Heterogena dvorazinska autentifikacija

Ignjatov, Lucas

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Lucas Ignjatov

Heterogena dvorazinska autentifikacija

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Lucas Ignjatov

Matični broj: 0016135410

Studij: Informacijski sustavi

Heterogena dvorazinska autentifikacija

ZAVRŠNI RAD

Mentor/Mentorica:

Doc. dr. sc. Robert Kudelić

Varaždin, kolovoz 2022.

Lucas Ignjatov

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Ovaj završni rad bavi se heterogenom dvorazinskom autentifikacijom. Dvorazinska autentifikacija je autentifikacija putem dva dokaza. U ovom slučaju su dva dokaza lozinka i mobilni uređaj. Prvi dokaz uključuje korisničko registriranu lozinku. Druga koristi bežićnu komunikaciju za prijenos podataka (*Near Field Communication)* (NFC) i čitač kako bi se registrirao i prijavio mobilni uređaj. Podaci se šalju kriptirani i u te svrhe se koristi RSA algoritam za asimetrično kriptiranje podataka te SHA256 algoritam za šifriranje istih podataka. U slučaju zaboravljene lozinke ili mobitela postoji i rezeverni dokaz u obliku odgovaranja na sigurnosna pitanja prepoznavanja lica.

Ključne riječi: Dvorazinska autentifikacija; mobilni uređaj; Near Field Communication; RSA; prepoznavanje lica; Lozinka; asinkrona enkripcija;

Sadržaj

1. Uvod

Autentifikacija je proces identificiranja pojedinca (Željko Panian, str. 44). Kada koristimo usluge ili web stranice na internetu kao što su Facebook, Instagram i Twitter traži se da prijavimo račun kako bi koristili stranice. U studentskoj menzi prislanjamo naše studentske iskazice na čitač kartica kako bi ovjerili da smo studenti i koji smo student. Pri izradi osobne iskaznice moramo donijeti fotografiju sebe i napraviti potpis koji će biti vidljiv na osobnoj („Ministarstvo unutarnjih poslova Republike Hrvatske.“, n.d.).

Sve od navedenoga su različiti načini autentifikacije. Ovaj rad će se primarno baviti implementacijom autentifikacije na računalu.

Najčešći način autentifikacije na internetu i računalu je putem lozinke. Korisnici preferiraju jednostavnost lozinke više od sigurnosti te se time stavljaju u opasnost od mogućih hakerskih napada (Maayan, n.d.). 2019. godine je centar za cyber security NCSC objavio listu sa 100.000 najčešćih lozinki koje hakeri koriste kako bi pokušali ući u korisničke račune korisnika usluga. Dobro konstruirane lozinke znatno smanjuju mogućnost provale u račune, ali koštaju korisnike jednostavnost prijave lakšom lozinkom.

Kako bi zadržali jednostavnost prijave korisnika i istovremeno napravili sigurnost jačom na korisničkim računima, možemo koristiti dvorazinsku autentifikaciju. Dvorazinska autentifikacija je ekstra sloj sigurnosti koji služi da bi se identificirao pojedinac. Ako bi se pokušali prijaviti na aplikaciju Steam putem računa koji ima uključenu dvorazinsku autentifikaciju, onda bi nakon unošenih podataka za korisničko ime i lozinku dobili još jedan prozor koji od nas traži peteroznamenkasti kod koji je prikazan na aplikaciji na mobilnom uređaju. To je jedna vrsta implementacije dvorazinske autentifikacije.

Ovaj rad se odnosi na implementaciju dvorazinske autentifikacije koristeći mobilni uređaj na Android operacijskom sustavu kao hardverski token koji vrši komunikaciju sa računalom putem NFC čitača. Ideja je da nije potrebno pročitati poruku ili odabrati opciju na ekranu već samo prisloniti mobilni uređaj na čitač kako bi se provjerila identifikacija pojedinca. Dvorazinske autentifikacije viđamo sve češće što je internet veći i što više naš osobni život ovisi o njemu.

Rad će pratiti provedeno istraživanje. Započeti će se temom *„Near-Field Communication“* (NFC) i Android sustava i opisati implemetaciju komunikacije između računala i mobilnog uređaja. Zatim prelazi na sigurnost u kojem se opisuje sigurnost, simetrična i asimetrična enkripcija te opis RSA enkripcije, šifriranja i njihove implementacije u programu. Dalje ćemo detaljno razraditi implementaciju što obuhvaća opisivati klase i metode

stvorene i korištene za izradu rada te ćemo pregledati procese potrebne za slučajeve korištenja. Na kraju ćemo u zaključku ucijeloviti i ponoviti sve što je napravljeno.

