitiman izvor za distribuciju mobilnih aplikacija preuzeli odgovornost za veći broj problema i propusta koji mogu nastati njihovom upotrebom. Na programerima je da ispoštuju stroga pravila ali i ustanovljene dobre prakse, ukoliko žele da se njihove aplikacije distribuiraju ovim putem. Sa druge strane, omogućavanje instalacije bilo koje aplikacije (proces poznat kao jailbraking ili rooting) uglavnom poništava garanciju na uređaj, prebacujući svu odgovornost na korisnika. Istovremeno, to je slučaj koji se ne pokriva smernicama programerima, te u praksi malo ko pokušava da dodatno obezbedi podatke legitimnih aplikacija ukoliko su pokrenute na modifikovanom uređaju.

Indirektni (Side channel) napadi uglavnom zavise od konkretne hardverske ili infrastrukturne osnove implementiranog sistema i vrlo ih je teško predvideti, odnonso preventivno delovati na njihovom suzbijanju pri konstruisanju i izradi konkretnog informacionog sistema. Oni obuhvataju širok dijapazon tehničkih postupaka kojima se do štićenog podatka dolazi manipulacijom nekog spoljnog dela sistema. Iako najčešće zahtevaju temeljan istraživački rad da bi se otkrile, ove ranjivosti mogu biti veoma ozbiljne i teške za otklanjanje ukoliko pogode neki rasprostranjeni tip sistema. Dobar primer su ranjivosti Rowhammer i Drammer koje pogađaju Android uređaje, odnosno ARM procesore koje većina njih koristi. Ukratko, specifičnim postupkom brze promene sadržaja određene memorijske adrese, postiže se izmena na susednim adresama što može rezultovati podizanjem privilegija pod kojima je pokrenuta aplikacija, tj. dobijanje administratorskog (root) pristupa uređaju. S obzirom da je ovo moguće usled nedostataka samog hardvera, odnosno pojedinih verzija ARM procesora, softverske ispravke nisu uspele u potpunosti da ih otklone, te značajan broj Android uređaja i dalje ostaje ranjiv na ovaj napad.

Generisanje, raspodela i čuvanje ključeva predstavlja najveći problem pri primeni bilo koje kriptografske tehnike zaštite. Bez obzira koliko neki algoritam ili protokol bili bezbedni, činjenica da ključ mora biti u nekom trenutku prisutan u memoriji ili sačuvan na disku, otvara mogućnost mnogih praktičnih napada. Dobre prakse nalažu da se ključ nikada ne čuva u plaintext obliku, već da uvek bude zaštićen šifrom ili dodatnim ključem (key encryption key – KEK) i upotrebom za to predviđenog algoritma. Takođe, dok se sa ključem radi u memoriji, preduzimaju se posebne mere (uglavnom implementirane u bibliotekama naprednih programskih jezika) kojima se on čuva na nepredvidivoj lokaciji u memoriji ili se često menja njegova lokacija. Generisanje ključeva i dugih tajnih parametara uglavnom zavisi od kvaliteta generatora slučajnih (random) brojeva, koji ne postoje u zadovoljavajućem obliku na jednostavnijim platformama kao što su embedded odnosno IoT (Internet of things) uređaji.

Osiguranje poverenja pri prvoj upotrebi se i u veoma sofisticiranim protokolima (kao što je „Signal“, o kome će više reči biti kasnije) vrši na neki od opisanih, a u suštini nepraktičnih ili nebezbednih načina. Najčešće se praktikuje upoznavanje korisnika sa hash vrednošću ključa bezbednim kanalom (što može biti putem SMS-a ili dostavljanjem fizičke poruke), kod nekih aplikacija za šifrovane telefonske razgovore praktikuje se prethodna razmena ključnih reči na sličan način, dok se kod provere samopotpisanih sertifikata na web sajtovima uglavnom podrazumeva da je prva konekcija (prilikom koje se prihvata izuzetak – exception) bezbedna. Manjkavosti ovih metoda su očigledne, ali se usled male verovatnoće da će napad biti izvršen prilikom inicijalnog uspostavljanja poverenja, one uglavnom ignorišu, te se primenjuju one koje su najkomfornije za korisnika.

Slično je i sa potvrdom spoljnog autoriteta, jer s obzirom da bi ručno modifikovanje liste CA tela kojima sistem implicitno veruje bilo previše komplikovano većini korisnika, proizvođači softvera nastoje da preuzmu ažuriranje ovih podataka na sebe, te da korisnike štite automatski putem npr. blagovremene distribucije lista povučenih (revoked) sertifikata i sličnim mehanizmima.

S obzirom da smo u ovom poglavlju napravili pregled najčešće korišćenih kriptografskih tehnika zaštite i naveli neke od teškoća pri njihovoj praktičnoj implementaciji, u sledećem delu ćemo detaljnije razraditi ove teme u kontekstu mobilnih aplikacija, a na primeru Android sistema.

3. Mobilne aplikacije

3.1. Karakteristike Android sistema

3.1.1. Istorijat Android sistema

Prve najave razvoja Android sistema došle su relativno kasno u fazi razvoja tzv. „pametnih“ mobilnih uređaja. Takođe, to nije bio jedini projekat otvorenog koda koji je pretendovao da se pozicionira kao industrijski standard, mnogi slični projekti su pokušavali da postignu isti cilj, a pojedini su bili i napredniji u tehničkom smislu [46]. Ipak, Android je probudio široko interesovanje tehničke, medijske i korisničke javnosti nastojeći da suštinski prevaziđe konceptualna ograničenja koja su do tada važila za lične elektronske uređaje. Naime, osim za uobičajene funkcionalnosti kao što su razgovor, SMS poruke, e-mail, pretraživanje malobrojnih servisnih informacija sa Interneta, a koje su bile uobičajene za takve uređaje, Android je pretendovao da ukine bilo kakva ograničenja ovog tipa, te proizvede platformu čija bi funkcionalnost oslobodila maštu korisnika i programera.

Velika promena je nastupila 2005. godine kada je kompanija Google kupila softversku kuću Android koja se bavila razvojem operativnih sistema za mobilne telefone. Ubrzo nakon toga, 2007. godine, kreirana je grupa od 34 kompanije pod nazivom Open Handset Alliance, sa zadatkom ubrzavanja inovacija u mobilnim tehnologijama radi pružanja boljih, raznovrsnijih i jeftinijih usluga korisnicima [5]. Kao okosnicu svoje strategije odabrali su upravo Android sistem, što je označilo početak njegovog munjevitog uspeha i popularnosti kod krajnjih korisnika. Već do 2011. godine više od polovine svih „pametnih“ telefona su bili bazirani na Android sistemu.

Za razliku od iPhone ekosistema, koji je imao ustanovljenu tržišnu poziciju, odanu korisničku bazu i raspolagao velikim brojem lepo dizajniranih aplikacija, Android je ponudio prednosti koje pruža otvorena platforma koja iza sebe ima podršku najjačih faktora u industriji. Otvorenost Android sistema se ogleda kako na hardverskoj, tako i na softverskoj strani. Dok je iPhone nastojao da što više ograniči pristup samom sistemu, pa i platformi za razvoj aplikacija, Android je pružao olakšice za rad programerima u vidu iscrpne dokumentacije i zvanične podrške različitim razvojnim okruženjima. Takođe, iako zvanično nepodržano, modifikovanje sistemskih delova platforme je takođe dozvoljeno na najvećem broju uređaja, čime je ostavljena mogućnost za značajna unapređenja koja dolaze iz zajednice entuzijasta širom sveta. Takođe, sistem distribucije Android aplikacija, iako centralizovan, ima daleko liberalnije uslove koje svaki softverski paket mora da ispuni kako bi dobio zvaničnu podršku odnosno preporuku korisnicima [5].

Android platforma je prešla dug put od prve verzije (1.0) objavljene 2008. godine. Čak i tada su postojale aplikacije koje su pratile Google servise kao što su Gmail, Maps, Kalendar, osnovni pretraživač interneta i YouTube, ali su one bile integrisane u sam sistem i pružale tek osnove funkcionalnosti. Značajno je primetiti početnu orijentaciju kompanije na profilisanje telefona kao multimedijalnog uređaja koji će imati daleko više namena od dotadašnjih koje su se svodile na razgovor, slanje jednostavnih poruka i bazične aplikacije [43].

Od sledeće verzije 1.5 (Cupcake tj. kolačić) iz 2009. godine počeli su nazivi po slatkišima, koji su, osim interesantnog marketinškog momenta, po abecednom redu označavali redosled objavljivanja. Ova verzija je donela prvu virtuelnu tastaturu, koja je omogućila da se uštedi fizički prostor uređaja, odnosno poveća ekran na budućim aparatima. Takođe je predstavljen razvojni okvir za aplikacije (framework) čime je platforma privukla veliki broj programera.

U verziji 1.6 (Donut) uvedena je podrška za različite veličine ekrana i rezolucije, kao i za određene GSM standarde, čime su se stvorili uslovi za proboj na tržišta širom sveta. Takođe je uvedena opcija direktne pretrage Interneta putem Google-a sa početnog ekrana (Home screen) u cilju dublje integracije nove platforme sa postojećim ekosistemom ove kompanije.

Verzije sa prefiksom 2 (2.0, 2.1 Eclair, 2.2 Froyo i 2.3 Gingerbread) označile su početak globalnog proboja Androida na tržište mobilnih uređaja. Ubrzan razvoj i brojne nove mogućnosti praćeni su agresivnim marketinškim kampanjama, izgradnjom vizuelnog identiteta brenda kao i sponzorstvima sa pružaocima telekomunikacionih usluga. Sa softverske strane, značajni noviteti su bili mapa sa GPS navigacijom i glasovnim uputstvima, poboljšana interaktivnost sa sadržajima na ekranu, podrška za tada popularni Adobe Flash, kao i začeci glasovnih komandi.

Verzije 3.0 do 3.2 (Honeycomb) su predstavljale specifična izdanja namenjena tablet računarima. Dizajn i funkcionalnost su značajno izmenjeni, a što je kasnije poslužilo kao osnova za slične izmene glavne verzije softvera namenjene telefonima.

Sledeći značajan korak je bila verzija 4 (4.0 Ice Cream Sandwich, 4.1, 4.3 Jelly Bean, 4.4 KitKat) koja se pojavila 2011. godine, označivši prelazak na sasvim nove koncepte dizajna i funkcionalnosti, kakve poznajemo i danas. Takođe, integrisan je razvoj okruženja za tablete i telefone u jednu razvojnu granu. Od unapređenja na svim novima treba izdvojiti povećanu interaktivnost sa Google servisima, koji su koristeći podatke prikupljene od korisnika, osposobljeni da daju značajno bolje rezultate

pretrage i proaktivno obaveštavaju o relevantnim informacijama. Pojavile su se opcije za višekorisnički rad na tabletima, kao i servis "OK, Google", odnosno glasovna aktivacija interaktivnog pretraživača.

Novi veliki skok je bila verzija 5.0 (Lollipop) objavljena krajem 2014. godine, koja je usvojila obrazac Material Design, savremen i prepoznatljiv izgled uz intuitivnu funkcionalnost sveprisutnu u brojnim aplikacijama i Google servisima. Glasovne komande su značajno unapređene, višekorisnički rad je omogućen i na telefonima, sistem obaveštavanja korisnika o porukama i bitnim događajima je daleko inteligentniji, ali su brojni noviteti praćeni velikim brojem tehničkih nedostataka koju su uglavnom ispravljeni u verziji 5.1 sledeće godine.

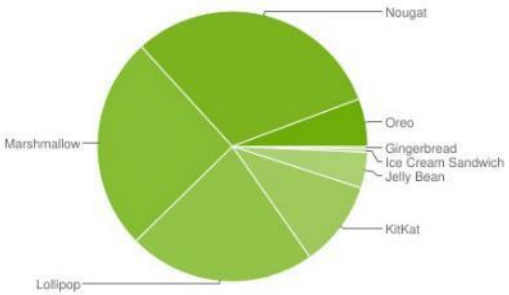
Verzija 6.0 (Marshmallow) nije donela značajne novine, ali je ustanovila praksu Google-a da svake godine objavljuje novu, celobrojnu iteraciju ovog softverskog paketa. Neke od novih opcija su podrška za čitače otiska prsta, USB-C protokol i detaljniji sistem dozvola za aplikacije.

U verzijama 7.0 i 7.1 (Nougat) uvedene su mogućnosti za deljenje ekrana (split screen), poboljšan je način na koji aplikacije prikazuju obaveštenja i predstavljena su poboljšanja upravljanja mrežnim protokolima kako bi se smanjila količina prenetih podataka. Prethodne glasovne komande su zamenjene sistemom Google Assistant, koji predstavlja adekvatan pandan sličnim rešenjima drugih kompanija (Amazon Alexa i Apple Siri), a oslanja se na impozantnu serversku infrastrukturu Google-a.

Verzije 8.0 i 8.1 (Oreo) donele su poboljšanja grafičkog interfejsa i korisničkog iskustva, nove multimedijalne opcije a takođe nastoje da poboljšaju funkcionalnost Google-ovih laptop računara (Chromebook), što je jedan od strateških prioriteta kompanije.

U sledećim verzijama možemo očekivati dalja poboljšanja postojećih funkcija, kao i brojne inovacije radi poboljšanja autonomije, iskorišćenja mrežnih kapaciteta i unapređenja sveukupnog korisničkog iskustva [43]. Takođe, najavljuju se značajne novine na polju bezbednosti informacija, odnosno zaštite privatnosti korisničkih podataka.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.5%
4.2.x		17	2.2%
4.3		18	0.6%
4.4	KitKat	19	10.3%
5.0	Lollipop	21	4.8%
5.1		22	17.6%
6.0	Marshmallow	23	25.5%
7.0	Nougat	24	22.9%
7.1		25	8.2%
8.0	Oreo	26	4.9%
8.1		27	0.8%



Data collected during a 7-day period ending on May 7, 2018.
Any versions with less than 0.1% distribution are not shown.

Slika 2 Rasprostranjenost Android verzija na tržištu [65]

3.1.2. Tehničke karakteristike

S obzirom se suštinski ne razlikuju od ostalih savremenih računarskih sistema, Android uređaji raspolažu sličnom strukturom operativnog sistema u arhitekturalnom smislu. Osnovna karakteristika savremenih operativnih sistema je slojevitost, odnosno vertikalna razdvojenost komponenti koje se bave različitim funkcijama koje sistem vrši, od najviših nivoa koji se bave prezentacijom i prikupljanjem podataka pri interakciji sa korisnikom, do najnižih procesa koji se bave komunikacijom sa hardverom i ne zahtevaju direktnu korisničku intervenciju [5].

Na najnižem nivou operativnog sistema nalazi se jezgro (kernel), koje se izvršava direktno na hardverskom sloju, odnosno procesoru, a kroz koje se direktno ili indirektno upućuju svi zahtevi ostalih aplikacija na sistemu [5]. Android je razvijen na osnovu Linux kernela verzije 2.6, a specifične implementacije u različitim Android sistemima predstavljaju manje ili veće modifikacije odnosno rekonfiguracije ovog dobro poznatog softverskog sistema. Iako je u suštini varijacija Linux sistema, Android je osim na kernelu, uveo brojne značajne modifikacije na skoro svim sistemskim alatkama, a pogotovo u upravljačkim programima (drajverima) za određene podsisteme. Usled toga, nije rasprostranjeno pominjanje Androida kao još jedne u nizu Linux distribucija (iako to suštinski jeste), već se o njemu, zbog specifične platforme za koju je namenjen, kao i značajnih modifikacija na svim nivoima, najčešće govori kao o zasebnom operativnom sistemu.

Slojevita arhitektura Android sistema prisutna je na svim nivoima ove platforme. Osim već pomenutog kernela, koji je odgovoran za direktnu komunikaciju sa hardverom putem drajvera i implementaciju osnovnih bezbednosnih protokola za pokretanje aplikacija, sledeći sloj na kome funkcionišu osnovne biblioteke se takođe smatra sistemskim delom Android platforme [49]. Ovi moduli su kompajlirani direktno u mašinski jezik i pružaju neke uobičajene servise koji su potrebni drugim sistemskim programima i korisničkim aplikacijama. Neki od njih su „surface manager“ koji upravlja grafičkim okruženjem odnosno displejom, biblioteke za 2d i 3d grafiku, WebKit koji aplikacijama pruža mogućnosti Internet pretraživača, SQLite baza podatka koja je osnovni i najjednostavniji način pohranjivanja podataka na Android platformi, itd.

Sledeći sloj, koji se takođe pokreće kao direktni proces Linux kernela je izvršno okruženje (app runtime), koje se kreira za svaku aplikaciju, a u čijoj osnovi se nalazi Dalvik virtuelna mašina. Ona u suštini predstavlja optimizovani interpreter Java aplikacija, prilagođen brzom radu na platformama ograničenih mogućnosti. U novijim verzijama Androida zamenjen je sistemom ART (Android runtime) koji ima istu funkcionalnost, ali je vrši na način adekvatan novijim uređajima koji raspolažu sa više memorijskog prostora i procesorske snage. Na ovom sloju se takođe nalaze i sistemske Android biblioteke, paketi funkcija koji olakšavaju rad sa izlazno-ulaznim periferijama i slični programski paketi neophodni za rad svake aplikacije.

Na sledećem nivou se nalazi okruženje za rad aplikacija (Application framework), u kome se nalaze kompajlirane verzije aplikativnog koda prilagođene pokretanju u virtualnim mašinama. Neki od ovih modula su upravljač paketima (package manager) koji služi za instalaciju i slične konfiguracione postupke sa korisničkim aplikacijama, menadžer aktivnosti (activity manager) koji upravlja “aktivnostima”, odnosno posebnim instrukcijama koje jedna aplikacija može uputiti nekom svom ili modulu druge aplikacije, a koje moraju biti kontrolisane iz centralizovanog procesa (activity stack).

Na najvišem nivou se nalaze same korisničke aplikacije, bez obzira da li su ih napisali nezavisni programeri ili su instalirane od strane Google-a ili drugih korporativnih tela (npr. operatera telefonske usluge za koju je vezan Android uređaj. Aplikacije na ovom sloju se uglavnom sastoje od više komponenti: aktivnosti (activity), podsistema za primanje obaveštenja (broadcast Receivers), raznih servisa (services) i isporučilaca sadržaja (Content Providers).

Osim ove vertikalne podeljenosti Android sistema na funkcionalne slojeve, od podjednakog značaja za razumevanje i adekvatno korišćenje svih mogućnosti ove platforme je i segmentirani bezbednosni model privilegija aplikacija koji je u potpunosti preuzet od Linux sistema.

Koncept korisnika i grupa kojim se ovaj model implementira, zasnovan je na dodeljivanju jedinstvenog identifikatora svakom korisniku (UID) kao i postojanju grupa kojima korisnici mogu pripadati, a koje su takođe jedinstveno određene putem identifikatora grupe (GID). Ovi jedinstveni brojevi služe da razlikuju korisnike, prikažu pripadnost korisnika određenim grupama, dok je njihova primarna upotreba vezana za posedovanje adekvatnih dozvola odnosno prava pristupa određenim resursima sistema.

Sistem dozvola u Linux sistemu je opširna tema koja prevazilazi okvire ovog rada, pa ćemo samo napomenuti da je izuzetno granularan i restriktivan, što znači da se dozvole pristupa svakom fajlu (a u fajlove se računaju i logičke putanje do praktično svih resursa Linux sistema) mogu podesiti na nivou vlasništva, prava čitanja, pisanja i izvršavanja za svakog pojedinačnog korisnika ili grupu korisnika. Iako zamišljen kao sistem upravljanja privilegijama u višekorisničkom okruženju, veoma lako je prilagođen Android platformi u kojoj, sa stanovišta prethodno pomenutih identifikatora, svaka aplikacija predstavlja zasebnog "korisnika". Ove bezbednosne politike predstavljaju osnovnu i vrlo značajnu liniju odbrane celokupnog Android sistema u smislu neovlašćenog pristupa korisničkim podacima od strane "zlonamerne" aplikacije.

Stroga segregacija ovlašćenja aplikacija se osim na nivou fajl sistema primenjuje i tokom izvršenja koda u memoriji. Pored već opisanog modela u kome se svaka aplikacija pokreće u zasebnom procesu, koji se takođe izvršavaju u odvojenim virtualnim (Dalvik) mašinama, čak i delovi koda koji se izvršavaju direktno na hardverskoj platformi poštuju sva ograničenja Linux sistema privilegija procesa [49]. U praksi to znači da tokom normalnog rada aplikacije, njoj, bez posebnih dozvola, nije omogućeno da iščitava memorijski prostor neke druge aplikacije.

3.1.3. Razvoj i distribucija aplikacija

Arhitektura Android aplikacija je nešto drugačija od one uobičajene za desktop i web aplikacije. Specifični uslovi korišćenja i distribucije mobilnih aplikacija zahtevaju posebne pristupe u njihovoj izradi, kao i naročita pravila i procedure koje garantuju njihov bezbednu isporuku i eksploataciju. Od programera se očekuje da prate ove smernice, a velike kompanije (pre svega Google) i standardizaciona tela nastoje da ih blagovremeno prilagođavaju potrebama korisnika i mogućnostima tehnologije.

Modularni pristup je prisutan u razvoju svih vrsta savremenih aplikacija, ali je kod mobilnih on

apsolutno neophodan. Ponovno korišćenje delova koda, elemenata interfejsa ili celih funkcionalnosti ubrzava razvoj, smanjuje mogućnost grešaka i pruža korisnicima poznato okruženje na koje se lako navikava [46]. Android odlazi korak dalje, omogućavajući da se delovi web aplikacija (pogotovo iz Google ekosistema, npr. Google mape) veoma lako integrišu u nove aplikacije, zadržavajući svoju prvobitnu funkcionalnost.

U drugim razvojnim okruženjima za mobilne aplikacije, postoji ograničenje po pitanju korišćenja specifičnih servisa koje pružaju postojeći delovi sistema ili druge aplikacije. Kod Androida imamo sasvim drugačiji pristup, jer je mehanizmom "namera" (intent) moguće odabrati koja aplikacija će preuzeti izvršenje operacije koju smo zadali, dok se o ispravnom prosleđivanju upita i odgovora između aplikacija, brine sam operativni sistem.

Android aplikacije se sastoje iz četiri osnovna tipa komponenti koji su definisani samom arhitekturom sistema:

Aktivnosti (**activity**) se mogu uporediti sa jednostavnijim desktop aplikacijama, koje imaju jednu uglavnom jednostavnu namenu [46]. Aktivnosti su delovi programskog koda koji se pokreću po korisničkom ili sistemskom zahtevu i ostaju aktivni dok za njima postoji potreba. One mogu interagovati sa korisnikom i razmenjivati podatke sa drugim aktivnostima, servisima ili "namerama" (intent). Većina koda koji se napiše prilikom kreiranja Android aplikacije pripada aktivnostima, a one se najčešće, delimično pojednostavljeno, izjednačavaju sa aktivnim prikazom, odnosno ekranom. Operativni sistem može uništiti (destroy) aktivnost kada proceni da više nije potrebna, kako bi oslobodio memorijski prostor.

Servisi (**service**) su koncept analogan pozadinskim procesima u desktop i serverskim okruženjima [46]. Na Android platformi karakteriše ih automatsko instanciranje pri pokretanju sistema, krajnjem korisniku transparentan i neprimetan rad a takođe uglavnom nemaju korisnički interfejs kojim bi bilo moguće uticati na parametre njihovog rada. Važno je napomenuti da njihova uloga ne mora biti samo sistemska, već one mogu biti komponenta neke korisnički orijentisane aplikacije, npr. program za puštanje muzike koji je aktivan bez obzira na trenutni ekranski prikaz i aktivnosti korisnika.

Prijemnici poruka i namera (**broadcast and intent receivers**) odgovaraju na zahteve drugih aplikacija koje putem sistemskih objava (announcement) šalju informacije o svojim potrebama [46]. Njih može poslati neki sistemski program ili bilo koja korisnička aplikacija, a o ispravnom povezivanju izlaznih i ulaznih interfejsa za razmenu podataka na ovaj način brine se operativni sistem.

Provajderi sadržaja (**content providers**) prosleđuju podatke aktivnostima ili servisima putem standardizovanog interfejsa u vidu URI-ja (Uniform Resource Identifier), odnosno jedinstvenog identifikatora [46]. S obzirom da aplikacija šalje zahtev na ovako specificiran način, ona ne mora da zna koji će konkretan provajder odgovoriti, već se o ispravnoj alokaciji traženog resursa brine operativni sistem. Takođe, sve aplikacije koje nude podatke određenog tipa registruju ih na isti način, te je moguće ostaviti korisniku izbor koji od ponuđenih izvora podataka da iskoristi. Iako nije nužno pristupati podacima na ovaj način, opisani mehanizam je preporučeni i podržani način funkcionisanja Android aplikacija, te programeri koji žele maksimalan komfor u razvoju i fleksibilnost u korišćenju, uvek nastoje da ga iskoriste.

Okosnica bezbednosnog modela Android sistema na aplikativnom sloju su dozvole (permission), koje se dodeljuju aplikacijama, a kojima se omogućava izvršenje neke funkcije. Lista ovih dozvola sastavlja se pri kreiranju aplikacije, a one se pri instalaciji predočavaju korisniku koji može odabrati da neke od njih onemogući, što ipak nije preporučljivo sa stanovišta stabilnosti same aplikacije [49]. Neke od tih dozvola obuhvataju pristup mrežnim resursima, Internetu, kameri, listi kontakata, podsistemu za upućivanje telefonskih poziva ili poruka koje se mogu dodatno naplaćivati, statičkim fajlovima na uređaju itd. Ovaj sistem je veoma jednostavan i za korisnike i za programere, a relativno je efikasan, bar u smislu obaveštavanja korisnika o mogućim rizicima koje prihvata instalacijom aplikacije.

Posebno važan segment garancije pouzdanosti i bezbednosti Android ekosistema predstavlja zvanični sistem distribucije aplikacija, odnosno Google Play [46]. Iako je podešavanjem posebne opcije moguće instalirati bilo koju kompatibilnu aplikaciju na svaki Android uređaj, zvanični kanal distribucije je najčešće korišćen način instalacije aplikacija. Dostupan je sa svakog uređaja i veoma jednostavan za korišćenje, a putem njega je moguće puštati u promet i besplatne i aplikacije koje se plaćaju, dok je samo procesiranje uplate najčešće integrisano sa telefonskim računom korisnika, čime je proces kupovine značajno olakšan i ubrzan. Da bi neka aplikacija bila prihvaćena za postavljanje na Google Play sistemu, neophodno je da zadovoljava određene kriterijume bezbednosti, stabilnosti i transparentnosti.

Aplikacija mora da bude temeljno testirana, na najvećem mogućem broju različitih uređaja i u svim predviđenim uslovima eksploatacije.

Poželjno je definisati neku vrstu “ugovora” sa krajnjim korisnikom (End User License Agreement –

EULA) kojim se regulišu prava i obaveze obe strane, te definišu pravni postupci ukoliko neko navedenih načela bude narušeno. Iako nije obavezan, ovaj deo aplikacije pokazuje ozbiljnost izdavača i pruža dobru osnovu za razrešenje eventualnih nesuglasica pravne prirode koje mogu nastati kao rezultat rada sa aplikacijom.

Kreiranje jedinstvenog vizuelnog identiteta aplikacije (ikonica, naziv, logotip itd.) doprinosi prepoznatljivosti proizvoda i dodatno informiše korisnika o softveru koji će instalirati.

Aplikacija koja se prezentuje krajnjem korisniku treba da sadrži samo neophodan kod, prečišćen od eventualnih zaostalih artefakta iz perioda razvoja. Osim na veličinu aplikacije i log fajlova, neke od ovih opcija mogu drastično narušiti bezbednost ili brzinu izvršenja programskog koda.

Verzionisanje aplikacije je bitno pre svega zbog praćenja povratnih informacija, odnosno povezivanja korisničkih iskustava ili problema sa konkretnom verzijom aplikacije, čime se značajno povećava efikasnost ispravljanja grešaka i unapređuje ukupan kvalitet aplikacije.

Digitalno potpisivanje aplikacije adekvatnim sertifikatom izdavača je ne samo potrebno zbog bezbednosnih razloga i dokazivanja autorstva, već predstavlja neophodan korak, s obzirom da je većina Android uređaja konfigurisana tako da će sprečiti pokretanje nepotpisane aplikacije. Nema posebnih zahteva za dodatnim garancijama nekog poznatog sertifikacionog tela, već je dovoljan i samopotpisani (selfsigned) sertifikat, koji se kreira pri kompajliranju aplikacije.

Takođe je poželjno dodatno testirati verziju aplikacije namenjene krajnjim korisnicima, jer je moguće da je u procesu pripreme za distribuciju došlo do neke greške koja nije bila vidljiva tokom razvoja.

Iako su uglavnom u pitanju neobavezujuće preporuke, pravila koja važe u Google Play sistemu distribucije značajno doprinose kvalitetu aplikacija i diseminaciji dobrih praksi među programerima. Nema razloga da se bilo koje od ovih pravila zaobiđe ni pri razvoju najjednostavnijih aplikacija za ličnu upotrebu ili onih namenjenim zatvorenom, užem krugu krajnjih korisnika.

3.2. Ograničenja mobilnih uređaja

3.2.1. Način korišćenja

Savremeni, "pametni" uređaji se definišu kao oni koji raspolažu inteligencijom sličnom čovekovoj [61], što znači da se od njih očekuje da adekvatno reaguju na ponašanje korisnika i vremenom mu se sve više samostalno prilagođavaju. Takođe korisnici očekuju da ovakav uređaj može da ih odmeni u donošenju nekih uobičajenih svakodnevnih odluka i obavljanju monotonih radnji. Istraživanja [61] su takođe pokazala da su prilagodljivost i multifunkcionalnost karakteristike koje korisnici najviše cene kod pametnih uređaja, odnosno da u odnosu na njihovu izraženost kreiraju odluku o izboru i kupovini uređaja.

