Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Dean Koštomaj

Nespletna uporaba varnostnega elementa pametne kartice na mobilnih napravah

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: prof. dr. Viljan Mahnič

Ljubljana, 2018

Zahvala

Zahvalil bi se vsem, ki so kadarkoli pripomogli pri izdelavi te magistrske naloge. Posebna zahvala gre mentorju prof. dr. Viljanu Mahniču, za pomoč in usmerjanje pri izdelavi ter punci, staršem in prijateljem.

Dean Koštomaj, 2018

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
AES	advanced encryption standard	napredni kriptografski standard
APDU	application protocol data unit	protokol za prenos podatkov med čitalcem in kartico
API	application programming interface	programski vmesnik
APK	Android package	Android paket
CMAC	cipher-based message authentication code	avtentikacijska koda zgrajena na podlagi zakodiranega sporočila
CPE	central processing unit	centralna procesna enota
EEPROM	electrically erasable programmable read-only memory	električno izbrisljivi programirljivi bralni pomnilnik
HCE	host-based card emulation	tehnologija na mobilnih napravah za oponašanje kartic
IEC	International Electrotechnical Commission	mednarodna elektrotehnična komisija
IMEI	international mobile equipment identity	identifikator mobilne naprave

ISO	International Organization for Standardization	mednarodna organizacija za standardizacijo
MAC	message authentication code	avtentikacijska koda sporočila
MSB	most significant bit	bit z največjo težo
NFC	near field communication	komunikacija kratkega dosega
OSPT	Open Standard for Public Transit	organizacija za odprt standard v javnem prometu
PIN	personal identification number	osebna identifikacijska številka
RAM	random access memory	bralno-pisalni pomnilnik
REST	representational state transfer	protokol za prenos podatkov preko interneta
RFID	radio-frequency identification	radiofrekvenčno prepoznavanje
ROM	read-only memory	bralni pomnilnik
SDK	software development kit	paket za razvoj programske opreme
SHA	secure hash algorithms	zgoščevalna funkcija
SIM	subscriber identification module	modul za identifikacijo naročnika
SMS	short message service	sistem pošiljanja kratkih sporočil
TEE	trusted execution environment	okolje varnega izvajanja
TSP	token service provider	ponudnik storitev za grajenje žetonov
UUID	universally unique identifier	splošni enolični identifikator

Povzetek

Naslov: Nespletna uporaba varnostnega elementa pametne kartice na mobilnih napravah

Leta 2013 je Google s predstavitvijo tehnologije za posnemanje pametnih kartic na mobilnih napravah odprl vrata novim možnostim za uporabo tehnologije komunikacija kratkega dosega ali NFC. Ena izmed rešitev, ki je postala mogoča, je virtualni varnostni element. To je sistem, ki poskuša zagotoviti varnost občutljivih podatkov, čeprav se ti nahajajo v pomnilniku mobilnih naprav, ki je lahko dostopen napadalcu. V tem delu je predstavljena rešitev, ki poskuša zagotavljati varnost z uporabo različnih tehnik in sistemov, kot so naprimer *Android SafetyNet Attestation API*, časovno omejeni in poosebljeni kriptografsi ključi, sistem *Android Keystore*, itd. Naša rešitev je namenjena predvsem ponudnikom javnega prevoza, kjer je hranjenje podatkov v pomnilniku naprave bistvenega pomena, saj se transakcije med karticami in čitalci kartic dogajajo v nespletnem načinu. Primerjava s sorodnimi rešitvami je pokazala, da je naša rešitev po hitrosti komunikacije primerljiva s fizičnimi pametnimi karticami ter bistveno hitrejša od virtualnih pametnih kartic v oblaku.

Ključne besede

varnostni element, mobilne naprave, pametna kartica

Abstract

Title: Offline use of smart card secure element on mobile devices

In year 2013 Google has introduced technology for enabling mobile devices to emulate smart cards and doing so opened a way to a lot of new possibilities. One of them is implementation of virtual secure element which is a system that tries to protect sensitive data even though it is located in memory of a mobile device. Solution in this document tries to secure sensitive data with use of *Android SafetyNet Attestation API*, time limited and personalized cryptographic keys, *Android Keystore*, etc. This solution is intended for public transporters who need a solution to securely store data in mobile device memory since transactions on their terminals are done offline. In comparison with similar solutions we found out that our solution works with approximately same speed as physical smart cards and a lot faster than virtual smart card implemented in cloud.

Keywords

secure element, mobile device, smard card

Poglavje 1

Uvod

1.1 Motivacija

V današnjem času se je že skoraj vsak srečal z uporabo pametne kartice. Uporabljamo jih za najrazličnejše namene. Pogosto se uporabljajo za dostop do prostorov ali območij, kjer je za to potrebno imeti dovoljenje. V zadnjih letih se vse bolj uporabljajo pri plačevanju, podjetja jih izdajajo kot kartice zvestobe, vgrajene pa so tudi v naše potne liste. Ena izmed najbolj pogostih uporab pametnih kartic pa se pojavi tudi v javnem prometu, pa čeprav ne gre za novo tehnologijo [1]. Uporabljajo se za plačevanje oziroma dokazovanje plačila vožnje. V javnem prometu so tako zelo razširjene večinoma zaradi lažjega plačila vožnje, hitrejšega pregledovanja vozovnic in hitrejšega vkrcanja [2]. Poleg tega njihova zasnova omogoča branje, pisanje in shranjevanje podatkov na zelo varen način [2]. Zasluge za varno obdelavo podatkov lahko pripišemo varnostnemu elementu (angl. *secure element*), ki zagotavlja varnost z uporabo simetričnih in asimetričnih kriptografskih ključev [6]. Vse naštete lastnosti pametnih kartic so razlogi za njihovo razširjeno uporabo, kar je privedlo do tega, da so se začele kopičiti v naših denarnicah.

S predstavitvijo tehnologije HCE (angl. *host-based card emulation*) na mobilnih napravah nam je Google odprl nove možnosti za izdelavo rešitev, ki temeljijo na komunikaciji kratkega dosega (angl. Near Field Communi-

cation - NFC) [6]. Ena izmed njih je tudi virtualni varnostni element, ki bi zmanjšala število plastičnih kartic in jih nadomestila z virtualnimi pametnimi karticami. Ker veliko javnih prevoznikov uporablja pametne kartice kot način plačila vožnje, je naša motivacija znebiti se nepotrebnih fizičnih kartic in jih nadomestiti z virtualnimi. S tem bo uporabniška izkušnja ob uporabi javnega prometa dosti bolj prijetna. Prijetnejše pa bo tudi plačevanje vožnje, za katero bo možno plačati kar preko mobilne naprave.

1.2 Predlagana rešitev

Na trgu že obstajajo nekatere rešitve, ki omogočajo komunikacijo mobilne naprave preko tehnologije NFC z drugimi napravami in s tem nadomeščajo nekatere pametne kartice. Rešitev, predlagana v tem delu, je izdelava varnostnega elementa na mobilnih napravah, ki bo zmožen komuniciranja s čitalcem pametnih kartic na enak način, kot to počnejo fizične pametne kartice. To pomeni, da bomo s tem omogočili sočasno uporabo tako mobilnih naprav kot tudi pametnih kartic na istih čitalcih. Rešitev bo zasnovana na generični način in s tem omogočila poljubno strukturo virtualne pametne kartice. V okviru tega dela bo implementirana rešitev podprla protokol Cipurse, ki je eden izmed najbolj razširjenih odprtokodnih standardov za komunikacijo med čitalcem pametnih kartic in pametnimi karticami v svetu javnega prevoza [7]. Prav tako bo omogočala enostavno kasnejše dodajanje na novo implementiranih protokolov za komuniciranje s čitalcem kartic.

Ker pametne kartice predstavljajo simbol dobro zavarovanih podatkov in komunikacije, bo tudi naša rešitev stremela proti varnosti. Uporabili bomo nekaj orodij in tehnik, s katerimi bomo zagotovili varnost podatkov. Pri naši rešitvi bo varnost igrala še toliko večjo vlogo, saj bo delovala v nespletnem (angl. *offline*) načinu. To pomeni, da bodo občutljivi podatki shranjeni na napravi in kot taki tudi lahko dostopni napadalcu. Rešitev bo izdelana tako, da bo pridobivanje občutljivih podatkov za napadalca zelo oteženo, hkrati pa bodo pridobljeni podatki veljavni samo za kratek čas in bodo s tem za

napadalca skoraj popolnoma nekoristni. Sistem bo zasnovan tudi tako, da bo ob morebitni zlorabi zaznal storjeno dejanje in razkril uporabnika, ki je odgovoren za nastalo škodo.

1.3 Struktura naloge

V naslednjem poglavju se bomo seznanili s pametnimi karticami in tehnologijo, ki omogoča varno realizacijo le-teh na mobilnih napravah. Tretje poglavje se osredotoča na varnostni element, njegove vrste in različne načine dostopa do podatkov, shranjenih na njem. Poleg tega je v tem poglavju predstavljenih tudi nekaj že obstoječih rešitev, ki so podobne rešitvi, opisani v tej nalogi. Sledi poglavje, ki predstavi našo rešitev in vse komponente, ki so potrebne za varno delovanje le-te. V petem poglavju se naloga osredotoči na analizo in vrednotenje realizirane rešitve, šesto poglavje pa povzema sklepne ugotovitve.

Poglavje 2

Pregled področja

2.1 Pametne kartice

Pametne kartice so kartice, ki so po navadi v velikosti bančnih kartic z vgrajenim čipom. Ti čipi so navadno pomnilniški ali mikroprocesorski čipi z notranjim pomnilnikom [4]. Spremljajo nas že ker nekaj časa. Prvi patent na tem področju je bil objavljen leta 1968 s strani dveh nemških izumiteljev, Dethloffa in Grotruppa, ki sta razvila koncept plastične kartice z vsebovanim mikročipom [1]. Zaradi pospešene rasti informacijske tehnologije so se pametne kartice dobro uveljavile kot identifikacijsko orodje, ki učinkovito odgovarja na vprašanji 'Kdo si?' in 'Lahko dokažeš, da si to res ti?' [3]. Za odgovor na prvo vprašanje poskrbi vsebina, shranjena na kartici, medtem ko je za drugo potrebna interakcija uporabnika, ki mora ponavadi podati osebno identifikacijsko številko oziroma PIN (angl. Personal Indentification Number).

Glede na način komuniciranja lahko pametne kartice v grobem delimo na stične (angl. *contact*) in brezstične (angl. *contactless*) [4]. Stične kartice za komunikacijo potrebujejo fizični stik in jih je zato potrebno vstaviti v čitalec kartic, medtem ko brezstične kartice komunicirajo preko vgrajene antene in ne potrebujejo fizičnega stika z čitalcem [5]. Ne glede na to, ali so kartice stične ali brezstične, so sposobne shranjevanja podatkov, manipulacije ozi-

roma obdelave podatkov, nadzorovanja dostopa (angl. *controlling access*), shranjevanja biometrik itd [5].

Ena izmed najbolj pogostih oblik pametnih kartic se pojavlja v naših telefonih. Čeprav niso velikosti bančnih kartic, pa so kartice SIM (angl. *Subscriber Identification Module*) prav tako vrsta pametnih kartic. Prvi telefon, ki je uporabljal predplačniški način plačevanja klicev s pametno kartico, se je pojavil leta 1984 [8]. Ker te kartice tako olajšajo storitve, ki vključujejo informacije in finančne transakcije, jih je bilo leta 1992 izdanih že več kot 200 milijonov. Do leta 1995 je ta številka poskočila že na 600 milijonov, od tega je bilo 500 milijonov pomnilniških kartic in 100 milijonov mikroprocesorskih kartic [8].

2.1.1 Pomnilniške kartice

Prve pametne kartice, uporabljene v telekomunikacijske namene, so bile pomnilniške kartice. To so bile predplačniške kartice, ki so imele v spominu elektronsko shranjeno vrednost, ki se je znižala vsakič, ko je bila kartica uporabljena. Da bi preprečili uporabniku nedovoljeno popravljanje vsote na kartici, te kartice uporabljajo varnostno logiko, ki onemogoča brisanje podatkov, ko so ti enkrat zapisani. S tem dosežemo, da je na primer število porabljenih enot za telefonske klice nepovratno [9].