2. Metode i tehnike rada

Tijekom izrade rada su korištena 3 uređaja. Prvotno je bio potreban mobilni uređaj s mogučnošću korištenja NFC tehnologije, zatim računalo na kojem se nalazi aplikacija kojom će mobilni uređaj interkatirati/komunicirati i medij između ta dva uređaja u obliku NFC čitača ACR1252U-M1 koji služi za komunikaciju između prijašnje navedena uređaja. Za programiranje su se koristila dva programska alata: Android Studio za programiranje u Java programskom jeziku na Android mobilnom sustavu i Visual Studio za programiranje u C# programskom jeziku koristeći .NET Framework i Windows forms na strani računala.

Za istraživanje teme su korištene knjige, pouzdane web stranice i stručne dokumentacije Google Android sustava, Microsoft dokumentacija za .NET i Winscard.h te dokumentacija ACR1252U čitača.

3. NFC i Android

3.1. NFC

Pravilni redoslijed obrađivanja tema smatram da je onaj koji će se prije u kodu reprezentirati. Nakon odabira teme prvo je bilo bitno shvatiti što je, kako radi i kako ću iskoristiti tehnologiju prijenosa podataka prije nego li krenem analizirati i implementirati sigurnost. U ovom poglavlju ćemo proći teme općenitog znanja, kako programirati uređaj i na kraju primjer.

3.1.1. NFC Općenito

NFC tehnologija je beskontaktna komunikacijska tehnologija. Radi na temelju radio frekvenciji od 13.56MHz i omogućuje prenošenje energije kako bi uređaje za spajanje kao što su NFC tagovi i beskontaktne kartice mogli koristiti bez zasebnih baterija (NFC Forum, n.d.). Često danas viđamo plaćanje u dućanima beskontaktnim bankovnim karticama. Mastercard Hrvatska procjenjuje da je 8 od 10 kupnji karticama u Hrvatskoj u 2020. godini provedeno beskontaktno te bilježe rast od 7-8% svake godine (Vrdoljak, 2020.).

NFC omogućuje prijenos podataka. Brzina i veličina poslanih podataka su male i transakcije su kratke. Prijenos se može izvršavati u 3 različita načina rada na čitaču i na Androidu:

- Čitač/pisač način rada
- Peer-2-peer (P2P) način rada
- Emulacija kartice

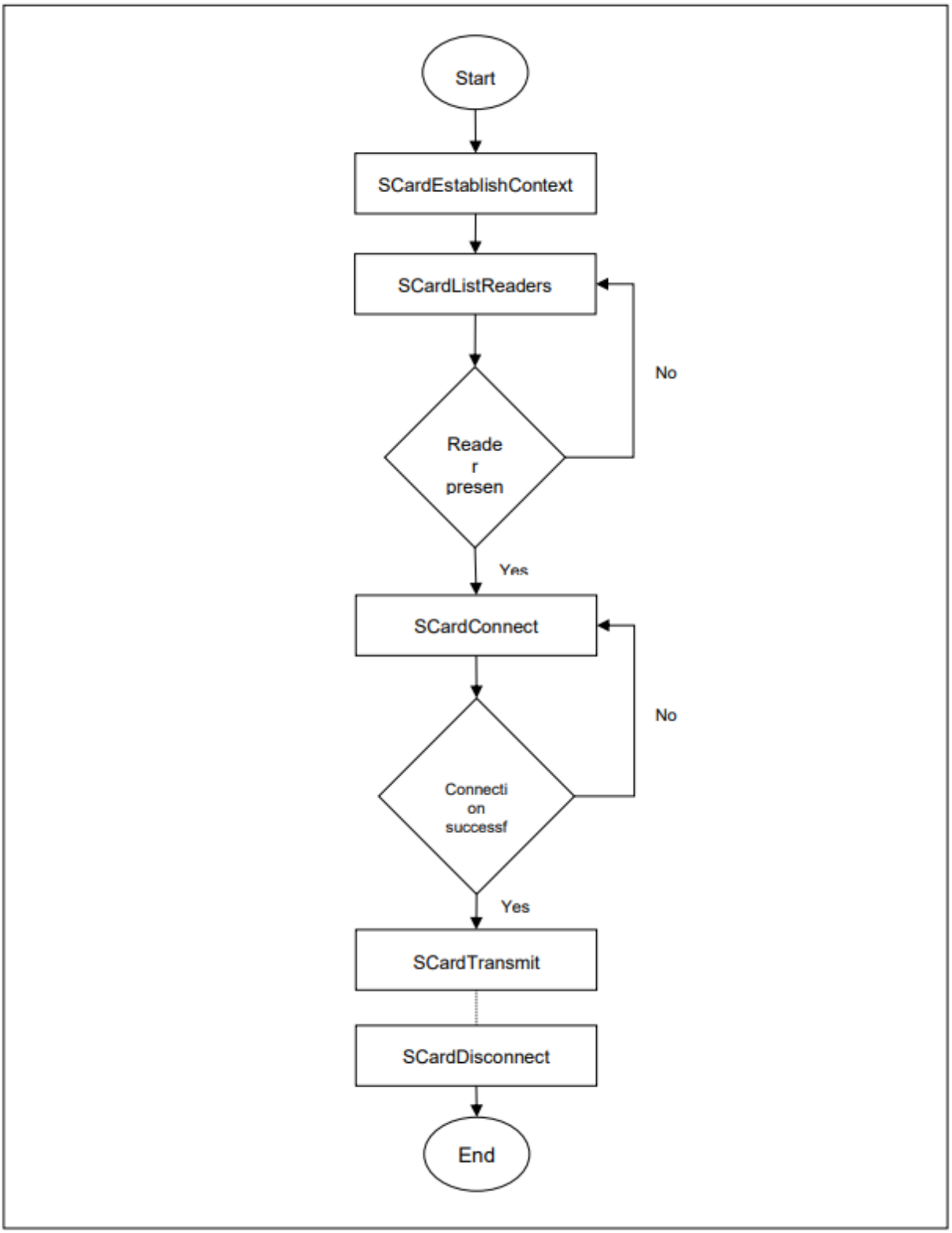
Prema NFC Forumu postoji i više načina rada, ali mi ćemo se fokusirati na ova tri jer su to ona koja su omogućena na Android uređajima i na našem čitaču. Čitač/pisač načina rada omogućuje NFC uređaju da čita i piše podatke sa drugog NFC pasivnog uređaja poput beskontatnih kartica ili NFC naljepnice. P2P način rada omogućuje razmjenu podataka između dva „peer“-a. Emulacija kartice omogućuje NFC uređaju da se prikazuje i ponaša kao beskontaktna kartica. (Google, n.d.). Nama u interesu su čitač/pisač način rada i emulacija kartice .