Ipak, najčešća upotreba mobilnih uređaja se svodi na zabavu i pasivnu konzumaciju sadržaja sa društvenih mreža [32]. Kada ovaj način korišćenja pređe u naviku (habituacija), zadovoljstvo upotrebom dodatno slabi i javlja se osećaj besmisla. Da bi se to izbeglo, nastoji se da svaka interakcija (mikrointerakcija) sa korisnikom ima neku svrhu, te da održava osećaj postignuća odnosno smisla.

Uprkos sveprisutnoj tehnologizaciji svih sfera savremenog života, istraživanja [54] usmerena na otkrivanje stvarnih potreba korisnika mobilnih uređaja, odnosno aplikacija, ukazuju da oni od svog odnosa sa tehnologijom žele nivo interakcije najpribližniji onom koji imaju sa drugim čovekom. Ukoliko potrebe korisnika za neposrednom komunikacijom, razmenom emocija i građenjem odnosa zasnovanih na iskrenosti i poverenju budu adekvatno zadovoljene, može se računati na njegovu dugoročnu lojalnost proizvodu odnosno brendu. Preneti na realan plan izrade mobilnih aplikacija, ovi koncepti se ogledaju u zahtevima za većom interaktivnošću, odgovorima praćenim adekvatnim emocijama (emotikoni), dodavanju funkcija koje simuliraju ljudsko ponašanje (npr. trenutni odgovori na pitanja postavljena uobičajenim jezikom u slobodnoj formi), itd.

Personalizacija ponašanja aplikacije [54] je takođe na veoma visokom nivou prioriteta pri dizajniranju i izradi kompetitivnih mobilnih aplikacija, što odmah povlači pitanje privatnosti korisničkih podataka koji su prikupljeni i obrađeni da bi se ona postigla. S obzirom na neophodnost čuvanja i sofisticirane obrade velike količine ovih krajnje privatnih podataka, nastaju problemi njihovog transfera na udaljene servere, odnosno pohranjivanja na samom uređaju. Korisnici retko uviđaju neposrednu vezu ovih rasprostranjenih korporativnih praksi sa problemima ugrožavanja privatnosti kada se podaci prikupljeni radi ciljanog marketinga iskoriste na drugi način, često protiv njihovih interesa.

Još jedan otkriven trend u očekivanjima korisnika [54] je želja da se smanji ili potpuno ukine upotreba lozinki i sličnih mehanizama koji smanjuju komfor interakcije. Biometrijske tehnologije koje se nameću kao logičan izbor su sve prisutnije na savremenim mobilnim uređajima, a njihova implementacija kao jedine tehnike autentifikacije je sve izvodljivija. Ipak, iako korisnici i prodavci

usluga imaju veliki interes da ova tehnologija zaživi, sa bezbednosne tačke gledišta i dalje postoje brojni izazovi u smislu njene pouzdanosti kao i obrade i čuvanja biometrijskih podataka.

Osim podataka prikupljenih direktno od korisnika, savremeni mobilni uređaji raspolažu brojnim senzorima koji primaju informacije iz svog neposrednog okruženja. Kombinacijom ovih parametara otvorene su mogućnosti za kvalitativna unapređenja brojnih do sada popularnih servisa kao što su društvene mreže, aplikacije orijentisane na praćenje zdravstvenog stanja pojedinca i zajednica, video igre, informacije o saobraćaju, životnoj sredini itd. [23]. S obzirom na izuzetnu popularnost ovih proširenih (augmented) aplikacija, za očekivati je da se trend nastavi, te će korisnici želeti još veću integraciju sa okruženjem posredovanu mobilnim uređajima. Najviše se odmaklo u sferi interakcija sa tzv. biosenzorima koji prikupljaju podatke o zdravstvenom stanju kao što je krvni pritisak, puls, elektrokardiogram, elektroencefalogram itd., a to čine putem integrisanih ili spoljnih uređaja. Pored brojnih koristi koje široka upotreba ovih uređaja može doneti, očigledni su povećani rizici po bezbednost krajnje ličnih podataka korisnika.

Pored novih mogućnosti, očekuje se da će one već prisutne napredovati u kvantitativnom smislu, što će se najviše odraziti na potrebu za bržim i pouzdanijim mogućnostima povezivanja na Internet. Razvojem mreža četvrte generacije (4G) praktično se izjednačila mogućnost pružanja sadržaja mobilnim i tradicionalnim (žičnim) multimedijalnim uređajima, ali ipak na ograničenim područjima, pod najpovoljnijim uslovima i u ograničenom vremenskom trajanju [63]. Zahtevi za velikom brzinom protoka, brzom odgovoru na upite, istovremenim opsluživanjem brojnih korisnika, visokom pouzdanošću i energetskom efikasnošću ostaju kao ciljevi koje nova, peta generacija (5G) treba da postigne.

Povezivanje na udaljene (cloud) servise, video pozivi visoke rezolucije, prikupljanje informacija iz automobila u realnom vremenu, IoT (Internet of things), pristup velikim bazama podataka (big data), kontrola dronova i robota, samo su neki od brojnih servisa koji će postati dostupni korisnicima mobilnih platformi na 5G mrežama. Inovacije na polju infrastrukture će doneti brojne mogućnosti ali i nove izazove pri izradi bezbednijih mobilnih aplikacija, koje će programerska zajednica morati da prihvati.

3.2.2. Ograničenja hardvera i infrastrukture

Postoje različite tehnologije i standardi pomoću kojih se mobilni uređaji povezuju sa Internetom i drugim mrežama. To su pre svega globalni sistem mobilne komunikacija (GSM), paketni radio servis

opšte namene (GPRS), mreže treće i četvrte generacije (3G i 4G) kao i još nekoliko sličnih manje rasprostranjenih sistema. Proizvođači uređaja su bili prinuđeni da prihvate ove razlike u standardima, tako da je danas teško naći savremene telefone koji su vezani isključivo za jedno tržište [69]. Srećom, deo Android sistema koji se bavi hardverskim slojem protokola je apstrahovan na takav način da navedena raznovrsnost standarda ne utiče direktno na razvoj aplikacija, odnosno ne unosi dodatnu kompleksnost.

Slično važi i za ostale bežične tehnologije, WLAN, NFC, InfraRed i Bluetooth čije su brojne iteracije najčešće podržane u novim uređajima, dok programeri imaju veću slobodu (i odgovornost) pri njihovoj upotrebi odnosno implementaciji u aplikacijama. Iako standardizovani, ovi komunikacioni protokoli unose značajan nivo kompleksnosti u razvoj ali i upotrebu aplikacija, jer često dolazi do nekompatibilnosti i nestabilnosti u radu među uređajima različitih proizvođača, odnosno verzijama protokola.

Sama fizička priroda mobilnih uređaja unosi određene teškoće u njihovu svakodnevnu upotrebu, a pogotovo u korporativnim okruženjima. Najveći rizik je svakako onaj od krađe ili gubljenja telefona koji sadrži osetljive podatke. S obzirom da je većini kompanija neisplativo da razvijaju specifična (bezbednija) rešenja za svoje zaposlene, oni uglavnom koriste iste uređaje za svoje privatne i za poslovne potrebe, što podržano potrebom za stalnom dostupnošću, dodatno povećava rizik od fizičke kompromitacije uređaja [69]. Slično, politika unošenja i korišćenja ličnih uređaja radi smanjenja troškova kompanije i povećanja komfora zaposlenih (Bring your own device - BYOD), u slučaju prethodne kompromitacije istih, može ugroziti interne sisteme organizacije, čak i kada su oni adekvatno obezbeđenih od spoljnih pretnji.

Savremeni mobilni telefoni i pored činjenice da predstavljaju moćne računare, ipak imaju brojna ograničenja. Osnovno ograničenje je kapacitet baterije, koji direktno utiče na autonomiju odnosno komfor u radu [46]. Svaki takt procesora, osvežavanje piksela na ekranu ili izmena sadržaja memorije koristi energiju iz istog izvora, odnosno baterije čije su fizičke dimenzije, a samim tim i kapacitet ograničeni. Takođe, često punjenje, pa makar ono bilo i ubrzano naročitim tehnološkim postupcima, predstavlja opterećenje za korisnika odnosno narušava "mobilnost" samog uređaja. Ova fizička ograničenja imaju veliki uticaj na projektovanje hardvera i softvera koji se koristi u mobilnim telefonima, odnosno zahtevaju konstantne proračune potrošnje koju će neka nova opcija uneti u sistem.

Pri razvoju aplikacija uvek trba imati u vidu da su mobilni telefoni značajno ograničeni resursima.

Ograničena procesorska snaga, veličina ekrana, mrežni protok i pokrivenost kao i autonomija uređaja imaju velikog uticaja na odluke koje programer donosi, odnosno kompromise koje je prinuđen da napravi [21]. Povrh toga, različite platforme i verzije operativnih sistema prisutne njima su suprotstavljene tržišnim zahtevima za globalnim prisustvom, odnosno podrškom za što veći broj uređaja. Da bi se izbegle brojne poteškoće i visoka cena razvoja iste aplikacije za različite platforme, nova paradigma nalaže razvoj specifičnih Internet (WEB) aplikacija koje uz minimalne modifikacije mogu raditi na svim uređajima uz minimalne gubitke performansi.

Ovaj trend ipak unosi nove probleme, od kojih je najveći neophodnost konstantne veze sa Internetom, ali i dodatnu kompleksnost koja se ogleda u prilagođavanju različitih interfejsa za kontrolu funkcionalnosti uređaja putem ugrađenih Internet pretraživača odnosno njihovih komponenti (WebViev). U praksi, softverska razvojna okruženja koja podržavaju ovaj pristup (PhoneGap, Ionic, Cordova, React Native itd.) tek delimično omogućavaju kreiranje višeplatformskih aplikacija sa performansama bliskim izvornim (Native), a taj problem se uvećava kako raste kompleksnost same aplikacije [21]. Takođe, nekonzistentne interpretacije HTML-a od strane različitih Internet pretraživača, što je problem odavno prisutan ali još uvek nerešen u desktop okruženjima, se u slučaju WEB aplikacija prenose i na mobilne uređaje, gde ih je znatno teže razrešiti usled komplikovanog testiranja na različitim uređajima.

3.3. Izazovi i pretnje bezbednosti mobilnih aplikacija

3.3.1. Ugroženost podataka

Korisnici mobilnih uređaja se susreću sa drugačijom vrstom rizika u odnosu na one prisutne pri upotrebi desktop računara. Osim očigledno uvećanog rizika od krađe ili gubljenja uređaja, ovi događaji, pored materijalne štete, nose mogućnost ugrožavanja privatnosti ili krađe bitnih podataka. I sama priroda informacija koje se pohranjuju na mobilnim uređajima je drugačija od onih zabeleženih u standardnom računaru [18]. Može se reći da su one "ličnije" jer osim e-mail poruka sadrže i SMS/MMS poruke, imenik, često sa adresama i ličnim podacima poznanika, fotografije, govorne poruke, posredne ili direktne podatke o kretanju vlasnika itd. Zbog česte upotrebe mobilnih uređaja ovi podaci su uvek svežiji i brojniji nego oni na desktop računaru.

Lične informacije možemo definisati kao one poznate korisniku i ograničenom broju ljudi, uglavnom iz najužeg kruga članova porodice i prijatelja. Ukoliko ove informacije postanu dostupne drugim

licima, uglavnom ne mogu biti zloupotrebljene na način koji bi izazvao više od neprijatnosti za korisnika.

Osetljive informacije su one koje se uglavnom ne dele ni sa kim. To mogu biti lozinke, PIN kodovi za bankarske usluge, brojevi telefona dece ili bliskih lica, matični broj (u pojedinim državama) itd. Šira dostupnost ovih podataka može biti zloupotrebljena na način koji korisniku može da nanese trajnu i nenadoknadivu štetu, te se oni uz posebnu pažnju moraju čuvati u svakom segmentu obrade.

Posebno su ugroženi korisnici koji čuvaju poslovne podatke na svojim mobilnim uređajima. Rasprostranjena je praksa slanja lozinki i drugih pristupnih parametara izolovanim korporativnim mrežama putem e-maila, a isto tako često je neoprezno čuvanje ovakvih poruka na privatnom mobilnom uređaju [18]. Slično važi i za podatke koji mogu predstavljati poslovnu tajnu, odnosno uvidom u koje, potencijalni konkurent može ostvariti korist odnosno naneti štetu kompaniji korisnika. Podaci koje mobilni uređaj sadrži nisu ugroženi samo u slučaju fizičkog pristupa uređaju, već postoje mnogi drugi vektori napada čijom realizacijom se do njih može doći.

Posebnu kategoriju potencijalno osetljivih podataka čine takozvani metapodaci, odnosno oni koji pružaju informacije o drugim podacima. Neočekivano su dospeli u fokus pažnje šire javnosti nakon nedavnih otkrića o zloupotrebama istih od strane raznih korporacija, državnih agencija i sofisticiranih napadača. Metapodaci nastaju prilikom uobičajene upotrebe komunikacionih sredstava i drugih tehničkih uređaja odnosno aplikacija [60]. Oni nose informacije o kreiranju, prenošenju i distribuciji poruke, kao i o lokaciji odakle je komunikacija izvršena. Takođe, moguće ih je tumačiti i u drugim kontekstima, jer svaka aktivnost na nekom elektronskom uređaju (npr. otvaranje fajla) najčešće kreira dodatne (meta)podatke, a da korisnik toga nije ni svestan. Ovo je naročito izraženo prilikom bilo kakvih aktivnosti na Internetu.

Savremena shvatanja ovog pojma naglašavaju neodvojivost metapodataka od samog sadržaja komunikacije, predstavljajući ih kao kontinuum, što ima značajne tehnološke i pravne posledice tokom manipulacije istima. Mišljenja eksperata iz sfera boraca za slobodu izražavanja, kriptografske zajednice, pravnih i bezbednosnih organa su uglavnom nepodeljena oko tvrdnji da metapodaci često mogu odražavati precizniju sliku o životu pojedinca, od konkretnog sadržaja njegovih neposrednih komunikacija. Gomilanjem ovih podataka raste i njihova upotrebljivost, jer se lakše uviđaju obrasci koje je moguće interpretirati na individualnom ali i na širem planu, društvenih grupa pa i čitavih populacija. Takođe, dokazano je [60] da se sa dovoljnom količinom relevantnih metapodataka obesmišljavaju mnoge mere anonimizacije i očuvanja privatnosti koje sprovode sami korisnici ili

servisi kojima pristupaju. Analizom društvenih mreža (Social network analysis), odnosno automatskim kreiranjem grafikona povezanosti konkretnog pojedinca, moguće je relativno jednostavno otkriti njegove neposredne veze čak i u slučajevima kada one sprovode određene mere da do toga ne dođe.

Većina ovih podataka biva u nekom obliku sačuvana na samom uređaju, čega korisnik uglavnom nije svestan. Šta više, najčešće ih nije moguće ukloniti bez posebnih sofisticiranih postupaka na administratorskom nivou, a čak i kada se to učini, prestaće da funkcioniše većina aplikacija kojima su ti podaci neophodni, ili će jednostavno baza ovih metapodataka biti ponovo kreirana.

3.3.2. Vektori napada i matrice pretnji

Da bi efikasno odbranila svoje informacione sisteme, svaka organizacija mora da okarakteriše svoje eventualne protivnike, njihovu motivaciju i vrste, odnosno vektore, napada koje mogu da izvrše. Napadači mogu biti konkurenti, kriminalci, hakeri, teroristi pa čak i grupe sponzorisane od strane neke države. Njihova motivacija varira od prikupljanja podataka, krađe intelektualne svojine, uništavanja infrastrukture, onemogućavanja određene usluge pa sve do tretiranja teškog oštećenja tuđeg informacionog sistema kao svojevrsnog postignuća za sebe, tj. trofeja u određenim hakerskim krugovima. Sredstva koja u te svrhe koriste su raznovrsna, ali ipak prilagođena sistemu koji se napada i mogućnostima, tehničkim, pravnim i finansijskim samog napadača. To mogu biti pasivno nadziranje komunikacija, aktivni napadi na mrežnu i softversku infrastrukturu, korišćenje lica sa neposrednim pristupom sistemu, uspostavljanje kontrole nad tehničkim i komunikacionim sredstvima u vlasništvu treće strane a od kojih meta neposredno zavisi itd. [1]. Od faktora koji su pod kontrolom organizacije ili pojedinca koji želi da zaštiti svoj informacioni sistem posebno treba izdvojiti izbegavanje ili makar blagovremeno otkrivanje propusta u aplikacijama i sistemima koji ih pokreću.

Bezbednosni propusti u softveru mogu ugroziti finansijsku, zdravstvenu, odbrambenu, energetsku, kao i bilo koju drugu kritičnu infrastrukturu [42]. Sa uvećanjem kompleksnosti i povezanosti aplikacija, postizanje bezbednosti celokupnog sistema postaje eksponencijalno komplikovanije. Takođe, zahtevi tržišta za što bržim razvojem softvera povećavaju mogućnost za greške i smanjuju verovatnoću njihovog otkrivanja i otklanjanja pre implementacije.

Jedan od kvalitetnih alata koji može pomoći pri projektovanju i testiranju aplikacija je lista deset najzastupljenijih tipova ranjivosti koje su prisutne u savremenim softverskim sistemima. Ovu listu održava i ažurira neprofitna zajednica OWASP (The Open Web Application Security Project), koja

je posvećena pružanju podrške organizacijama i pojedincima koji žele da razvijaju, održavaju i koriste softverske pakete na bezbedan način.

U dokumentu koji se objavljuje svake godine, OWASP prezentuje saznanja uvaženih svetskih eksperata iz različitih oblasti informacione bezbednosti, nastojeći da uspostavi konsenzus oko aktuelnih kritičnih rizika prema kojima sve odgovorne kompanije trebaju da modifikuju svoje bezbednosne i razvojne strategije. Usled relevantnosti ove liste, detaljnije ćemo opisati svaku stavku, ukazujući na njenu relevantnost u kontekstu mobilnih aplikacija i kriptografskih tehnika zaštite koje se u njima mogu implementirati. Takođe, bitno je napomenuti da osim navedenih, postoje i brojni drugi mogući propusti koje ova lista ne obuhvata, a koji podjednako mogu biti iskorišćeni za narušavanje bezbednosti informacionih sistema.

1) Ranjivosti prilikom izvršenja dodatnih upita u bazi podataka (**injection**) se javljaju kada se poslati podaci maliciozno pripreme na poseban način tako da ih interpreter na serverskoj strani tumači kao komandu, a ne kao informaciju koju treba uneti u bazu. Zavisno od sistemskih podešavanja, na ovaj način se može izvršiti nedozvoljeno čitanje, upis, izmena ili brisanje podataka, a nije retko da se jednom komandom može uništiti veći deo sadržaja cele baze.

Ove probleme je relativno jednostavno uočiti i otkloniti tokom izrade ili analize koda klijentske, a pogotovo serverske strane aplikacije, u čemu mogu pomoći i specijalizovani alati koji automatizuju deo tog procesa. Konkretne ranjivosti se manifestuju u slučajevima kada podaci dostavljeni od strane korisnika nisu validirani, filtrirani ni prečišćeni od strane aplikacije, a njihova obrada odnosno interpretacija nije podvrgnuta strogoj separaciji na komande i parametre. Ponekad propusti ovog tipa ipak nisu očigledni, jer mogu postojati duboko u sistemu za objektno relaciono mapiranje (ko što je npr. Hibernate u Java okruženju), ili se nesanitizovani ulazi mogu naći kao parametri dinamički kreiranih upita ili snimljenih procedura u samoj bazi podataka.

Mere zaštite podrazumevaju rigoroznu kontrolu korisničkih podataka pre svega na serverskoj strani, striktnu separaciju komandi i parametara u delovima aplikacije koji komuniciraju sa bazom podataka, kao i strukturisanje samih upita na način koji umanjuje ili onemogućava zloupotrebe ove vrste.

2) Mehanizmi autentifikacije i upravljanja sesijama su često loše implementirani (**broken authentication**) što omogućava napadačima da dođu do lozinki, ključeva, sesijskih identifikatora (tokena) ili da preuzmu identitet nekog drugog korisnika ili administratora, te da u njegovo ime i sa njegovim ovlašćenjima izvrše određene radnje na sistemu.

Ukoliko je autentifikacija realizovana na neadekvatan način, moguće su razne vrste napada kojima se mogu dobiti ili povećati privilegije određenih korisnika sistema. Najjednostavniji, a ponekad ipak efikasni su napadi primenom grube sile (bruteforce), kada se pokušavaju brojne kombinacije korisničkih imena i lozinki automatskim putem i velikom brzinom. Nešto sofisticiranije varijacije podrazumevaju pripremljene liste najčešće korišćenih lozinki ili pak pokušaje pogađanja aktivnih sesijskih tokena na sličan, automatizovan način.

Neki od najčešćih propusta su korišćenje slabih lozinki, njihovo čuvanje u otvorenom (plaintext) obliku, nepostojanje višefaktorske autentifikacije, neadekvatno kreiranje i postupanje sa identifikatorima sesije, propusti u kodu koji omogućavaju saznavanje korisničkih imena ili identifikatora drugih korisnika, ostavljanje aktivnim podrazumevanih sistemskih ili administratorskih naloga, dozvoljavanje neograničenih pokušaja logovanja, itd.

Problemi ovog tipa su veoma rasprostranjeni u savremenim web aplikacijama, jer autentifikacija često predstavlja nezaobilazan deo svake korisničke interakcije sa sistemom. Da bi se autentifikacija ispravno izvršila, neophodno je potvrditi identitet korisnika, utvrditi nivo prava pristupa koji mu je dodeljen i adekvatno voditi računa o uspostavljanju, praćenju i isteku sesije koju je uspostavio sa sistemom. Takođe, pravila za postavljanje lozinki moraju odražavati kompromis između komfora korisnika i bezbednosti, te se više ne preporučuje forsiranje izuzetno kompleksnih lozinki koje se moraju često menjati, već je najbolje koristiti dvofaktorsku autentifikaciju, najčešće putem SMS poruke ili token aplikacije na mobilnom telefonu. Što se tiče upravljanja sesijom, osim ustanovljenih dobrih praksi kreiranja i vremenski ograničene invalidacije na serverskoj strani, preporučuje se što restriktivniji pristup ovom problemu sa klijentske strane, odnosno korisnik biva automatski izlogovan kada isključi browser tj. kada mobilna aplikacija izgubi fokus.

3) Mnoge web i mobilne aplikacije ne pridaju dovoljno pažnje zaštiti osetljivih ličnih podataka, pa u pojedinim slučajevima veoma lako može doći do njihovog otkrivanja trećim licima koja ih mogu zloupotrebiti (**sensitive data exposure**). U poslednjih nekoliko godina ovo je postao najrasprostranjeniji tip napada kojim se postiže najveći efekat, odnosno čini šteta po ciljanu kompaniju ili pojedinca. Bilo da se radi o podacima u prenosu ili u mirovanju, većina problema koji prouzrokuju ovakve propuste potiču iz činjenice da u aplikacijama nisu implementirane nikakve kriptografske tehnike zaštite, ili u nešto ređim slučajevima, da su primenjene na neadekvatan način.

Veoma su retki direktni napadi na propuste u kriptografskim algoritmima pa i protokolima, mnogo

su češći pokušaji krađe ključeva i razne vrste napada tipa "čovek u sredini" (man in the middle), dok je u najvećem broju slučajeva moguće doći do podataka u otvorenom (plaintext) obliku zaobilaznim putem, odnosno kroz fajl sistem servera, tokom nezaštićenog prenosa nekim komunikacionim kanalom ili iz privremenih (cache) fajlova browsera ili aplikacije. Često fizički pristup nekim inače zaštićenim podacima (npr. šifrovane lozinke ili osetljivi podaci u bazi) otvara mogućnosti drugih, znatno efikasnijih napada, kao što je korišćenje snažnih računarskih resursa (GPU) radi otkrivanja tih podataka, a što je inače onemogućeno drugim tehnikama zaštite kada sistem nije kompromitovan.

Naročito štetna posledica ove grupe ranjivosti ogleda se u tome što u slučaju uspešnog napada najčešće dolazi do otkrivanja osetljivih podataka svih korisnika datog sistema, što može imati ozbiljne ekonomske i pravne posledice po kompaniju koja ih je prikupljala, obrađivala i čuvala. Detaljan opis konkretnih napada i tehnika zaštite koje spadaju u ovaj segment OWASP klasifikacije dat je na drugim mestima u ovom radu.

4) Mnoge starije web aplikacije, odnosno serverski deo istih, i dalje koriste podatke u XML formatu (**XML external entities**), a to čine na prevaziđen i nebezbedan način. Zavisno od konkretne implementacije, postoje slučajevi kada se postavljanjem specifičnog upita, serverska strana se može navesti na prikazivanje podataka u čiji uvid korisnik nema pravo, ili može dati informacije o konfiguraciji sistema odnosno topografiji lokalne mreže. Takođe, ukoliko dođe do nefiltrirane interpretacije XML upita, moguće je izvršavanje arbitrarnih komandi na serveru, ili izazivanje nestabilnosti sistema radi uskraćivanja usluge drugim korisnicima (denial of service).

Iako specifičan i ne preterano rasprostranjen, ovaj napad može biti efikasan, a njegovo predupređivanje komplikovano i skupo jer podrazumeva detaljne provere velike količine serverskog koda i eventualne migracije na savremenije formate obrade podataka kao što je JSON. Kao privremena rešenja mogu poslužiti implementacija strogog filtriranja i sanitizacije korisničkih upita na serverskoj strani i instalacija sofisticiranih sistema za detekciju pretnji na aplikativnom nivou (web application firewall – WAF).

5) Pri razvoju ili implementaciji web aplikacija često dolazi do propusta po pitanju kontrole pristupa (**broken access control**), pa eventualni napadači mogu pristupiti tuđim korisničkim nalozima, videti osetljive fajlove, modifikovati prava pristupa itd. Do ovoga najčešće dolazi usled teškoća automatizacije testiranja radi blagovremenog otkrivanja ovih slučajeva. Pažljivom proverom svakog upita koji korisnička strana može poslati serveru detektuju se pogrešna podešavanja i loše tretirani izuzeci koji mogu dovesti do izvršenja funkcija sa povišenim privilegijama.

Sve zaštite od ovih vrsta napada se mogu implementirati isključivo na serverskoj strani, dok se filtriranjem i sanitizacijom korisničkih upita pri slanju većina ozbiljnih napada može samo delimično usporiti ali ne i zaustaviti. Kontrola pristupa mora biti osmišljena i ugrađena u sistem uz poštovanje jasno definisanih dozvola i prava za svakog korisnika odnosno grupu korisnika u odnosu na dostupne resurse, dok podrazumevano pravilo treba da bude zabrana pristupa, koja se naknadno liberalizuje po potrebi. Kao i kod mnogih drugih ranjivosti, ispravna konfiguracija servera i višeslojna provera parametara pristupa može značajno pomoći u prevenciji napada ovog tipa.

6) Loša konfiguracija parametara sistema koji utiču na bezbednost (**security misconfiguration**) je veoma česta i može imati dosta različitih uzroka. Nepotpuna ili podrazumevana konfiguracija, preopširne poruke o greškama u produkcionom okruženju, odavanje previše informacija o postavkama sistema korisnicima, su veoma rasprostranjeni jer se mogu nalaziti u bilo kom segmentu sistema, ne samo konkretnoj aplikaciji.

Na decentralizovanim (cloud) platformama koje pokreću složene aplikacije ili mikroservise, a koji opet mogu biti sastavljeni iz brojnih softverskih paketa neujednačenog kvaliteta i nivoa primene bezbednosnih preporuka, ovi problemi postaju još izraženiji. Osim na aplikativnom nivou, propusti su mogući i na infrastrukturnom, te mrežni servisi, web serveri, baze podataka pa i delovi samog operativnog sistema mogu biti pokrenuti sa parametrima koji nisu namenjeni upotrebi u potencijalno rizičnom okruženju.

Iako najčešće samo dovode do prikaza informacija, koje mogu pomoći u naknadnoj fazi napada, ovi propusti sami po sebi takođe mogu dovesti do kompromitacije sistema. Loše podešene dozvole fajl sistema, aktivirani nepotrebni, a ranjivi servisi, podrazumevani nalozi i lozinke, kao i neblagovremena primena unapređenja instaliranih programa i biblioteka (security updates) mogu biti dovoljni za izvođenje veoma jednostavnih a apsolutno efikasnih napada.

7) Mogućnost integrisanja skripti i sadržaja sa drugih sajtova (**cross-site scripting – XSS**) od strane napadača ili drugih korisnika, može dovesti do kompromitovanja podataka sačuvanih u pretraživaču, odnosno mobilnoj aplikaciji ili do preusmeravanja, često neprimetnog, na sajt pod kontrolom napadača, radi sprovođenja drugih malicioznih tehnika, npr. krađe login podataka ili podmetanja modifikovanih verzija izvršnih fajlova koji se snimaju.

Ukoliko se blagovremeno sprovede, automatsko skeniranje aplikacija daje dobre rezultate, odnosno

otkriva većinu ovakvih propusta. Sa druge strane, ako se takva kontrola propusti, napadač isto tako može iskoristiti ova pomoćna sredstva. Osim uobičajenog načina, kada je propustom napadaču omogućeno da kontroliše nefiltrirani sadržaj koji će biti prikazan nekom drugom korisniku, postoji varijanta kada on sebi to isto omogući primenom nekog drugog napada, npr. presretanjem komunikacije ka nekoj lokaciji odakle se povlače skripte ili drugi fajlovi potrebni za prikaz sajta odnosno aplikacije.