V tovrstnih karticah se po navadi uporablja pomnilnik EEPROM (angl. *Electrically Erasable Programmable Read-Only Memory*). Sposobne so tudi poganjanja preprostejših algoritmov, kot je na primer šifriranje podatkov. Ker je prilagodljivost teh kartic zelo omejena, so ponavadi optimizirane za točno določeno aplikacijo [8]. Poleg tega imajo še eno slabo lastnost, to pa je enkratna uporaba. Te kartice so uporabne samo, dokler uporabnik ne porabi vseh naloženih sredstev na kartici [9].

2.1.2 Mikroprocesorske kartice

Kot že ime pove, gre za kartice, katere imajo vgrajen mikroprocesor, ki je povezan s pomnilniškimi komponentami, kot so ROM (angl. *Read Only Memory*), RAM (angl. *Random Access Memory*) in EEPROM [8].

V pomnilniku ROM se nahaja operacijski sistem za mikroprocesor, njegova vsebina pa je določena že ob izdelavi kartice. Vse kartice iz iste serije imajo enako vsebino pomnilnika, podatke v njem pa je nemogoče prepisati [8].

RAM predstavlja delovni spomin za mikroprocesor, tako kot pri namiznih računalnikih, podatki v njem pa se izbrišejo vsakič, ko kartica izgubi napajanje, ki ga prejema preko čitalca [8].

EEPROM je spomin, namenjen aplikacijam. V njem se nahajajo podatki aplikacije in njena programska koda. Branje in pisanje vanj je kontrolirano s strani operacijskega sistema [8].

Tovrstne kartice so bile najprej uporabljene v Franciji v obliki bančnih kartic. Njihova ključna prednost je zmožnost varnega shranjevanja zasebnih ključev in izvajanja naprednejših kriptografskih algoritmov, kar omogoča visoko varnost pri nespletnem plačevanju [9].

2.1.3 Komunikacija z čitalcem kartic

Tako stične kot tudi brezstične pametne kartice uporabljajo enak protokol za komunikacijo z čitalcem kartic, ki je definiran s standardom ISO/IEC 7816-4. Komunikacija teče po načinu ukaz (angl. *command*) in odgovor (angl. *response*). Ta par imenujemo APDU (angl. *application protocol data unit*). Komuniciranje vedno poteka tako, da ukaze pošilja čitalec kartic, medtem ko odgovarja vedno kartica [10].

2.2 Varnostni element

Varnostni element (angl. *secure element*) je kombinacija strojne in programske opreme, vmesnikov in protokolov, ki omogočajo varno shranjevanje po-

datkov [12]. V varnostnem elementu so nameščene aplikacije, ki jih je pri nekaterih možno prilagajati, nameščati in upravljati na daljavo (angl. *over-the-air*) [11]. Naloga varnostnega elementa je zagotavljanje varnega območja, ki ščiti izvajanje aplikacije in elemente, s katerimi razpolaga aplikacija (npr. podatki plačila, kriptografski ključi itd.) [13]. Čeprav pogosto slišimo, da se uporablja v svetu bančništva, pa se poleg uporabe za bančne aplikacije lahko varnostni element uporablja tudi za potrebe preverjanja identitete [12], dokazovanje plačila vožnje v javnem prometu, omogočanje dostopa do raznih stavb itd. Njegova uporaba je primerna povsod, kjer potrebujemo varno okolje za obdelavo in shranjevanje občutljivih podatkov.

2.3 Tehnologija NFC

NFC (angl. *Near Field Communication*) je način brezstičnega komuniciranja med dvema napravama, ki odpira vrata najrazličnejšim storitvam, kot so plačila, prodaja in preverjanje veljavnosti vozovnic, navigaciji itd [14]. Izhaja iz standarda RFID (angl. *Radio-Frequency Identification*), vendar je omejena na maksimalno razdaljo 10 cm med napravama [15]. NFC deluje na visoko frekvenčnem pasu pri 13.56 Mhz po standardih ISO 14443, ISO 18092 in FeliCa. Hitrosti, ki jih lahko doseže med komuniciranjem, so 106, 212, 424 in 848 kbps (*kilobitov na sekundo*) [15]. Za svoje delovanje izrablja koncept elektromagnetnega polja [17]. Ker je NFC zmožen tako branja kot tudi pisanja na naprave, bo v prihodnosti najverjetneje po uporabi prehitel standardne pametne kartice [15].

Tehnologija NFC ne podpira samo komunikacije med aktivnim čitalcem in pasivnimi značkami tako kot RFID, pač pa podpira tudi komunikacijo med dvema aktivnima napravama [16]. To je razlog, da se naprave NFC delijo na aktivne in pasivne. V primeru komuniciranja dveh aktivnih naprav gre za aktivno komuniciranje, ko komunicirata pasivna in aktivna naprava, pa gre za pasivno komunikacijo [18]. Dve pasivni napravi nista zmožni medsebojne komunikacije.

Kot že rečeno, pasivna komunikacija poteka med aktivno in pasivno napravo. Razlika med tema dvema napravama je, da pasivna naprava nima lastnega vira napajanja in ji ga zato priskrbi aktivna naprava preko elektromagnetnega polja [17].

Aktivna komunikacija poteka med dvema napravama, ki imata vsaka svoj vir napajanja. Ti napravi ustvarjata vsaka svoje elektromagnetno polje in ga vklapljata izmenično takrat, ko pošiljata podatke [18].

2.4 Mobilne naprave

V letu 2017 je imelo že dve tretjini celotne populacije mobilne naprave [19]. Veliko ljudi ima tudi po več kot eno mobilno napravo. Mejo petih milijard uporabnikov mobilnih naprav na svetu smo presegli v drugi četrtini lanskega leta [19]. Že leta 2014 pa je bilo na svetu več mobilnih naprav kot ljudi [20]. Iz teh podatkov je razvidno, da mobilne naprave postajajo vse bolj razširjene. Njihove zmožnosti se izboljšujejo zelo hitro, zlasti na področjih procesorjev, pomnilnikov in komunikacije [21]. Te lastnosti jim omogočajo opravljati najrazličnejša opravila, kar pripelje do dejstva, da postajajo vse bolj pomembne za učinkovito komunikacijo, ki ni vezana na kraj ali čas [22].

Ko govorimo o tem, da postajajo pametne naprave ključnega pomena za komunikacijo, ne mislimo samo na komunikacijo med ljudmi, temveč tudi med napravami. Kot smo že v prejšnjem poglavju omenili, so naprave zmožne komunicirati po protokolu NFC in mobilne naprave niso tukaj nobena izjema. Čeprav so bili nekateri na začetku skeptični glede te tehnologije [23], se je dobro uveljavila tudi v svetu mobilnih naprav.

2.5 Tehnologija HCE

Tehnologija HCE (angl. *Host-based Card Emulation*) omogoča mobilnim napravam in aplikacijam, ki so na njih nameščene, da posnemajo delovanje brezstičnih pametnih kartic [24]. To pomeni, da so mobilne naprave združljive

z obstoječo brezstično infrastrukturo [25], kot so čitalci pametnih kartic na prevoznih sredstvih javnega prometa. To tehnologijo je predstavil Google v svojem operacijskem sistemu Android z verzijo 4.4 (KitKat) [6], ki je bila izdana v oktobru leta 2013. Že leto pred tem pa je imel vgrajeno tehnologijo HCE mobilni telefon Bold 9900 proizvajalca Blackberry, ki je s tem postal prvi telefon na svetu s to tehnologijo [24].

Pred pojavom tehnologije HCE so bili varnostni elementi fizično vgrajeni v mobilne naprave, kar pomeni, da so bile aplikacije nameščene nanje že med njihovo izdelavo. To je razlog, da manjša podjetja niso imela možnosti razvijanja aplikacij, ki bi uporabljale varnostni element, saj je bilo skoraj nemogoče prepričati velika podjetja, da namestijo njihovo rešitev v svoj varnostni element. Tehnologija HCE je omogočila, da lahko razvijalci aplikacij zaobidejo fizični varnostni element in uporabijo tako imenovani mehki varnostni element (angl. *soft secure element*) [25]. Ker si aplikacije, ki uporabljajo tehnologijo HCE, delijo CPE (*centralna procesna enota*) z drugimi aplikacijami [25], je pri razvoju le-teh, če operirajo z občutljivimi podatki, potrebno biti še posebej pazljiv.

2.6 Protokol Cipurse

Protokol Cipurse je bil razvit s strani združenja OSPT (*Open Standard for Public Transit*). Je odprt standard, ki je namenjen javnim prevoznikom kot varna rešitev za plačevanje in preverjanje plačila vozovnic [26]. Cipurse omogoča platformo za varno uporabo tako novih kot tudi starih aplikacij, ki se uporabljajo v javnem prometu, in ima zato velik potencial za uporabo z obstoječo infrastrukturo po celotnem svetu [26]. Cipurse je tudi generična rešitev, ki omogoča, da razvijalcem ni potrebno izdelovati majhnih programov (angl. *applet*) za vsakega prevoznika posebej, ampak lahko samo personalizirajo vsebino kartice Cipurse [27].

Protokol definira nekaj različnih tipov (profilov) organiziranja spomina na generičnih brezstičnih karticah in protokol, po katerem je možno komunici-

ranje s kartico. Ta protokol vključuje kriptografsko specifikacijo, ki definira, kako poteka avtentikacija in enkripcija med kartico in čitalcem kartic [28]. Omenjene funkcionalnosti pri protokolu Cipurse temeljijo na standardu AES (angl. *Advanced Encryption Standard*) s 128 bitnimi ključi, ki jih morata biti kartica in čitalec kartic sposobna varno shraniti [28].

2.7 Android Keystore

Operacijski sistem Android ima vgrajen sistem Keystore od svoje verzije 4.3 naprej. Ta sistem omogoča varno shranjevanje kriptografskih ključev na način, ki otežuje nepooblaščeno izluščevanje (angl. *extraction*) le-teh. Poleg tega omogoča tudi določanje pravil za uporabo shranjenih ključev [29].

Sistem ima v svoje delovanje vgrajeni dve varnostni politiki, s katerima ščiti kriptografske ključe. S prvim ukrepom Keystore prepreči, da bi kriptografski ključ kadarkoli med izvajanjem aplikacije vstopil v njen proces delovanja. S tem je napadalcu, ki je vrdl v izvajanje aplikacije, onemogočeno pridobivanje ključa in posledično tudi preprečena uporaba ključa zunaj naprave. Drugi ukrep pa lahko določen ključ veže na varovano strojno opremo, kot je TEE (angl. *Trusted Execution Environment*) ali varnostni element. Če vklopimo to funkcijo, ključ ne bo nikoli uporabljen zunaj določene strojne opreme [29].

Poglavje 3

Varnostni element in mobilne naprave

3.1 Fizični in virtualni varnostni element

Varnostni element je lahko fizično vgrajen v mobilno napravo, kar omogoča napravi opravljanje transakcij in komunikacijo s čitalci kartic [30], ki jih lahko najdemo v trgovinah, podzemnih železnicah, avtobusih, ne nazadnje tudi v službi za dostop do prostorov. Poleg fizično vgrajenega varnostnega elementa ima lahko mobilna naprava tudi sistem, ki omogoča rabo virtualnega varnostnega elementa, vendar mora biti v tem primeru vsaj del podatkov, navadno je to enkripcijski ključ za podatke, shranjenih v varnostnem elementu [31]. Tako kot z vgrajenim varnostnim elementom je tudi z virtualnim mogoče opravljati transakcije s pomočjo čitalca kartic.

Fizični varnostni elementi se na mobilnih napravah lahko pojavijo v različnih oblikah, v grobem pa jih lahko delimo na odstranljive in neodstranljive [12]. Eden izmed neodstranljivih fizičnih varnostnih elementov je procesor, ki upravlja procese, kateri za svoje izvajanje potrebujejo anteno (angl. *baseband processor*) [12]. Primer odstranljivega varnostnega elementa pa je varna pomnilniška kartica (angl. *seruce memory card*), ki jo je možno odstraniti iz naprave. Vgrajen ima mikroprocesor, ki omogoča dostop do ključev

šele po uspešni avtentikaciji [32].