3.1.2. Programiranje za NFC

Za interakciju između čitača i pametne kartice koristi se format podataka zvan jedinični aplikacijski protokolni podaci (Application Protocol Data Unit) (APDU). Sve kartice koje prate ISO 14443-4 standard za pametne kartice komuniciraju i razumiju APDU poruke. Određuju ga prvih 5 bajta poruke. Prvi bajt je klasa, drugi bajt je instrukcija, treći je prvi paramtera, četvrti je

drugi paratemar i u peti bajt se unosi duljina podataka. Nakon prvih 5 bajta možemo, ali ne moramo, unijeti podatke za slanje kartici („Advanced Card Systems Ltd.“ [acs], 2020.).

U ovome radu će android uređaj emulirati karticu, a čitač će biti u čitač/pisač načinu rada. Programiranje za NFC se ovdje radi na Windows operacijskom sustavu i koristi se biblioteka „winscard.h“. Ona je sučelje za programiranje aplikacija tipa osobno računalo/pametne kartice (PCSC). Za izradu rada pratimo APDU tok. On je definiran u dokumentaciji čitača ACR1252U i prikazan je dijagramom tijeka.



Slika 1. APDU tijek (Izvor: ACS, 2020. str. 18)

Proći ćemo kroz funkcije prikazane u dijagramu tijeka kako bi bolje razumijeli što svaka od njih radi.

Tablica 1: Opis osnovnih funkcija 1

Naziv	Parametri	Izlaz	Opis
SCardEstablishContext	DWORD dwScope, LPCVOID pvReserved1, LPCVOID pvReserved2, LPSCARDCONTEXT phContext	U slučaju uspjeha vraća: SCARD_S_SUCCESS U slučaju neuspjehu nam vraća u hex formatu grešku	Parametri pvReserved1 i 2 nisu upotrijebljeni još i rezervirani su za daljnja unaprijeđenja. Funkcija stvara kontekst pomoću kojeg se definiraju i rade upiti u bazu podataka. Za parametar dwScope možemo postaviti konstante SCARD_SCOPE_USER ili SCARD_SCOPE_SYSTEM ovisno o u kojoj domeni želimo raditi. U parametar phContext stavljamo referencu na koju će se spremati rukovatelj
SCardListReaders	SCARDCONTEXT hContext, LPCSTR mszGroups, LPSTR mszReaders, LPDWORD pcchReaders	U slučaju uspjeha vraća: SCARD_S_SUCCESS U slučaju da čitači ne postoje u traženoj grupi: SCARD_E_NO_READERS_AVAILABLE U slučaju da odabrani čitač nije dostupan: SCARD_E_READER_UNAVAILABLE U slučaju neuspjehu nam vraća u hex formatu grešku	Funkcija u referencu mszReaders sprema nazive svih čitača pronađenih trenutno spojenih u sustav. Funkcija se mora 2 puta pozvati. Prvi puta postavlja se vrijednost za mszReaders na null kako bi se veličina teksta spremila u referencirani pcchReaders. Drugi puta možemo u mszReaders postaviti referencu na polje u koje će se spremati nazivi pronađenih čitača. U mszGroups se postavlja vrijednost null ako želimo ispis svih čitača, inaće koristimo ponuđene konstante za odabir grupa. Ranije stvoren rukovatelj se stavlja na mjesto hContext