Prevencija napada podrazumeva strogu separaciju podataka unetih od strane nekog korisnika od aktivnog prikaza u pretraživaču ili aplikaciji. Filtriranje ovog sadržaja pre prikazivanja moguće je automatizovati, pa mnoga savremena okruženja za razvoj softvera to i čine. Ipak, u delovima koda gde se ovim podacima ručno manipuliše, neophodno je primeniti adekvatnu sanitizaciju ulaza, bilo da dolaze od korisnika ili od eksternih dobavljača pomoćnih sadržaja kojima se implicitno veruje (npr. mreže za distribuciju sadržaja – CDN).

8) Nešto manje poznat i rasprostranjen, ali i dalje potencijalno ozbiljan napad je deserijalizacija podataka na nebezbedan način (**insecure deserialization**). Ukoliko se uspešno izvede, što je inače prilično komplikovano uraditi i praktično nemoguće automatizovati, ova tehnika može proizvesti teške posledice po bezbednost sistema, kao što su izvršavanje koda na serveru, izmena podataka u bazi i ostvarivanje administratorskog pristupa.

Odbrana je podjednako komplikovana kao i samo izvođenje napada, a podrazumeva detaljnu analizu serverskog koda koji se bavi deserijalizcijom, odnosno sprovođenje ozbiljnih arhitekturinih izmena aplikacije kako bi se tej deo izdvojio i izvršavao u virtuelnom okruženju sa nižim privilegijama. Takođe, kao i kod mnogih drugih napada, malo toga se može uraditi izmenama isključivo na strani odakle se šalju podaci jer taj segment nikada nije pod apsolutnom kontrolom.

9) Već pomenuta značajno uvećana kompleksnost savremenih aplikacija nosi sa sobom dodatne rizike u smislu korišćenja komponenti sa poznatim bezbednosnim propustima (**using components with known vulnerabilities**). Sve komponente jedne aplikacije, bilo da su u pitanju posebni moduli, biblioteke ili pomoćni delovi razvojnog sistema (framework) se uglavnom pokreću pod istim privilegijama, te kompromitacija bilo kog dela ugrožava integritet celokupne aplikacije. Šta više, ako su druge bezbednosne prakse dosledno sprovedene, jedan propust ovog tipa, koji se u principu lako eksploatiše, može onesposobiti čitav sistem. Takođe, programski kod pojedinih komponenti može biti javno dostupan, čime se potencijalnom veoma motivisanom napadaču može ostaviti mogućnost da ciljano konstruiše napad (exploit) baš za taj sistem, odnosno njegov deo.

Iako očigledne, tehnike zaštite se otežano sprovode usled kompleksnih načina distribucije i eksploatacije ovakvih segmentiranih sistema. Takođe nekompatibilnost pojedinih komponenti aplikacije sa novim verzijama nekih drugih delova, politika nabavke i licenciranja određenih paketa, kao i neprihvatljivi periodi nedostupnosti usled čestih obnavljanja softvera mogu značajno usporiti ili onemogućiti implementaciju zaštitnih mera. Čak i u idealnim uslovima, briga o poštovanju aktuelnih bezbednosnih preporuka ne može se u potpunosti prepustiti automatskim sistemima, već predstavlja celodnevni posao eksperata iz te oblasti, što je najčešće van budžeta mnogih kompanija.

10) Nedovoljno detaljno beleženje događaja i nadzor sistema (**Insufficient logging and monitoring**), pogotovo ako je propraćeno nepostojećim ili neefikasnim planovima odgovora na bezbednosne incidente, ostavlja mogućnost napadaču da pošto obezbedi uporište, odnosno probije zaštitu jednog, neometano nastavi sa kompromitacijom drugih sistema na toj ili drugim mrežama. Podaci istraživanja pokazuju da se uspešni napadi ne otkriju i po dve stotine dana od napada, ukoliko ne postoje jasne i adekvatne politike detektovanja odnosno logovanja sumnjivih aktivnosti na sistemima.

S obzirom da je prva faza svakog napada prikupljanje podataka, najbolje je postaviti mehanizme koji upozoravaju na ove aktivnosti i primenjuju odrađene protivmere (npr. blokiranje IP adrese) u realnom vremenu. Dobra praksa za implementaciju ovih metoda je primena aktivnog testiranja na poznate ranjivosti (penetration testing), jer se prilikom ovog procesa, koji se u suštini ne razlikuje od pravog napada, kreiraju logovi sa svim relevantnim podacima o pokušanim napadima i njihovom eventualnom uspehu.

Rizik	Tehnika zaštite	Implementacija	Praktični problemi
Fizički pristup uređaju	Enkripcija podataka u mirovanju.	Na sistemskom i aplikativnom nivou. Full device encryption ili različita userspace rešenja.	Čuvanje ključa na samom uređaju nepouzdano. Izvođenje ključa iz lozinke, biometrije ili PIN-a problematično.
Fizički pristup uređaju	Drugi faktori autentifikacije, nešto što korisnik zna ili jeste. PIN, lozinka ili biometrija.	Na sistemskom i aplikativnom nivou. PIN, lozinka, pattern i biometrija podržani na mnogim novijim uređajima.	Pamćenje lozinke. Procedure u slučaju zaboravljanja. Nepouzdana biometrija i problemi privatnosti prilikom prikupljanja i čuvanja biometrijskih podataka.
Napadi na nivou sistema (root pristup)	Enkripcija podataka u mirovanju. Drugi faktori autentifikacije. Tehnike zaštite koda i defanzivnog programiranja.	Na aplikativnom nivou.	Komplikovan sistem prava i privilegija otežava implementaciju i korišćenje aplikacija.
Napadi na aplikativnom sloju.	SSL/TLS napredne mogućnosti. Dodatni slojevi enkripcije.	Poštovanje dobrih bezbednosnih praksi i preporuka pri implementaciji viših mrežnih protokola.	Komplikovana implementacija, zahteva izmene na serverskoj strani.
Side channel napadi.	Napredne mogućnosti kriptografskih klasa i funkcija.	Poštovanje dobrih bezbednosnih praksi pri implementaciji kriptografskih funkcija.	Nemogućnost predviđanja svih vektora napada. Komplikovane i spore izmene hardvera i sistemskog softvera.
Napadi na mrežnu infrastrukturu.	SSL/TLS napredne mogućnosti. Dodatni slojevi enkripcije.	Poštovanje dobrih bezbednosnih praksi i preporuka pri implementaciji transportnih protokola. Adekvatna (bezbedna) konfiguracija mrežnog okruženja.	Nekompatibilnost protokola, komplikovana implementacija, zahteva izmene na serverskoj strani.

Tabela 1 Tehnike zaštite

3.3.3. Zapostavljanje bezbednosti pri razvoju aplikacija

Greške u implementaciji kriptografskog softvera najčešće obezvređuju sve mogućnosti obezbeđenja podataka putem kriptografskih tehnika zaštite. Analizom 269 prijavljenih ranjivosti putem CVE baze

u periodu od januara 2011. godine do maja 2014. godine utvrđeno je da samo 17% odlazi na greške u samim kriptografskim bibliotekama, dok je preostalih 83% posledica loše implementacije u specifičnim aplikacijama, pre svega u delu koji se tiče zaštite komunikacionih kanala [29].

Postoje primeri aplikacija (Apple TLS implementacija, GnuTLS) u kojima je samo jedna pogrešno napisana linija koda rezultovala greškom koja je u potpunosti obesmislila implementirani sistem zaštite podataka, čime je bilo pogođeno više miliona korisnika na duži vremenski period. Tehnike za prevenciju ovakvih grešaka podrazumevaju testiranje, statičku analizu, formalnu potvrdu i redizajn aplikacionog interfejsa (API). Čak i kada su SSL/TLS biblioteke ispravno implementirane na sistemskom nivou, postoji mogućnost da ih sama aplikacija neće adekvatno iskoristiti. Ovaj slučaj nije ograničen na proveru serverskih sertifikata, odnosno zaštitu komunikacionog kanala, takođe je rasprostranjeno neproveravanje digitalnih potpisa aplikacija koje se instaliraju na sistem.

Iako su greške u samim kriptografskim bibliotekama retke, ukoliko do njih dođe one po pravilu imaju dalekosežne posledice usled njihove rasprostranjene upotrebe i komplikovanog postupka izmene implementiranih verzija. Neke od ovih grešaka obuhvataju slabe algoritme za enkripciju, loše generatore pseudoslučajnih brojeva (PRNG), slabe (kratke) ključeve i slično. Interfejsi biblioteka koje pretenduju da reše neke od navedenih problema moraju u svojim podrazumevanim režimima rada pružati jake šifre i generatore pseudoslučajnih brojeva, imati dovoljne izvore entropije i izbegavati ponavljanje ključeva i drugih kriptografskih parametara. Takođe, neophodno je da pružaju mogućnosti autentifikacije poruka i provere sertifikata.

S obzirom da su greške u kriptografskim bibliotekama retke, te da je njihovo otklanjanje najčešće van mogućnosti inženjera angažovanih na izradi konkretne aplikacije koja ih koristi, rezultati mnogih studija ukazuju da se najznačajniji efekat povećanja bezbednosti mobilnih aplikacija može postići izbegavanjem i predupređivanjem grešaka i propusta na nivou implementacije kriptografskih tehnika zaštite komunikacionih kanala odnosno protokola.

Upotreba HTTPS protokola u aplikacijama se u velikom broju slučajeva može naknadno implementirati bez značajnijih intervencija u samom kodu. Šta više, istraživanje [57] je pokazalo da je u 90% posmatranih aplikacija bilo dovoljno dodati slovo "S" u postojeće HTTP linkove, da bi postojeće biblioteke realizovane zadovoljavajuće bezbednu konekciju ka serveru.

Problem nastaje kada se vrše određene izmene sistemskih biblioteka, odnosno menadžera poverenja (trust manager) kako bi se izbegla pojedina ograničenja koja nameće SSL-TLS standardizacija. Ovo

se najčešće čini privremeno, u fazi razvoja, pa se ne izmeni prilikom objavljivanja finalnog proizvoda, dok takođe nisu retki slučajevi kada se mehanizmi poverenja svesno isključuju radi olakšanja rada li se to događa usled neznanja programera. Najčešće greške koje se prave su prihvatanje svih sertifikata, tolerisanje neusklađenosti sertifikata sa domenom na kojem se servis realno nalazi kao i ignorisanje ostalih grešaka koje se mogu javiti prilikom SSL-TLS komunikacije. Može se reći da je izrada aplikacije na ovaj način suštinski ista kao i ponašanje neodgovornog korisnika koji ignoriše upozorenja browser-a [57]. Ovi postupci obesmišljavaju celokupan sistem poverenja i čine raznovrsne napade na privatnost komunikacija mogućim pa i trivijalnim.

Ne treba gubiti iz vida činjenicu da postoje brojna scenarija kada prilikom razvoja mobilne aplikacije nije moguć nikakav uticaj na serversku stranu, odnosno neophodno je prilagoditi se postojećem stanju i implementiranim tehnologijama. Očigledno je da se ova situacija ne može ispraviti bilo kojim postupcima sa isključivo klijentske strane. Iako broj sajtova zaštićenih HTTPS protokolom raste, ovaj pozitivan trend se nešto sporije prihvata kada su u pitanju backend sistemi, odnosno API tačke pristupa (endpoint) koji nisu vidljivi krajnjem korisniku, već im pristupaju samo aplikacije.

Nevezano za isključivo kriptografske tehnike, nije retka praksa da se aplikacije sa uključenim opcijama za detaljno praćenje grešaka (debugging) nađu u produkcionom okruženju, što može biti izvor ključnih informacija potencijalnom napadaču. Ove i slične programerske greške su očekivane u kompleksnom razvojnom okruženju kao što je Android, mada uprkos tome, postoje brojni pojedinci i kompanije koje odbijaju da ih prihvate kao neminovnost, te odbijaju da otklone bezbednosne propuste na koje im skrenu pažnju korisnici ili istraživači.

Još jedan rasprostranjen tip loših praksi, koji možemo podvesti pod ljudski faktor, je nespremnost programera i menadžera da inoviraju znanje u smeru aktuelnih bezbednosnih izazova, ili da čak i kad to znanje poseduju, nevoljno menjaju ustaljene obrasce rada i kodifikovane protokole kontrole.

3.3.4. Mogućnosti zloupotrebe PKI modela

U sistemu zasnovanom na poverenju u sertifikacione autoritete (CA) postavlja se pitanje da li je odgovorno verovati da oni uvek savesno obavljaju svoj posao, odnosno da im je poštovanje interesa korisnika prioritet. Nekoliko poznatih primera pokazuje da postoje institucije ovog tipa koje su zloupotrebile ovo poverenje radi realizacije kratkoročnih finansijskih interesa. Npr. kompanija Trustwave je 2012. godine javno priznala da je prodavala posredničke (intermidiate) sertifikate firmama koje se bave proizvodnjom i prometom opreme za vršenje nadzora nad Internet

komunikacijama [45], pružajući im mogućnost da svoje aktivnosti vrše bez dodatnih intervencija nad ciljanim sistemima.

Osim svojevoljnog, postoje brojni primeri nenamernog narušavanja poverenja od strane CA, do koga dolazi usled tehničke greške ili uspešnog ciljanog napada. Falsifikovani sertifikati izdati na ovaj način mogu ostati neotkriveni veoma dugo, ukoliko se koriste radi nadzora ograničenog broja meta. Posebno zabrinjava činjenica da kompromitacija CA tela može pomoći sofisticiranim napadačima da zaobiđu napredne tehnike odbrane kao što su kačenje (pinning) sertifikata ili ključa (ukoliko se proverava neki od sertifikata viših po hijerarhiji) i DNSSEC sa DANE ekstenzijom, (u slučajevima kada je i pravi i lažni sertifikat izdao isti CA).

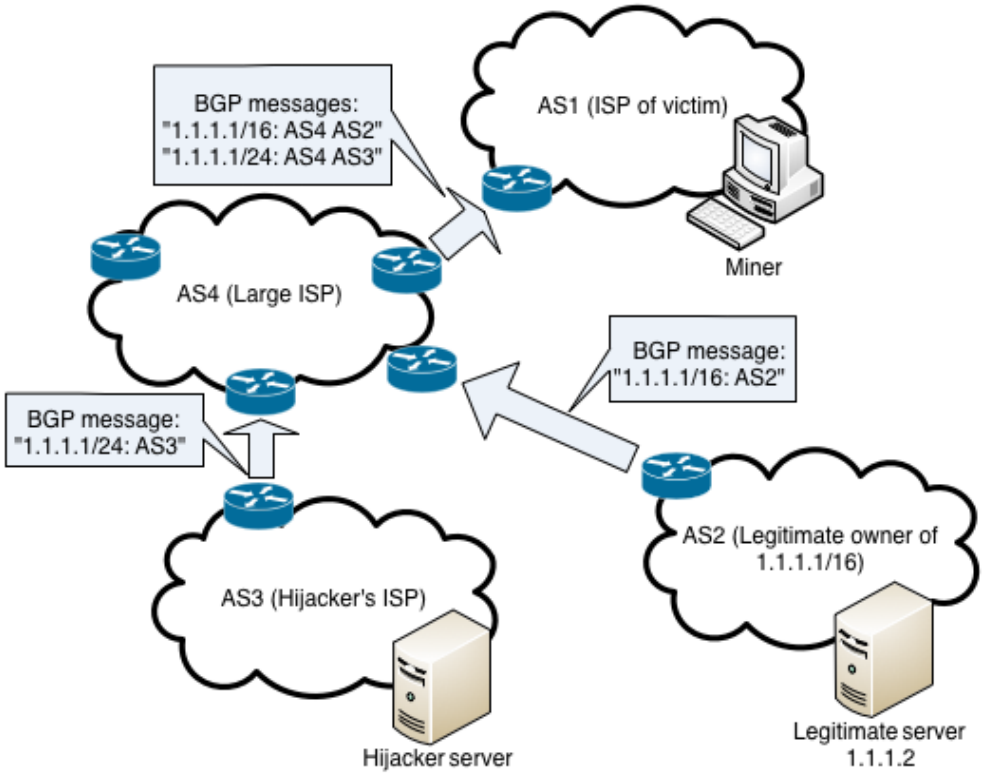
Još 2003. godine prikazani su izvodljivi napadi grubom silom (bruteforce) na 1024-bitne sertifikate, koje bi mogao izvršiti motivisani napadač sa pristupom adekvatnoj hardverskoj infrastrukturi (tada u vrednosti od oko deset miliona dolara, a danas mnogo manje) [45]. Iako je većina ovakvih sertifikata povučena iz upotrebe, ne treba isključiti mogućnost da su neki od njih bili "provaljeni" i nenamenski iskorišćeni.

Mnogo verovatniji i praktičniji napadi na PKI infrastrukturu su oni koji se izvode u lokalnim (najčešće korporativnim) mrežama, a vrše se putem zlonamernog softvera, drugih korisnika ili administratora sistema. Postoje opravdana scenarija kada se vrše manipulacije sistemom poverenja, npr. prilikom automatske inspekcije saobraćaja radi uklanjanja virusa, ali čak i tada se korisnici veoma retko obaveštavaju o prisustvu uređaja koji te provere vrše [45]. Ova praksa faktički predstavlja izvođenje Man in the middle napada u kontrolisanom okruženju, što je vrlo slično scenariju koji ćemo detaljno obraditi u praktičnom delu ovog rada. Čak i kada se vrši bez prethodne kompromitacije ili preinstalacije CA sertifikata, većina korisnika (i aplikacija) ignoriše upozorenja Internet pretraživača, čime ovaj napad postaje izvodljiv i privlačan velikom broju potencijalnih napadača.

Jedan od manje poznatih napada na sistem poverenja infrastrukture javnih ključeva (PKI) je preuzimanje ruta komunikacije, odnosno presretanje saobraćaja koji se između Internet provajdera odvija putem BGP protokola, a u cilju manipulacije procesom verifikacije domena za koji CA izdaje sertifikat. Ukoliko je uspešan, napadač će imati validan sertifikat za domen koji realno ne poseduje [2].

S obzirom da BGP protokol služi za razmenu informacija o logičkoj lokaciji i dostupnosti autonomnih sistema (AS) putem adresnih prefixa koji su im dodeljeni, očigledan vektor napada je preuzimanje

saobraćaja namenjenog domenu mete putem objavljivanja specifičnijeg IP prefixa koji sadrži IP adresu ciljanog domena. Na ovaj način sva komunikacija upućena ka meti biva preusmerena ka napadaču jer BGP ruteri uvek daju prednost specifičnijim odrednicama krajnje tačke.



Slika 3 BGP redirekcija [70]

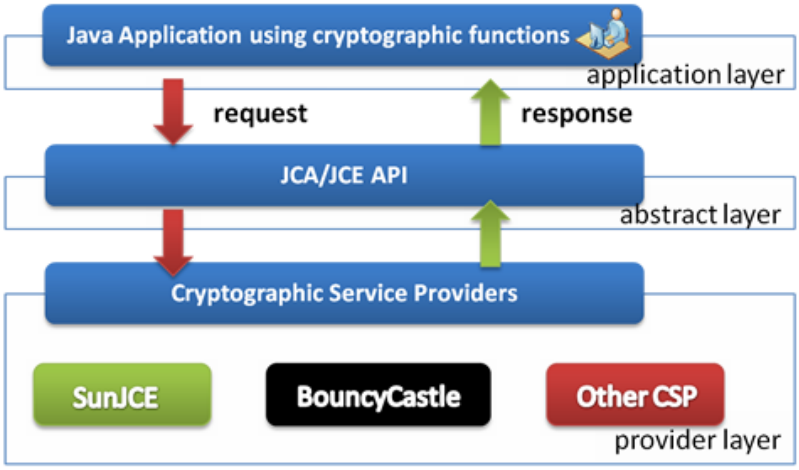
Istraživanje [cl-Birge-Lee] je pokazalo da nijedno autorizaciono telo ne primenjuje nikakve mere za sprečavanje ovog veoma sofisticiranog, ali izvodljivog i efikasnog napada.

3.4. Primena kriptografije u mobilnim aplikacijama

3.4.1. Dostupna rešenja

Osnovne kriptografske funkcionalnosti u Java, odnosno Android, okruženju dostupne su kroz biblioteke JCA (**Java Cryptography Architecture**) i JCE (**Java Cryptography Extension**) [20]. Njih karakteriše izostanak strogih definicija podržanih algoritama pa čak i interfejsa ka klasama koje ih implementiraju, odnosno, one predstavljaju sloj apstrakcije (abstraction layer) izložen programeru putem kojeg se mogu kreirati potrebne klase. Ovo je tzv. provajderska (provider) arhitektura, što znači da su ove biblioteke zadužene za instanciranje objekata koji neposredno primenjuju kriptografske funkcije, te da oni nisu direktno dostupni kodu koji piše programer. Ove kolekcije klasa koje se sadrže u JCA i JCE bibliotekama se i nazivaju provajderi, a njihov izbor ili izmena je

omogućena putem jednostavnih mehanizama.



Slika 4 JCA/JCE Arhitejtura [67]

Osnovni razlog ovakve arhitekture nije tehničke već političke prirode. Naime, u periodu razvoja JCA i JCE biblioteka bile su na snazi veoma stroge izvozne restrikcije po pitanju softverskih proizvoda koji sadrže kriptografske funkcije. Šta više, sredinom 90-ih godina iz SAD nije bilo moguće izvoziti ni softver koji se lako može prilagoditi, odnosno proširiti, kriptografskim funkcijama, već se isti mogao objavljivati samo u sklopu naučnih publikacija. Iz navedenih razloga, JCA je u prvoj verziji sadržala samo komponente koje se bave autentifikacijom, dok su kriptografske funkcije došle kasnije u vidu ekstenzije JCE.

Takođe treba imati u vidu da u pojedinim državama postoje uvozne restrikcije na ovu vrstu softverskih proizvoda, te da je u njima veoma teško ili nemoguće na legalan način koristiti komercijalne aplikacije koje sadrže jaku kriptografiju. Zato je i danas, kada je otklonjena većina izvoznih ograničenja, čest slučaj da se kriptografske biblioteke podrazumevano distribuiraju u oslabljenoj verziji (pre svega po dužini ključa), te da ih je neophodno naknadno unaprediti kako bi se iskoristio sav njihov potencijal pri izradi kriptografski zaštićenih aplikacija.

Sledeći set ograničenja koji je usporavao razvoj i distribuciju kriptografskih alata u Java i drugim softverskim okruženjima bili su patenti, kojima su najviše bili pogođeni algoritmi, kao npr. RSA. Patentima je onemogućena njihova direktna implementacija ili adaptacija te su mogli da se koriste samo u izvornom obliku putem priloženih API interfejsa [20]. Ovo je upravo i bio ključni razlog da se biblioteke struktuiraju na način koji omogućava da se neophodni objekti instanciraju isključivo u propisanom obliku putem factory obrasca (pattern), a povoljni propratni efekat je mogućnost da se provajderi izmenjuju prema potrebama konkretne aplikacije odnosno uslova tržišta na koje se plasira.

U praksi to znači da se izmenom jednog stringa u konfiguracionom delu koda aplikacije može izmeniti verzija kriptografskog algoritma koji će se koristiti, bez pojedinačnog podešavanja svih neophodnih parametara za njegov pravilan i bezbedan rad. Slično, novi algoritmi ili njihove verzije se lako mogu dodavati jer ovakav arhitekturalni koncept omogućava visoku modularnost koda.

Biblioteka **Bouncy Castle** je dobar primer efikasno sprovedene modularizacije i implementacije provajderskog modela, jer u svojoj jednostavnijoj, (lightweight) verziji sadrži 23 klase koje neposredno izvršavaju kriptografske funkcije, dok ih programer može pozivati odnosno koristiti na jednostavan način, bez potrebe za detaljnim poznavanjem njihove interne strukture. Bouncy Castle paket kriptografskih funkcija razvijen je od strane neprofitne organizacije Legion of the Bouncy Castle sa ciljem da olakša implementaciju kriptografskih funkcija u Java okruženju, nezavisno od ograničenja koja su imali slični paketi a uz poštovanje FIPS standarda.

Iako se sastoji od skoro pola miliona linija koda, biblioteka Bouncy Castle je relativno laka za integraciju u Java, odnosno Android aplikacijama. Funkcionalnostima koje nudi može se pristupiti putem tri grupe API-ja. Prva grupa omogućava direktno korišćenje kriptografskih servisa, druga predstavlja provajdera tih funkcija koji se poziva putem postojeće JCA/JCE arhitekture, dok je treća grupa zadužena za rad sa kriptografskim protokolima kao što su OpenPGP, S/MIME, TSP itd., kao i za generisanje, iščitavanja i proveru sertifikata u različitim formatima i standardima [59]. Bitno je napomenuti da fleksibilnost koju ova biblioteka nudi po pitanju odabira algoritama, njihovih verzija i protokola u kojima će se primenjivati ne narušava bezbednost sistema u kome se implementira, jer suštinski ni u jednom trenutku ne dolazi do izmene samih algoritama, a što je najčešći izvor problema i ranjivosti u implementaciji kriptografskih tehnika zaštite.

Usled određenih odluka donetih u ranim fazama razvoja Android platforme, kada su tržištem dominirali uređaji slabijih performansi, a pojedini patenti još uvek bili na snazi, uz većinu Android razvojnih softverskih okruženja i dalje dolazi oslabljena verzija Bouncy Castle biblioteke [59]. Usled navedenog, preporuka je koristiti modifikovanu i potpuno kompatibilnu verziju Spongy Castle, koja se takođe distribuira po sistemu otvorenog koda, a koja nema ograničenja po pitanju dostupnih algoritama i njihovih verzija.

3.4.2. Enkripcija uređaja i statičnih podataka

Kao što smo već pomenuli, Android sistem implementira određene mehanizme segregacije podataka različitih aplikacija, te ih one mogu razmenjivati samo putem naročitih kodifikovanih protokola

poštujući pritom sistem dozvola i određivanja prava pristupa svakom resursu. Sve ove zaštite podrazumevaju da je sistem u aktivnom i ispravnom stanju, te de se podacima pristupa regularnim, kontrolisanim, kanalima.

S obzirom da opisano stanje nije nužno uvek ispoštovano, moguće su brojne zaobilazne tehnike, odnosno ranjivosti koje omogućavaju pristup podacima u mirovanju (data at rest) na android platformi. Svi podaci na ovim uređajima su pohranjeni na nekom obliku uređaja za snimanje velike količine podataka. To može biti interna memorija, odnosno memorijski čip integrisan na matičnu ploču uređaja, ili eksterna memorijska kartica, najčešće dosta većeg kapaciteta.

U podrazumevanoj konfiguraciji, podaci koji se upisuju na ove medije nisu posebno zaštićeni od direktnog iščitavanja na drugom uređaju, u slučaju memorijske kartice, ili neposrednim pristupom hardveru, što je komplikovanije ali i dalje dostupno napadaču koji ima fizički pristup uređaju. Podsetimo da ovi podaci mogu obuhvatati mnoge privatne i osetljive informacije, za koje krajnji korisnik ne mora biti ni svestan da su sačuvane od strane neke aplikacije koju je koristio u jednom trenutku. Potreno je napomenuti da memorijske kartice uglavnom koriste jednostavne fajl sisteme koji ne podležu strogim separacijama privilegija i kontroli prava pristupa, što znači da svaka aplikacija kojoj se odobri pristup kartici, ima pristup svim podacima na kartici, bez obzira koja aplikacija ih je kreirala.

Takođe na većini android uređaja je relativno jednostavno pribaviti administratorski (root) nalog, bilo putem aplikacije ili specijalnim postupkom koji zahteva priključivanje na računar, odnosno fizički pristup. Iako se generalno ne ohrabruje, i povlači poništavanje garancije na uređaj, ovaj postupak se ne sprečava i ne ograničava od strane Google-a i drugih regulatora Android standarda, jer je često neophodan prilikom servisiranja ili testiranja novih funkcionalnosti.

S obzirom sa se radi o Linux derivatu, jasno je da administrator (root) ima apsolutni pristup svakom delu sistema, a da isto važi i za aplikacije pokrenute pod ovim privilegijama. Naravno, korisnik ne mora uvek biti svestan da je na njegovom uređaju omogućen administratorski nalog, jer je isto mogla izvršiti specijalno napisana zlonamerna aplikacija ili njen deo. Shodno navedenom, iako u suštini efikasan i proveren, bezbednosni model Android sistema zasnovan na Linux interpretaciji koncepta privilegija i prava pristupa, ipak nije dovoljan da pruži adekvatnu zaštitu u slučaju kompromitacije administratorskog naloga ili neovlašćenog fizičkog pristupa uređaju [15]. Iz tog razloga, relativno rano u razvoju Android platforme pristupljeno je uvođenju pojedinih kriptografskih tehnika zaštite statičnih podataka.

Enkripcija celog diska (Full disk encryption – FDE) uvedena je u verziji 3.0 i nije se značajnije menjala sve do verzije 5.0. Za ovaj postupak iskorišćena je postojeća infrastruktura Linux sistema bazirana „device mapper“ okruženju, koje omogućava transparentnu enkripciju korisničkih podataka, odnosno u slučaju Androida, deo interne memorije koji se mapira (mount) kao /data particija. Transparentnost podrazumeva da se svi podaci šifruju pre upisa i dešifruju nakon čitanja sa diska, a bez potrebe za posebnom interakcijom od strane korisnika. U ovoj verziji, glavni (master) ključ koji se koristi je 128-bitni, nasumično generisan, a koristi se u AES algoritmu u CBC modu. Sam master ključ je zaštićen enkripcijom putem posebnog ključa (key encryption key – KEK), koji se izvodi iz lozinke za zaključavanje ekrana, što je ujedno i najslabija tačka celog kriptosistema [15].