O primerih virtualnih varnostnih elementov bomo govorili v poglavju 3.4, kjer si bomo ogledali nekaj že obstoječih rešitev na tem področju. Kar je dobro vedeti, je to, da se virtualni varnostni elementi ne nahajajo fizično na napravah, ampak jih s pomočjo različnih orodij poskušamo simulirati v programski kodi.

3.2 Dostopnost varnostnega elementa

Dostop do varnostnega elementa na mobilnih napravah NFC je mogoč na dva načina. Pri zunanjem (angl. *external*) dostopu mobilna naprava oponaša brezstično pametno kartico in se s tem predstavi čitalcu kartic. Tako lahko čitalec dostopa do podatkov, shranjenih v varnostnem elementu. V primeru notranjega (angl. *internal*) dostopa pa nekatere podatke iz varnostnega elementa bere mobilna aplikacija, ki teče na napravi [10].

Ko govorimo o dostopu podatkov, ki so shranjeni v varnostnem elementu, pa seveda ne mislimo na vse podatke. Tako zunanji kot tudi notranji dostop imata svoja pravila, ki strogo določajo, katere podatke je možno brati in katere ne.

3.3 Prednosti varnostnega elementa na mobilni napravi

Prva in največja prednost varnostnega elementa na mobilnih napravah je ta, da ima veliko mobilnih naprav že vgrajeno tehnologijo NFC, ki se že uporablja v svetu pametnih kartic za komuniciranje s čitalci kartic [33].

S prisotnostjo varnostnega elementa na mobilni napravi so aplikacijam dostopni tudi nekateri podatki, kot so na primer podatki o opravljenih transakcijah za prikazovanje računov na napravi [30], kar ima lahko tudi marketinško prednost pred pametnimi karticami, saj ni potrebno izdajanje računa v fizični obliki. Poleg tega so mobilne naprave že same po sebi sposobne

komuniciranja preko mobilnega omrežja z zalednimi sistemi, kar odpira nove možnosti za inovativnost. Tako se lahko na primer uporabniku na napravi prikazuje njegovo vsakodnevno potovanje v službo z javnim prevoznikom in morebitne predlagane optimizacije, s katerimi bi prihranil nekaj časa.

3.4 Pregled obstoječih rešitev

3.4.1 Varnosti element v oblaku

Ena izmed rešitev, pri katerih je varnostni element v celoti v oblaku je bila predstavljena v delu Tima Smoleta z naslovom *Implementacija brezstičnih pametnih kartic z varnostnim elementom v oblaku* [34]. Podobna rešitev je bila omenjena tudi v [35].

Tu gre za komunikacijo čitalca kartic z varnostnim elementom preko mobilne naprave, ki ima vgrajeno anteno NFC in podpira tehnologijo HCE. Kot je opisano v diplomskem delu, mobilna naprava oponaša pametno kartico in hkrati deluje kot posrednik, ki posreduje na oblak ukaze APDU, ki prihajajo iz čitalca. Na oblaku se nato v varnostnem elementu izvede ukaz in sestavi odgovor, ki je nato posredovan preko mobilne naprave nazaj do čitalca kartic. Slika 3.1 prikazuje delovanje take rešitve.

Ta rešitev hrani vse občutljive podatke v oblaku, kjer je varnost veliko lažje doseči, vendar zato potrebuje hitro povezavo 4G ali celo 5G med mobilno napravo in oblakom [35]. Tako hitro povezavo je težko zagotoviti v vsakem trenutku. Na primer na podzemni železnici dostikrat ni mobilnega omrežja ali pa je zelo oslabljeno. Pri oslabljeni povezavi bi v tem primeru čitalec in mobilna naprava konstantno izgubljala povezavo, saj bi med komunikacijo nenehno prihajalo do prekoračitve časovne omejitve (angl. *timeout*).

3.4.2 Tokenizacija

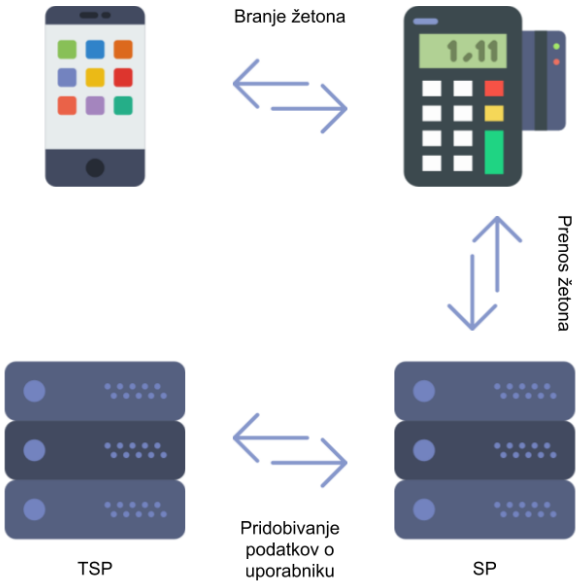
Tovrstne rešitve so se začele pojavljati po predstavitvi tehnologije HCE skupaj z operacijskim sistemom Android 4.4 [36]. Kot je opisano v [36], gre

Slika 3.1: Slika prikazuje potek komunikacije med čitalcem kartic in varnostnim elementom v oblaku preko mobilne naprave.

za rešitev, kjer se občutljivi podatki nahajajo v oblaku in ne na mobilni napravi. Rešitev za svoje delovanje uporablja koncept tokenizacije, njena arhitektura pa je sestavljena iz treh entitet. Prva entiteta je uporabnik, ki mora imeti mobilno napravo z vgrajeno anteno NFC in podporo tehnologiji HCE. Naloga uporabnika je naložiti aplikacijo, ki ob prvem zagonu ustvari tako imenovan uporabniški žeton. To je naključno zgrajen niz znakov, ki unikatno predstavlja uporabnika in njegovo napravo. Za tem sledi proces registracije, pri katerem mora sodelovati druga entiteta, ki se imenuje TSP (angl. *Token Service Provider*). Cilj registracije je, da TSP pridobi podatke o uporabniku vključno z njegovim žetonom ter oboje shrani v podatkovno bazo. Po končanem procesu registracije lahko uporabnik prične z uporabo aplikacije.

Tretja entiteta v tem sistemu je SP (angl. *Service Provider*), ki ob predstavitvi uporabnikove naprave čitalcu kartic preko le-tega prejme uporabnikov žeton. S to informacijo lahko preko entitete TSP nato pridobi informacije o uporabniku, ki so bile zapisane v podatkovno bazo TSP ob registraciji. Na ta način lahko SP preveri, kateri uporabnik se je predstavil čitalcu s svojo mobilno napravo, in mu pošlje odgovor. Slika 3.2 prikazuje delovanje take rešitve.

Rešitev, opisana v [36], elegantno reši problem shranjevanja občutljivih podatkov na mobilni napravi, kjer je varnost le-teh težko doseči. Cena tega je, da rešitev za delovanje vedno potrebuje povezavo s strežnikom na strani



Slika 3.2: Slika prikazuje potek komunikacije med čitalcem kartic in varnostnim elementom po konceptu tokenizacije.

čitalca. To je včasih težko doseči, še posebej v svetu javnega prometa, saj čitalci nimajo možnosti konstantne povezave z zalednimi sistemi preko mobilnega omrežja.

Za podobno rešitev gre tudi v delu [37], ki je nekakšna kombinacija rešitev predstavljenih v [36] in [34]. Tu se mobilna naprava čitalcu predstavi z žetonom v primeru, da nima povezave s strežnikom. V nasprotnem primeru mobilna naprava deluje kot posrednik med čitalcem in strežnikom.

Poglavje 4

Opis rešitve

4.1 Problem

Do zdaj smo se seznanili s konceptom virtualnega varnostnega elementa in njegove rabe na mobilnih napravah, ki podpirajo tehnologijo HCE. Čeprav so omenjene rešitve dobre, pa je njihova pomanjkliovost konstantna potreba po povezavi s strežniškim delom sistema. Rešitev, predlagana v [34], potrebuje povezavo z oblakom za prenos vsakega ukaza, ki ga pošlje čitalec kartic na mobilno napravo. Pri nestabilni povezavi z oblakom je taka rešitev skoraj neuporabna. Druga rešitev, ki je omenjena v [36], je potrebo po internetni povezavi prestavila na stran čitalca, kar omogoča lažjo uporabo sistema tudi pri nekoliko nestabilni povezavi. Nobena izmed omenjenih rešitev pa ne deluje dovolj dobro tam, kjer se pojavi potreba po hitrem branju podatkov iz varnostnega elementa. Poleg tega se nam poraja še dodatno vprašanje *'Kaj pa območja in prostori, kjer povezava s strežnikom ni mogoča?'*. Potreba po hitrem branju in nedostopnost internetne povezave sta dva problema, ki ju srečamo v javnem prometu. V nadaljevanju bo predstavljena predlagana rešitev, ki naslavlja omenjena problema.

4.2 Ideja

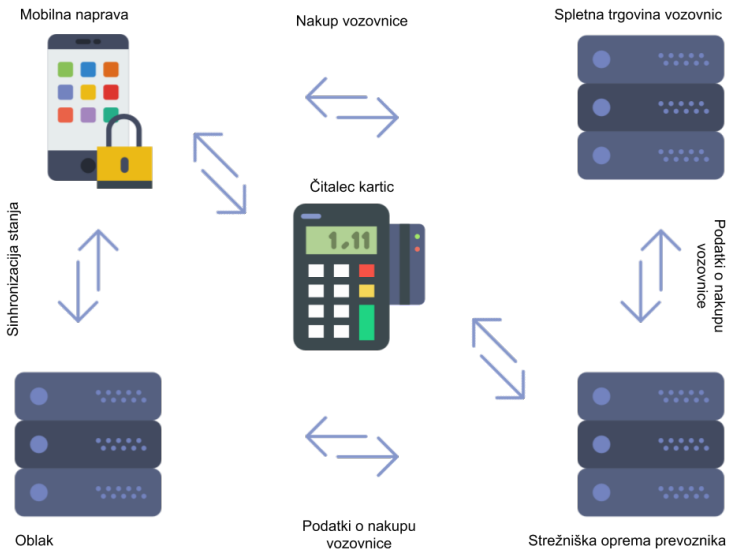
Kot že prej omenjeno, sta največja problema, ki ju srečamo v svetu javnega prevoza, hitrost branja iz varnostnega elementa in zahteva po komuniciranju z oblakom. Pametne kartice dobro rešujejo omenjena problema. Kaj pa mobilne naprave?

Ideja rešitve, ki je predlagana v tem delu, je implementacija virtualnega elementa na mobilni napravi, ki v pomnilniku hrani tudi občutljive podatke. Tako je komunikacija s čitalcem kartic hitrejša, saj se ukazi izvajajo kar na mobilni napravi in klici na odročne sisteme niso potrebni. S tako realizacijo se poveča možnost napadov, saj so občutljivi podatki lahko dostopni napadalcu. Ker je potrebno napade preprečiti, se mora tudi naša rešitev občasno povezati z oblakom za preverjanje in sinhronizacijo stanja virtualnih kartic na mobilni napravi. Tako oblak vodi evidenco o uporabljenih in še veljavnih vozovnicah. Na ta način lahko zazna zlorabe sistema in mobilni napravi ter uporabniku onemogoči njegovo nadaljnjo uporabo.

V tem delu bodo predstavljeni varnostni ukrepi, ki so bili uporabljeni za preprečevanje zlorab. Zavedati se moramo, da stoprocentna varnost v svetu računalništva ne obstaja. Tako kot pametne kartice tudi naša rešitev ne more biti popolnoma varna, vendar se bomo vseeno hoteli temu čim bolj približati.

4.3 Arhitektura sistema

Kot že omenjeno je v opisani rešitvi celoten varnostni element realiziran na mobilni napravi. Kljub temu, da se logika za izvajanje transakcij s čitalcem kartic nahaja na mobilni napravi, pa sistem še vseeno potrebuje strežnik, ki bo nadziral vse naprave in jih po potrebi tudi označil kot potencialno nevarne. Ker želimo, da je naša rešitev generična, v arhitekturi nastopata še dodatni dve komponenti. Prva predstavlja ponudnika prevoznih storitev, ki ima verjetno že postavljeno svojo strežniško infrastrukturo. Druga pa je spletna trgovina za vozovnice. Obe omenjeni entiteti sta izven področja tega dela, vendar je vseeno pomembno, da se zavedamo, da sta del arhitekture



Slika 4.1: Slika prikazuje arhitekturo sistema.

celotnega sistema.