SCardConnect	SCARDCONTEXT hContext, LPCSTR szReader, DWORD dwShareMode, LPSCARDHANDLE phCard, DWORD dwPreferredProtocols, LPDWORD pdwActiveProtocol	U slučaju uspjeha vraća: SCARD_S_SUCCESS U slučaju neuspjehu nam vraća u hex formatu grešku U slučaju da nije bilo moguće se spojiti na karticu zbog bilo kojeg razloga: SCARD_E_NOT_READY	Funkcija stvara vezu između aplikacije i NFC uređaja spojenog na čitač i sprema rukovatelj u referencu na rukovatelj phCard Ranije stvoren rukovatelj se stavlja na mjesto hContext U vrijednost szReader stavljamo naziv odabranog čitača dwShareMode se koristi kako bi definirali smiju li druge aplikacije koristiti trenutno spojenu pametnu karticu dwPreferredProtocols definira dopuštene načine prijenos podataka. Dva načina prijenosa podataka su T=0 i T=1. U T=0 protokolu, podaci se šalju znak po znak dok se u T=1 protokolu šalju u blokovima. U pdwActiveProtocol navodimo referencu u koju će se spremiti korišteni protokol
SCardTransmit	SCARDHANDLE hCard, LPCSCARD_IO_REQUEST pioSendPci, LPCBYTE pbSendBuffer, DWORD cbSendLength, LPSCARD_IO_REQUEST pioRecvPci, LPBYTE pbRecvBuffer, LPDWORD pcbRecvLength	U slučaju uspjeha vraća: SCARD_S_SUCCESS U slučaju neuspjehu nam vraća u hex formatu grešku	Funkcija za izvršavanje prijenosa podataka. Ona se ne bude izvršila dok ne dobije odgovor od pametne kartice. pbSendBuffer i cbSendLength su parametri u kojima spremamo podatke koje želimo poslati i veličinu tih podataka pbRecvBuffer i cbRecvLength su parametri u kojima spremamo podatke koje dobijemo te veličinu dobivenih podataka pioSendPci i pioRecvPci su

			reference na korišteni protokol. Tipa su strukture LPCSCARD_IO_REQUEST unutar kojih se nalazi veilčina strukture i korišteni protokol.
SCardDisconnect	SCARDHANDLE hCard, DWORD dwDisposition	U slučaju uspjeha vraća: SCARD_S_SUCCESS U slučaju neuspjehu nam vraća u hex formatu grešku	Funkcija prekida vezu i komunikaciju sa pametnom karticom. U parametar hCard proslijeđujemo rukovatelj karticom U parametar dwDisposition definiramo koja će se akcija odviti nad karticom. Definirane su akcije za ostavljanje kartice na miru, resetiranja(nemam bojlu riječ) kartice, izbacivanja kartice ili gašenja kartice

(Izvor: Microsoft, n.d.)