Naime, iako se za zaštitu master ključa koristi za to predviđena kriptografski jaka funkcija (PBKDF2) u dve hiljade iteracije, skup mogućih ulaznih podataka, odnosno lozinka koju korisnik unosi u vidu PIN-a ili grafičke interpretacije sheme za zaključavanje, nema dovoljan broj mogućih kombinacija, te je podložan napadu putem grube sile (bruteforce). Nakon demonstracije prvih praktičnih napada na ovaj sistem, od verzije 4.4 promenjen je algoritam zaštite master ključa, pa se sada u te svrhe koristi „scrypt“, koji je značajno teže prilagoditi paralelizovanim napadima na široko dostupnom hardveru.

U novijim verzijama Androida, pristup enkripciji celog diska je fundamentalno izmenjen, te se sada uvodi hardverska zaštita ključeva, odnosno njihovo pohranjivanje u delu memorije koji nije dostupan ostatku sistema osim putem posebnog API-ja, u procesoru se integriše akceleracija kriptografskih operacija, a uvodi se koncept adaptabilnog memorijskog prostora (adoptable storage) koji omogućava transparentnu enkripciju eksternih uređaja kao što su memorijske kartice i USB memorije [15].

3.4.3. End to end enkripcija komunikacije

End to end enkripcija je pojam koji se često koristi u marketinške svrhe, odnosno postojanje određenih kriptografskih tehnika implementiranih u pojedinim aplikacijama se pogrešno predstavlja kao rešenje koje nudi apsolutnu bezbednost komunikacije između dva korisnika. Najčešće je u pitanju slobodno tumačenje konvencionalnih kriptografskih tehnika upotrebljenih u više segmenata kanala veze, koji iako pružaju značajan nivo bezbednosti, ne obezbeđuju komunikaciju u slučajevima kada je neka od ključnih tački u mrežnom saobraćaju kompromitovana, što podrazumeva da i sam provajder usluge može zloupotrebiti svoju poziciju i narušiti privatnost komunikacije za koju garantuje [6]. Nedavna otkrića o masovnom nadzoru komunikacija širom sveta podigla su svest

javnosti o ovom problemu, pa je porasla potreba za komunikacionim sredstvima koja mogu garantovati privatnost učesnika u komunikaciji čak i u slučajevima apsolutno nebezbednog, odnosno kompromitovanog kanala komunikacije.

Signal je novi komunikacioni protokol, prvobitno primenjen u istoimenoj aplikaciji koji garantuje end-to-end enkripciju poruka, te ćemo ga iskoristiti kao ilustrativan primer implementacije ove tehnike zaštite u mobilnim aplikacijama. Nakon munjevitog proboja na tržište aplikacija za slanje poruka, drugi značajni faktori kao što su WhatsApp, Facebook Messenger i Google Allo tvrde da su implementirali neke delove ovog protokola, ali usled zatvorene prirode ovih aplikacija, detaljna analiza ovih tvrdnji, odnosno sistematska provera njihove bezbednosti nije moguća od strane nezavisnih istraživača [6].

Inovativnost signal protokola ogleda se u adekvatnoj primeni kriptografskih koncepata koji se ne koriste toliko često u savremenim protokolima i aplikacijama, a takođe je kreirana nova tehnika uvezivanja ključeva (key ratcheting) koja omogućava da se u svakoj sesiji koristi novi ključ, stvoren i razmenjen na bezbedan način, tako da eventualna kompromitacija u jednom trenutku ne narušava poverljivost prethodnih ni budućih komunikacija [6]. Usled izuzetne složenosti ovog protokola, pomenućemo samo njegove najznačajnije karakteristike koje ilustruju adekvatnu primenu pojedinih kriptografskih tehnika zaštite.

Prvobitna autentifikacija strana u komunikaciji i razmena tajnog ključa koji će se koristiti za šifrovanje samih poruka, vrši se putem proširenog trostrukog Difi-Helman protokola za razmenu ključeva (Extended Triple Diffie-Hellman - X3DH). U pitanju je varijanta autentifikacije bazirana na kriptografiji javnih ključeva, koja osnovne parametre izvodi iz eliptičnih krivih (eliptic curve) pružajući buduću tajnost (forward secrecy) i neporecivost razmenjenih poruka (u ovom slučaju tajnog ključa) [34].

X3DH je prilagođen slučajevima asinhrone međusobne autentifikacije, koja je neophodna u scenarijima kada jedan od učesnika nije trenutno aktivan na mreži, što je česta situacija prilikom upotrebe mobilnih aplikacija za komunikaciju putem poruka. U tom slučaju se za uspostavljanje odnosa poverenja i razmenu tajnog ključa koristi jedan od javnih ključeva koje je taj korisnik ostavio na serveru, ali se nakon što se oba korisnika nađu "online" autentifikacija ponovo vrši putem ključeva koji postoje samo na korisničkim uređajima [34].

Dvostruka "reza" (Double Ratchet) algoritam se upotrebljava za šifrovanje samih poruka inicijalnim

korišćenjem ključa koji je ustanovljen nakon X3DH faze uspostavljanja veze [33]. Specifičnost ovog algoritma se ogleda u tome što se svaka poruka šifruje posebnim ključem, koji se delimično izvodi iz metapodataka (proisteklih iz Difi-Helman parametara) poslatih uz samu poruku. Na ovaj način se postiže da se u slučaju kompromitacije nekog od ključeva ne mogu dešifrovati prethodne ili buduće poruke u toj komunikaciji.

Iako zbog svoje kompleksnosti nije bio podvrgnut mnogobrojnim analizama, Signal protokol je bio predmet nekoliko temeljnih studija izvršenih od strane vrhunskih eksperata iz oblasti matematike i kriptografije, te su one potvrdile većinu tvrdnji autora protokola o njegovoj pouzdanosti, dok nisu otkrile značajnije nedostatke. Čak i površan pogled na arhitekturu Signal protokola otkriva da se u svakoj fazi primene koristi veliki broj kriptografskih tehnika, a shodno činjenici da je aplikaciju moguće bez teškoća koristiti i na starijim mobilnim uređajima, možemo zaključiti da nije validan često pominjani argument da jaku kriptografiju nije moguće implementirati u uređajima slabijih performansi. Takođe, dosledna primena end-to-end koncepta faktički izmešta vršenje kriptografskih operacija sa servera na klijentske uređaje, čime se postiže mogućnost skaliranja sistema na veliki broj korisnika bez eksponencijalnog rasta potrebe za serverskom procesorskom snagom.

3.4.4. HTTPS kao de facto standard zaštite komunikacionih kanala

Promena težišta korisničke elektronike ka mobilnim uređajima uvećala je protok ličnih, osetljivih i poverljivih informacija putem ovih platformi. Ovi uređaji se često povezuju putem javno dostupnih bežičnih mreža, koje i u slučajevima kada su ispravno konfigurisane, predstavljaju najslabiju tačku u sistemu komunikacije. Svi paketi koji putuju bežičnim mrežama (WLAN) mogu biti presretnuti ili izmenjeni uz minimalne intervencije napadača koji se nalazi na istoj toj mreži. Takođe, nedavnom proliferacijom softverski definisanih radio uređaja (SDR), velikom broju potencijalnih napadača otvara se mogućnost nadzora i manipulacije samim mobilnim mrežama koje uređaj koristi (GSM, UMTS, 4G LTE, itd.), što ipak i dalje nije trivijalan postupak, te ga nećemo detaljno obrađivati u ovom radu.

Da bi se sprečili napadi na komunikacioni kanal, pribegava se implementaciji adekvatnih kriptografskih tehnika zaštite, koje se u slučaju komunikacije mobilnih uređaja u najvećem broju svode na korišćenje HTTPS protokola. Za razliku od implementacije HTTPS-a na javno dostupnim web sajtovima, koja se najčešće sprovodi od strane za to stručnih lica, uz poštovanje formalnih protokola i dobrih praksi, integracija ovog protokola u mobilnim, pre svega Android aplikacijama često je usputni postupak u razvoju kojem se ne pridaje dužna pažnja [57].

Kada želimo da zaštitimo podatke koji se razmenjuju sa nekim javno dostupnim servisima, moramo imati u vidu da u najvećem broju slučajeva nemamo nikakvu kontrolu nad serverima koji ih pružaju. U tim slučajevima, podrazumevano ponašanje Android sistema je provera SSL/TLS sertifikata putem ugrađene podrške, odnosno njihova validacija korišćenjem korenih (root) sertifikata preinstaliranih na sistemu [49].

Okosnica HTTPS-a, kao i svih ostalih protokola koji funkcionišu nad SSL/TLS-om je mogućnost da klijent van svake sumnje može potvrditi da komunicira sa pravim serverom. Da bi se ustanovio ovaj odnos poverenja, vrši se kompleksan postupak razmene i potvrde javnih ključeva, dogovor oko tajnog sesijskog ključa a tek nakon čega se može smatrati da je buduća komunikacija bezbedna [57].

Digitalni sertifikati na kojima je baziran ovaj sistem funkcionišu prema X.509 protokolu, koji se najčešće koristi u sprezi sa protokolom za proveru statusa sertifikata (Online Certificate Status Protocol - OCSP). Zajedno oni omogućavaju da se sertifikati kreiraju, proveravaju i poništavaju na bezbedan i pouzdan način. Ove provere se vrše u više koraka: poverava se validnost izdavača sertifikata, usklađenost sertifikata sa domenom i konsultuje lista poništenih odnosno povučenih sertifikata. Kada ove provere vrši web browser, očekuje se da će korisnik na osnovu prezentovanih informacija odnosno upozorenja, doneti adekvatnu odluku da li da nastavi sa konekcijom prema konkretnom serveru ili da je prekine [57]. Kod implementacije ovih provera u mobilnim aplikacijama sva odgovornost je na programeru, koji, svesno ili ne, odlučuje da li će ih primeniti [57]. Najveći problem leži u činjenici da je dovoljno izostaviti ili pogrešno implementirati samo jednu od brojnih komponenti SSL/TLS protokola, pa da se obesmisli ceo sistem zaštite.

U standardnoj Android implementaciji HTTPS protokola postoje tri faze kreiranja bezbedne konekcije. To su: postavljanje osnovnih podešavanja, kreiranje konekcije (socket) i upravljanje sertifikatima. U prvoj fazi se podešavaju zaglavlja HTTP protokola, što se radi sistemskim klasama HttpParams ili ClientConnectionManager, a takođe se mogu izmeniti podrazumevani algoritmi za šifrovanje. Nakon toga, konekcija se otvara odgovarajućom metodom SSLSocketFactory klase. X509TrustManager je entitet koji takođe postoji u okviru ove klase, a njegovom inicijalizacijom će automatski biti izvršena provera pristiglog sertifikata u odnosu na listu postojećih sistemski preinstaliranih [57]. Tek nakon uspešnog okončanja svih ovih faza, uspostavlja se bezbedna komunikacija. Iako veoma složen, ovaj proces je apstrahovan i pojednostavljen radi lakše upotrebe, što u praksi znači da je dovoljno nekoliko linija koda da bi se implementirao u standardnom Android okruženju.

Ovaj model se može proširiti funkcionalnošću koja je neophodna kada želimo da naša aplikacija komunicira sa samo jednim, ili manjim brojem servera koji su pod našom kontrolom i čija svrha je da opslužuju isključivo tu aplikaciju. Potrebno je modifikovati konfiguraciju Android SSL/TLS implementacije kako bi se ograničilo poverenje na samo određene root sertifikate, eventualno odobrilo prihvatanje naših samopotpisanih (self-signed) sertifikata ili omogućila upotreba klijentskih sertifikata radi identifikacije korisničke strane [49].

S obzirom da je lista sertifikata kojima Android sistem implicitno veruje veoma dugačka, te da se na pojedinim verzijama ona vrlo retko ili nikad ne ažurira, smatra se veoma dobrom praksom ograničavanje ili potpuno izbegavanje korišćenja ove liste kada infrastruktura ostatka sistema to dozvoljava [49]. Šta više, odobravanje samo jednog konkretnog i to samopotpisanog sertifikata pruža najmanju moguću zavisnost od poverenja prema nekom spoljnom autoritetu.

3.5. Kriptografska zaštita komunikacionih kanala primenom infrastrukture javnih ključeva

3.5.1. Standardizacija PKI

Infrastruktura javnih ključeva (PKI) koja se koristi na Internetu zasnovana je na X.509 međunarodnom standardu. Od 1995. godine kreće ubrzan razvoj ovog standarda, a u ovaj proces se uključuju mnoga stručna tela u partnerstvu sa značajnim privrednim subjektima [45]. Između ostalih, standardizaciju su kreirali Međunarodna telekomunikaciona unija ITU (International telecomunication union), ISO, Internet Engineering Task Force (IETF), itd. svaki u svojim oblastima ekspertize [38].Od samog početka, standard je razvijan tako da prati realne potrebe tržišta i tehnološke inovacije koje su menjale strukturu Interneta kao i način na koji se koristi.

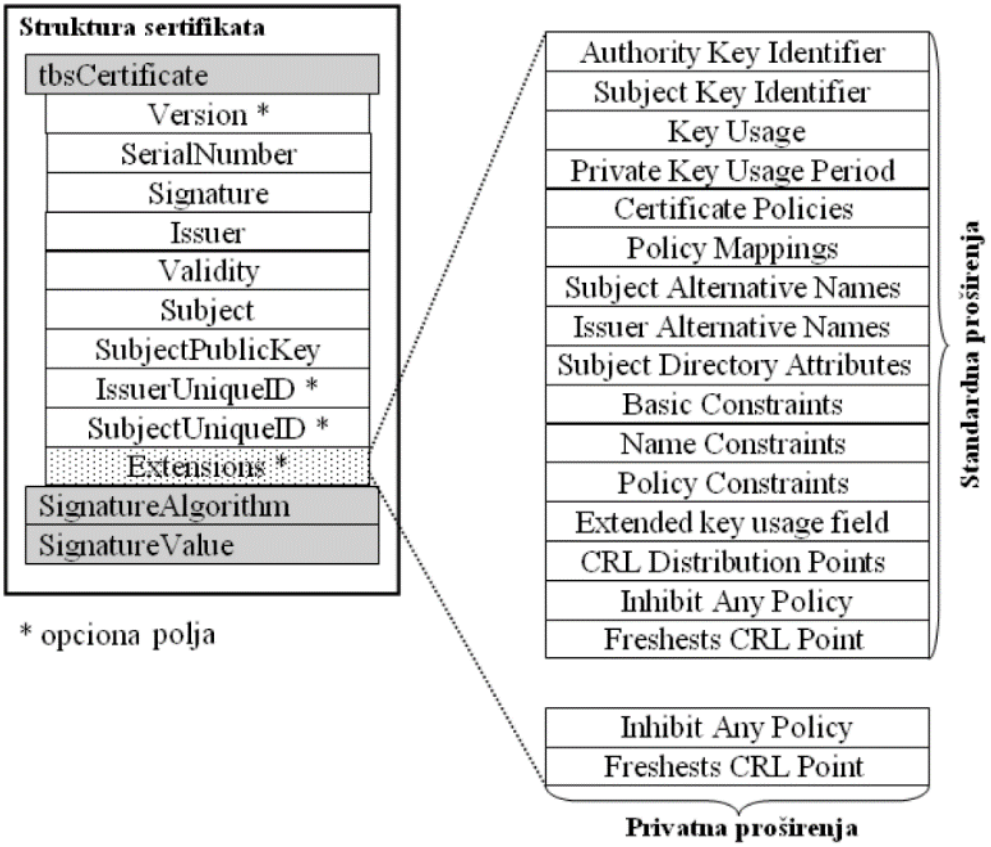
Ovi standardi se tiču različitih oblasti primene PKI, kao što su: format sertifikata, upravljanje sertifikatima, odnosno interakcija između klijenata i sertifikacionih autoriteta (CA), distribucija sertifikata, prevremeno povlačenje (revocation), garancija valjanosti potpisa i nakon što je istekao originalni sertifikat kojim su potpisani (time-stamping), definisanje politika upotrebe sertifikata od strane CA koji ga izdaje itd.

Sami sertifikati, koji su osnova PKI sistema predstavljaju digitalne dokumente koji sadrže javni ključ, informacije o entitetu za čiji identitet garantuju i digitalni potpis izdavača [45]. Oni suštinski

omogućavaju razmenu, čuvanje i upotrebu javnih ključeva, što im je osnovna svrha. Među najvažnijim delovima sertifikata su svakako oni koji se bave definisanjem kriptografskih tehnika koje se koriste pri upotrebi, odnosno verifikaciji istog [38]. Definišu se pre svega dve najosnovnije kriptografske funkcije, a to su vrsta digitalnog potpisa i kriptografski jaka jednosmerna (hash) funkcija. Za njihovo detaljno određivanje koriste se brojni specifični standardi.

ASN notacija (Abstract Syntax Notation One - ASN.1) predstavlja skup pravila za definisanje, transport i razmenu kompleksnih struktura podataka. Prva verzija ovog standarda nastala je još 1988. godine sa idejom da olakša mrežnu komunikaciju između raznovrsnih uređaja, bez obzira na arhitekturu i operativne sisteme [45]. ASN opisuje podatke na apstraktan način, dok se način njihovog enkodiranja definiše posebnim standardima kao što su BER (Basic Encoding Rules), DER (Distinguished Encoding Rules), PEM (Privacy Enhanced Mail) itd.

Sertifikati se pretežno distribuiraju u PEM formatu, a on predstavlja ASCII (Base64) interpretaciju DER enkodiranih podataka. Pri razvoju aplikacija ili administriranju složenih sistema, često se javlja potreba za konverzijom iz jednog formata u drugi, što se lako postiže adekvatnim alatima [45].



Slika 5 Struktura sertifikata [48]

Osim osnovnih polja, od verzije 3 X.509 standarda, sertifikate je moguće proširiti određenim ekstenzijama (extensions) [38]. Neke od najvažnijih su: korišćenje ključa (Key usage) koja definiše u koje svrhe se ključ može isključivo koristiti, npr. šifrovanje, potpisivanje podataka, potpisivanje sertifikata; polise sertifikata (Certificate policies), koje čine skup pravila prema kojima se dati sertifikat mora upotrebljavati; alternativno ime subjekta (Subject alternative name) koje dozvoljava povezivanje dodatnih identiteta sa nosiocem sertifikata, npr. IP adrese, e-mail adrese, ime u drugačijem obliku ili alfabetu, URL u lokalnoj mreži itd.

Pored međunarodnih standarda za sertifikate, postoje i brojni drugi formati koji se definišu u posebnim domenima primene [38]. Npr. "EMV certificate" je standard razvijen od strane konzorcijuma Europay, MasterCard Visa, a kojim su definisani sertifikati za upotrebu u pametnim platnim karticama.

3.5.2. PKI - model poverenja (CA)

U kontekstu infrastrukture javnih ključeva (PKI) termin poverenje treba upotrebljavati u veoma strogom smislu, gde on označava da se konkretni sertifikat može potvrditi (validirati) putem nekog od autoritativnih sertifikata (CA) kojima raspolaže korisnički sistem [45]. Proširenje ovog pojma unelo bi zabune, što bi posledično moglo izazvati neadekvatnu implementaciju protokola koji su na njemu zasnovani.

Osnovni zadatak infrastrukture javnih ključeva (PKI) je da definiše mehanizme koji omogućavaju primaocu i pošiljaocu da međusobno potvrde autentičnost, poverljivost i integritet poruka koje razmenjuju. Sastoji se od elemenata neophodnih za implementaciju kriptografskih tehnika baziranih na sistemu parova javnih i tajnih ključeva, posebno prilagođenih upotrebi od strane velikog broja korisnika. Iako prvenstveno zamišljena kao tehnologija obezbeđivanja standardnih korisničkih interakcija sa internet sajtovima, PKI se danas koristi kao glavni mehanizam obezbeđivanja komunikacije mobilnih aplikacija [16]. Ipak, da bi se postigao željeni nivo bezbednosti u novom okruženju, nije dovoljno koristiti samo elementarne mogućnosti ovog sistema, kao što je to slučaj pri zaštiti internet prezentacija, već ih treba koristiti kao osnovu nad kojom se gradi adekvatan bezbednosni sistem.

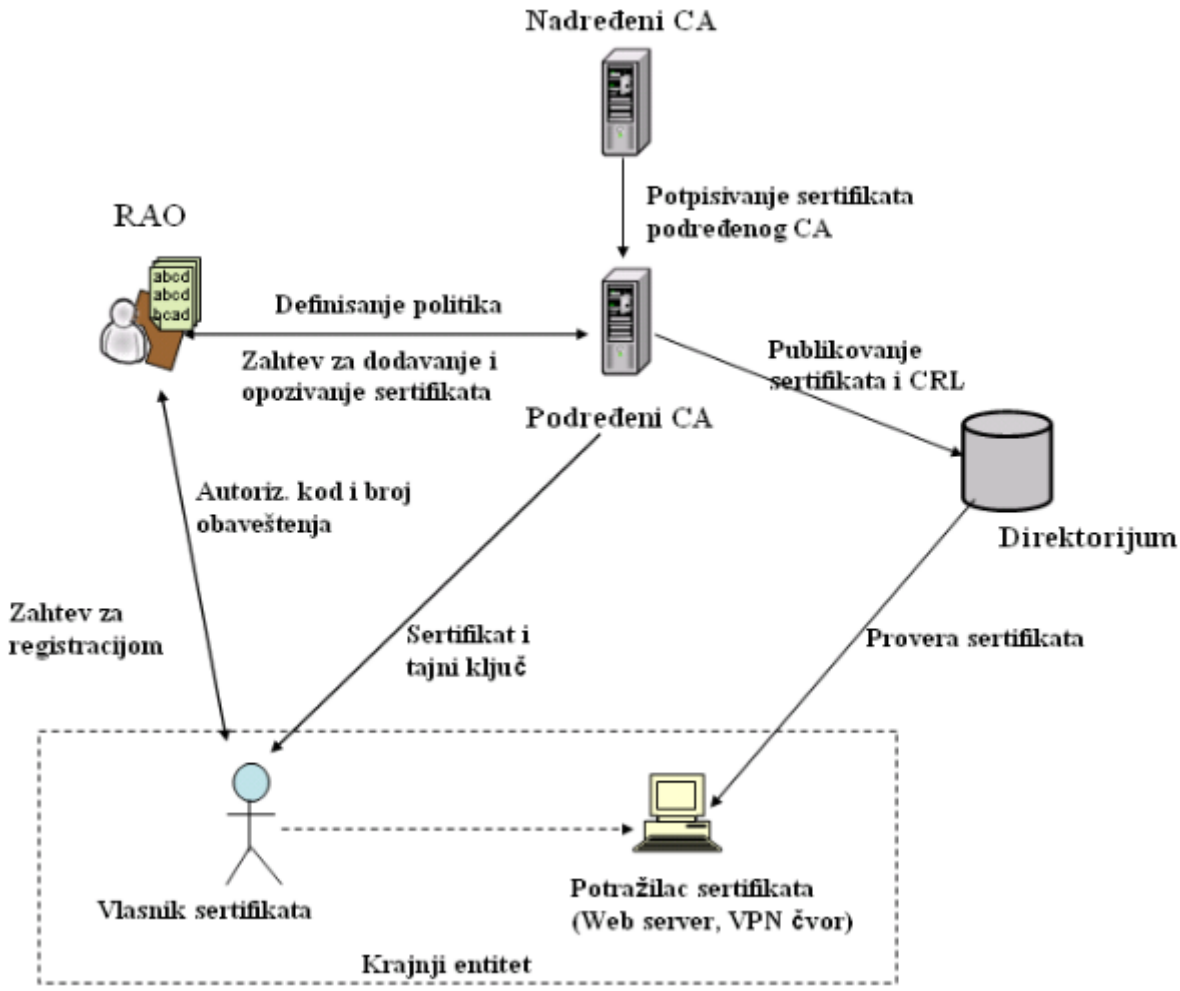
PKI sadrži komponente koje omogućavaju autentifikaciju korisnika, očuvanje integriteta, poverljivosti i neporecivosti poruka koje razmenjuju, a putem sistema distribucije javnih ključeva, ova funkcionalnost je dostupna korisnicima koji nisu prethodno uspostavljali komunikaciju odnosno

ostvarili odnos poverenja [16]. Upravo je delegiranje poverenja na nezavisni entitet, odnosno autoritet, glavna karakteristika PKI sistema, jer ovi entiteti potvrđuju (sertifikuju) pripadnost pojedinačnih javnih ključeva konkretnim učesnicima u komunikaciji, korišćenjem za to podesnih, ranije pomenutih, kriptografskih tehnika.

Može se reći da je infrastruktura javnih ključeva (PKI) skup hardvera, softvera, ljudi i procedura neophodnih radi kreiranja, upravljanja, čuvanja, distribucije i poništavanja sertifikata o validnosti javnih ključeva (public key certificates) [16]. Konkretni entiteti koji realizuju ove aktivnosti su sertifikaciono telo ili autoritet (certification authority – CA), registraciona tela, centralni server (directory server), a sistem se može proširiti tako da obuhvata i neke dodatne delove kao što su pametne (smart) kartice, serveri koji vode precizne časovnike (time-stamp servers),
OCSP responderi itd.

Najvažniji segment ovog sistema je ključ osnovnog sertifikata nekog sertifikacionog autoriteta (root CA key), koji služi za izdavanje i validaciju sertifikata nižeg ranga. Stariji ključevi, čiji sertifikati su opšteprihvaćeni od strane brojnih sistema imaju značajnu finansijsku vrednost, jer je njihova izmena u svim uređajima koji im veruju praktično nemoguća. Takođe, ukoliko bi ovi ključevi bili kompromitovani, njihovo povlačenje (revocation) bi uticalo veliki broj izdatih sertifikata, čineći ih neupotrebljivim.

Iako i dalje postoje pojedina CA tela koja koriste osnovne (root) sertifikate za izdavanje klijentskih, ovaj postupak se smatra suviše opasnim, te se uvode brojne procedure i standardi radi suzbijanja ove i sličnih loših praksi [45]. Takođe, pojedini standardi za dobijanje CA statusa propisuju neophodnost ručne manipulacije (putem unetih komandi, bez automatizacije) osnovnim sertifikatima odnosno njima pripadajućim tajnim ključevima, što implicira da konkretni terminali ne smeju biti povezani na Internet već im operater mora neposredno pristupati.



Slika 6 PKI infrastruktura [48]

Neposredni učesnici u procesu izdavanja i validacije serverskih sertifikata su pretplatnik (odnosno sistem koji želi da pruža HTTPS servis), registraciono autorizaciono telo (registration authority - RA) koje obavlja određene administrativne radnje kao što je potvrda identiteta pretplatnika, autorizaciono telo (CA) koje može biti osnovno ili posredno (intermidiate) a koje izdaje, potvrđuje i povlači sertifikate, dok se na kraju lanca nalazi korisnik, odnosno Internet pretraživač, operativni sistem ili konkretna aplikacija koja proverava predstavljene sertifikate putem lokalnog repozitorijuma CA sertifikata [45].

Broj aktivnih sertifikata u upotrebi se meri milionima, dok se smatra da CA autoriteta kojima sistemi implicitno veruju ima preko stotinu [45]. Na ovaj broj treba dodati posredničke (intermidiate) CA koji takođe imaju mogućnost izdavanja serverskih sertifikata. Nije poznat broj konkretnih organizacija koje kontrolišu sva ova CA tela, ali se zna da za 90% izdatih sertifikata garantuje neki od deset najvećih CA kao što su Symantec, GoDaddy, Comodo, GlobalSign, DigiCert, StartCom i Entrust.

3.5.3. Napredne mogućnosti HTTPS-a

Većina uobičajenih konfiguracija SSL-TLS protokola se zadržava na korišćenju sistema PKI za potvrdu identiteta serverske strane, odnosno serveru ne pruža nikakve informacije niti potvrde o identitetu klijenta koji zahteva usluge [49]. Iako se ova funkcionalnost najčešće delegira aplikativnom sloju, gde se primenjuju neki od brojnih ranije pomenutih mehanizama autentifikacije, u posebno osetljivim okruženjima, ko što su kompanijski ili bankarski sistemi, dobra praksa nalaže klijentsku autentifikaciju na nivou transportnog protokola.

Naravno, implementacija ovakvog rešenja zahteva potpunu kontrolu nad serverskom stranom, pre svega na nivou web server softvera, ali najčešće i samog backend dela aplikacije. Što se tiče klijentske strane, odnosno Android aplikacije, nju je takođe neophodno konstruisati na specifičan način, odnosno izmeniti konfiguracione parametre JCA kriptografskih provajdera na način koji omogućava upotrebu konkretnog klijentskog sertifikata [49]. Ovo je jedna od naprednijih tehnika koju ćemo primeniti u našoj aplikaciji, jer oslikava mogućnost dramatičnog povećanja bezbednosti komunikacije relativno jednostavnim modifikacijama podrazumevanih parametara postojećih komponenti Android sistema.

Digitalni klijentski sertifikati se takođe mogu koristiti kao mehanizam za autentifikaciju [16]. U tom scenariju, svakom za svakog korisnika se kreira jedinstven set tajnih i javnih ključeva, dok se sertifikat, koji pored javnog ključa, uglavnom sadrži i dodatne informacije o korisniku, potpisuje od strane pouzdanog sertifikacionog tela, ili onog koje je pod kontrolom sistema na koji se korisnik autorizuje.

Da bi se to postiglo, iskorišćen je ranije pomenuti mehanizam inicijalizacije kriptografskih klasa unutar JCA biblioteke putem dodatnih parametara, koji specifikuju ključeve koje aplikacija može koristiti, bez obzira na podrazumevana sistemska podešavanja [49]. Na ovaj način se postiže najviši mogući nivo bezbednosti transporta podataka, u smislu poverljivosti informacija i potvrde identiteta obe strane. Ovaj metod ćemo iskoristiti kao dodatnu meru autentifikacije u aplikaciju koju razvijamo za potrebe ilustracije koncepata iznetih u ovom radu.