Slika 4.1 prikazuje celotno arhitekturo sistema. Komponenta, na katero se bomo podrobno osredotočili v nadaljevanju, je mobilna naprava z varnostnim elementom.

Skupno je sistem sestavljen iz petih delov. Kot že rečeno, je oblak zadolžen za nadziranje mobilnih naprav in odkrivanje zlorab, vendar to ni njegova edina zadolžitev. Oblak je prav tako zadolžen za izdajo virtualnih pametnih kartic in vstavljanje virtualnih vozovnic v le-te. Mobilna naprava, oziroma natančneje varnostni element na mobilni napravi poskuša prejete virtualne kartice čim bolj zavarovati in jih nato uporabiti ob komunikaciji s čitalcem kartic. Slednji je zadolžen za preverjanje vozovnic na virtualnih karticah in shranjevanje zapisov oziroma dnevnika (angl. *log*), ki je lahko dodatno orodje za odkrivanje zlorab. Spletna trgovina v tej arhitekturi igra vlogo prodajalca vozovnic in je lahko tudi združena z infrastrukturo ponudnika prevoznih storitev, ta pa je zadolžena za izdajo vsebine vozovnic, kupljenih preko spletne trgovine.

4.4 Oblak

Čeprav oblak ni bistvo rešitve, ki jo opisujemo v tem delu, pa je vseeno prav, da mu namenimo nekaj besed.

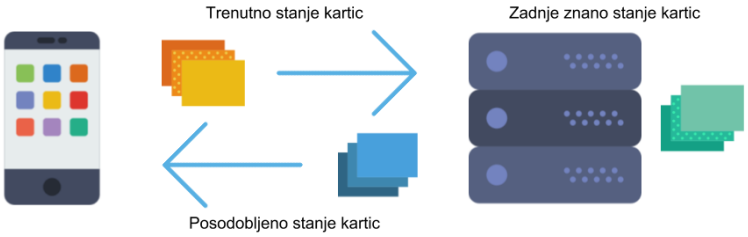
Zadolžen je za komunikacijo z virtualnim varnostnim elementom na mobilni napravi in skrb za njegovo pravilno delovanje. Njegova naloga je tudi blokirati naprave, ki se izkažejo za potencialno nevarne. Zgrajen je iz storitev REST (angl. *Representational State Transfer*), spletnih vtičnic (angl. *websocket*), avtentikatorja, združevalca virtualnih kartic in podatkovne baze.

4.4.1 Komuniciranje z drugimi deli arhitekture

Za komuniciranjo z ostalimi deli arhitekture, oblak uporablja tehnologijo REST in spletne vtičnice. Tehnologija REST se uporablja ob registraciji uporabnika v sistem in ob nakupu vozovnice preko spletne trgovine. Z izjemo registracije se ostalo komuniciranje z mobilno napravo opravi izključno preko spletnih vtičnic. Razlog za to se skriva v zagotavljanju varne in učinkovite storitve, saj vsi zahtevki, ki prihajajo na oblak z mobilne naprave, razen registracije, v odgovoru pričakujejo posodobljeno stanje uporabnikovih virtualnih kartic. Ker je s tehnologijo REST nemogoče biti prepričan, da je uporabnik poslane podatke v odgovoru res prejel, naša rešitev za te namene uporablja spletne vtičnice.

4.4.2 Posodabljanje stanja virtualnih pametnih kartic

Omenili smo, da je za komunikacijo med posodabljanjem stanja virtualnih kartic na mobilni napravi uporabljena tehnologija spletnih vtičnic. Razlog za to se skriva v načinu posodabljanja virtualnih pametnih kartic. Vsakič, ko mobilna naprava na oblak pošlje zahtevo za posodobitev kartic, mora priložiti svoje trenutno stanje le-teh. Na podlagi tega oblak primerja prejeto stanje z nazadnje znanim. S tem lahko preveri katere vozovnice so bile porabljene. Poleg tega oblak preveri tudi, ali je stanje kartice sumljivo. Stanje je sumljivo, če so na primer na kartici vozovnice, ki jih uporabnik ni kupil. V



Slika 4.2: Slika prikazuje posodobitev stanja virtualnih pametnih kartic.

tem primeru se uporabnikov račun blokira in označi kot sumljiv. Če oblak ne ugotovi nepravilnosti, posodobi stanje kartic, kar pomeni, da s pomočjo združevalca kartic odstrani porabljene vozovnice, doda na novo kupljene in odgovori mobilni napravi z novim stanjem. Na ta način združi stanje, ki ga pošlje mobilna naprava, in svoje stanje ter ga shrani v podatkovno bazo. S tem je cikel sinhronizacije, ki je tudi prikazan na sliki 4.2, zaključen. Ravno zato se je tu pojavila potreba po uporabi spletnih vtičnic, saj mora biti oblak prepričan, da je mobilna naprava prejela zadnje stanje kartic in so njegovi podatki v podatkovni bazi pravilni. V nasprotnem primeru bi lahko bil uporabnikov račun blokiran zaradi izgubljene internetne povezave, saj se njeno stanje ne bi ujemalo s stanjem na oblaku.

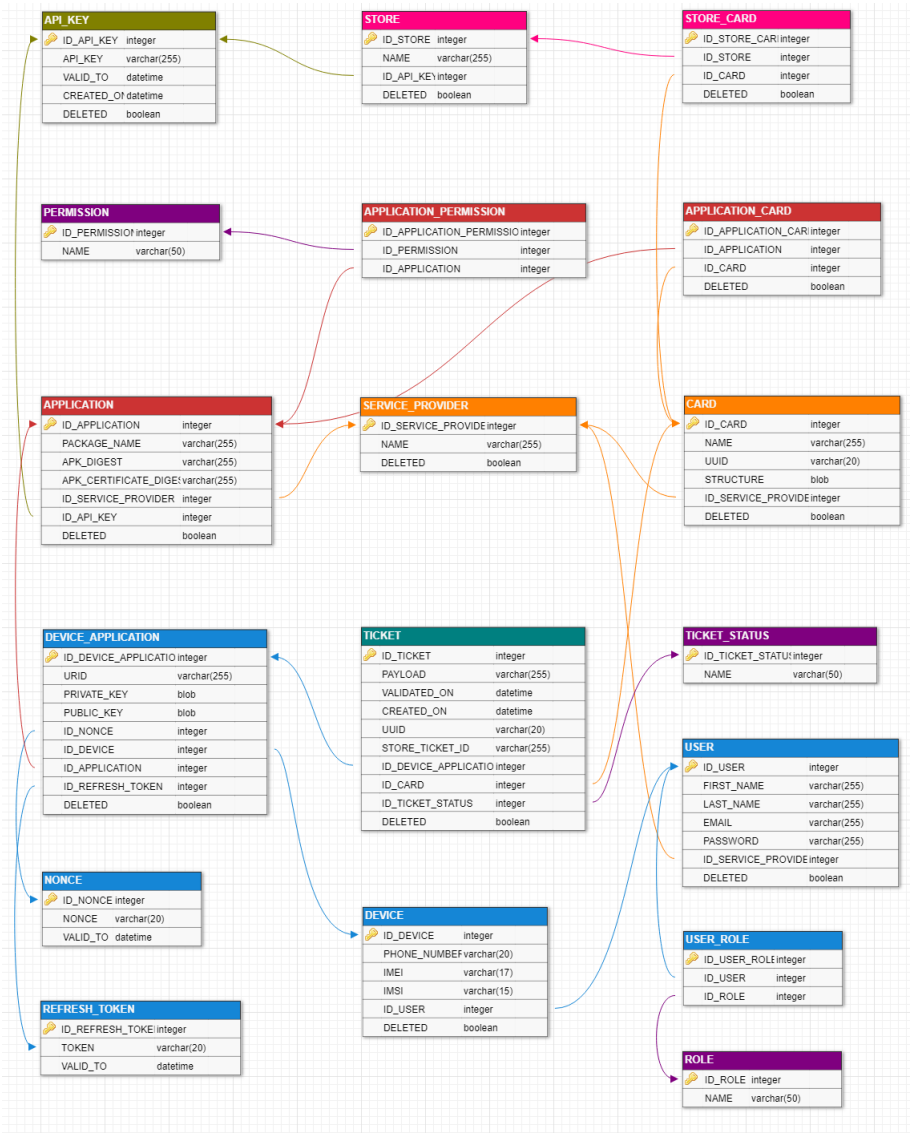
4.4.3 Avtentikator

Avtentikator je del strežnika, ki skrbi, da imajo dostop do podatkov samo aplikacije, ki jim zaupa. Vsak ponudnik javnega prevoza dobi ključ API (angl. *Application Programming Interface*), ki ga vgradi v aplikacijo in ji s tem omogoči dostop do oblaka. Poleg tega mora aplikacija ob vsakem zahtevku poslati tudi dostopni žeton uporabnikovega računa in potrdilo o legitimnosti. Več o slednjih dveh bomo govorili v nadaljevanju.

4.4.4 Podatkovna baza

Oblak uporablja MySQL podatkovno bazo, ki hrani vse potrebne podatke za njegovo delovanje. Njeno shemo sestavlja osemnajst tabel, ki so prikazane na sliki 4.3. Najpomembnejše so:

- **SERVICE_PROVIDER**, ki predstavlja ponudnika javnega prevoza, ki uporablja naš sistem,
- **APPLICATION**, ki predstavlja aplikacije, ki imajo dostop do sistema,
- **API_KEY**, ki predstavlja ključe, s katerimi lahko aplikacije dostopajo do sistema,
- **CARD**, ki predstavlja strukturo virtualne pametne kartice,
- **APPLICATION_CARD**, ki predstavlja strukture kartic, ki jih uporabljajo aplikacije,
- **DEVICE**, ki predstavlja napravo, na kateri teče ena ali več aplikacij,
- **DEVICE_APPLICATION**, ki predstavlja aplikacijo, ki teče na določeni napravi,
- **TICKET**, ki predstavlja vozovnice, ki pripadajo specifični aplikaciji na določeni napravi,
- **TICKET_STATUS**, ki predstavlja status vozovnice,
- **STORE**, ki predstavlja spletno trgovino za vozovnice in
- **STORE_CARD**, ki predstavlja kartice, za katere lahko trgovina prodaja vozovnice.



Slika 4.3: Slika prikazuje shemo podatkovne baze sistema.

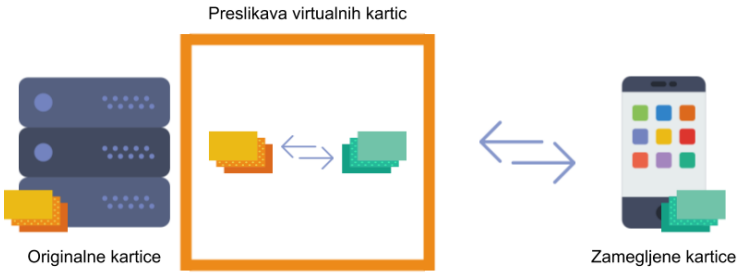
4.5 Varnostni element na mobilni napravi

Ker se celotna implementacija varnostnega elementa nahaja na mobilni napravi, je potrebno dobro poskrbeti za varnost, saj je lahko uporabnik naše rešitve hkrati tudi napadalec. V tem primeru je to še posebej nevarno, saj so napadalcu na voljo orodja, s katerimi je možno aplikacijo na mobilni napravi povrniti v skoraj identično stanje, kot je bila med razvojem [38]. To pomeni, da lahko napadalec točno vidi, kako se programska koda obnaša in kaj opravlja. Možnost ima tudi povezati razhroščevalnik (angl. *debugger*) in aplikacijo ter korak po korak spremljati njeno izvajanje. S tem ima dostop do vseh skrivnosti, ki jih je poskušal razvijalec uvesti z namenom, da bi zavaroval aplikacijo. Eno izmed orodij, ki to omogoča je *Apktool* [39]. Prav zaradi tega je potrebno programsko kodo zavarovati še pred izdajo.