3.1.3. Primjer programa

Program treba uneseni tekst poslati pametnoj kartici. Napisan je u C++ programskom jeziku. Prvo unosimo tekst te će se taj tekst proslijediti kao APDU [AA, AA, AA, AA, 00, (uneseni tekst)] pametnoj kartici. Pametna kartica će primiti poruku te će pročitati naš tekst ako na sebi prepozna instruckiju navedenu u prvih 4 bytea.

```
#include <iostream>
#include <winscard.h>
#define SCARD_SCOPE_USER 0

int main()
{
    char tekst[15];
    do
    {
        std::cin >> tekst;
    } while (strlen(tekst)>15 || strlen(tekst)==0);


    //handler za context
    SCARDCONTEXT    cards;
    SCARDHANDLE     my_card;
    unsigned long   aktivni_protokol;
    LONG            povratni_kod;
```

```
    LPTSTR          citaci = NULL;
    DWORD           velicina = SCARD_AUTOALLOCATE;

    DWORD   SendLen; // Dužina APDU poruke
    DWORD   RecvLen; // Dužina APDU dobivene poruke
    BYTE    RecieveBuffer[262]; // Mjeste spremanja APDU odgovora

    //Postavljen context
    //ako je postavljen SCOPE_USER onda radimo na razini korisnika, može biti
i SYSTEM

    povratni_kod=SCardEstablishContext(SCARD_SCOPE_USER, NULL, NULL,&cards);
    if (povratni_kod == SCARD_S_SUCCESS)
    {
        //char nazivi_sc_readera[512];

povratni_kod=SCardListReaders(cards,SCARD_ALL_READERS,(LPTSTR)&citaci,
&velicina);

        int input = 0;
        switch(povratni_kod)
        {
            case SCARD_S_SUCCESS:
            do
                {
                    //koristimo  wcout  iz  razloga  što  koristimo  wide
charactere (ilitiga charactere sa UNICODE tablicom)
                    std::wcout << "Postoji citac:" << citaci << std::endl;


                    //SCARD_PROTOCOL_T1 određuje da će se podaci izmjenjivati
u paketima (packet based),
                    //ako bi stavili T0, onda bi se podaci izmjenjivali u
characterima byte po byte (character based)

                  do
                  {
             povratni_kod=SCardConnect(cards,citaci,SCARD_SHARE_EXCLUSIVE,
SCARD_PROTOCOL_T1, &my_card, &aktivni_protokol);

                    } while (povratni_kod != SCARD_S_SUCCESS);

                    if (povratni_kod == SCARD_S_SUCCESS)
                    {


                        BYTE SendCommand[262];
                        //Pokušavam slati poruku bilo kakvu
                        SendCommand[0] = 0xAA;

                        //instruction
                        SendCommand[1] = 0xAA;

                        //parametar 1
                        SendCommand[2] = 0xAA;

                        //parametar 2
                        SendCommand[3] = 0xAA;

                        //length
                        SendCommand[4] = 0xAA;
```

```
                        //Poruka
                        SendCommand[5] = 0x00;
                        for(int i=0; i<strlen(tekst); i++)
                        {
                            SendCommand[i + 6] = tekst[i];
                        }
                        int velina_poruke = 6 + strlen(tekst);
                        povratni_kod = SCardTransmit(my_card, SCARD_PCI_T1,
SendCommand, velina_poruke, NULL, RecieveBuffer, &RecvLen);


                        char primanje[255];
                        for (int i = 0; i < RecvLen; i++)
                        {
                            primanje[i] = RecieveBuffer[i];
                        }

                        primanje[RecvLen] = '\0';
                        std::cout << primanje;
                    }


                    if (povratni_kod == SCARD_E_NOT_READY)
                    {
                        std::cout << "ECARD nije spreman" << std::endl;
                    }

                    SCardDisconnect(my_card, SCARD_LEAVE_CARD);
                    std::cin >> input;

                }while (input != 0);
                SCardFreeMemory(cards, citaci);
                break;


            case SCARD_E_NO_READERS_AVAILABLE:
                std::cout << "Nema citaca";
                break;

            case SCARD_E_READER_UNAVAILABLE:
                std::cout << "Ne mozemo pristupiti citatcu";
                break;
        }
        SCardReleaseContext(cards);
    }
    else
    {
        std::cout << "No world\n";
    }
}
```

3.2. Android

3.2.1. Android općenito

Android je operacijski sustav za mobilne uređaje baziran na Linuxu. Izdan je 2008. godine od Googlea (Tutorialspoint, n.d.) i *open-source* je. U 2022. godini 75% korisnika globalno koristi Android operacijski sustav (David Curry, 2022.). Operacijski sustav se koristi na više vrsta uređaja nego samo za mobitele, no u drugim oblicima. Možemo Android Wear operacijski sustav naći na pametnim satima kao što je LG G Watch ili čak u frižiderima kao Samsungov F9000.

3.2.2. Programiranje za Android

Kako bi programirali Android aplikacije, važno je razumjeti njezinu arhitekturu. Sastoji se od 4 softverska sloja i ona su Linux kernel, biblioteke, aplikacijsko sučelje i aplikacije. Unutar kernela se nalazi *driver* za hardver, sloj biblioteke uključuje sve Java biblioteke specifične za Android, sloj aplikacijskog sučelja nudi Java klase za servise više razine i sloj aplikacije na kojem se instaliravaju naše aplikacije. Aplikacije za Android se pišu u Java programskom jeziku, iako se u novo vrijeme može pisati i u programskom jeziku Kotlin (Tutorialspoint, n.d.)

Android aplikacije imaju 4 komponente koje služe kao ulaz u aplikaciju za korisnika ili sustav: aktivnosti, servisi, prijemnik emitiranja i pružatelj sadržaja.

Aktivnosti služe za interakciju korisnika s aplikacijom. Jedan ekran u Android aplikaciji reprezentira jednu aktivnosti i skup tih aktivnosti surađuje i služi kako bi aplikacija imala bolje korisničko iskustvo. Svaka aktivnost je samostalna i treba biti samostalno napravljena. Zbog načina na koji su napravljene aktivnosti, moguće je istu aktivnost koristiti u različitim aplikacijama uz dopuštenje kao primjerice putem pritiskom tipke „Podijeli“ na Redditu otvara se mogućnost odabira načina ili aplikacije kojom ćemo proslijediti objavu te odabirom WhatsAppa otvara nam se aktivnost WhatsAppa za prosljeđivanje objave. Ako nam je klasa podklasa klase `Activity`, onda je klasa implementirana kao aktivnost (Google, n.d.).