Zarad neporecive autentifikacije servera klijentu, jedno od najčešće predlaganih rešenja je učestalije korišćenje "kačenja" sertifikata (SSL Pinning), a što je u stvari implementacija ranije opisanog koncepta poverenja pri prvoj upotrebi (TOFU) [57]. Sertifikat se pri prvoj konekciji na uobičajen način preuzima od servera, a zatim se kešira kako bi se svaki sledeći put proveravalo da li je došlo do

eventualnih izmena istog. Rizik da prva konekcija bude kompromitovana (a zatim i svaka naknadna, i to od strane istog napadača) je veoma nizak, pa se ova tehnika smatra prihvatljivom, a razvijeno je više protokola koji automatizuju ovaj proces od kojih su neki integrisani u najpopularnije internet pretraživače.

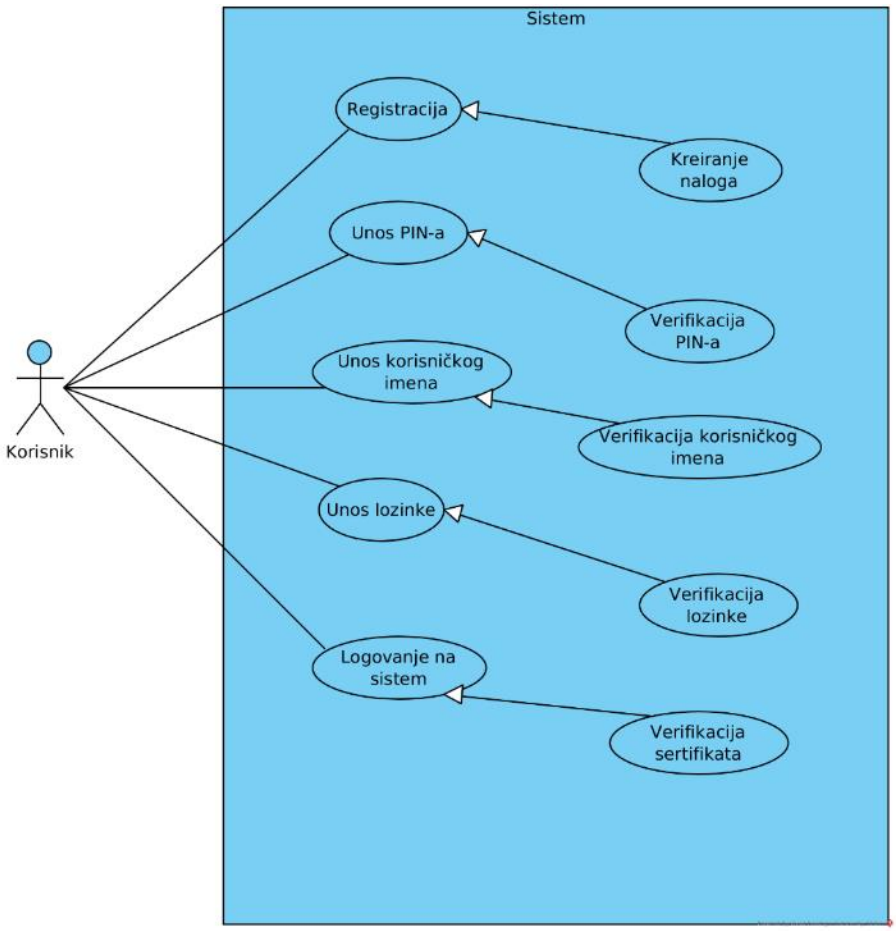
U Android aplikacijama se ova tehnika uglavnom implementira ručno, korišćenjem naprednih mogućnosti postojećih kriptografskih biblioteka i polako postaje standard za zaštitu komunikacija sa serverskom stranom. Zaobilazi se veoma teško, isključivo ciljanom intervencijom (dekompajliranjem i modifikacijom) same aplikacije, što obesmišljava većinu rasprostranjenih napada na HTTPS protokol, ma koliko oni bili sofisticirani. Sa druge strane, mora se voditi računa o kontinuiranoj validnosti sertifikata koji su pinovani, jer ukoliko neki od njih postane nevažeći iz bilo kog razloga, neće ga biti moguće zameniti dok se hash sledećeg ne implementira u novoj verziji aplikaciji, odnosno svim njenim instancama.

4. Implementacija kriptografskih tehnika zaštite u konkretnoj mobilnoj aplikaciji

4.1. Arhitektura sistema (u kome je implementirana konkretna mobilna aplikacija)

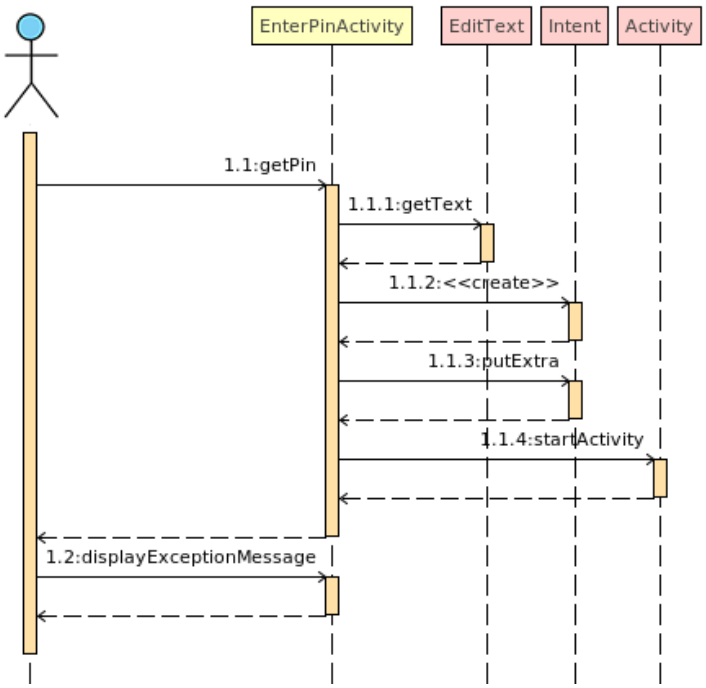
4.1.1. Arhitektura Android aplikacije

Korisnik dolazi na sajt za otvaranje naloga i skidanje aplikacije (https://master01.duckdns.org/). Korisnik unosi korisničko ime, lozinku i broj telefona. Ukoliko su podaci validni, sistem kreira korisnički nalog i omogućava skidanje aplikacije. Istovremeno, sistem kreira klijentski sertifikat za tog konkretnog korisnika i pohranjuje detalje o istom u bazu. Sertifikat se zaključava šestocifrenim nasumično generisanim PIN kodom. Korisniku se omogućava preuzimanje zaključanog sertifikata, a PIN se šalje putem SMS-a (ili e-mail-a) i ne čuva se na serveru.



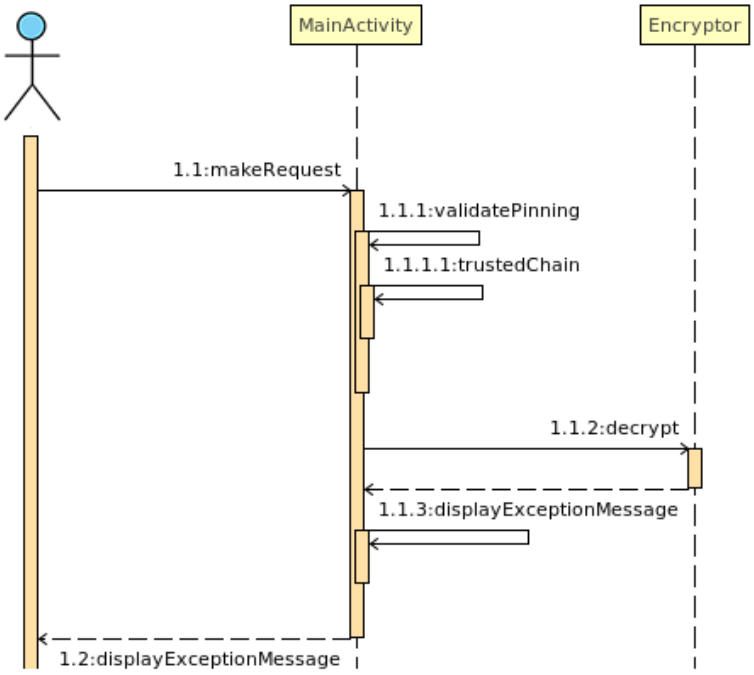
Slika 7 Dijagram slučajeva korišćenja

Korisnik skida aplikaciju, pokreće je, unosi PIN, ime i lozinku. Aplikacija proverava validnost prezentovanog serverskog sertifikata, uključujući i proveru hash vrednosti ključa poslednjeg (*leaf*) sertifikata (tehnika *certificate pinning*).

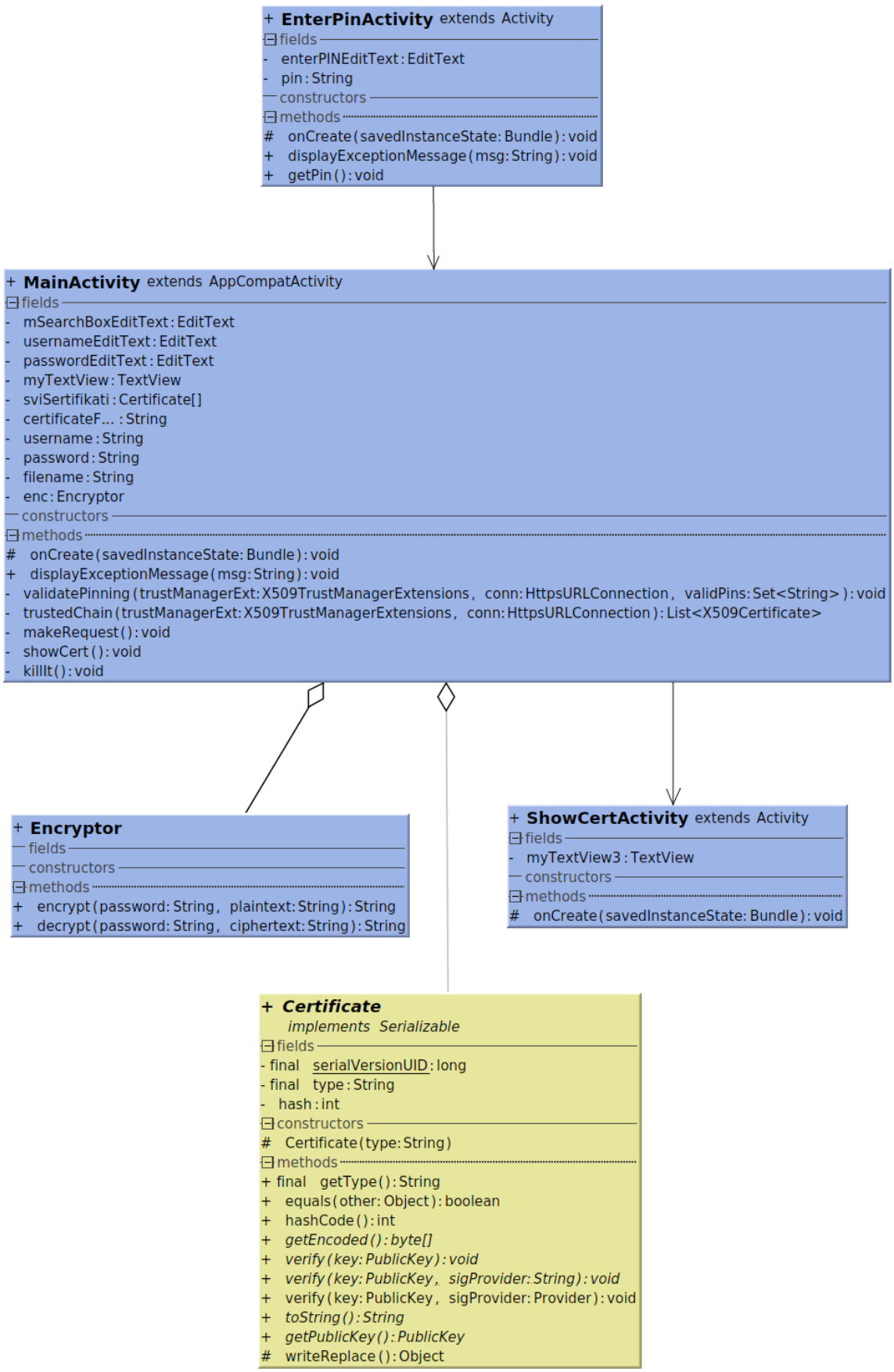


Slika 8 Dijagram sekvenci 1

Nakon međusobnog potvrđivanja identiteta i uspostavljanja konekcije, aplikacije prima podatke od servera koji su zaštićeni simetričnom šifrom. Odmah po prijemu podataka, osnovna klasa MainActivity poziva klasu Encryptor i odgovarajućom metodom decrypt() dešifruje podatke.



Slika 9 Dijagram sekvenci 2



Slika 10 Dijagram klasa

Podaci između klijenta (aplikacije) i servera se dodatno štite enkripcijom na nivou aplikacije. Algoritam koji se koristi je AES256, a ključ se izvodi iz lozinke za to predviđenom kriptografski jakom sistemskom funkcijom. Podaci primljeni sa servera mogu se sačuvati u lokalnom fajlu (kontejneru) koji je takođe šifrovan AES256 algoritmom, dok se ključ za pristup istima takođe izvodi iz lozinke na bezbedan način. Prilikom kompajliranja mobilne aplikacije koristiće se su tehnike zaštite koda od reverznog inženjeringa (obfuskacija). Izvorni kod aplikacije se nalazi na adresi https://github.com/batica81/Master01 .

4.1.2. Arhitektura mrežnih protokola

Mrežni protokoli predstavljaju skup pravila za upravljanje i ostvarivanje interakcije (komunikacije) između sistema mreže. Svakako najvažniji od njih, TCP/IP (Transmisson control protocol / Internet protocol) predstavlja skup protokola koji omogućuju normalno funkcionisanje Interneta i drugih mreža zasnovanih na istom principu. Najvažnije karakteristike ovog protokola su slojevitost i interoperabilnost sa raznim vrstama hardvera i softvera.

OSI model

Layer	Name	Example protocols
7	Application Layer	HTTP, FTP, DNS, SNMP, Telnet
6	Presentation Layer	SSL, TLS
5	Session Layer	NetBIOS, PPTP
4	Transport Layer	TCP, UDP
3	Network Layer	IP, ARP, ICMP, IPSec
2	Data Link Layer	PPP, ATM, Ethernet
1	Physical Layer	Ethernet, USB, Bluetooth, IEEE802.11

Slika 11 OSI model [68]

Detalji fizičkog i data-link sloja nisu od značaja za našu aplikaciju i testove koje ćemo sprovoditi, jer mogu funkcionisati bez obzira na konkretne implementacije. Tek na mrežnom i transportnom sloju ćemo uspostaviti kontrolisane uslove, odnosno radi testiranja ćemo postaviti uređaj sa Android aplikacijom i napadača (laptop) na istu lokalnu mrežu. Napominjemo da aplikacija inače funkcioniše u bilo kojoj mrežnoj konfiguraciji, a slične napada je moguće izvesti i na drugim tačkama kroz koje prolazi komunikacija do servera.

Integracija kriptografskih tehnika zaštite u mobilnoj aplikacije će imati najviše uticaja na komunikaciju putem sesijskog i prezentacionog sloja. Najveći deo preliminiranih aktivnosti, odnsono

inicijalizacije SSL/TLS protokola se odvija na sesijskom, dok se same kriptografske funkcije vrše na prezentacionom sloju [68].

Transakcija korišćenjem SSL/TLS protokola uključuje sledeće aktivnosti:
- klijent inicira konekciju
- server šalje svoj digitalni sertifikat klijentu
- klijent šalje svoj digitalni sertifikat serveru
- klijent proverava da li je sertifikat izdat od strane validnog CA
- server proverava validnost klijentskog sertifikata
- klijent i server razmenjuju javne ključeve
- klijent generiše tajni ključ koji se koristi samo u toj transakciji
- klijent šifruje generisani tajni ključ korišćenjem serverovog javnog ključa i šalje ga serveru
- u daljnjem toku transakcije server i klijent koriste isti tajni ključ metodom simetričnog kriptovanja

Na višem nivou, komunikacija aplikacije sa serverom se obavlja uobičajenim HTTP protokolom.

4.1.3. Sistemska arhitektura

Na serverskoj strani koja obrađuje zahteve Android aplikacije nalazi se NginX server koji pokreće dve php aplikacije neophodne za rad celog sistema. Prvu koja služi kao portal za registraciju i skidanje aplikacije i sertifikata, i drugu sa kojom će aplikacija neposredno komuniciarati. Ova PHP aplikacija proverava korisničko ime i lozinku, a ako se slažu sa određenim postojećim nalogom, proverava da li je predstavljeni klijentski sertifikat bas onaj koji je dodeljen tom korisniku, odnosno vezan za isti nalog. Ova provera se vrši u tabeli baze u koju je informacija o hash vrednosti upisana prilikom kreiranja klijentskog sertifikata. Ako se svi podaci slažu, dozvoljava se pristup podacima klijenta, a ukoliko u bilo kom koraku dođe do greške, prikazuje se detaljna poruka o istoj.

Na serverskoj strani, Nginx server terminiše SSL/TLS kanal i proverava validnost klijentskog sertifikata koji šalje android aplikacija, a takođe prosleđuje informacije o klijentskom sertifikatu (hash) PHP aplikaciji.

Softver koji ćemo koristiti, NginX, je pre svega web server, što znači da isporučuje statičke sadržaje, dok za dinamičke prosleđuje upite određenom interpreteru (u ovom slučaju php-fpm) od koga, nakon obrade, prihvata rezultate koje šalje klijentu u za njega prihvatljivom formatu. Konkretna putanja na koju se upit prosleđuje, kao i eventualne parametre, NginX iščitava iz URL-a, ako je tako

konfigurisan u adekvatnom bloku. Neke takođe česte upotrebe Nginx-a su isporučivanje statičnih fajlova, keširanje, SSL terminacija, kompresija i sl. Osim standardnih web protokola (HTTP, HTTPS, WebSocket), postoji podrška i za e-mail protokole IMAP, POP3, SMTP kao i za TCP [50].

Iako je uobičajena praksa da same aplikacije vode računa o autentifikaciji i autorizaciji korisnika, odnosno ograničavanju pristupa pojedinim delovima aplikacije, u pojedinim scenarijima se mogu koristiti i mogućnosti samog web servera. Mi ćemo se u potpunosti osloniti na proveru klijentskog SSL sertifikata od strane NginX servera, dok ćemo dodatne podatke o istom, pre svega hash vrednost proslediti php aplikaciji radi dodatne potvrde identiteta korisnika.

```
autoindex off;
root /var/www/master01_backend;
index index.php index.html index.htm;
server_name api.master01.duckdns.org;

### SSL Configuration ###
listen 9443 ssl http2;
ssl_dhparam /etc/ssl/certs/dhparam.pem;
ssl_certificate /etc/letsencrypt/live/api.master01.duckdns.org/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/api.master01.duckdns.org/privkey.pem;
ssl_client_certificate /etc/ssl/client/ClientCA.crt;
ssl_verify_client on;
ssl_crl /etc/ssl/client/ClientCA.crl;

### PHP Configuration ###
location ~ \.php$ {
        fastcgi_param X-SSL-CLIENT-CERT-SHA1 $ssl_client_fingerprint;
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php7.2-fpm.sock;
}
```

Opšteprihvaćena tehnologija zaštite web aplikacija kojom se odgovara na najveći broj postavljenih izazova je SSL enkripcija (Secure Socket Layer), odnosno njena implementacija u vidu HTTPS protokola. S obzirom da operacije enkripcije i dekripcije sadržaja troše puno procesorskih resursa, potrebno je posvetiti posebnu pažnju podešavanjima koja optimizuju ovaj proces. Najznačajnija je opcija održavanja konekcije (keepalive) koja sprečava da se najzahtevnija operacija otpočinjanja SSL komunikacije (handshake) izvršava po svakom zahtevu.

Slično, SSL sesije se čuvaju u keš memoriji koja se deli između radnih procesa, te je uglavnom preporučljivo povećati njenu vrednost odgovarajućom opcijom (ssl_session) u http bloku konfiguracionog fajla NginX-a [50].

```
# SSL Settings
ssl_protocols TLSv1.2 TLSv1.1 TLSv1;
ssl_prefer_server_ciphers on;
ssl_ciphers
EECDH+ECDSA+AESGCM:EECDH+aRSA+AESGCM:EECDH+ECDSA+SHA512:EECDH+ECDSA+SHA384:EECD
H+ECDSA+SHA256:ECDH+AESGCM:ECDH+AES256:DH+AESGCM:DH+AES256:!aNULL:!eNULL:!LOW:!
RC4:!3DES:!MD5:!EXP:!PSK:!SRP:!DSS:!AES128;

# OCSP stapling
ssl_stapling on;
ssl_stapling_verify on;
resolver 1.1.1.1;
ssl_session_cache shared:SSL:5m;
ssl_session_timeout 1h;
add_header Strict-Transport-Security "max-age=15768000" always;
```

4.1.4. Prihvaćena ograničenja

Kao glavni ograničavajući faktor pri implementaciji tehnika zaštite, javlja se dodatna kompleksnost prilikom upotrebe. Korisnici nisu spremni da pamte duge lozinke ili izvode posebne radnje svaki put kada pokreću aplikaciju, čak i ako bi im to garantovalo znatno veći stepen bezbenednosti. Usled navedenog, pokušaćemo da dodatne tehnike implementiramo na način koji je prihvatljiv najvećem broju korisnika, odnosno transparentan sa njegove tačke gledišta.

Ograničićemo se na slanje aktivacionog PIN koda putem SMS ili e-mail poruke (što će predstavjati drugi faktor autentifikacije, poslat bezbednim kanalom) a koji će nam poslužiti za otključavanje klijentskog sertifikata.

Limitirajući faktor je pre svega prihvatljiva dužina PIN koda, jer bez obzira na algoritam kojim eventualno možemo izvesti neki ključ iz njega, savremeni procesori i grafičke karte mogu ga "provaliti" metodom grube sile za nekoliko sati. Takođe postoje jeftini uređaji (npr. USB Rubber Ducky i slični) ili čak softverski emulatori USB tastature koji imitiraju korisnika koji ručno unosi sve moguće kombinacije. Zato ćemo koristiti uobičajeni postupak dodatne zaštite ugradnjom internog brojača, koji će posle tri neuspešna unosa blokirati aplikaciju, odnosno obrisati sertifikat.

```
voja@lurker:~$ crackpkcs12 -b -m 4 -M 6 -c n -v batica+1434@gmail.com.p12

Brute force attack - Starting 4 threads

Alphabet: 0123456789
Min length: 4
Max length: 6
Use -m and -M flags to modify these values.

Performance:               113111 passwords [    5336 passwords per second]
**********************************************************
Brute force attack - Thread 3 - Password found: 204700
**********************************************************
```

Slika 12 Pronalaženje PIN-a bruteforce metodom

Iako je moguće automatski kreirati drugačiji primerak aplikacije za svakog korisnika, odnosno integrisati klijentski sertifikat u .apk fajl, to bi onemogućilo distribuciju uobičajenim kanalima (Google Play) i primoralo korisnika na dodatne radnje prilikom instalacije. Takođe, morala bi se posvetiti posebna pažnja serveru za distribuciju aplikacije, odnosno načinu na koji bi klijent utvrđivao njegovu autentičnost. Iz navedenog razloga odlučili smo da aplikaciju i sertifikat distribuiramo odvojeno, što je simulacija scenarija gde se aplikacija dobavlja na uobičajen, proveren način putem Google Play servisa, a automatska personalizacija sertifikata predstavlja deo autentifikacionog procesa.

Takođe, s obzirom da su jedini tajni elementi sa kojima operišemo lozinka i PIN, njih ćemo iskoristiti i za izvođenje drugih zaštitinih tehnika. Šifrovani kontejner fajl u kome će se čuvati sertifikat biće zaštićen ključem izvedenim iz lozinke, kao i sloj simetrične enkripcije razmenjivanih podataka, s tim što će ovaj drugi biti menjan prilikom svake konekcije. Ostale opcije (npr. sistemska baza podataka ili integrisana enkripcija uređaja) nisu suštinski bezbednije u slučaju kada napadač ima fizički ili administratorski pristup uređaju. Ovi postupci su potpuno transparentni za korisnika, ukoliko ne unose primetna usporenja u radu.

4.2. Opis implementacije tehnika i metoda zaštite u konkretnoj mobilnoj aplikaciji

4.2.1. Korišćene biblioteke

Osim standardnih Java biblioteka, dodate su pomoćne biblioteke za JSON formatiranje i olakšani IO rad. Aplikaciji su dodeljene privilegije INTERNET, ACCESS_NETWORK_STATE,

READ_EXTERNAL_STORAGE i WRITE_EXTERNAL_STORAGE. Radi jednostavnosti i preglednosti koda, isključana je opcija provere networking on main thread, što je preporučljivo raditi samo u test okruženju jer može izazvati neželjeno ponašanje aplikacije u realnim uslovima.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Korišćene su standardne klase za rad sa HTTPS protokolom, te je provera većine parametara komunikacije, (osim validnosti sertifikata) prepuštena sistemu i ne vrši se u kodu aplikacije. Aplikacija je kompajlirana sa podrškom za Android uređaje od verzije 4.4 (KitKat, Sdk verzija 19).

```
dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation 'com.android.support.constraint:constraint-layout:1.1.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-
core:3.0.2'
    implementation 'commons-io:commons-io:2.5'
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.cedarsoftware:json-io:4.9.10'
}
```

4.2.2. Implementacija HTTPS protokola

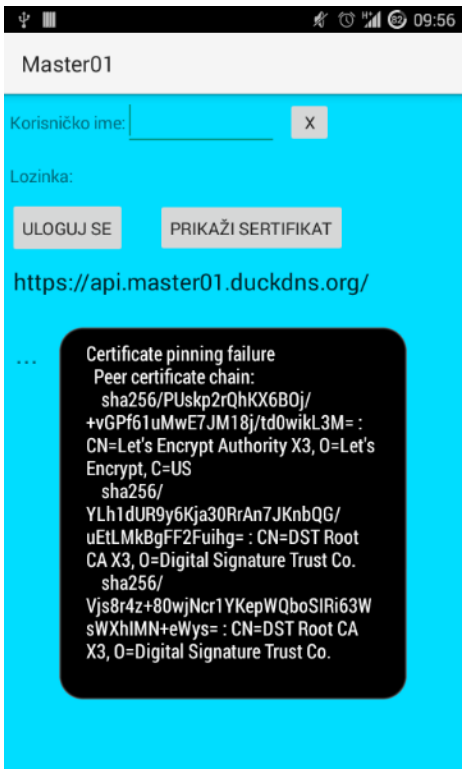
Napredne provere serverskog sertifikata na strani aplikacije se vrše korišćenjem sistemskih klasa za rad sa HTTPS protokolom. Inicijalizacija neophodnih klasa i menadžera poverenja je urađena na podrazumevani način, bez modifikacija koje bi eventualno narušile integritet protokola.

```
TrustManagerFactory trustManagerFactory =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
trustManagerFactory.init((KeyStore) null);
X509TrustManager x509TrustManager = null;
for (TrustManager trustManager : trustManagerFactory.getTrustManagers()) {
    if (trustManager instanceof X509TrustManager) {
        x509TrustManager = (X509TrustManager) trustManager;
        break;
    }
}
```

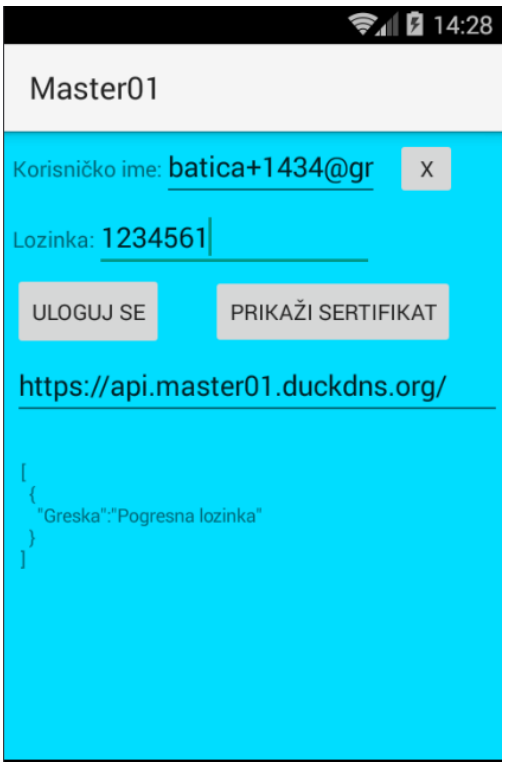
```
X509TrustManagerExtensions trustManagerExt = new
X509TrustManagerExtensions(x509TrustManager);
```

Osim standardne, već integrisane provere validnosti sertifikata u odnosu na preinstaliranu (ili proširenu) listu CA od poverenja, dodali smo proveru hash vrednosti domenskog (leaf) sertifikata, koju smo predefinisali u samoj aplikaciji. Tek ukoliko je verifikacija sertifikata pozitivna, omogućava se konekcija ka aplikaciji gde se proverava uneta lozinka.

```
Set<String> validPins =
Collections.singleton("PUskp2rQhKX6BOj/+vGPf61uMwE7JM18j/td0wikL3M=");
validatePinning(trustManagerExt, urlConnection, validPins);
.
.
.
private void validatePinning(){
MessageDigest md = MessageDigest.getInstance("SHA-256");
    List<X509Certificate> trustedChain = trustedChain(trustManagerExt, conn);
    for (X509Certificate cert : trustedChain) {
        byte[] publicKey = cert.getPublicKey().getEncoded();
        md.update(publicKey, 0, publicKey.length);
        String pin = Base64.encodeToString(md.digest(), Base64.NO_WRAP);
        certChainMsg += "    sha256/" + pin + " : " +
cert.getIssuerDN().toString() + "\n";
        if (validPins.contains(pin)) {
            return;
        }
    }
} catch (NoSuchAlgorithmException e) {
    throw new SSLException(e);
}
throw new SSLPeerUnverifiedException("Certificate pinning failure\n  Peer
certificate chain:\n" + certChainMsg);
```