4.6 Zamegljevanje programske kode

Da bi preprečili napadalcu vpogled v programsko kodo in s tem tudi izdajanje skrivnosti, je bilo uporabljeno orodje Proguard [40]. Proguard je orodje za zamegljevanje (angl. *obfuscation*) javanske kode. Ker lahko aplikacije Android prav tako tečejo v tem jeziku, ga je možno uporabiti tudi za zamegljevanje Android aplikacij. Proguard je za lažjo uporabo že vgrajen v Android SDK (angl. *Software Development Kit*). Njegova glavna funkcionalnost je zamenjava imen paketov (angl. *package*), razredov (angl. *class*), metod (angl. *method*) in atributov (angl. *fields*). To naredi programsko kodo skorajda neberljivo, saj so imena, ki jih omenjenim elementom dodeli Proguard namesto originalnih, naključna. S tem zabrišemo kontekst programske kode, saj so po navadi ti elementi poimenovani v skladu z njihovo funkcijo.

Zaradi zamegljene kode na mobilnih napravah pride do novega problema. Razredi, paketi, metode in atributi, ki predstavljajo strukturo virtualne kartice, se zdaj razlikujejo od tistih, ki se nahajajo na oblaku. Ker sistem uporablja serializacijo (angl. *serialization*) za prenos virtualne kartice v varnostni element na mobilni napravi, se morajo imena zgoraj omenjenih elementov na



Slika 4.4: Slika prikazuje preslikavo originalnih virtualnih kartic v zameglene in obratno.

mobilni napravi ujemati s tistimi na oblaku. Če temu ni tako, je nemogoče preslikati vrednosti iz originalnega v zamegljen objekt in obratno. Seveda bi lahko uporabili zamegljene razrede tudi na oblaku, vendar bi to zelo otežilo nadaljnji razvoj in razhroščevanje programske koda tako med razvojem kot tudi v produkciji.

Ko Proguard konča s svojim delom, med drugim ustvari tudi tekstovno datoteko, v kateri so zapisana vsa originalna imena in imena v katera so bila le-ta preslikana. Ta datoteka služi kot slovar, kar nam je omogočilo implementacijo preslikovalnika, prikazanega na sliki 4.4, ki zna preslikati originalne objekte v zamegljene in obratno s pomočjo funkcionalnosti *Java Reflection*.

Odsev (angl. *reflection*) je sposobnost izvajajočega programa, da raziskuje samega sebe in svoje okolje ter prilagodi izvajanje glede na ugotovljeno [41]. Če malo pomislimo, je to točno to, kar počne naš preslikovalnik. S pomočjo slovarja poišče nova imena originalnih razredov in nato preslika vrednosti njihovih atributov v atribute zamegljenih razredov in obratno. S tem razišče ene in druge razrede in prilagodi svoje delovanje tako, da zna operirati z obojimi.

Čeprav smo s preslikovalnikom dosegli, da je programska koda na mobilni napravi skoraj popolnoma neberljiva, še vedno ni nemogoče iz nje izluščiti občutljivih podatkov, kot so na primer kriptografski ključi. Ti se uporabljajo za kriptiranje podatkov, ki jih varnostni element pošilja čitalcu kartic.

4.7 Varovanje kriptografskih ključev

Kriptografski ključi se uporabljajo za zavarovano in zaupanja vredno komunikacijo med varnostnim elementom in čitalcem kartic. V primeru protokola Cipurse so to simetrični ključi, na podlagi katerih se dinamično izračunajo sejni ključi (angl. *session key*). To pomeni, da imata tako čitalec kartic kot tudi varnostni element enak ključ, s katerim se posredno predstavita drug drugemu in preverjata, ali je komunikacija zaupanja vredna. S tem omenjeni protokol varuje komunikacijo pred prisluškovanjem (angl. *eavesdropping*), napadom ponovitve (angl. *replay*) in prestrezanjem komunikacije (angl. *man in the middle*).

Kriptografski ključi so najbolj občutljivi podatek v našem virtualnem varnostnem elementu. Ker so shranjeni na mobilni napravi, kjer ima napadalec dostop do programske kode, jih je potrebno ustrezno temu zavarovati. Ključi morajo biti v taki obliki, da napadalec s pridobljenim ključem ne more narediti škode celotnemu sistemu.

4.7.1 Časovno omejeni kriptografski ključi

Ker so kriptografski ključi shranjeni v pomnilniku mobilne naprave in ima napadalec možnost dostopa do njih, je potrebno ključe zavarovati tako, da napadalec z njimi ne more narediti škode. Tako se je porodila ideja o časovno omejenih kriptografskih ključih.

Sistem, ki je bil implementiran, na vnaprej določen interval menja trenutno veljaven kriptografski ključ za avtentikacijo virtualne pametne kartice s čitalnikom. Da se lahko virtualna kartica predstavi čitalniku s trenutno veljavnim ključem, mora biti sinhronizirana s stanjem na oblaku. To pomeni, da bo naša rešitev zahtevala internetno povezavo z oblakom vsaj enkrat na interval. S tem bodo kartice na mobilni napravi imele vedno veljaven kriptografski ključ za varno komunikacijo.

Sistem, ki računa časovno omejene ključe, teče tako na oblaku kot tudi na čitalniku kartic. Trenutno veljavni ključ se izračuna s pomočjo glavnega

Tabela 4.1: Primer zaokroževanja časovnega intervala za izračun trenutno veljavnega kriptografskega ključa.

Dolžina intervala (min)	Trenuten čas	Zaokrožen čas
2	13:37	13:36
60	13:37	13:00

ključa (angl. *master key*) in trenutnega časa, ki je zaokrožen na začetek trenutnega intervala. Primer zaokroževanja intervala je prikazan v tabeli 4.1.

Algoritem CMAC

Trenutno veljaven ključ je izračunan s pomočjo algoritma *CMAC* [42]. S pomočjo tega algoritma izračunamo avtentikacijsko kodo sporočila *MAC*, ki se nato uporabi kot kriptografski ključ. Algoritem *CMAC* je sestavljen iz dveh delov. Prvi del izračuna dva podključa (angl. *subkey*), ki se nato v drugem delu skupaj s podatki uporabita za izračun na novo veljavnega ključa.

Psevdo programska koda 4.1 prikazuje izračun podključev K_1 in K_2 s pomočjo glavnega ključa K, kjer funkcija MSB_1 vrača skrajno levi bit podanega parametra, funkcija AES_K vrača kriptirano vsebino podanega parametra s ključem K in R_{128} predstavlja konstanto $0^{120}10000111_{\mathrm{b}}$.

$$
\begin{aligned}
&Naj\ bo\ L = AES_K(0_b^{128});\\
&\check{C}e\, MSB_1(L) = 0,\\
&potem\, K_1 = L << 1\\
&drugače\, K_1 = (L << 1) \oplus R_{128};\\
&\check{C}e\, MBS_1(K_1) = 0,\\
&potem\, K_2 = K_1 << 1\\
&drugače\, K_2 = (K_1 << 1) \oplus R_{128};
\end{aligned}
\tag{4.1}
$$

Drugi del algoritma *CMAC* in izračun trenutno veljavnega ključa (K_C) je prikazan s psevdo programsko kodo 4.2. Tu M predstavlja vhodno sporočilo

v algoritem, v tem primeru je to trenutni čas, ki je lahko razdeljeno na cele bloke takrat, kadar je njegova dolžina večkratnih števila 128.

$$
\begin{aligned}
&Naj\ bo\ M = M_1 \parallel M_2 \parallel ... \parallel M_{n-1} \parallel M'_n;\\
&Če\ je\ M'_n\ cel\ blok\\
&potem\ M_n = K_1 \oplus M'_n,\\
&drugače\ M_n = K_2 \oplus (M'_n \parallel 10_b^{127});\\
&Naj\ bo\ C_0 = 0_b^{128};\\
&Od\ i = 1\ do\ n,\ C_i = AES_K(C_{i-1} \oplus M_i);\\
&K_C = MSB_{128}(C_n);
\end{aligned}
\tag{4.2}
$$

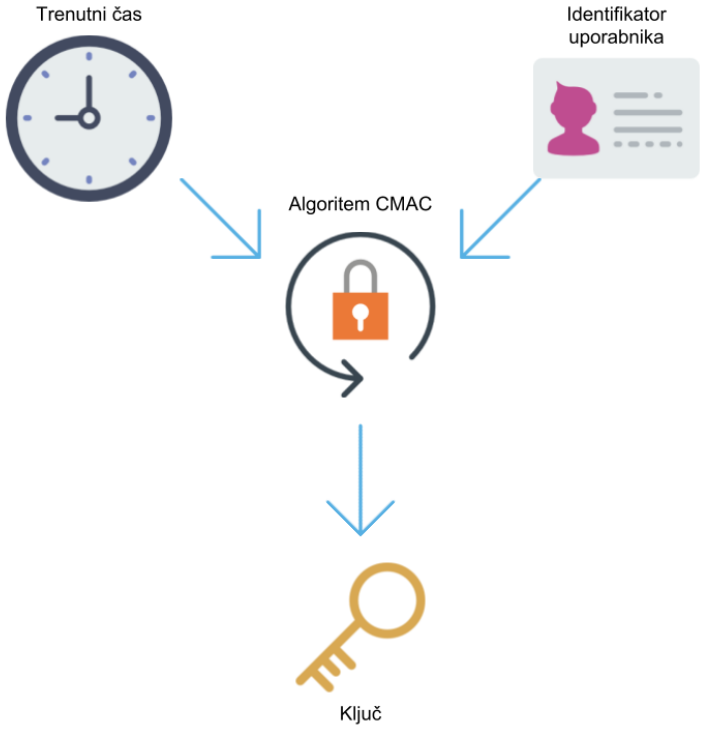
Prekrivanje ključev

S prej opisanim algoritmom naša rešitev ne uporablja samo enega ključa za avtentikacijo virtualnih kartic, ampak se ti menjajo na vnaprej določen interval. Pri tem lahko pride do problemov, saj ima tako čitalec kartic kot tudi mobilna naprava svojo interno uro, ki lahko včasih odstopa od točnega časa. To je razlog za vpeljavo koncepta prekrivanja kriptografskih ključev.

Protokol Cipurse omogoča ob izbiranju aplikacije na pametni kartici, pošiljanje dodatnih parametrov v tako imenovani znački *TAG86*. Te parametre lahko nato čitalec kartic uporabi za izpeljavo sejnega ključa iz glavnega ključa. V našem primeru lahko virtualna kartica pošlje začetni čas intervala, za katerega ima v svoji strukturi shranjen ključ. Odločitev je nato na strani čitalca, da odloči ali je razlika med trenutnim časom in časom, ki ga pošilja kartica še sprejemljiva. Na podlagi ugotovitev se čitalec kartic odloči, ali bo nadaljeval komunikacijo in za izračun ključa uporabil čas, ki ga pošilja virtualna kartica, ali pa bo komunikacijo zavrnil.

4.7.2 Poosebljeni kriptografski ključi

S prekrivanjem ključev smo odpravili težave z morebitnim odstopanjem internih ur med različnimi napravami. Poleg tega pa je potrebno rešiti še en



Slika 4.5: Slika prikazuje izračun poosebljenega ključa z uporabo algoritma *CMAC* ter časom in unikatnim identifikatorjem kot podana parametra temu algoritmu.

problem. Z do sedaj opisanim postopkom generiranja kriptografskih ključev imajo vse virtualne pametne kartice v svoji strukturi isti ključ za avtentikacijo. To je lahko velika varnostna težava, če bi napadalec prišel do ključa in se začel predstavljati drugim mobilnim napravam kot čitalec kartic. Poleg tega bi napadalec lahko s pridobljenim ključem izvajal tudi napad prisluškovanja komunikaciji. To so razlogi, zakaj naša rešitev uporablja poosebljene kriptografske ključe.