Servisi su ulazne točke za aplikacije koje služe kako bi se dijelovi aplikacije izvršavali dok je aplikacija u pozadini. Oni nemaju korisničko sučelje za interakciju, već se koriste kako bi se mogle izvršavati dugotrajne operacije, obrađivati rad za druge procese ili prikupljati podatke sa interneta kao na primjer prikupljanje novih podataka video igre bez da je otvorena cijelo vrijeme ili sviranje muzike putem Spotify aplikacije bez da je otvorena (Google, n.d.).

Dvije vrste servisa koje postoje su pokrenuti servesi i povezani servisi. Pokrenuti servisi su oni servisi koji postoje sve dokle posao nije završen. Povezani (*bound*) servisi se pokreću

kada druga aplikacija ili sustav ima korist od tog servisa. Ako nam je klasa podklasa klase `Service`, onda je klasa implementirana kao servis. (Google, n.d.)

Prijemnik emitiranja je komponenta koja služi kako bi sustav mogao slati poruke neovisno o trenutnoj aplikaciji. Događaji koji se šalju su neovisni o korisničkom iskustvu te omogućuju da se događaji šalju i od aplikacija koje trenutno nisu pokrenute. Primjer toga je sustavski događaj koji šalje poruku da je baterija skoro prazna. Ti događaji ne moraju se pokazati, ali i mogu putem statusne notifikacije. Ako je nama klasa podklasa klase `BroadcastReciever`, onda je klasa implementirana kao prijemnik emitiranja. (Google, n.d.)

Pružatelj sadržaja daje mogućnost upravljanja podacima koje tvoja aplikacija ima i može spremati neovisno o lokaciji ili načinu spremanja. Druge aplikacije mogu koristiti te iste podatke u slučaju da je dopušteno. Tutorialspoint (n.d.) opisuje način postavljanja upita u pružatelj sadržaja u obliku: `<prefix>://<authority>/<data_type>/<id>` gdje je prefix uvijek „content://“, authority postavlja pružatelja podataka (npr. contacts), data_type opisuje vrstu podataka i id je traženi podatak unutar tipa traženih podataka. (Google, n.d.)

Sve komponente koje se stvaraju nisu odmah prepoznate od android sustava, već se moraju zapisati da postoje kako bi ih sustav pročitao. U tu svrhu služi manifest datoteka koju svaka aplikacija ima. Unutar manifest.xml možemo definirati komponente koje smo napravili i dopustiti aplikaciji da koristi određene dijelove operacijskog sustava ako su dostupne. Može se definirati da aplikacija samo radi ako je verzija androida dovoljna, te se mogu postaviti dopuštenja pristupačnosti servisima. (Google, n.d.) Tu mogućnost ćemo koristiti kako bi omogućili samo android sustavu da se može povezati na servis.

Za izradu ovog rada potreban nam je samo jedan od ovih komponenti, a to je servis. Mobilni uređaj će služiti kao emulirana kartica te u te svrhe koristimo „*Host-based card emulation*“ (HCE) servis.