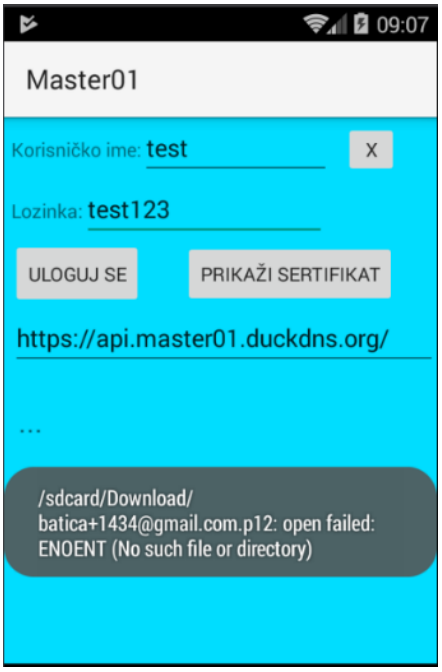
Slika 13 Greška pri proveri serverskog sertifiakta



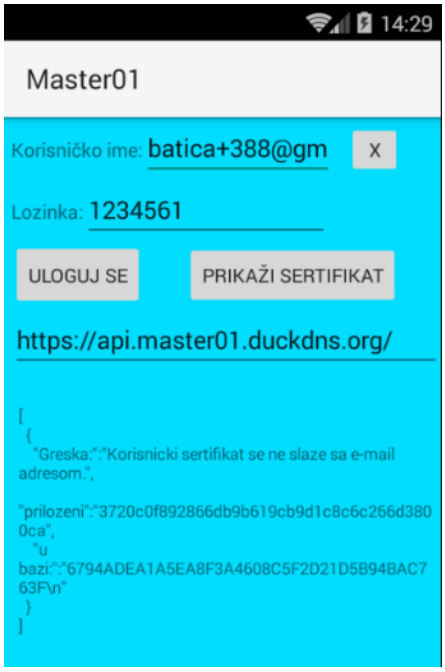
Slika 14 Greška: pogrešna lozinka

Klijentski sertifikat se učitava sa eksterne memorije dostupne korisniku za upis, a posle ispravno unetog PIN koda kojim je zaključan, on se koristi za uspostavljane konekcije i autentifikaciju. Sertifikat se mora slagati sa onim dodeljenim tom korisniku prilikom registracije, u protivnom prijaviće se greška.

```
certificateFile = "/sdcard/Download/" + username + ".p12";

KeyStore keyStore = KeyStore.getInstance("PKCS12");
InputStream fis = new FileInputStream(certificateFile);

Intent intent = getIntent();
String enteredPin = intent.getStringExtra("message");

keyStore.load(fis, enteredPin.toCharArray());
KeyManagerFactory kmf = KeyManagerFactory.getInstance("X509");
kmf.init(keyStore, enteredPin.toCharArray());
KeyManager[] keyManagers = kmf.getKeyManagers();

SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
sslContext.init(keyManagers, null, new SecureRandom());
```

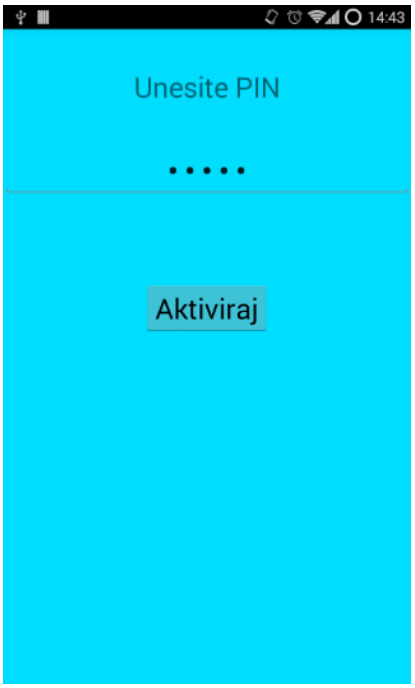


Slika 15 Greška: sertifiakt nije prisutan

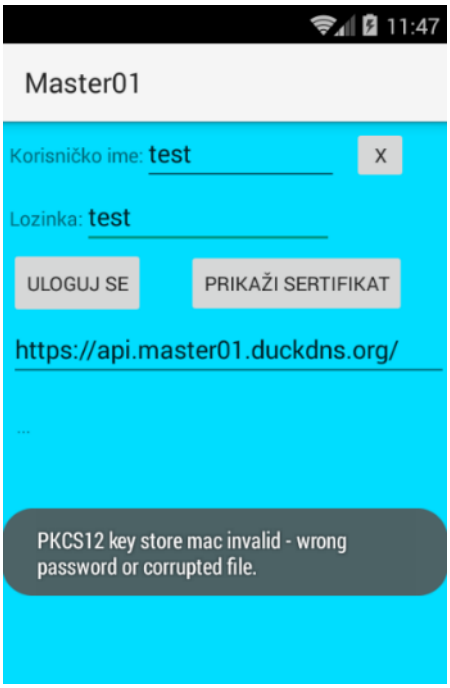


Slika 16 Greška: sertifikat se ne slaže

Ukoliko nije unet ispravan PIN, aplikacija će prijaviti grešku.



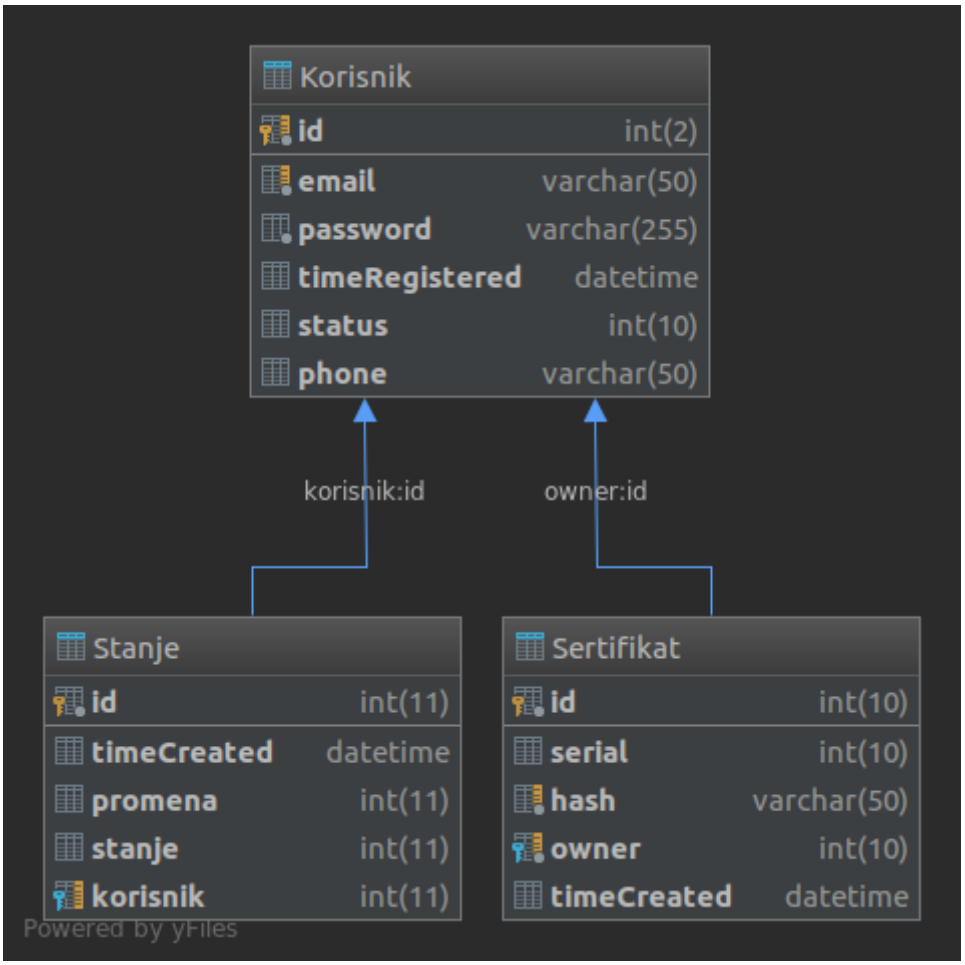
Slika 17 Unošenje PIN-a



Slika 18 Greška: pogrešan PIN

4.2.3. Dinamička distribucija klijentskih sertifikata

Klijentski SSL sertifikati su osnova modela bezbednosti implementiranih u ovoj aplikaciji. Da bi poslužili kao dodatni faktor autentifikacije, moraju biti personalizovani, odnosno povezani sa svakim korisničkim nalogom, i distribuirani korisniku na adekvatan način. PHP aplikacija koja služi za registraciju korisnika, istovremeno kreira i sertifikat putem openssl alata, upisuje hash sertifikata u bazu, pravi dinamički link i QR kod za download i šalje PIN kojim je sertifikat zaključan na SMS i e-mail korisnika (koristeći PHP integracije sa servisima Twillio i Sendgrid.

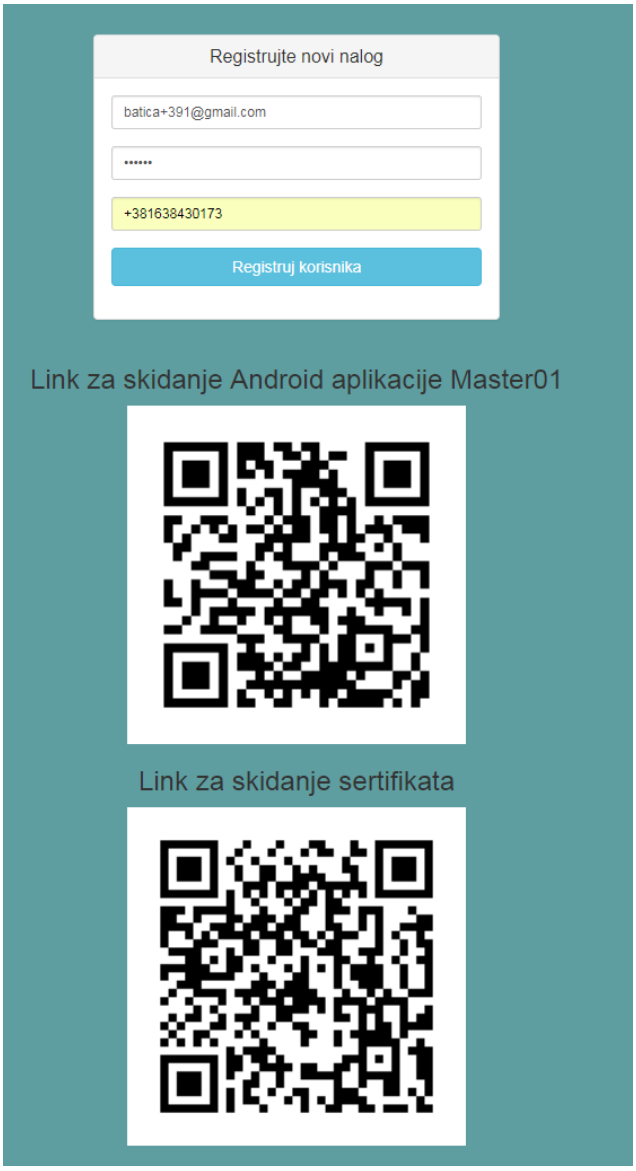


Slika 19 Šema baze podataka na serveru

Izvorni kod aplikacije se nalazi na adresi https://github.com/batica81/master01_backend , a portalu za registraciju se može pristupiti na https://master01.duckdns.org/ .

```
$email = $_POST['email'];
$pin = rand_num_pass(6);
chdir('../cert');
```

```
if (ENV_OS == 'windows') {
    $output = shell_exec('autocert.bat '. $email .' '. $pin);
} else {
    $output = shell_exec('./autocert.sh '. $email .' '. $pin);
    copy('../cert/certs/'.$email.'.p12',
'../register/tempcert/'.$email.'.p12');
}

$sha1temp = explode( 'Fingerprint=' , $output  )[1];
$sha1 = str_replace (':', '', $sha1temp);

$database->insert('Sertifikat', [
    'hash' => $sha1,
    'owner' => $userId
]);
```



Slika 20 Registracija korisnika

```
printf 'Generisanje klijentskog sertifikata:\n'
openssl req -new -passout pass:$2 -config client.conf -out certs/$1.csr -keyout
certs/$1.key

printf 'Potpisivanje klijentskog sertifikata:\n'
openssl ca -config ca.conf -batch -in certs/$1.csr -out certs/$1.crt -
extensions client_ext

printf 'Izvoz klijentskog sertifikata u p12 format:\n'
openssl pkcs12 -export -clcerts -in certs/$1.crt -inkey certs/$1.key -passin
pass:$2 -out certs/$1.p12 -passout pass:$2

printf 'Izvoz klijentskog sertifikata u pem format:\n'
openssl pkcs12 -in certs/$1.p12 -passin pass:$2 -out certs/$1.p12 -out
certs/$1.pem -passout pass:$2 -clcerts

printf 'Prikaz hash vrednosti klijentskog sertifikata:\n'
openssl x509 -noout -fingerprint -sha1 -inform pem -in certs/$1.pem
```

4.2.4. Simetrična kriptografija na aplikativnom sloju (AES256 end to end)

Radi ilustracije implementacije simetrične kriptografije na aplikativnom sloju poslužićemo se sistemskim funkcijama dostupnim u standardnim Java (Android) kriptografskim bibliotekama, a kojima ćemo radi jednostavnosti korišćenja pristupati kroz namenski kreiranu Encryptor klasu.

Usled teškoće distribucije tajnog ključa, a što je ujedno najveća mana simetrične kriptografije, pribeći ćemo kompromisnom rešenju i ključ ćemo izvoditi iz lozinke kojom korisnik inače pristupa sistemu. Takođe, šifrovaćemo samo jednu stranu komunikacije (od servera ka klijentu).

```
Encryptor enc = new Encryptor();
String jsonString2 = IOUtils.toString(inputStream, StandardCharsets.UTF_8);
String tempString = enc.decrypt(password, jsonString2);
String jsonString3 = JsonWriter.formatJson(tempString);
myTextView.setText(jsonString3);
```

S obzirom da PHP od verzije 7 u potpunosti podržava integraciju sa OpenSSL paketom, a imajući u vidu visoku sofisticiranost i prilagođenost standardima ovog paketa, interoperabilnost sa Java (Android) klijentom je moguća bez ikakvih dodatnih intervencija.

Sistemske metode PHP-a za inicijalizaciju AES algoritma prihvataju nekoliko uobičajenih parametara, a mi smo dodali Base64 enkodiranje radi kompatibilnosti sa već postojećom Encryptor klasom koju koristimo na Android strani.

```
define('AES_256_CBC', 'aes-256-cbc');
$encryption_key = hex2bin( hash('sha256', $password));
$iv = openssl_random_pseudo_bytes(openssl_cipher_iv_length(AES_256_CBC));
$data = json_encode($jsondata);
$encrypted = openssl_encrypt($data, AES_256_CBC, $encryption_key,0 , $iv);
$encrypted = $iv. base64_decode($encrypted);
echo base64_encode($encrypted);
```

Šifrovani podaci primljeni od servera se prevode u čitljiv oblik obrnutim postupkom, a cela procedura je potpuno transparentna za korisnika.

4.2.5. Enkripcija statičnih podataka

Kao primer enkripcije statičnih podataka, u aplikaciji smo implementirali šifrovanje AES algoritmom. Ključ se izvodi iz lozinke koja služi za pristup aplikaciji, a podaci se čuvaju u kontejner fajlu koji se nalazi na sistemskoj particiji, odnosno predstavlja interno skladište aplikacije. Pri uobičajenom korišćenju uređaja, ovom delu sistema može pristupiti samo konkretna aplikacija, dok je to takođe omogućeno i administratorskom (root) nalogu.

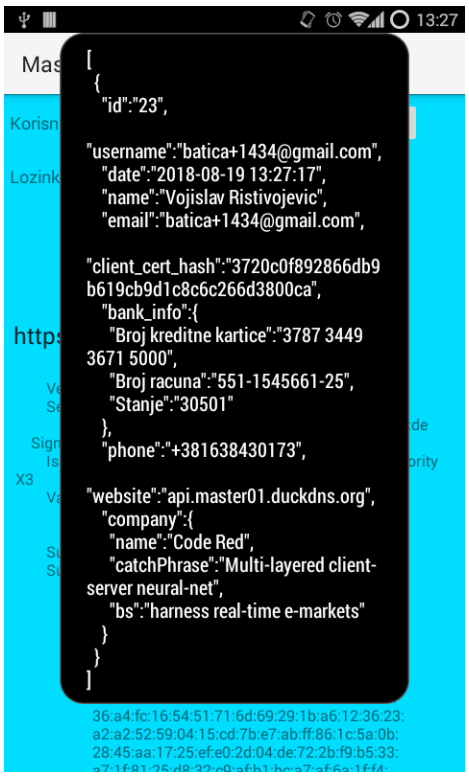
```
Encryptor enc = new Encryptor();
String cyphertext = enc.encrypt(password, plaintext);
outputStream = openFileOutput(filename, Context.MODE_PRIVATE);
outputStream.write(cyphertext.getBytes());
outputStream.close();
```

Encryptor klasa sadrži konkretnu implementaciju šifrovanja i dešifrovanja:

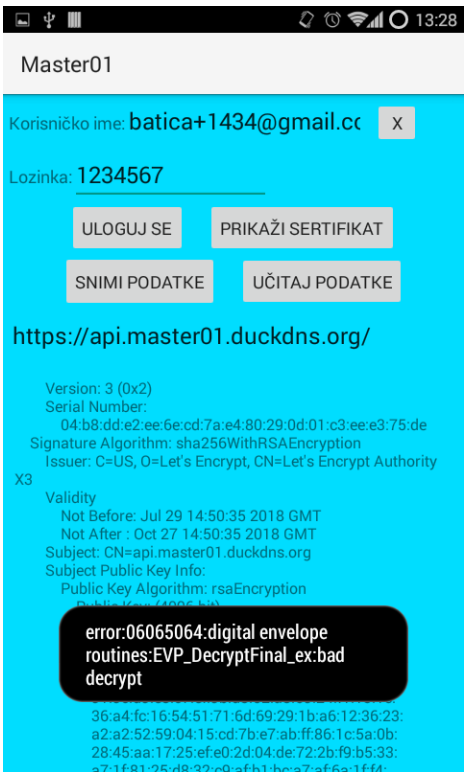
```
byte [] bytePlaintext = plaintext.getBytes();

// Use password hash as a key
MessageDigest md = null;
try {
    md = MessageDigest.getInstance("SHA-256");
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
}
byte[] key = (md.digest(password.getBytes()));
```

```
// Create Initialisation vector
SecureRandom sr = new SecureRandom();
byte[] values = new byte[16];
sr.nextBytes(values);
byte[] encIV = values;

// Initialise cipher
Cipher aes = null;
aes = Cipher.getInstance("AES/CBC/PKCS5Padding");
SecretKeySpec k = new SecretKeySpec(key,"AES_256");
aes.init(Cipher.ENCRYPT_MODE, k, new IvParameterSpec(encIV));
byte [] tempByteCiphertext = aes.doFinal(bytePlaintext);

// Concatenate IV
ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
outputStream.write(encIV);
outputStream.write(tempByteCiphertext);
byte[] byteCiphertext = outputStream.toByteArray();

return Base64.encodeToString(byteCiphertext, Base64.DEFAULT);
```

Unošenjem ispravne lozinke sadržaj enkriptovanog fajla se prikazuje u „toast“ prozoru, a u slučaju pogrešne ispisuje se poruka o grešci:

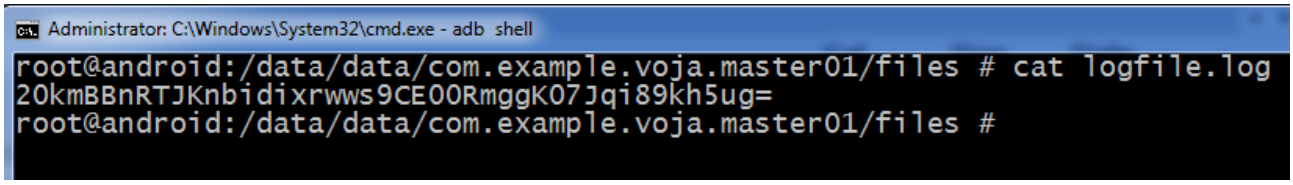


Slika 21 Uspešno učitani podaci



Slika 22 Greška: pogrešna lozinka

Ukoliko se na bilo koji način dođe do šifrovanog fajla, uvid u njegov sadržaj nije moguć bez poznavanja lozinke i konkretnog algoritma koji je upotrebljen.



Slika 23 Sadržaj šifrovanog fajla

4.3. Opis testnog sistema

4.3.1. Test infrastruktura

Testni sistem se sastoji od računara (laptopa) koji se nalazi na istoj lokalnoj mreži odakle aplikacija pristupa serveru za razmenu podataka. Na ovaj način je moguće pokusati većinu relevantnih Man In The Middle tehnika, a koje su najverovatniji vektor napada u ovakvom scenariju. Aplikacije koje će biti korišćene su: Wireshark, MitmProxy, SSLStrip, ArpSpoof itd. Takođe će biti isprobana scenarija kada napadač raspolaže ključem CA sertifikata kome sistem (Android) implicitno veruje.

Od alata ćemo koristiti programe mitmproxy, iptables i arpspoof na Linux platformi. Mitmproxy je konzolni alat koji omogućava uvid i modifikaciju HTTP i HTTPS saobraćaja. Za razliku od Wireshark-a i sličnih alata koje možemo koristiti za analizu običnog mrežnog saobraćaja, mitmproxy ima mogućnost presretanja SSL/TLS komunikacije.

Očekivani rezultat ponašanja naše, ojačane, aplikacije je nemogućnost ostvarivanja konekcije kada je bilo koji element sistema poverenja narušen napadom, kao i adekvatno prijavljivanje greške od strane aplikacije. Takođe će radi poređenja biti prikazano ponašanje nekih drugih mobilnih aplikacija koje ne primenjuju tehnike zaštite implementirane u našoj aplikaciji.

4.3.2. Test alati

Wireshark je najpoznatiji I najrapstranjeniji alat za analizu mrežnih protokola. Njime se do detalja može prikazati ponašanje mrežnih servisa I protokola, pa se smatra standardnim softverskim paketom za ove svrhe. Razvija se pod licencom otvorenog koda, a rad na njemu traje od 1998. godine. Pre nego što se Wireshark uspostavio kao industrijski standard, programi ili specijalizovani uređaji ovog tipa bili su retki, skupi I uglavnom nedostupni široj javnosti. Koristi se za dijagnostifikovanje problema na mreži, utvrđivanje poštovanja bezbednosnih protokola, proveru ponašanja aplikacija I protokola koji koriste mrežu, a ima I veoma zapaženu ulogu u obrazovnoj sferi gde dolazi do izražaja njegov detaljan grafički interfejs I obimna ažurna dokumentacija. Mi ćemo ga u ovom radu pre svega koristiti za potvrđivanje ispravnosti mrežne konfiguracije I osnovnog funkcionisanja HTTPS protokola.

S obzirom da Wireshark u osnovnoj konfiguraciji ne može da pomogne kod analize sadržaja kriptografski zaštićenih komunikacija putem HTTPS protokola, u te svrhe ćemo koristiti poseban alat **MitmProxy**. Običan posredni (proxy) server takođe nema mogućnost uvida u šifrovane konekcije, već ih samo prosleđuje, eventualno menjajući zaglavlje paketa. MitmProxy postiže ovu funkcionalnost vršeći posebnu vrstu aktivne izmene mrežnog saobraćaja, poznatiju kao napad "čoveka u sredini" (man in the middle). Ono što karakteriše ovaj softverski paket je da veoma sofisticiranu varijantu tog napada prilagođenu HTTPS protokolu vrši transparentno, te je veoma pogodan za analizu šifrovanog saobraćaja I od strane korisnika koji nisu verzirani u primeni kriptografskih tehnika, ali takođe I od strane stvarnih napadača kojima znatno olakšava aktivnosti.

Osnovna funkcionalnost koju MitmProxy implementira podrazumeva da, pošto je aktiviran na za to pogodnoj "lokaciji" odnonsno putu između klijenta I servera, aktivno presreće njihovu komunikaciju

I to tako što se pretvara kao da je ona druga strana (prikazuje se kao klijent serveru I kao server klijentu). S obzirom da je ceo sistem infrastrukture javnih ključeva sa centralnim autoritetom od poverenja uspostavljen baš da bi se sprečili napadi ovog tipa, MitmProxy pokušava da automatizje neke sofisticirane napade na ovaj sistem. U podrazumevanom načinu rada, on dinamički kreira serverske sertifikate za sve HTTPS adrese koristeći svoj CA sertifikat (koji je neophodno importovati na klijentu čiji saobraćaj se analizira), ali se lako može podesiti da kreira samopotpisane sertifikate (koji prihvataju mnoge aplikacije I skoro svi korisnici) dok postoji I opcija korišćenja validnog CA sertifikata kojem ciljani korisnik implicitno veruje (česta praksa u korporativnim okruženjima). MitmProxy se može koristiti u interaktivnom ili pasivnom modu, a takođe se može konfigurisati I korz Internet pretraživač. Ovaj alat će biti od ključnog značaja za ilustraciju koncepata predstavljenih u ovom radu.

SSLStrip ima nešto jednostavniji pristup pokušaju analize šifrovanog Internet saobraćaja. Iako vrlo sofisticiran I kompleksan u smislu softverske arhitekture, mehanizam na koji radi je jednostavan. Naime, njegov cilj je da presretne inicijalnu konekciju ka servisu zaštićenom HTTPS-om I pokuša da je uspostavi bez upotrebe kriptografskih protokola. Ukoliko je traženi servis dostupan u takvom obliku, to je veoma jednostavno, ali čak I ako nije, SSLStrip uspostavlja šifrovanu konekciju ka njemu, dok pokušava da korisniku predstavi vezu kao nezaštićenu. S obzirom da mali broj korisnika obraća pažnju na upozorenja pretraživača o nebezbednosti konekcije, ovi napadi su bili veoma uspešni dok nizu preduzete sistemske mere za njihovo suzbijanje, u vidu HSTS ekstenzije HTTPS potokola. Takođe, većina mobilnih aplikacija više ne dozvoljava dinamičku promenu protokola, te ovaj napad I sam alat više nema toliki značaj u tom kontekstu, što ćemo I prikazati.

Da bi se napadi tipa “man in the middle” izveli van potpuno kontrolisanog okruženja, najčešće je neophodno preusmeriti komunikaciju korisnika na način koji nije u skladu sa aktuelnom konfiguracijom mreže, osdnosno napadač želi da mu se predstavi kao adekvatno prvo odredište ka daljim mrežama (gateway). U te svrhe koristi se dobro poznati mehanizam falsifikovanja keša ARP protokola (adress resolution protocol cache poisoning). Alat **ArpSpoof** omogućava manipulaciju paketima na najnižem logičkom sloju, te se njime uređaji na mreži mogu navesti da pakete šalju na drugačija odredišta od onih inicijalno konfigurisanih. Budući da se ovi specijalni paketi suštinski ne razlikuju od legitimnih, napade ovog tipa je veoma teško sprečiti, te se u većini mreža to ni ne pokušava. Konkretno, ArpSpoof meti šalje informaciju da se IP adresa čvorišta (gateway router) nalazi na MAC adresi napadača, a čemu većina operativnih sistema u podrazumevanoj konfiguraciji implicitno veruje. Nakon ovog koraka, komunkacija je prusmerena kroz napadačev uređaj I može se njome manipulisati korišćenjem nekih od gore navedenih alata. Treba napomenuti da za ovaj alat

postoje legitimne upotreba, kao što je analiza saobraćaja mobilnih aplikacija koje ignorišu sistemska podešavanja vezana za podrazumevano čvorište (gateway). ArpSpoof ćemo koristiti u kombinaciji sa MitmProxy alatom radi ilustracije realnih scenarija napada.

4.3.3. Očekivani rezultati

S obzirom da ćemo testiranje izvršiti u više faza, opisaćemo očekivano ponašanje aplikacije i sistema u svakoj od njih.

Budući da ruter u lokalnoj mreži nema implementirane nikakve zaštite od ARP redirekcije (spoofing), što je uobičajena konfiguracija, prva faza napada će biti uspešna bez obzira na to koja aplikacija je aktivna na telefonu, jer se izvršava na nižem sloju IP protokola. Preusmereni paketi će biti čitljivi sa računara napadača, s tim što će naravno biti enkriptovani.

Saobraćaj preusmeren kroz MitmProxy ćemo prvo presecati korišćenjem CA sertifikata kome android sistem ne veruje, a radi ilustracije adekvatnog podrazumevanog nivoa zaštite u sistemskim klasama na novijim Android platformama. Očekujemo da se pojavi poruka o grešci u obe aplikacije, jer sistem odbija da uspostavi konekciju.

U sledećem slučaju, kada prihvatimo sertifikat na sistemskom nivou, očekujemo da prva aplikacija uspostavi konekciju kroz MitmProxy, dok će je druga odbiti jer se hash vrednost javnog ključa prikazanog serverskog sertifikata ne poklapa sa onom koja postoji u aplikaciji.