Za izračun poosebljenih ključev, naš sistem uporablja algoritem, ki je bil omenjen že v prejšnjem podpoglavju 4.7.1 z minimalnim popravkom. V drugem koraku algoritma se za vhodno sporočilo M ne uporabi samo čas, temveč tudi unikatni identifikator uporabnika. Čas in unikatni identifikator

uporabnika sta podana kot parameter algoritemu *CMAC*. Slika 4.5 prikazuje postopek izračuna poosebljenih ključev.

Za unikaten identifikator uporabnika je uporabljen tako imenovan *UUID* (angl. *Universally Unique Identifier*), ki je samodejno dodeljen uporabniku ob prvi uporabi naše rešitve. Ob začetku komuniciranja virtualne pametne kartice s čitalcem kartic kartica vsakič pošlje *UUID* uporabnika, ki se nato uporabi skupaj s časom za izračun trenutno veljavnega kriptografskega ključa.

S tem smo dosegli, da ima vsak uporabnik svoj ključ in tako v primeru razkritja enega ključa, napadalec ne ogrozi celotnega sistema in ostalih uporabnikov.

4.8 Varovanje podatkov

Čeprav je bilo do zdaj vpeljanih že nekaj varnostnih ukrepov za zaščito sistema, se virtualna pametna kartica še vedno nahaja na mobilni napravi, ki je lahko last napadalca. S tem ima napadalec še vedno dostop do podatkov, ki so shranjeni na napravi, vključno s strukturo in podatki virtualne pametne kartice.

Operacijski sistem Android podpira uporabo podatkovnih baz SQLite, ki so po navadi shranjene v temu namenjene direktorije na mobilni napravi. Čeprav ima razvijalec možnost shraniti podatke skoraj kamorkoli (v primeru *root* aplikacij lahko razvijalec shrani podatke kamorkoli), se ti podatki še vedno nahajajo na napravi. To za našo rešitev predstavlja nov problem, saj lahko napadalec brez posebnega znanja pride do podatkov, ki jih naša rešitev poskuša zavarovati. Ker je podatkovna baza SQLite shranjena kot ena datoteka, ki deluje na različnih operacijskih sistemih in ne podpira različnih uporabnikov [43], jo je možno poiskati in dostopati do njenih podatkov brez posebnega napora, raziskovanja programske kode ali omogočenega *root* dostopa na napravi.

4.8.1 Kriptiranje podatkovne baze

Kot smo ugotovili, je ves naš trud zaman, če ne poskrbimo za varno shranjevanje podatkov na mobilni napravi. Ena izmed rešitev je uporaba sistema Android Keystore [29]. Kot je bilo že omenjeno, je Keystore idealna rešitev za varno shranjevanje kriptografskih ključev, do katerih ima nato dostop samo aplikacija, ki jih je ustvarila.

Podatek, ki ga je potrebno zavarovati pred napadalci je struktura in vsebina virtualne kartice. Tam se namreč nahajajo kriptografski ključi za avtentikacijo s čitalcem in vsebine vozovnic, ki jih kartica pošlje čitalcu kot odkazilo o plačilu vožnje.

Celotno kartico in njeno vsebino mobilna naprava prejme v serializirani obliki in kot tako tudi shrani v svojo lokalno podatkovno bazo. Čeprav je njena struktura zamegljena, pa je njena vsebina še vedno predstavljena z golim besedilom (angl. *plain text*). To predstavlja potencialno varnostno luknjo, ki jo naš sistem rešuje z uporabo sistema Android Keystore.

Ob prvem zagonu aplikacije se izvede tudi generiranje kriptografskega ključa znotraj sistema Keystore, ki je nato uporabljen za kriptiranje in dekriptiranje virtualne kartice ob vsakem shranjevanju in pridobivanju kartic iz podatkovne baze. To varuje občutljive podatke na kartici, saj bi moral napadalec za pridobivanje kriptografskih ključev ali vozovnic na virtualni kartici najprej uganiti ključ, s katerim je bila kriptirana celotna virtualna kartica.

4.9 Izločanje nelegitimnih naprav

Nad mobilnimi napravami z opercijskim sistemom Android je mogoče pridobiti tako imenovani dostop *root*, kar pomeni, da ima neka aplikacija ali uporabnik popoln nadzor nad napravo. Čeprav je to lahko zelo uporabno, lahko tudi predstavlja potencialno nevarnost. Če ima uporabnik popoln nadzor nad napravo in njenim delovanjem, razvijalec težko naredi aplikacijo, ki bi bila varna oziroma odporna na morebitne zlorabe.

V primeru, da ima uporabnik dostop *root*, lahko aplikacijo poveže z raz-

hroščevalnikom in spremlja delovanje aplikacije vrstico za vrstico. Z *root* dostopom ima možnost tudi spreminjanja oziroma prilagajanja delovanja same aplikacije. S tem bi lahko uporabnik izključil enkripcijo naše virtualne pametne kartice ali pa pregledoval njeno vsebino kar med izvajanjem aplikacije.

Do sedaj smo posvetili veliko časa varovanju podatkov, shranjenih na mobilnih napravah z operacijskih sistemom Android. Vse to pa je zaman, če ima uporabnik dostop *root*, zato bi bilo dobro take naprave blokirati.

4.9.1 Android SafetyNet Attestation API

Android SafetyNet Attestation API je sistem, ki pregleda okolje na katerem teče naša aplikacija. Uporablja tehniko, pri kateri analizira programsko in strojno opremo na napravi, ki je naložila našo aplikacijo z namenom ugotavljanja integritete naprave. S tem omogoča detekcijo naprav, katerih operacijski sistem je bil na kakršenkoli način prilagojen ali spremenjen od tistega, ki je bil na napravi ob nakupu. Poleg tega pa pregleda tudi podpis aplikacije, s katerim je mogoče ugotoviti, ali je bila naša aplikacija na kakršenkoli način prilagojena ali spremenjena [44].

Uporaba

Z omenjenim sistemom lahko s strani operacijskega sistema Android naprava pridobi potrdilo o svoji integriteti, katerega podpis je nato potrebno overiti. Ker sta lahko naprava in aplikacija že spremenjena s strani napadalca, je potrdilo nesmiselno overjati na napravi. Zato naša rešitev upošteva napotke podjetja Google in potrdilo overi v oblaku.

Omenjeno potrdilo bi bilo zlahka shraniti na napravi in ga nato pošiljati vedno znova na oblak. Ta napad se imenuje napad s ponovitvijo (angl. *replay*), ki ga Android SafetyNet Attestation API rešuje s pomočjo parametra NONCE (angl. *one occasion*), katerega je potrebno podati ob vsakem klicu tega sistema. Tako kot overjanje potrdila je generiranje parametra *NONCE* nesmiselno na napravi, zato to opravi strežnik.

Notacija 4.1 Spodnja notacija predstavlja potrdilo o legitimnosti naprave v obliki JSON (angl. *JavaScript Object Notation*). Prvi trije parametri predstavljajo *NONCE*, čas in ime paketa aplikacije. Četrti parameter predstavlja SHA-256 zgoščevanje (angl. *hash*) certifikata s katerim je bila podpisana nameščena verzija aplikacije. Peti parameter predstavlja SHA-256 zgoščevanje aplikacije, ki je nameščena na napravi. Vrednost šestega parametra nam pove ali je naprava prestala preizkus CTS (angl. *Compatibility Test Suite*). Zadnji parameter pa je nastavljen na *true*, če naprava najverjetneje ni bila spremenjena ali prilagojena, ampak ni prestala preizkusa CTS.

```
{
  "nonce": "Zuo435hzUg6uiG8zuG8UI",
  "timestampMs": 9860437986543,
  "apkPackageName": "si.primer.ime.aplikacije",
  "apkCertificateDigestSha256": ["zgoščevanje
     certifikata"],
  "apkDigestSha256": ["zgoščevanje APK datoteke"],
  "ctsProfileMatch": true,
  "basicIntegrity": true
}
```

Naša rešitev je zasnovana tako, da strežnik komunicira samo z napravami, ki se mu predstavijo kot legitimne z veljavnim potrdilom sistema Android SafetyNet Attestation API. To zagotavlja tako, da ob vsakem klicu s strani aplikacije zahteva potrdilo. Če potrdila ni, je pretečeno ali neveljavno zahteva obnovitev le-tega in napravi odgovori z naključno zgrajenim parametrom *NONCE*, ki ga shrani v podatkovno bazo za kasnejše preverjanje. Potrdilo, ki ga pošlje mobilna naprava je veljavno, če:

- se *NONCE* v potrdilu ujema s tistim v podatkovni bazi,
- je razlika med trenutnim časom in časom v potrdilu manjša od vnaprej določene vrednosti,
- se ime paketa aplikacije ujema z imenom paketa originalne aplikacije,
- se zgoščevanje certifikata, s katerim je bila podpisana aplikacija ujema z zgoščevanjem certifikata, s katerim je bila podpisana originalna verzija aplikacije,
- se zgoščevanje nameščene APK datoteke ujema z zgoščevanjem originalne APK datoteke,
- je parameter *ctsProfileMatch* nastavljen na *true*,
- je parameter *basicIntegrity* nastavljen na *true* in
- je bilo potrdilo podpisano z Android certifikatom.

Ker je za pridobitev potrdila potreben klic na strežnike podjetja Google, naša rešitev dopušča, da si naprava za nekaj časa shrani potrdilo in ga uporablja za dostop do oblaka. Ko potrdilo poteče, je naslednji klic s strani aplikacije na oblak zavrnjen in potrebna je pridobitev obnovljenega potrdila.

4.10 Generična zasnova

Kot je bilo že omenjeno, je naša rešitev zasnovana na način, da omogoča uporabo na različnih infrastrukturah ponudnikov javnega prevoza. Ta pod-

jetja imajo tudi najverjetneje že obstoječe aplikacije ali pa jih želijo morda razviti na svoj način. To je razlog, da je naša rešitev v obliki knjižnice Android, ki jo je mogoče umestiti v aplikacijo med razvojem. S tem imajo ponudniki javnega prevoza proste roke pri razvoju aplikacije, hkrati pa se jim ni potrebno ukvarjati z vsemi varnostnimi težavami, ki se jih morda ne zavedajo. Čeprav je ideja o generičnosti zelo privlačna, pa po drugi strani s seboj prinese določene težave.

4.10.1 Registracija uporabnika brez gesla

Ena izmed težav je enolično določanje uporabnika, saj bo najverjetneje ponudnik javnega prevoza tisti, ki bo hranil podatke o uporabniku, katere bo pridobil skozi postopek registracije. Tako kot sistemi teh podjetij, mora tudi naš sistem razlikovati uporabnike med seboj. Možna rešitev je avtomatična registracija uporabnika s strani prevoznika. Čeprav se na začetku to zdi lepa rešitev, pa se moramo zavedati, da samodejna registracija s strani ponudnika javnega prevoza ne more uporabljati gesla za registracijo, saj bi bilo potrebno hranjenje gesla na strani ponudnika javnega prevoza v golem besedilu. Namesto registracije uporabnika v naš sistem z geslom, predstavljanim z golim besedilom, bi lahko uporabili zgoščevanje gesla, ki je hranjeno v podatkovni bazi prevoznika, vendar je tudi to slaba zamisel. Dobra praksa je namreč, da shranjena gesla in njihova zgoščevanja, ne zapuščajo sistema, kjer so hranjena.

Zaradi omenjenih razlogov je bila razvita rešitev, ki za registracijo ne uporablja občutljivih podatkov, ki bi jih zahtevala s strani prevoznika. Postopek registracije v naš sistem je opravljen brez uporabniškega imena, e-poštnega naslova ali gesla. Naša rešitev za uspešno registracijo potrebuje samo IMEI (angl. *International Mobile Equipment Identity*) naprave in telefonsko številko uporabnika. Slednjo je nemogoče zanesljivo pridobiti samodejno, zato jo mora uporabnik vnesti med registracijo.

Postopek registracije uporabnika bo znotraj končne aplikacije narejen s strani prevoznika, zato je v naši knjižnici že pripravljen razred, ki ga lahko

razvijalci razširijo, prilagodijo in vgradijo v svoj postopek registracije. S tem je bila dosežena registracija v dva sistema, čeprav je z uporabnikove perspektive videti kot običajen postopek registracije.