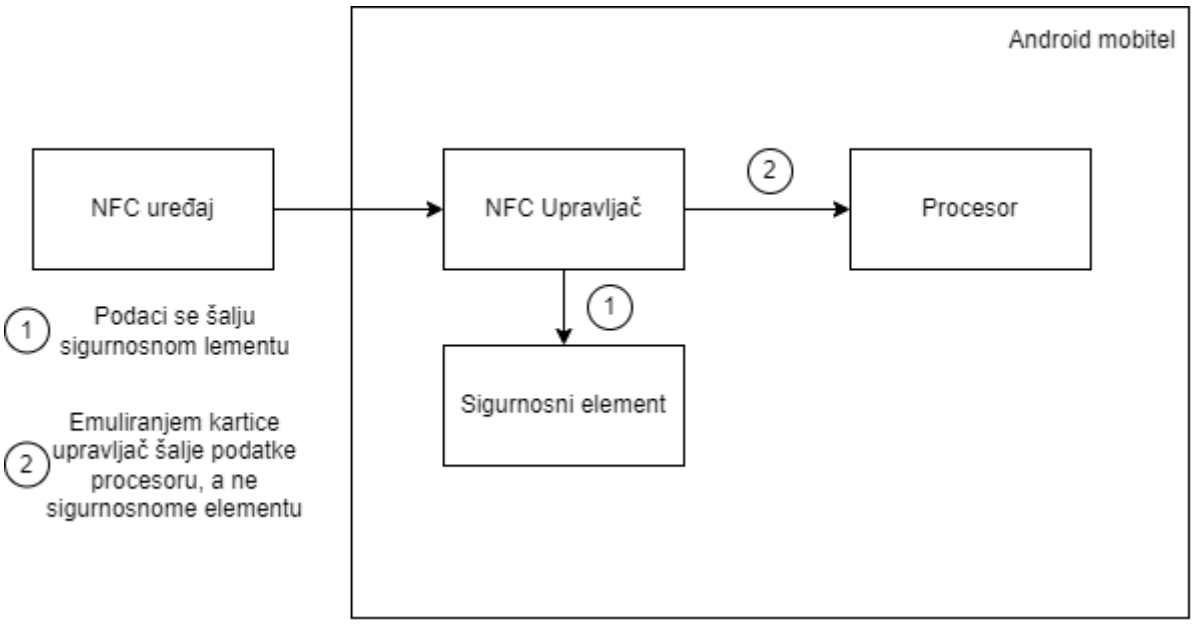
3.2.3. HCE

Emuliranje kartice na android uređajima je bilo inicijalno izvođeno putem čipa u uređaju zvan sigurnosni element. Direktan pristup tom čipu nemamo. Možemo samo raditi upite u njegu da vidimo status transkacija. Kasnije je omogućen način emuliranja kartice putem aplikacija.Takvu vrstu emulacije zovemo emuliranje kartice na temelju domaćina (Host-based Card Emulation)(HCE).

Uspoređivanjem procesa izvođenja NFC transakcija između korištenja HCE i sigurnosnog elementa možemo vidjeti kako i zašto koristimo HCE. U procesima se nalaze 4 elementa:

- NFC čitač
- NFC upravljač
- Procesor
- Sigurnosni element

Kod emuliranja kartice pomoću sigurnosnog elementa, podaci se šalju u sigurnosni element i definira se koja se kartica emulira. Prilikom prislanjanja uređaja na čitač, NFC upravljač prosljeđuje podatke sigurnosnom elementu. Kada to radimo pomoću HCE, podaci prilikom transakcije šalju se direktno procesoru i onda ih možemo koristiti za izradu aplikacije. Sigurnosni element se ne koristi u tom slučaju.



Slika 2: Prikaz razlike rada između emuliranja kartice putem sigurnosnog elementa i na temelju domaćina (Prema: Google, n.d.)

HCE servisi su servisi koji su identificirani svojim aplikacijskim identifikatorom (Application Identifier) (AID). AID je definiran sa 16 bajtova i unosimo ga u manifest. Služi kako bi sustav identificirao kojom aplikacijom mora komunicirati pri NFC transkaciji. Zadajemo ga sami iz razloga što stvaramo vlastitu infrastrukturu između čitača i uređaja. Moguće je definirati ih u grupe kako bi više aplikacijskih identifikatora mogli voditi prema jednoj aplikaciji. Ne smiju postojati dvije aplikacije koje koriste isti identifikator. Kada prislonimo Android mobilni uređaj na naš NFC čitač, šalju se APDU poruke te se spajamo na naš servis i daljnje operacije se mogu izvrašavati i poruke mogu biti poslane.

Kako bi definirali da servis koristi HCE, moramo napraviti klasu za servis koji proširuje klasu `HostApduService`. Unutar nje postoje dvije apstraktne metode koje se moraju implementirati zvane `processCommandApdu` i `OnDeactivate`. Unutar `processCommandApdu` prihvaćamo poslani APDU i izvršavamo određene funkcije koje mi želimo. Metoda `OnDeactivate` se poziva kada se makne NFC uređaj sa čitača ili kada se na čitaču pozove funkcija `ScardDisconnect`. Kao što je ranije prikazano kod čitača, kada čitač pošalje APDU mobitelu, očekuje se da bude android mobitel vratio natrag bilo kakav odgovor. Transakcije se odvijaju serijski, a ne paralelno. (Google n.d.)