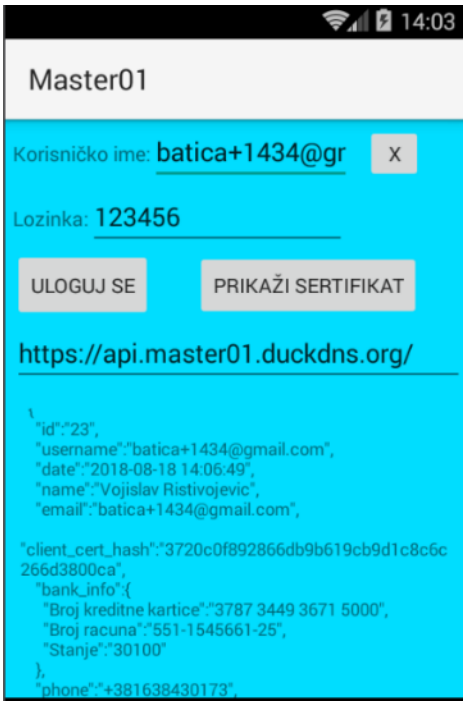
Takođe u ovom slučaju, konekcija prve aplikacije će biti prekinuta odmah nakon što ona ne bude u mogućnosti da prezentuje klijentski sertifikat serveru. Konekcija pod istim uslovima ka istom backend-u kroz drugačije konfigurisani server (koji ne zahteva klijentski sertifikat) će uspeti, u kom slučaju će napadač imati potpun uvid u istu.

U slučaju koji ovde nećemo obrađivati, a to je kada napadač poseduje validan klijentski sertifikat (čest slučaj je korišćenje istog sertifikata u svim instancama aplikacije), MitmProxy bi uz manje izmene konfiguracije mogao da ga iskoristi za dešifrovanje oba smera presretnute komunikacije.

Takođe, konekcija sa povučenim (revoked) klijentskim sertifikatom neće biti moguća, jer NginX vrši proveru pre prosleđivanja zahteva.

Još jedna varijacija sofisticiranog napada je kada se poseduje validan sertifikat nekog konkretnog korisnika. U ovom slučaju konekcija (manipulisana od strane napadača) će stići do backend aplikacije, ali će biti prekinuta čim se ustanovi neslaganje hash-a predstavljenog sertifikata sa onim u bazi, vezanim za tog korisnika. Opciono, posle tri takva pokušaja povezivanja, sertifikat će biti automatski povučen.

Jedino ojačana aplikacija, koja proverava hash serverskog sertifikata (pinn-ovanog), prilaže validan klijentski sertifikat, korisničko ime i lozinku, će moći da razmeni podatke sa serverom u regularnim uslovima, dok će u slučaju različitih napada prikazati odgovarajuću grešku.



Slika 24 Uspešna konekcija

4.4. Testiranje implementacije korišćenjem PenTest alata

4.4.1. Postupak testiranja

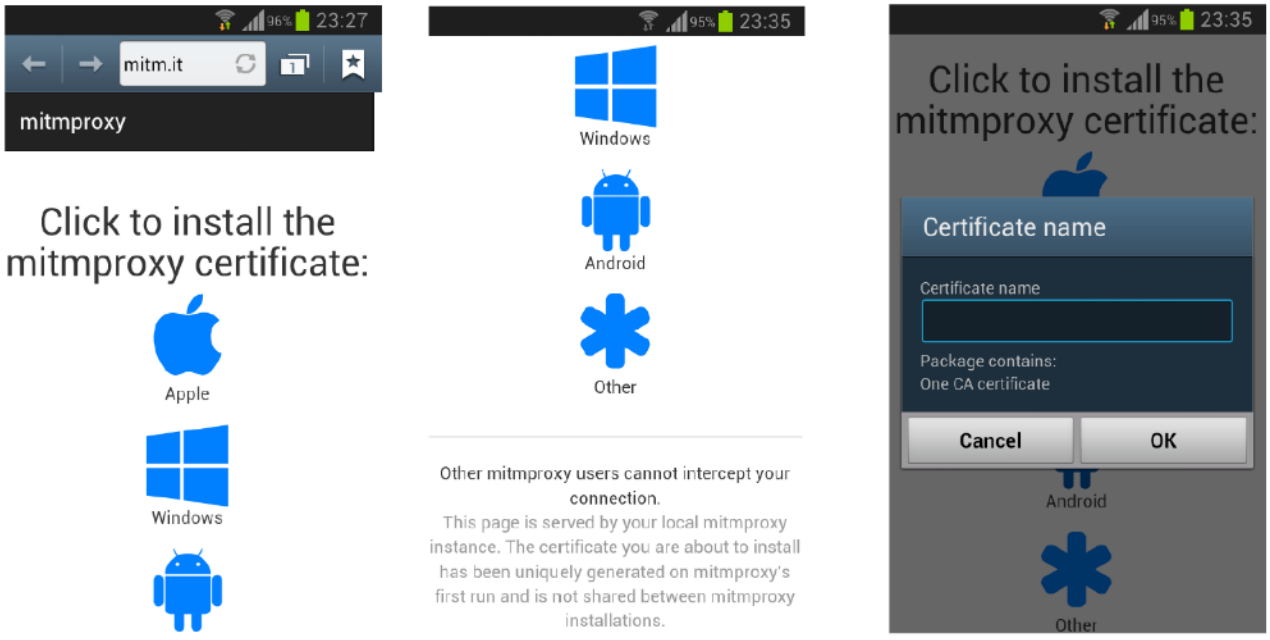
Postupak testiranja izvršićemo u više faza. Za potrebe istraživanja ćemo koristiti namenski kreirane Android aplikacije Tzrm01 (bez implementiranih tehnika zaštite, kod dostupan na: https://github.com/batica81/Tzrm01) i aplikaciju Master01 koju smo prethodno opisali. Obe komuniciraju isključivo putem HTTPS protokola sa PHP aplikacijom na NginX serveru. Serverskoj aplikaciji se može pristupiti putem dva domena: api.master01.duckdns.org i

cinamontest.duckdns.org/master01_backend/. U prvom slučaju NginX je konfigurisan tako da zahteva i proverava klijentski sertifikat, dok se drugom može pristupiti bez ove vrste autentifikacije.

Od alata ćemo koristiti mitmproxy, iptables i arpspoof na Linux platformi, a radi postizanja ponovljivih rezultata, izolovanu lokalnu mrežu koju čine laptop (napadač) i Android telefon (meta) povezanu na Internet.

S obzirom da ćemo demonstraciju izvoditi u kontrolisanim uslovima, neophodno je izvršiti određene pripremne radnje na svim uključenim elementima. Oba uređaja ćemo povezati na lokalni WiFi Access Point, koji iako ima zaštitu u vidu WPA protokola na Data Link sloju, nije dodatno konfigurisan da pokuša da spreči arpspoof napade koji se vrše na višem, mrežnom sloju.

Ključni element za uspešnu simulaciju izvođenja MITM napada na HTTPS protokol je navođenje klijent uređaja da prihvati Root sertifikat koji će biti korišćen za automatsko izdavanje serverskih sertifikata za adrese koje klijentske aplikacije posećuju. Softverski paket mitmproxy sadrži efikasan način za postizanje ove funkcionalnosti u kontrolisanim uslovima, a uz manje modifikacije web interfejsa, isti se može primeniti i u realnom scenariju. Naime, pošto je komunikacija mete preusmerena kroz mitmproxy, odlaskom na adresu mitm.it program će korisniku predstaviti formu za instalaciju sertifikata za odgovarajuću platformu.



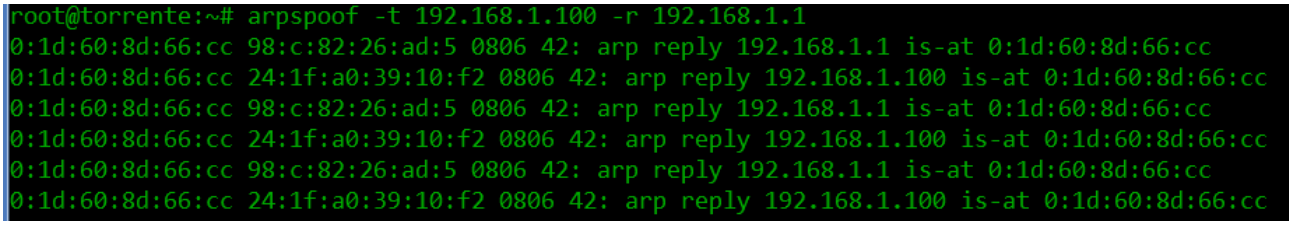
Slika 25 MitmProxy portal

Na laptopu kojim će napad biti izvršen instalirani su programi mitmproxy i arpspoof, uključena je opcija prosleđivanja paketa na druge IP adrese, a u firewall tabelu su dodata pravila za preusmeravanje dolaznog saobraćaja ka portovima 80 i 443 na port 8080 lokalne mašine.

```
#!/bin/sh
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8080
iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 8080
```

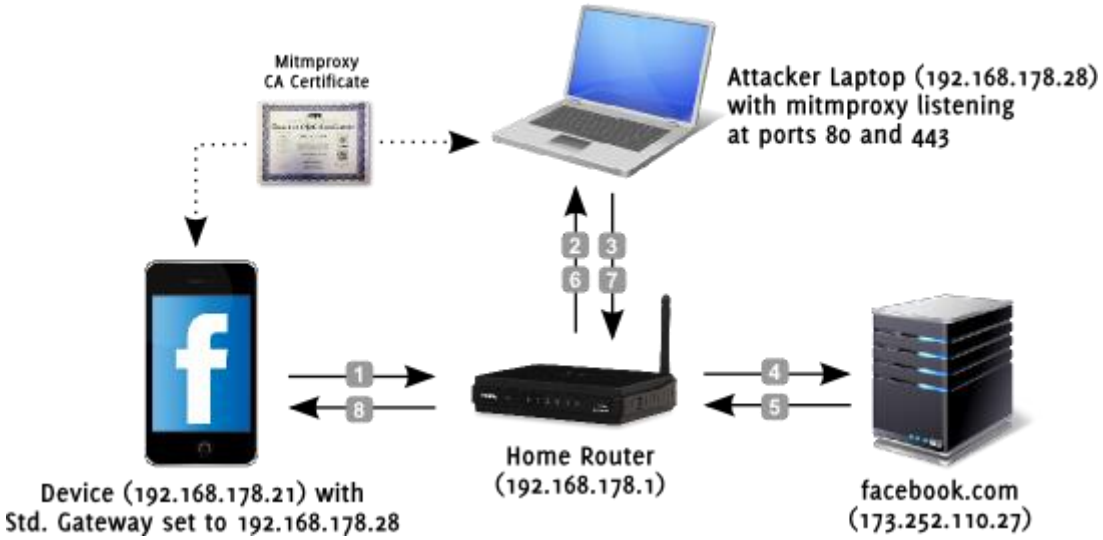
U realnom napadu, DNS redirekcijom je moguće automatski navesti korisnika na drugačije dizajniranu stranu koja objašnjava zašto je neophodno instalirati sertifikat. Drugi načini na koje je ovo moguće postići navedeni su u uvodu, a bitno je napomenuti da je ipak vrlo teško izvesti ovaj korak bez znanja i delimičnog učešća samog korisnika. Pošto je i ovaj poslednji preduslov ispunjen, pokrenućemo mitmproxy u transparent modu i arpspoof kojim ćemo izvršiti redirekciju saobraćaja namenjenog serveru sa klijenta na laptop:

```
mitmproxy --mode transparent --showhost
arpspoof -t 192.168.1.100 -r 192.168.1.1
```



Slika 26 Arpspoof redirekcija

Komunikacija koja se odvija od trenutka pokretanja programa arpspoof preusmerena je kroz laptop, prihvaćena od strane iptables-a i upućena na lokalni port 8080 gde je automatski dešifrovana, te se može vršiti uvid u podatke i njihova eventualna izmena.

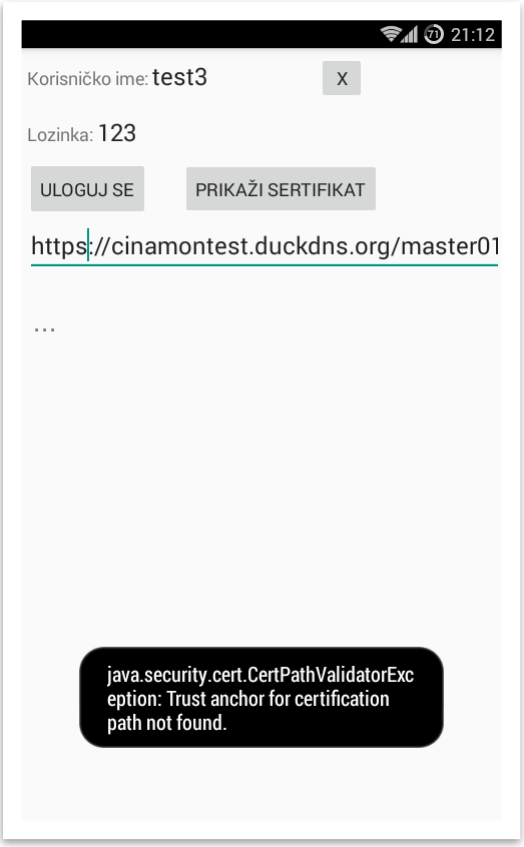


Slika 27 MITM šema

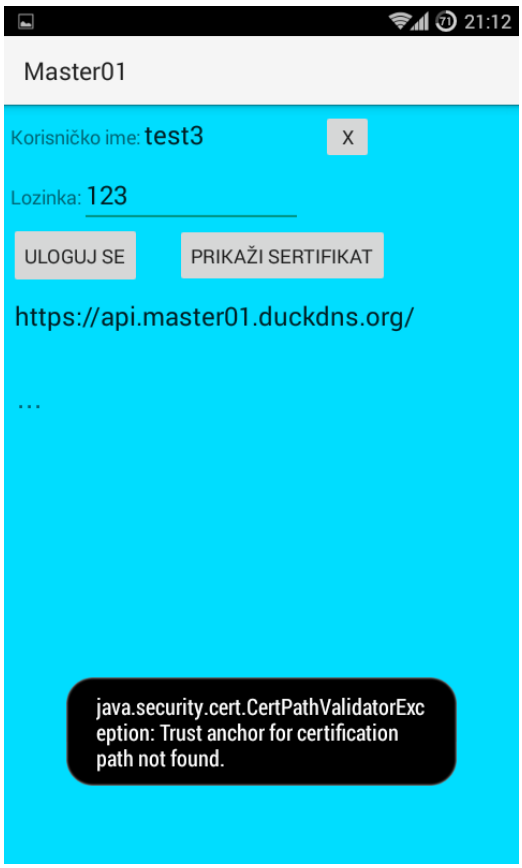
4.4.2. Poređenje ponašanja standardne i ojačane aplikacije

Da bismo jasnije ukazali na razlike u ponašanju obične i ojačane aplikacije, prikazaćemo paralelno rezultate pojedinih aktivnosti izvršenih na obe, u istim ili (približnim) uslovima.

U prvom scenariju, iskoristili smo CA sertifikat kome sistem ne veruje, što je rezultovalo odbijanjem konekcije uz istu grešku na obe aplikacije.

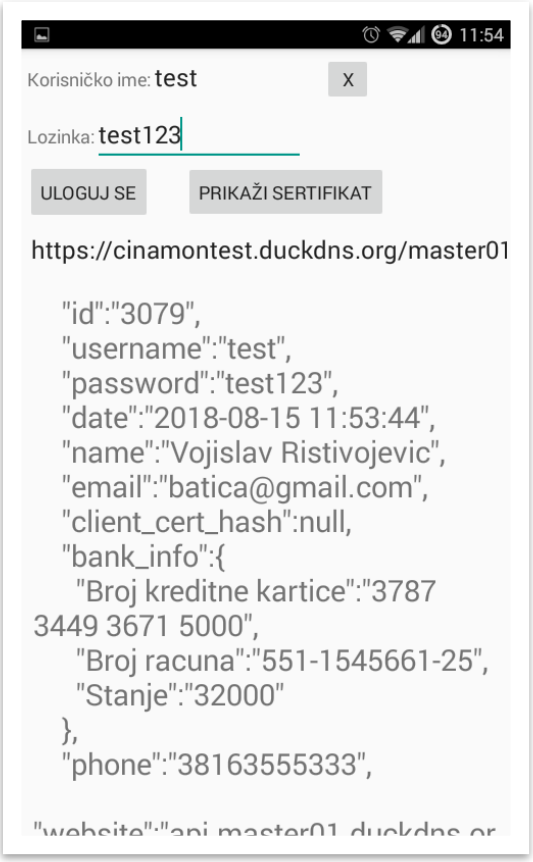


Slika 28 Sistem ne veruje CA

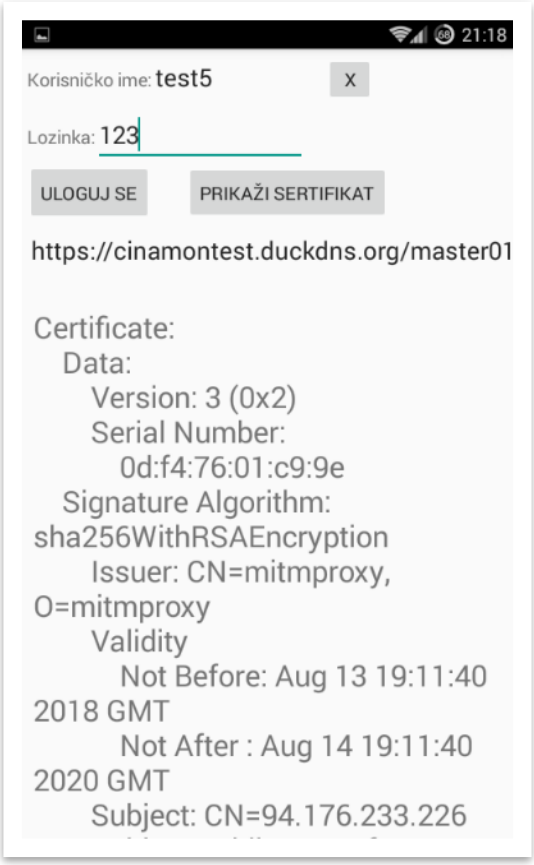


Slika 29 Sistem ne veruje CA

U sledećem koraku, importovali smo sporni sertifikat u sistemski repozitorijum, nakon čega je prva aplikacija uspostavila konekciju. Detalji sertifikata ukazuju da je prihvaćeni sertifikat potekao od strane napadača. Ojačana aplikacija odbija konekciju uz prethodno prikazanu poruku o grešci (certificate pinning failure).

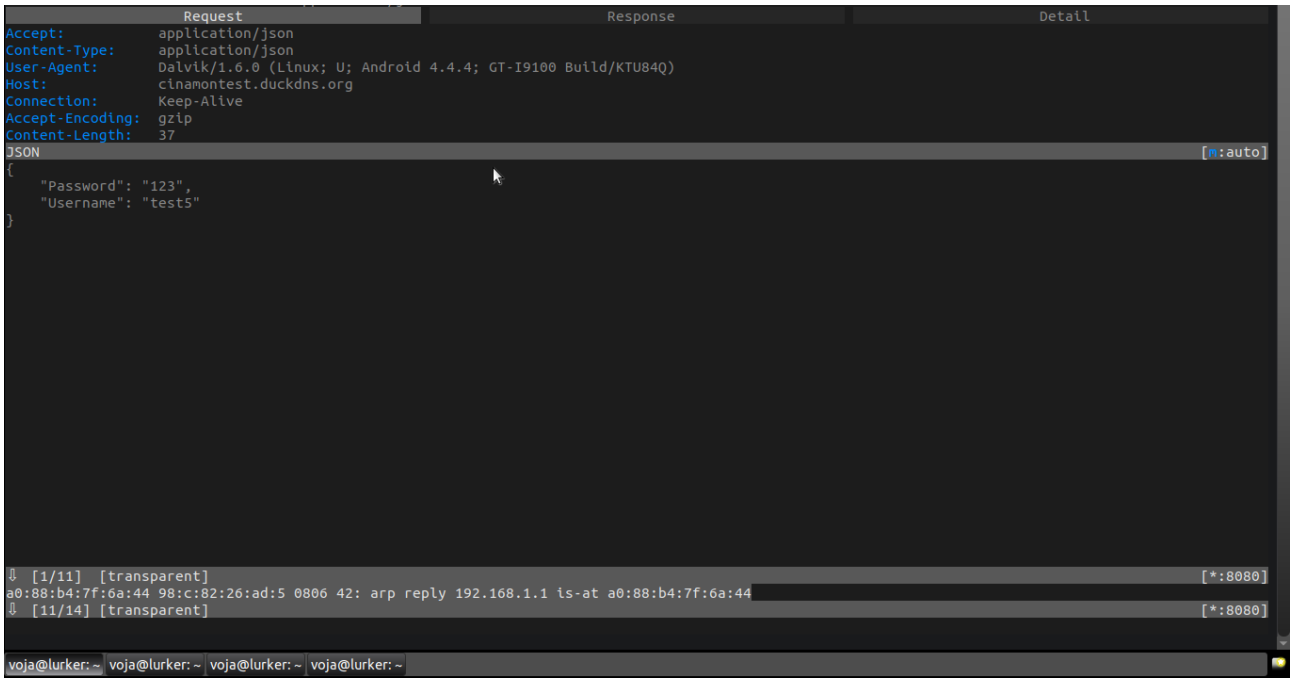


Slika 30 Konekcija bez klijentskog sertifikata

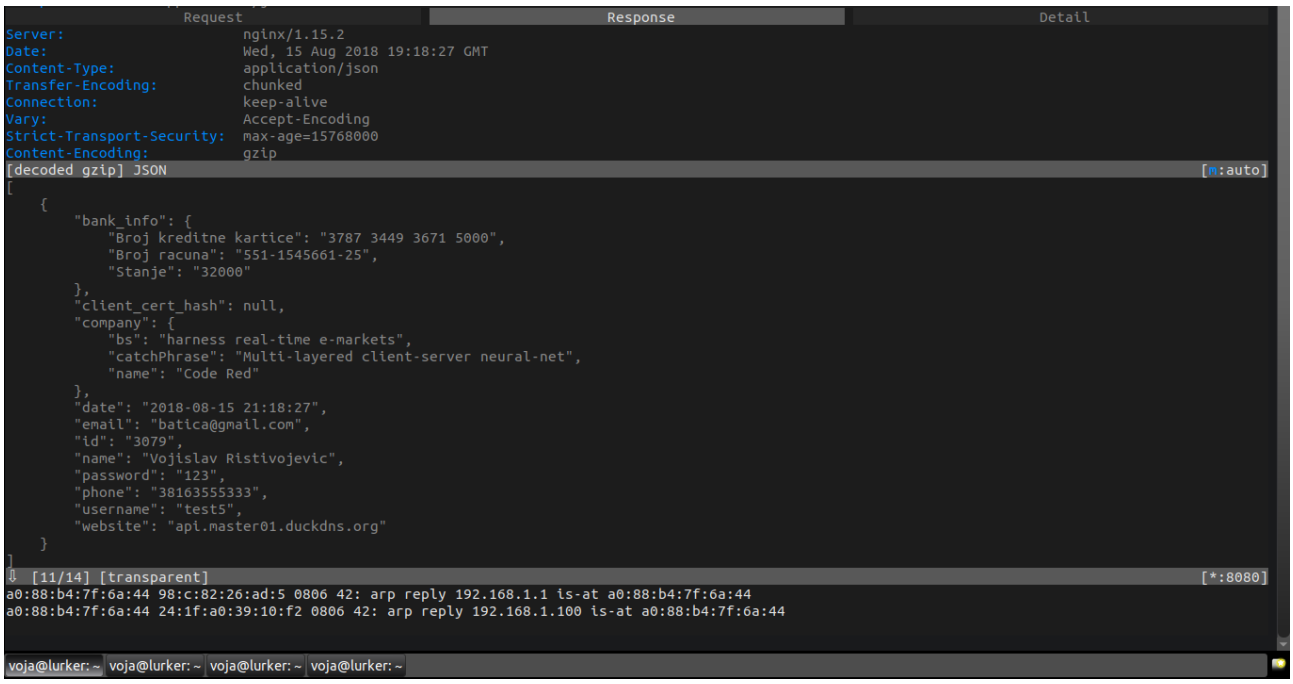


Slika 31 Detalji MitmProxy sertifikata

Napadač je u mogućnosti da vrši uvid ili izmene u oba smera komunikacije zaštićene HTTPS-om.

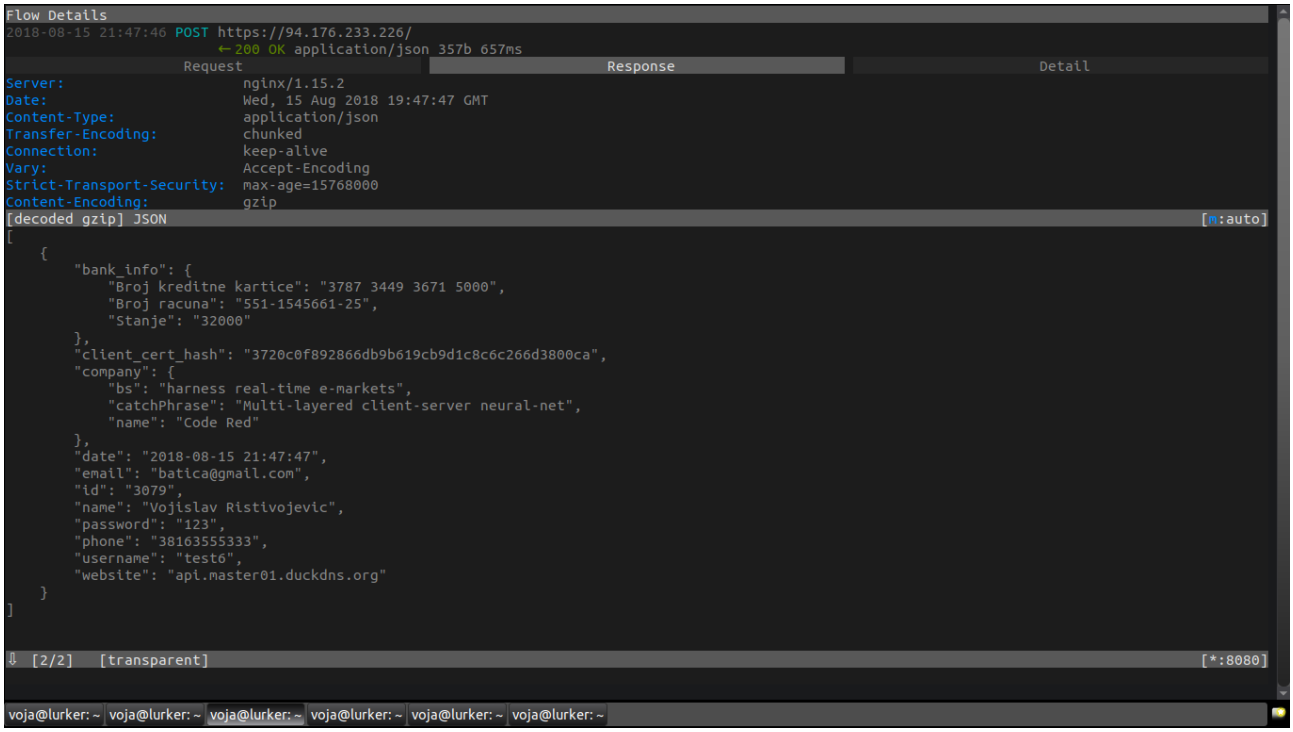


Slika 32 Zahtev klijenta



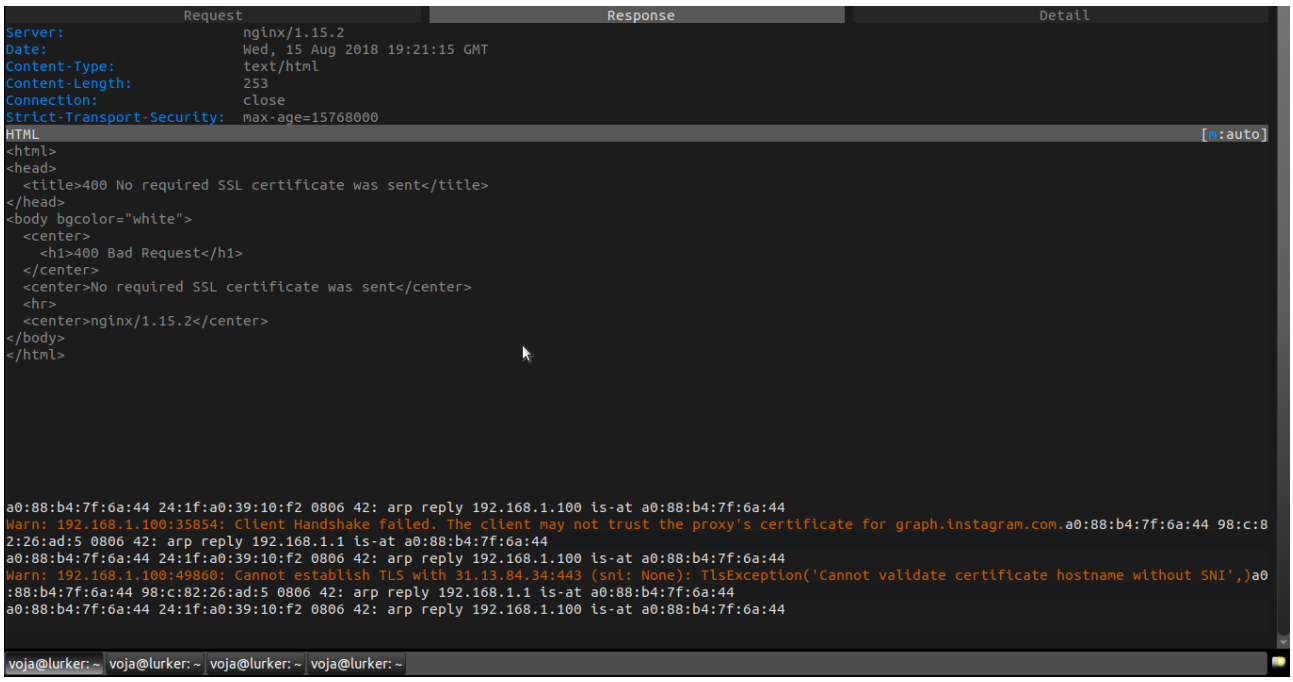
Slika 33 Odgovor servera

U malo verovatnom scenariju kada napadač ima pristup validnom klijentskom sertifikatu, takođe će moći da presretne komunikaciju (verzija aplikacije bez certificate pinning-a).