4.10.2 Legitimnost registracije

Z omenjenimi podatki, ki jih naš sistem zbere med postopkom registracije, je mogoče račun uporabnika vezati na specifično mobilno napravo in kartico SIM (angl. *Subscriber Identity Module*). Ti podatki so bili izbrani z namenom preprečevanja zlorab. Če bi za enolično določanje uporabnika uporabili samo telefonsko številko, bi zlahka prišlo do zlorab. Tako bi lahko več uporabnikov med registracijo vneslo isti podatek. V tem primeru bi bilo uporabnike nemogoče ločiti med seboj. Nezmožnost ločevanja uporabnikov pa pomeni, da bi ob nakupu vozovnice na eni napravi, vsi z enako vneseno telefonsko številko ob registraciji prejeli identično vozovnico na svojo mobilno napravo. S tem bi bilo mogoče pretentati sistem in koristiti več vozovnic za ceno ene.

Med postopkom registracije uporabnika in njegove naprave se zgodi tudi dvostopenjsko preverjanje (angl. *2-step verification*) s pomočjo potrditvene kode, ki jo uporabnik prejme preko SMS (angl. *Short Message Service*) sporočila. S tem je zagotovljeno, da je uporabnik vnesel veljavno telefonsko številko. Čeprav je nemogoče preveriti, ali je to res njegova telefonska številka, pa na ta način izločimo napadalce, ki bi se skušali predstaviti z neveljavnimi številkami.

4.10.3 Koraki registracije

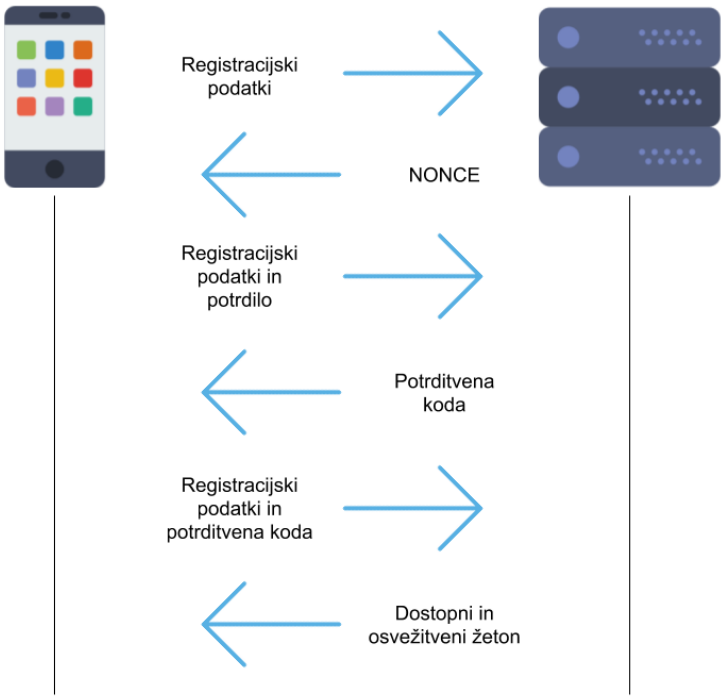
Slika 4.6 prikazuje postopek registracije vključno s procesoma, ki sta opisana v poglavjih 4.9 in 4.10. Z nje je možno razbrati kako se uporabnik in njegova mobilna naprava prijavita v naš sistem. Registracija poteka v šestih korakih. V prvem koraku mobilna naprava vneseno telefonsko in samodejno pridobljeno številko IMEI pošlje na oblak, ki nato v drugem koraku na podlagi prejetih podatkov ustvari nov uporabniški račun, zgradi naključni *NONCE*

in ga pošlje nazaj na mobilno napravo. Sledi tretji korak, kjer je mobilna naprava zadolžena za pridobitev potrdila o svoji legitimnosti s strani sistema *Android SafetyNet Attestation API* in ga skupaj s telefonsko in številko IMEI poslati na oblak. V tem koraku je potrebno ponovno pošiljanje registracijskih podatkov zato, ker mobilna naprava še ni prejela dostopnega žetona (angl. *access token*), da bi jo strežnik lahko identificiral. V četrtem koraku je naloga oblaka preveriti prejeto potrdilo po postopku, ki je opisan v poglavju 4.9 in poslati SMS s potrditveno kodo na mobilno napravo. Ko SMS prispe na mobilno napravo, se s pomočjo sistema *Android SMS Retriever API* samodejno prebere in izlušči potrditvena koda, ki je nato v petem koraku skupaj z registracijskimi podatki poslana nazaj na oblak. Iz enakega razloga kot v tretjem je tudi v tem koraku potrebno ponovno pošiljanje registracijskih podatkov. Zadnji korak je preverjanje prejete potrditvene kode na strani oblaka in generiranje dostopnega in osvežitvenega žetona (angl. *refresh token*). Prejeta žetona nato naprava shrani in ju uporablja za dostop do oblaka. Za dostop do oblaka mora vedno v glavi (angl. *header*) zahtevka poslati dostopni žeton. Ko veljavnost tega poteče, lahko pridobi nova žetona z osvežitvenim žetonom. V primeru, da uporabnik aplikacije ni uporabil toliko časa, da veljavnost poteče tudi osvežitvenemu žetonu, je potrebno ponoviti celoten postopek registracije.

4.11 Detektiranje zlorab

Omenjeno je bilo že, da stoprocentna varnost v svetu računalništva ne obstaja. Čeprav smo našo rešitev dobro zavarovali, je še vedno potrebno imeti sistem, ki bo zaznal morebitne zlorabe. Naš primer je še posebno kompleksen zaradi okoliščin, ki zahtevajo, da se transakcije dogajajo brez internetne povezave. Prav zaradi te zahteve je detektiranje zlorab v realnem času v nekaterih primerih nemogoče. Iz tega razloga je bila razvita rešitev, ki pridobi dnevnik zapisov iz čitalnika kartic in pregleduje, ali je prišlo do zlorabe.

Ker so čitalci kartic večino časa v nespletnem načinu delovanja, je do-



Slika 4.6: Slika prikazuje postopek registracije uporabnika in njegove mobilne naprave ob prvem zagonu aplikacije.

stop do dnevnikov periodično ponavljajoč se proces, ki se zgodi ob vnaprej določenih intervalih oziroma takoj ko je to možno. Dolžina intervala je odvisna od čitalcev kartic, ki jih uporabljajo ponudniki javnega prevoza. Nekateri imajo možnost konstantnega dostopa do internetne povezave in komuniciranja s strežniki, med tem ko imajo drugi to možnost enkrat na daljše obdobje. To pomeni, da so lahko nekateri čitalci kartic brez internetne povezave tudi po več dni. Ravno to predstavlja največji problem za našo rešitev. V takih primerih bi lahko napadalec, ob predpostavki, da mu je uspelo zaobiti vse varnostne ukrepe, ki smo jih do sedaj predstavili, večkrat zapored uporabljal vozovnice za enkratno uporabo. S tem bi nastala škoda, ki jo je potrebno resno obravnavati.

Naša rešitev take težave odpravlja s periodičnim zbiranjem dnevnikov in pregledovanjem uporabljenih vozovnic. Če je bila ena vozovnica uporabljena večkrat, kot je to dovoljeno, gre za zlorabo. Ob ugotovljeni zlorabi je potrebno tudi izslediti odgovorno osebo. Čeprav naša rešitev nima pravic za terjatev odgovorne osebe, pa lahko vseeno ugotovi, kdo je povzročil nastalo škodo.

V poglavju 4.7.2 smo opisovali poosebljene ključe, ki se uporabljajo za avtentikacijo in zaupno komunikacijo med čitalcem kartic in virtualno pametno kartico. Ker se s tem ob vsaki komunikaciji med virtualno pametno kartico in čitalcem kartic predstavi tudi uporabnik, ga je mogoče izslediti. Tako sistem ugotovi, kater uporabniški račun je odgovoren za nastalo škodo in ga blokira. Poleg tega je v našem sistemu zabeležena tudi uporabnikova telefonska številka, ki jo je mogoče uporabiti pri izsledit vi uporabnika. Delo je še lažje, če ponudnik javnega prevoza med procesom registracije uporabnika zahteva njegove osebne podatke. V tem primeru, je mogoče povezati telefonsko številko z osebnimi podatki in tako točno vedeti za katerega uporabnika gre.

Poglavje 5

Analiza in rezultati

5.1 Primerjava rešitve s pametnimi karticami

Da bomo znali našo rešitev primerno vrednotiti, jo je potrebno tudi primerjati z obstoječimi pametnimi karticami. Ker smo si na začetku zadali cilj, da bo naša rešitev delovala s podobno hitrostjo kot fizične pametne kartice, je potrebno primerjati hitrosti delovanja le-teh s hitrostjo delovanja naše rešitve. Testiranja so bila opravljena s čitalcem kartic podjetja Planeta Informatica model RC700 SAM REV053. Za vsako kartico je bilo opravljenih 20 testiranj validiranja vozovnic in 20 testiranj preverjanja stanja kartic.

V tabeli 5.1 so prikazani povprečni časi v milisekundah od začetka do

Tabela 5.1: Tabela prikazuje primerjanje rezultatov testiranja hitrosti naše rešitve in fizičnih pametnih kartic.

Ime kartice	Čas validacije (ms)	Čas preverjanja stanja (ms)
Virtualna pametna kartica	**293,7**	**258,9**
Virtualna pametna kartica v oblaku	1220,9	1196,8
Tubitak Bilgem AKiS Bilet	295,25	240,5
ZeitControl ZCPurse T Profile	185,6	152,7
Infineon SLM 10TLC002L S	152,7	155,1
Oberthur cityGo CIPURSE S	153,7	139,7

konca komunikacije med kartico in čitalcem. Iz rezultatov lahko vidimo, da naša rešitev deluje v mejah fizičnih pametnih kartic. Čeprav so meritve naše rešitve zelo podobne tistim, ki so bile opravljene na najslabši kartici, lahko rečemo, da naša rešitev dosega hitrosti komuniciranja fizičnih pametnih kartic. Najbolj spodbudna primerjava je primerjava z virtualno pametno kartico, realizirano v oblaku. Naša rešitev je kar štirikrat hitrejša. Treba se je zavedati, da so bile meritve za pametno kartico v oblaku opravljene na lokalni mreži, kar pomeni, da bi v realnosti ta rešitev delovala še počasneje.

5.1.1 Analiza hitrosti komuniciranja

Da bi ugotovili, kateri deli naše rešitve zahtevajo največ časa, je bila narejena analiza procesiranja ukazov znotraj virtualnega varnega elementa. Ugotovljeno je bilo, da operacijski sistem Android v povprečju porabi 25 ms za predobdelavo prejetega ukaza *APDU*, ki ga prejme s strani čitalca, preden ga preda naši aplikaciji. To je približno dvakrat do šestkrat več kot naša rešitev potrebuje za obdelavo *APDU* ukazov. Ti v povprečju potrebujejo nekje od 4 do 13 ms. Ker je ozko grlo naše rešitve operacijski sistem Android, jo je brez pomoči podjetja Google nemogoče pohitriti.

5.2 Varnost sistema

Največja nevarnost našemu sistemu je to, da lahko napadalec pridobi izvorno kodo aplikacije, jo spremeni in ponovno namesti na svojo mobilno napravo. S tem bi lahko napadalec prilagodil delovanje našega virtualnega varnostnega elementa. Kot je bilo že omenjeno, naš sistem to preprečuje z uporabo sistema *Android SafetyNat Attestation API*.

V okviru tega dela je bilo preizkušeno, kako ta sistem deluje. Že naložena aplikacija je bila povrnjena nazaj v izvorno kodo, minimalno prilagojena in naložena nazaj na mobilno napravo. Pri spreminjanju aplikacije ni bil zamenjan aplikacijski paket (angl. *application package name*), da bi poskusili pretentati v poglavju 4.9 omenjen sistem. Naš sistem je vseeno ugotovil

neskladje med originalno in spremenjeno aplikacijo ter zavrnil komunikacijo. S tem lahko rečemo, da je naša rešitev odporna proti osnovnim napadom, pri katerih poskuša napadalec zaobiti varnostni sistem Android SafetyNet Attestation API.

5.3 Možne ranljivosti

Ranljivosti naše rešitve so že delno omenjene v poglavju 4.11. Ker se v naši rešitvi občutljivi podatki nahajajo v pomnilniku mobilne naprave, je možnost napadov zelo velika. Posebej problematične so mobilne naprave, kjer je bil omogočen *root* dostop. Čeprav naš sistem poskuša tovrstne naprave blokirati, pa bi prišlo do problema, če napadalcu uspe zaobiti ta sistem. V tem primeru bi lahko napadalec uporabljal že koriščene vozovnice, dokler na oblak ne prispejo dnevniki, ki jih zbirajo čitalci kartic. V tem trenutku bi bil račun uporabnika blokiran in bi za ponovno uporaba sistema potreboval novo telefonsko številko ali novo mobilno napravo. To bi se zgodilo v najslabšem primeru vsakih nekaj dni, kar pomeni, da bi imel napadalec kar veliko stroškov z menjavanjem SIM kartic in mobilnih naprav. Otežiti napadalcu delo do te mere pa je tudi cilj naše rešitve. Čeprav bi napadalec zaobšel vse varnostne ukrepe, ki smo jih omenili, bi na koncu najverjetneje ugotovil, da se ne izplača toliko dela in stroškov za tako manjhno nagrado kot je vozovnica za avtobus.

Poglavje 6

Sklepne ugotovitve

V delu je predstavljena realizacija virtualne pametne kartice z varnostnim elementom na mobilnih napravah. Ker se celotna realizacija varnostnega elementa nahaja na napravi, lahko rešitev deluje v nespletnem načinu. S tem se ukazi izvajajo veliko hitrejše, kot to mogočajo rešitve, kjer je varnostni element realiziran v oblaku in je z njim potrebna zanesljiva in hitra internetna povezava. Zmožnost delovanja v nespletnem načinu je še posebej pomembena v situacijah, kjer internetna povezava ni na razpolago ali pa je ta šibka. To pride še posebej do izraza v javnem prometu, kjer je treba zagotoviti hitro validacijo vozovnic, spletna povezava pa ni vedno zagotovljena (npr. podzemna železnica). Po drugi strani pa ta rešitev prinaša vrsto varnostnih tveganj, saj se vsi podatki, ki jih potrebuje za svoje delo, nahajajo na mobilnem telefonu in so s tem lahko dostopni napadalcu. Ravno zato je potrebna občasna sinhronizacija podatkov na mobilni napravi s podatki v oblaku in čim bolj pogosta sinhronizacija podatkov med čitalci kartic in oblakom.

Omenjenim problemom smo v delu posvetili osrednjo pozornost. Problem varnosti smo rešili z zamegljevanjem programske kode, časovno omejenimi in poosebljenimi kluči in preverjanjem legitimnosti naprav. Čeprav smo ugotovili, da je varovanje podatkov na mobilnih napravah težak problem, nam je uspelo našo rešitev vseeno dobro zavarovati. Poleg tega nam je uspelo rešitev narediti dovolj hitro, da se lahko primerja s hitrostjo delovanja fizičnih pa-

metnih kartic. To je pomembno še posebej za ponudnike javnega prevoza, ki potrebujejo varne in hitre sisteme, ki delujejo brez uporabe internetne povezave. Problem bi bil veliko težje rešljiv, če bi poskušali zagotoviti varnost na prireditvah, kot so koncerti. Na takih dogodkih so cene vstopnic po navadi veliko višje, zato je nespletni način pregledovanja vstopnic zelo tvegan. Napadalci bi lahko eno vstopnico klonirali in z njo vstopili vsak na svojem vhodu. V nespletnem načinu je tako zlorabo nemogoče zaznati takoj, saj čitalci kartic niso zmožni komuniciranja drug z drugim. Ko bo tehnologija napredovala do te mere, da bodo imeli čitalci možnost zanesljivega in hitrega komuniciranja drug med drugim, bo problem varnosti podatkov na mobilnih napravah dosti lažji.

Rešitev, opisana v tem delu, je namenjena ponudnikom javnega prevoza in bi lahko zmanjšala uporabo plastičnih kartic. S takim sistemom bi poleg nakupa in validacije vozovnic, bila možna tudi realizacija sistemov za izmenjavanja vozovnic med uporabniki ali kupovanja vozovnic kot darilo. S tem bi uporabnikom omogočili veliko lažjo in bolj prijetno uporabo javnega prometa.

Literatura

[1] M. P. Pelletier, M. Trépanier, C. Morency, "Smart card data use in public transit: A literature review", *Transportation Research Part C: Emerging Technologies*, št. 19, zv. 4, str. 557–568, 2011.

[2] P. T. Blythe, "Improving public transport ticketing through smart cards", *Proceedings of the Institution of Civil Engineers - Municipal Engineer*, št. 157, zv. 3, str. 47–54, 2004.

[3] S. Louca, D. Nathanael, "The Introduction of Smartcards in Small European Countries: The Case of Cyprus", School of Business, Intercollege Nicosia, 2004.

[4] T. W. Buescher, D. L. Manchester, "Passive contactless smartcard security system", U.S. Patent No. 6,588,660. 8 Jul. 2003.

[5] R. Jones, "Contact smart cards having a document core, contactless smart cards including multi-layered structure, pet-based identification document, and methods of making same." U.S. Patent No. 6,843,422. 18 Jan. 2005.

[6] A Smart Card Alliance Mobile & NFC Council White Paper, "Host Card Emulation (HCE) 101" Dostopno na: `http://www.iqdevices.com/pdfFiles/HCE-101-WP-FINAL-081114-clean.pdf`

[7] L. Cremer, "Cipurse: An open standard for next-generation fare collection solutions.", IT-Trans, IT Solutions for Public Transport. No. Session 1. 2012.

[8] K. Finkenzeller, "RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication.", *John Wiley & Sons*, 2010.

[9] R. Wolfgang, E. Wolfgang, "Smart card handbook", *John Wiley & Sons*, 2004.

[10] M. Roland, J. Langer, J. Scharinger, "Practical attack scenarios on secure element-enabled mobile devices.", *Near Field Communication (NFC), 2012 4th International Workshop on.*, IEEE, str. 19–24, 2012.

[11] B. Choudhary, J. Risikko, "Mobile Financial Services Business Ecosystem Scenarios Consequences.", Mobey Forum, c/o Nordea Bank, Satamaradankatu. Vol. 3. 2006.

[12] M. Reveilhac, M. Pasquet, "Promising Secure Element Alternatives for NFC Technology", *Near Field Communication*, 2009. *NFC'09. First International Workshop on.* IEEE, 2009

[13] EMVCo, "Emv mobile contactless payment technical issues and position paper", EMVCo, Tech. Rep., 2007

[14] V. Coskun, B. Ozdenizci, K. Ok, "The survey on near field communication", *Sensors* Multidisciplinary Digital Publishing Institute, št. 6, zv. 15, str. 13348–13405, 2015.

[15] K. Curran, A. Millar, C. Mc Garvey, "Near field communication.", *International Journal of Electrical and Computer Engineering*, št. 2, zv. 3, str. 371, 2012.

[16] R. Want, "Near field communication.", *IEEE Pervasive Computing*, št. 10.3, str. 4–7, 2011.

[17] V. Sharma, P. Gusain, P. Kumar, "Near field communication.", *Conference on Advances in Communication and Control Systems 2013*, str. 342–345, 2013.

[18] A. Kumar, "Near field communication.", 2011.

[19] R. Hollander (2017) "Two-thirds of the world's population are now connected by mobile devices.", Dostopno na: `http://www.businessinsider.com/world-population-mobile-devices-2017-9`

[20] Zachary D. Boren (2014) "There Are Officially More Mobile Devices Than People in the World.", Dostopno na: https://www.independent.co.uk/life-style/gadgets-and-tech/news/there-are-officially-more-mobile-devices-than-people-in-the-world-9780518.html

[21] G. Huerta-Canepa, L. Dongman, "A virtual cloud computing provider for mobile devices.", *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing Services: Social Networks and Beyond.*, ACM, 2011.

[22] Hoang T. Dinh, "A survey of mobile cloud computing: architecture, applications, and approaches.", *Wireless communications and mobile computing*, št. 13, zv. 18, str. 1587–1611, 2013.

[23] J. Fischer, "NFC in cell phones: The new paradigm for an interactive world [Near-Field Communications].", *IEEE communications Magazine*, št. 46, zv. 6, str. 22–28, 2009.

[24] M. Alattar, M. Achemlal, "Host-based card emulation: Development, security, and ecosystem impact analysis.", *High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), 2014 IEEE Intl Conf on.*, IEEE, 2014.

[25] A. Umar, K. Mayes, K. Markantonakis, "Performance variation in host-based card emulation compared to a hardware security element.", *Mobile and Secure Services (MOBISECSERV), 2015 First Conference on.*, IEEE, str. 1–6, 2015.

[26] L. Cremer, "Cipurse: An open standard for next-generation fare collection solutions.", *IT-Trans, IT Solutions for Public Transport.*, No. Session 1. 2012.

[27] A. Andersen, A. Munch-Ellingsen, "Mobile device security: The role of NFC, UICC and secure elements.", *Norsk Informasjonssikkerhetskonferanse (NISK 2014)*, Fredrikstad, 2014.

[28] A. Munch-Ellingsen, R. Karlsen, A. Andersen, S. Akselsen, "Two-factor authentication for android host card emulated contactless cards.", *Mobile and Secure Services (MOBISECSERV), 2015 First Conference on.*, IEEE, str. 1–6, 2015.

[29] "Android keystore system", Dostopno na: `https://developer.android.com/training/articles/keystore`, 2018

[30] M. Fisher, "Social Media Marketing Based on Transactions Using a Mobile Device and Associated Secure Element." U.S. Patent Application No. 13/195,055. 2 Feb. 2012.

[31] D. Brudnicki, "System and Method for Providing a Virtual Secure Element on a Portable Communication Device." U.S. Patent Application No. 13/279,147. 17 May. 2012.

[32] Thomas O. Holtey, Peter J. Wilson, "Secure memory card." U.S. Patent No. 5,293,424. 8 Mar. 1994.

[33] M. Roland, J. Langer, J. Scharinger, S. Akselsen, "Relay attacks on secure element-enabled mobile devices.", *IFIP International Information Security Conference.*, Springer, Berlin, Heidelberg, 2012.

[34] T. Smole, "Implementacija brezstičnih pametnih kartic z varnostnim elementom v oblaku.", Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 2015.

[35] B. Ozdenizci, V. Coskun, K. Ok, T. Karlidere, "A secure communication model for HCE based NFC services.", *3rd International Conference on Creative Technology*, str. 1–4, 2015.

[36] B. Ozdenizci, V. Coskun, S. Gulsecen, K. Ok, "A Cloud Based Framework for HCE enabled NFC Services.", *64th The IIER International Conference, At Barcelona*, Spain, March, 2012.

[37] D. Gruntz, C. Arnosti, M. Hauri, "MOONACS: a mobile on-/offline NFC-based physical access control system.", *International Journal of Pervasive Computing and Communications*, Emerald Group Publishing Limited, št. 1, zv. 12, str. 2–22, 2009.

[38] Brij K. Misra, S. Tripathi, "Code and Data Security Risk in Android.", *HCTL Open International Journal of Technology Innovations and Research (IJTIR)*, 2015.

[39] "Apktool: A tool for reverse engineering Android apk files.", Dostopno na: `https://ibotpeaches.github.io/Apktool/`, 2018

[40] "Proguard", Dostopno na: `http://proguard.sourceforge.net`, 2018

[41] Ira R. Forman. N. Forman, J. Vlissides Ibm, "Java reflection in action.", 2004.

[42] M. Dworkin, "NIST special publication 800-38B." NIST special publication 800.38B, Mar 2005.

[43] S. Mutti, E. Bacis, S. Paraboschi, "Sesqlite: Security enhanced sqlite: Mandatory access control for android databases.", *In Proceedings of the 31st Annual Computer Security Applications Conference*, ACM, str. 411-420 December, 2015.

[44] "SafetyNet Attestation API", Dostopno na: `https://developer.android.com/training/safetynet/attestation`, 2018

[45] licence-cc.pdf. Dostopno na: https://ucilnica.fri.uni-lj.si/course/view.php?id=274