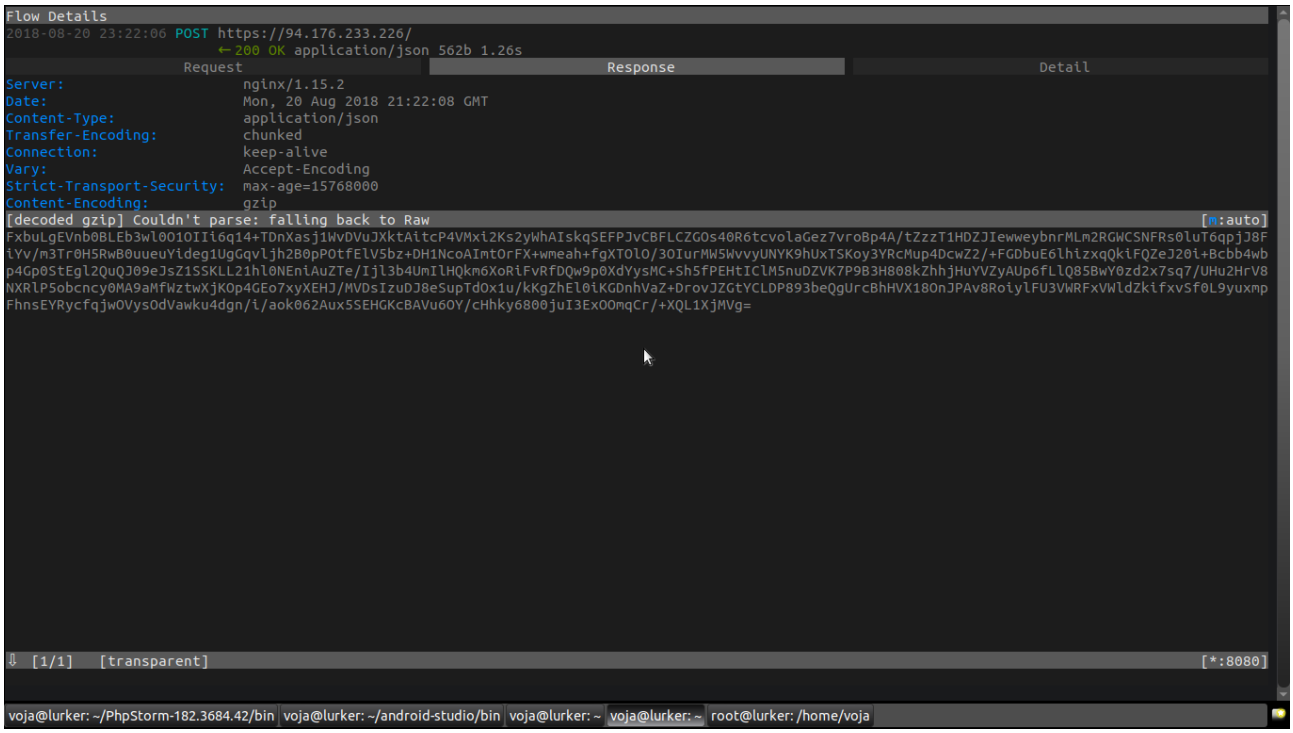
Slika 34 Napadač poseduje klijentski sertifikat

Kada pokuša da preseče komunikaciju ojačane aplikacije bez adekvatnog klijentskog sertifikata, konekcija će biti prekinuta od strane NginX servera.



Slika 35 Odbijena konekcija

U malo verovatnom scenariju kada napadač ima pristup tajnom ključu pinnovanog CA sertifikata i validnom klijentskom sertifikatu, moći će da presretne komunikaciju, ali će ona ostati zaštićena end to end enkripcijom na aplikacionom sloju.



Slika 36 Enkripcija na aplikacionom sloju

5. Diskusija i pravci daljeg razvoja

5.1. Komentari rezultata testiranja

U prethodnom poglavlju smo na konkretnom primeru prikazali neke od mogućih scenarija napada kojima mobilne aplikacije, odnosno njihovi korisnici, mogu biti izloženi u realnom svetu. Iako sprovedeni van laboratorijskih uslova, te se ne mogu tretirati kao eksperiment u strogom smislu, ovi testovi su pokazali da su neadekvatno ojačane aplikacije, odnosno one kod kojih nisu primenjene složene kriptografske tehnike zaštite veoma ranjive na pojedine, netrivijalne napade, a koji su ipak dostupni sve širem krugu potencijalnih napadača. Iako prikazani na primeru mobilnih aplikacija, kao što je već više puta navedeno, ovi napadi se mogu primeniti i u drugim situacijama, odnosno kada se komunikacija obavlja kroz internet pretraživač ili između dva nezavisna uređaja (npr. IoT senzor i server).

Specifični uslovi u kojima je moguće izvesti ove napade su dostupni moćnim napadačima, ili su prilagođeni automatskom izvršavanju u određenim (pre svega korporativnim) okruženjima kada se kao pomoćna sredstva koriste posebni uređaji za analizu mrežnog saobraćaja, a koji u suštini funkcionišu na osnovu postupka prikazanog u prethodnom delu rada. Oni su unapred opremljeni adekvatnim (uglavnom intermidate) sertifikatima i pripadajućim ključevima, koje je izdalo neko od CA tela kojima ciljani sistemi unapred raspolažu. Ova oprema je skupa i teško se nabavlja, ali ipak nije predviđena isključivo za korišćenje od strane državnih organa, te postoji značajna mogućnost njene zloupotrebe u korporativnim okruženjima.

Prvi deo napada, u kome se komunikacija mete preusmerava putem falsifikovanja MAC adrese rutera (arp spoofing) smatramo trivijalnim, jer većina mrežnih konfiguracija ne raspolaže nikakvim mehanizmima za sprečavanje istog, ili oni nisu implementirani. Slično se može postići i slanjem DNS odgovora koji vraćaju IP adresu napadača za svaki domen, što je donekle manje primenjivo u kontekstu testiranja mobilnih aplikacija koje ponekad ne koriste ovaj servis, odnosno komuniciraju direktno sa predefinisanom IP adresom servera.

Iako zamišljen kao ilustracija, odnosno simulacija realnog napada izvršenog od strane sofisticiranog napadača, scenario koji smo opisali, u kome se na uređaj preinstalira sertifikat koji je izdalo CA pod kontrolom napadača ne treba u potpunosti zanemariti. Savremeni Android sistemi (od verzije 4.0 i više) daju brojna upozorenja i drastično otežavaju manipulaciju listom CA sertifikata od strane

korisnika, dok izmene preinstaliranih sistemskih sertifikata predstavljaju veoma složenu radnju za koju je neophodan administratorski pristup uređaju. Ipak, ovaj isti napad je daleko lakše izvršiti u korporativnom okruženju, odnosno na desktop računarima koji se centralizovano administriraju. Ovaj postupak ne mora nužno biti izvršen uz znanje administratora, već napadač može biti bilo ko ko je uspostavio kontrolu nad centralnim serverom za administraciju radnih stanica (npr. domen kontroler u Windows okruženju), a što je najčešća meta spoljnih i unutrašnjih napada na infrastrukturu neke kompanije. Ne treba zanemariti ni situaciju kada se korisnik (mobilne aplikacije ili desktop računara) odluči na upotrebu nekog (uglavnom besplatnog ili vrlo povoljnog) VPN servisa, čime on faktički postaje deo mreže u koju polaže apsolutno poverenje. Tom prilikom postoji mogućnost navođenja korisnika (koji raspolaže nedovoljnim znanjima) na instalaciju CA sertifikata koji je pod kontrolom zlonamernog pružaoca VPN usluge, a pod izgovorom dodatnog obezbeđenja njegove privatnosti. S obzirom da se pretpostavlja da korisnici VPN usluga prenose osetljive podatke ovim putem, nije neverovatno da potencijalni napadač isfabrikuje čitavu VPN infrastrukturu da bi došao do istih.

Izvođenje MITM napada na HTTPS (SSL/TLS) konekcije, kao i adekvatna zaštita od istih zahteva temeljno poznavanje ovih protokola i njihovih kriptografskih osnova (PKI modela poverenja) kako od strane napadača, tako i od programera i administratora sistema. Ukoliko se izvede efikasno, u potpunosti obesmišljava većinu mera zaštite odnosno sadržaji komunikacija korisnika bivaju u potpunosti kompromitovani. Opisane protivmere su dovoljno efikasne da drastično uspore ili u potpunosti onemoguće ove napade, dok sa druge strane zahtevaju blagovremeno prethodno planiranje radi njihove implementacije u brojnim segmentima sistema komunikacije.

Ostali pomoćni mehanizmi zaštite koje smo primenili (čuvanje podataka u šifrovanom kontejneru, dopremanje PIN-a bezbednijim kanalom, automatska invalidacija klijentskog sertifikata) povećavaju bezbednost aplikacije, ali sami po sebi ne mogu garantovati isti nivo pouzdanosti kao adekvatno primenjene kriptografske mere zaštite na arhitekturalnom nivou aplikacije i serverske konfiguracije. Svaka od korišćenih tehnika može biti podvrgnuta nekom manje ili više uspešnom napadu, a šta više, uvek postoji mogućnost otkrivanje neke nove ranjivosti ili indirektnog (side channel) napada sa kojima se ne može unapred računati.

Ipak, slojevita primena odbrambenih mehanizama, usklađena sa ranije pomenutim "dubinskim" defanzivnim konceptima (defense in depth) praktično onemogućava trivijalne automatizovane napade, dok značajno usporava i otežava ciljane zlonamerne aktivnosti, čime deluje demotivišuće na potencijalnog napadača. Ono što je bitno imati u vidu prilikom projektovanja i izrade ojačane softverske i serverske infrastrukture je da uvek treba koristiti samo poznata i proverena rešenja, a

konkretne tehnike primenjivati svrhovito i racionalno, imajući u vidu njihovu adekvatnost za odbijanje verovatnih vektora napada kao i cenu koju njihova implementacija unosi u izradu, održavanje i korišćenje samog softverskog proizvoda.

5.2. Predlog razvoja biblioteke koja olakšava implementaciju dobrih kriptografskih praksi

Iako postoji više načina za adekvatnu primenu HTTPS-a u Android okruženju, te da je ista olakšana postojanjem sistemskih biblioteka i podrške zajednice, ne postoje mehanizmi koji bi primorali programere da ove tehnike implementiraju u svoje Aplikacije, bez obzira na značaj podataka sa kojima one operišu [57]. Takođe, prosta implementacija bilo kakve varijante HTTPS protokola štiti samo od najjednostavnijih napada, dok je za odbranu od aktivnih tehnika man in the middle, neophodno koristiti sve mogućnosti ovog protokola primenjene na celishodan način.

Kao što u višim programskim jezicima dolazi do apstrakcije koja olakšava posao programera, slično se pokušava sa prilagođavanjem i standardizacijom kriptografskih biblioteka, kako bi se procesi njihove integracije u aplikacije učinili pristupačnijim i razumljivijim [57]. Ovo je naročito važno u Android okruženju koje karakteriše veliki broj nezavisnih razvojnih timova, izdavača i programera, a koji nemaju ujednačen nivo znanja ili tehničkih mogućnosti da podjednaku pažnju posvete svakoj fazi procesa izrade aplikacija.

Idealna HTTPS biblioteka za Android bi omogućavala ispravnu upotrebu SSL/TLS protokola bez ulaganja dodatnog napora, dok bi istovremeno zabranjivala izmene bilo kog elementa protokola koje bi narušile sistem poverenja. U ovom modelu bi programer bio odgovoran jedino za kreiranje konekcije i selekciju sertifikacionih autoriteta kojima veruje. Osnovna ideja je maksimalno apstrahovati sve korake osim najneophodnijih za kreiranje bezbedne konekcije, jer svaka dodatna intervencija od strane programera može izazvati grešku. Takođe, postojale bi samo minimalne mogućnosti privremenog isključivanja nekih delova SSL/TLS protokola tokom izrade aplikacije, i veoma stroge kontrole radi sprečavanja puštanja razvojne verzije u produkciono okruženje. Istraživanja pokazuju [57] da trenutna rešenja, odnosno dostupne pomoćne biblioteke za implementaciju HTTPS protokola u Android aplikacijama ne zadovoljavaju ove kriterijume, odnosno da su ili previše komplikovana ili nedovoljno razrađena u smislu posedovanja svih neophodnih opcija.

Kao logičan sledeći korak pri razvoju savremene biblioteke za ove namene, bilo bi forsiranje komunikacije isključivo putem HTTPS protokola, što je delimično standard koji već nastoje da

nametnu najzastupljeniji pretraživači Interneta (Chrome, Mozilla). Potpunim prelaskom na bezbednu komunikaciju bi se onemogućio veliki broj lakih i trivijalnih napada, a istovremeno bi se značajno smanjio broj grešaka pri implementaciji protokola koje nastaju prilikom pokušaja da se održi kompatibilnost aplikacija sa oba protokola (HTTP i HTTPS).

Jedna od naprednih mogućnosti HTTPS protokola, "kačenje" sertifikata - SSL Pinning (o kojoj je bilo reči ranije), bi takođe trebala da bude obavezna u svakoj mobilnoj aplikaciji. Ona sprečava napade od strane zlonamernih CA autoriteta, uklanjanje SSL sloja (SSL stripping), preuzimanje sesije (session hijacking) i druge slične napade [57]. Zajedno sa drugim unapređenjima HTTPS protokola, kao što su pooštravanje uslova za poverenje određenim CA i obavezno korišćenje jakih kriptografskih parametara i šifara, postigla bi se pouzdana osnova na kojoj bi programeri izrađivali znatno kvalitetnije aplikacije u smislu bezbednosti komunikacije i privatnost podataka njihovih korisnika.

Jasna i opširna dokumentacija kao i jednostavna upotreba su najvažniji faktori koji utiču na odluku programera da iskoristi određenu biblioteku. Takođe, da bi kontinuirana poboljšanja svakog softverskog paketa imala smisla za korisnike, neophodna je stalna, dvosmerna komunikacija sa njima, što podrazumeva postojanje određene zajednice i otvorenost obe strane za prihvatanje novih rešenja. Ovaj princip se mora imati na umu u svakoj fazi razvoja biblioteke, pa tako i komponente kao što su prijavljivanje grešaka, protokoli za testiranje, imenovanje klasa i metoda trebaju biti prilagođeni korisnicima (programerima) koji nemaju iskustva sa implementacijom bezbednosnih protokola niti raspolažu domenskim znanjem iz oblasti kriptografije [cl-wei-https1]. Edukativnu dimenziju iz ovih oblasti treba proširiti i na ceo Android ekosistem, te značajno poboljšati postojeću zvaničnu dokumentaciju i razvojna okruženja (IDE) tako da podržavaju razvoj bezbednijih aplikacija istovremeno podižući svest zajednice o značaju doslednog sprovođenja dobrih praksi iz ove oblasti.

6. Zaključak

Istorijski posmatrano, kriptologija je uvek privlačila pažnju značajnog broja istraživača, pre svega lingvista i matematičara, dok je istovremeno zadržavala izvesnu distancu od šire prihvaćenosti u praksi, zbog specifične oblasti koju obrađuje odnosno problema koje rešava. Tek razvojem savremenih informaciono-komunikacionih tehnologija i njihovom globalnom proliferacijom, narasta svest o neophodnosti zaštite privatnosti prenetih poruka odnosno identiteta učesnika u komunikaciji. Prva konkretna softverska rešenja ove namene, dostupna širim krugovima korisnika, ipak nisu naišla na veću prihvaćenost, dok ona standardizovana u osnovne komunikacione protokole nisu pružala adekvatnu zaštitu. Tek proširenjem mobilnih komunikacija, sredinom devedesetih godina XX veka počinje da se posvećuje dužna pažnja ovoj problematici, kako u praktičnom smislu tako i u akademskoj zajednici. Korisnici su svoje razgovore i poruke sa pravom doživljavali kao mnogo osetljivije informacije od e-mail-a i Internet pretraga, što su pružaoci usluga blagovremeno uvideli, te su GSM i slični specifični standardi od početka bili adekvatna zaštita privatnosti od uobičajenih napada.

Dalju evoluciju tehnologije i potreba korisnika, ali i društvenog konteksta u kome se komunikacija odvijala (nastanak društvenih mreža) nije ispratio razvoj i implementacija adekvatnih mera zaštite, dok su sa druge strane napadi postali učestaliji a napadači sofisticiraniji i bolje organizovani. Tek pre nekoliko godina, kada su veliku i zasluženu medijsku pažnju privukla otkrića Edvarda Snoudena i Džulijana Asanža o globalnom prisluškivanju građana, narasla je svest o mogućnostima ne samo državno sponzorisanih, već i drugih naročito motivisanih grupa, odnosno mogućoj šteti koju njihova manipulacija privatnim podacima može izazvati. Bezbednost komunikacija i privatnost podataka postali su tema brojnih medijskih napisa ali i akademskih studija, što je rezultovalo konkretnim aktivnostima pružaoca komunikacionih usluga i kreatora aplikacija i servisa na implementaciji adekvatnih mera zaštite.

Polazeći od navedenog, pokušali smo da u ovom radu damo skromni doprinos sveobuhvatnom shvatanju problematike upotrebe kriptografskih tehnika zaštite u mobilnim aplikacijama, sa akcentom na najrasprostranjeniju platformu (Android). Naveli smo osnovne teorijske koncepte kriptologije, opisali konkretne kriptografske tehnike zaštite kao i segmente procesa razvoja mobilnih aplikacija u kojima se najčešće primenjuju. Obrađena je i lista (OWASP) najzastupljenijih pretnji po bezbednost podataka u mobilnim aplikacijama, koja se redovno ažurira i predstavlja de facto polazišnu osnovu pri konstrukciji i izgradnji ojačanih mobilnih aplikacija i drugih bezbednosnih komponenti i sistema. Takođe smo napravili kratak pregled pretnji bezbednosti mobilnih aplikacija i adekvatnih odgovora

na njih primenom kriptografskih tehnika zaštite, ukazujući na ograničenja koja u realnoj upotrebi nameću dometi tehnologije i očekivanja korisnika.

Predložena rešenja smo praktično proverili na primeru Android aplikacije koja integriše pojedine kriptografske tehnike zaštite, koju smo, u kontrolisanom testnom okruženju i primenom za to pogodnih alata, podvrgli uslovima aktivnog napada na bezbednost podataka (Man In The Middle koncept, prilagođen HTTPS protokolu). Ponašanje ojačane aplikacije smo uporedili sa onom koja ima bazičnu funkcionalnost i ne primenjuje nikakve dodatne mere bezbednosti, te smo rezultatom ove komparacije potvrdili polaznu pretpostavku o neophodnosti implementacije ovih mera, kako bi se sačuvala privatnost komunikacije prilikom relativno sofisticiranog napada.

Problem zanemarivanja bezbednosti podataka pri razvoju mobilnih aplikacija se može u najvećoj meri eliminisati. Ipak, biće potrebno dosta vremena, resursa i pre svega napora i dobre volje velikog broja ljudi. Podizanje nivoa svesti o potrebi implementacije najvišeg dostupnog stepena bezbednosti u najvećem broju mobilnih aplikacija je dugoročan proces, ali srećom čini se da programerska zajednica kao i sve veći broj korisnika prihvata savete eksperata iz ovih oblasti. Nakon usvajanja delimično birokratski sprovedenog i medijski neadekvatno plasiranog GDPR standarda, čak i oni subjekti koji su izbegavali temeljno bavljenje problemima privatnosti korisničkih informacija su na to sada praktično zakonski primorani.

Dodatne troškove koje stvara implementacija adekvatnih mera zaštite, a pre svega kriptografskih tehnika u savremene mobilne aplikacije treba posmatrati kao investiciju koja će se vratiti poverenjem i odanošću korisnika, a sa druge strane kao osiguranje od odgovornosti kompanije u slučaju eventualnih uspešnih, krajnje sofisticiranih, ili potpuno novih napada.

Značaj pitanja vezanih za računarsku bezbednost, a pogotovo privatnost podataka uviđa sve veći broj ljudi, pa ove donedavno opskurne i nepoznate teme zaokupljaju sve veću medijsku pažnju ali i interesovanje akademske zajednice. Uz sve rizike koje "demokratizacija" ovih usko specijalizovanih oblasti nosi, mišljenja smo da je aktuelni trend generalno pozitivan, odnosno da će pokretanje i održavanje javne debate motivisati, ili bar primorati, brojne aktere na ovom polju da prihvate i primene argumentovane sugestije eksperata. Nadamo se da smo ovim radom uspeli da doprinesemo ovom procesu.

7. Literatura

1. Barnum S., Michael Gegick, M., Michael, C.C., Defense in Depth, National Security Agency, 2005.
2. Birge-Lee, H., Sun, Y., Edmundson, A., Rexford, J., & Mittal, P., Using BGP to Acquire Bogus TLS Certificates, Princeton University, 2017.
3. Astolfi, F., Kroese, J., van Oorschot, J., I2P - The Invisible Internet Project, Leiden University, 2014.
4. Barrett, D., J., Silverman, R., E., Byrnes, R., G., SSH, The Secure Shell: The Definitive Guide, O'Reilly Media, 2005.
5. Burd, B., Android Application Development - All in one, Wiley, 2012.
6. Cohn-Gordon, K., A Formal Security Analysis of the Signal Messaging Protocol, University of Oxford, 2017.
7. Cohn-Gordon, K., Cremers, C.J., Garratt, L., Millican, J., & Milner, K., On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees, IACR Cryptology ePrint Archive, 2017.
8. Crist, E., F., Keijser, J., J., Mastering OpenVPN, Packt Publishing, 2015.
9. Davies, J., Implementing SSL TLS Using Cryptography and PKI, Wiley Publishing Inc., 2011.
10. De Smet, D., Pritchett, W. L., Kali Linux Cookbook, Packt Publishing, 2013.
11. Dingledine, R., Tor: The Second-Generation Onion Router, The Free Haven Project, 2018.
12. Dwivedi, H., Clark, C., Thiel, D., Mobile Application Security, The McGraw-Hill Companies, 2010.
13. Egele, M., Brumley, D., Fratantonio, Y., Kruegel, C., An Empirical Study of Cryptographic Misuse in Android Applications, DARPA, 2017.
14. Egger, C., Schlumberger, J., Kruegel. C., Vigna, G., Practical Attacks Against The I2P Network, Friedrich-Alexander University Erlangen-Nuremberg, 2013.
15. Elenkov, N., Android Security Internals, No Starch Press, 2014.
16. Gómez, A. F., Mart, G., Cánovas, Ó., New security services based on PKI, Future Generation Computer Systems, 19, 251–262., 2003.
17. Gueron, S., AES-GCM for Efficient Authenticated Encryption, Stanford University, 2013.
18. Gunasekera, S., Android Apps Security, Apress, 2012.
19. Gupta, A., Learning Pentesting for Android Devices, Packt Publishing, 2014.
20. Hook, D., Beginning Cryptography with Java, Wrox, 2005.
21. Kadhim, J., Developing a Multi Platforms Web Applications for Mobile Device Using HTML5, Journal of Information Technology & Software Engineering, br. 8., 2018.
22. Keijser, J., J., OpenVPN Cookbook, 2nd Ed., Packt Publishing, 2017.
23. Khan,W. Z., Xiang, Y., Arshad, Q., Mobile Phone Sensing Systems: A Survey, IEEE Communications Surveys & Tutorials, vol. 15, 2013.
24. Kim, D. & K., Bum J. & D., Certified Malware: Measuring Breaches of Trust in the Windows Code-Signing PKI., Tudor, 2017.
25. Koch, W., State of End To End Encryption, FSCONS 15, Gothenburg, 2015.
26. Kotipalli, S., R., Imran, M., A., Hacking Android, Packt Publishing, 2016.
27. Kozák, K., Kwon, B.J., Kim, D., Gates, C., & Dumitras, T., Issued for Abuse: Measuring the Underground Trade in Code Signing Certificate, CoRR, 2018.

28. Kurose, J., F., Ross, K., W., Computer Networking: A Top-Down Approach, 6th Ed., Pearson, 2012.
29. Lazar, D., Chen, H., Wang, X., Zeldovich, N., Why does cryptographic software fail? A case study and open problems, Proceedings of 5th Asia-Pacific Workshop on Systems, APSYS, 2014.
30. Lazarević, B., Marjanović, Z., Aničić, N., Babarogić, S., Baze Podataka, FON Beograd, 2016.
31. Lucas, M., W., SSH Mastery - OpenSSH, PuTTY, Tunnels and Keys, Tilted Windmill Press, 2012.
32. Lukoff, K., Hiniker, A., What Makes Smartphone Use Meaningful or Meaningless?, Interactive Mobile and Wearable Ubiquitous Technologies, 2018.
33. Marlinspike, M., Perrin, T. ed., The Double Ratchet Algorithm, Signal inc., 2016.
34. Marlinspike, M., Perrin, T. ed., The X3DH Key Agreement Protocol, Signal inc., 2016.
35. McMahon Stone, C., Chothia, T., Garcia, F., Spinner: Semi-Automatic Detection of Pinning without Hostname Verification. 176-188., 2017.
36. Menezes, A., van Oorschot, P., Vanstone, S., Handbook of Applied Cryptography, CRC Press, 1997.
37. Mishra, C., Mastering Wireshark, Packt Publishing, 2016.
38. Mitchell, C., PKI standards, Information Security Technical Report, br. 5, 2000.
39. Mohd, B. J., Hayajneh, T., & Vasilakos, A. V., A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues, Journal of Network and Computer Applications, 58, 73–93., 2015.
40. Neagu, A., Oracle 11g Anti-hacker's Cookbook, Packt Publishing, 2012.
41. Oppliger, O., Oppliger, Contemporary Cryptography, Artech House, 2005.
42. OWASP Top 10 - 2017: The Ten Most Critical Web Application Security Risks, The OWASP Foundation, 2017.
43. Raphael, J., R., Android versions: A living history from 1.0 to today, Computerworld, 2018.
44. Rayarikar, R., Upadhyay, S., Pimpale, P., SMS Encryption using AES Algorithm on Android, International Journal of Computer Applications, 50(19), 12–17., 2012.
45. Ristic, I., Bulletproof SSL and TLS, Feisty Duck, 2014.
46. Rogers, R., Lombardo, J., Medniek, Z., Meike, B., Android Application Development, O'Reilly Media, 2009.
47. Schneier, B., Applied Cryptography, 2nd, Ed., Wiley, 1996.
48. Simić, D., Tehnike zaštite u računarskim mrežama, slajdovi sa predavanja, 2017.
49. Six, J., Application Security for the Android Platform, O'Reilly Media, 2011.
50. Soni, R., Nginx: From Beginner to Pro, Apress, 2016.
51. Stallings, W., Cryptography and Network Security: Principles and Practice, 6th Ed., Pearson, 2013.
52. Stallings, W., Network Security Essentials: Applications and Standards, 4th Ed., Pearson, 2010.
53. Stamp, M., Information security: principles and practice, 2nd ed., Wiley, 2011.
54. Technical Report, UX Trends 2018, Zorraquino, 2018.
55. Velu, V., K., Mobile Application Penetration Testing, Packt Publishing, 2016.
56. Verma, P, Dixit, A., Mobile Device Exploitation Cookbook, Packt Publishing, 2016.
57. Wei, X., Wolf, M., A Survey on HTTPS Implementation by Android Apps: Issues and Countermeasures, Applied Computing and Informatics, 2016.

58. Wendlandt, D., Andersen, D. G., Perrig, A., Perspectives: Improving SSH-style Host Authentication with MultiPath Probing, Carnegie Mellon University, 2008.
59. White Paper, Bouncy Castle FIPS Java API, The Legion of the Bouncy Castle, 2015.
60. White Paper, Metadata and Privacy: A Technical and Legal Overview, Office of the Privacy Commissioner of Canada, 2014.
61. Won-jun, L., Seungjae, S., Effects of Product Smartness on Satisfaction, Journal of Theoretical and Applied Electronic Commerce Research, br. 13, 2018.
62. Yadav, R. K., Cryptography on Android Message Applications – A Review, International Journal on Computer Science and Engineering, 5(5), 362–367., 2013.
63. Yu Heejung, L., Hongbeom, H., What is 5G? Emerging 5G Mobile Services and Network Requirements, Sustainability, br. 9., 2017.
64. https://blog.filippo.io/the-ecb-penguin/ pristupljeno : 01.05.2018.
65. https://developers.google.com/android/ pristupljeno : 01.07.2018.
66. https://oit.colorado.edu/it-security/security-awareness/encryption/types-encryption, pristupljeno : 01.07.2018.
67. https://sites.google.com/site/lbathen/research/bouncy_castle pristupljeno : 01.07.2018.
68. https://thecybersecurityman.com/2017/11/08/the-osi-model-quick-example/ pristupljeno : 18.08.2018.
69. https://www.nibusinessinfo.co.uk, pristupljeno : 31.07.2018.
70. https://www.zdnet.com/article/hacker-hijacks-isps-steals-83000-from-bitcoin-mining-pools/ pristupljeno : 01.07.2018.