3.3. Interakcija između čitača i Android uređaja

Poruke između dvaju uređaja se šalju u obliku APDU poruka. Oni su polu-duplex strukture što znači da jedan NFC uređaj (u našem slučaju čitač) pošalje APDU android uređaju, HCE servis pročita APDU i vraća jedan APDU (Google, n.d.). Prvi APDU koji je poslan Android uređaju je `SELECT AID` APDU koji je opisan u ISO 7816-4 specifikaciji. Nakon odabranog HCE servisa, možemo početi stvarati set instrukcija. Za primjer možemo koristiti programski kod napisan u točki 3.1.3 uz dodatak poslanog `SELECT AID` APDU-a. Unutar tog koda je definiran i poslan APDU: [AA, AA, AA, AA, 00, (uneseni tekst)]. Ako bi htjeli da mobilni uređaji može prepoznati tu instrukciju i da npr. se taj tekst pošalje u log, moram na mobilnom uređaju prvo stvoriti aplikaciju, zatim stvoriti HCE servis, zadati mu jedinstven AID (na kojega se čitač poziva) i unutar `processCommandApdu` metode reći servisu što da napravi kada susretne tu specifičnu bajtnu sekvencu. Sljedeći kod prikazuje implementaciju navedenog scenarija.

```
public class hceClass extends HostApduService {
    public byte[] response;

@Override
public byte[] processCommandApdu(byte[] apdu, Bundle extras) {
    Integer velicina=apdu.length;
      //zadana bajtna sekvenca
    if(apdu[0]==(byte)0xAA && apdu[1]==(byte)0xAA && apdu[2]==(byte)0xAA &&
apdu[3]==(byte)0xAA)
    {
        byte[] tekst=Arrays.copyOfRange(apdu,5,velicina);
        String s = new String(tekst, StandardCharsets.UTF_8);
        Log.d("Tekst koji je dosao: ", s.toString());
        String odgovor = "Vracam poruku";
        response=odgovor.getBytes();
    }
    sendResponseApdu(response);
    return response;
}

@Override
public void onDeactivated(int reason) {
```

```
    Log.d("onDeactivated: ","Gasi poruka");
    }
}
```

U ranije navedenom C++ kodu će se poruka poslana od strane mobitela spremiti u varijablu primanje i ispisati. U pravilu bi odgovorni APDU prema ISO 7816-4 standardu trebao izgleda kao sljedeće: [(odgovor), SW1, SW2] pri ćemu su SW1 i SW2 statusni bajtovi. Oni se u jednom djelu krajnjeg koda rada koriste za provjeru da li postoji aplikacija na mobitelima

4. Računalna sigurnost

Dvorazinsku autentifikaciju smo ranije definirali kao dodatan sloj sigurnosti za identificiranje pojedinca. Lozinke i ključeve koje unosimo nikada ne smiju biti spremljeni, preneseni ili korišteni u čistome obliku zato što time ostavljamo rupu za potencijalni napad. Kako bi izbjegli propust u sigurnosti, moramo analizirati načine kako se kriptiranja podataka i kako očuvati i osigurati prijenos poruka između naša 2 uređaja. U ovom dijelu biti će riječi o računalnoj sigurnosti i kriptiranju podataka uz objašnjenje simetričnog i asimetričnog načina kriptiranja.

4.1. Općenito o računalnoj sigurnosti

Računalna ili informacijska sigurnost je nastojanje održavanja zaštite računala i sve povezano s računalom. Čuva hardware, software i informacije pohranjene u sustavu (Russel i sur., 1991, str. 8). Sigurnost informacijskih sustava štiti računala ne samo od vanjskih napadača koji žele učiniti štetu, već i od slučajnih ili namjernih grešaka korisnika poput dijeljenja lozinke, zanemarivanja važnosti backupa ili instaliranja zlonamjernog softvera (Russel i sur., 1991, str. 8).

Postoje brojne metode kojima se može narušiti sigurnost podataka. Russel i sur. (1991, str. 12) ističu neke od najčešćih ranjivosti računala. Tako navode fizičku ranjivost koja se odnosi na ranjivost prostora u kojem se nalazi naš uređaj gdje je najveća opasnost krađa uređaja, navode prirodnu ranjivost koja se odnosi na prijetnje okoliša i ljudsku ranjivost gdje je čovjek iliti zaposlenik koji može namjerno ili nenamjerno napraviti štetu. Postoji ranjivost hardvera i softvera koja se prvenstveno odnosi na metode kojima neautorizirana strana napada naš sustav.

U ovom radu prvenstveno će biti riječi o prijenosu i verifikaciji podataka zbog načina prijenosa podataka i vrsti podataka (u našem slučaju ključ). Povreda koja nastane prilikom narušavanja informacijske sigurnosti naziva se povreda podataka.

Povreda podataka želi se spriječiti zbog velikog broja razloga. Razmotrimo slučaj u kojem smo napravili aplikaciju koju neki korisnik koristi na dnevnoj bazi i unosi svoje privatne podatke. Nakon nekog vremena korištenja aplikacije, dogodi se krađa njegovih podataka od neautorizirane strane. Autorizirane strane su one koje su prošle proces dobivanja pristupa informacijskim resursima (Željko Panian, str 45.). U slučaju neautoriziranog prisupa, korisnik će se osjećati nesigurno dalje koristiti našu aplikaciju i vijest upada u aplikaciju proširila bi se na druge korisnike. Web trgovine znaju tražiti podatke o adresi i druge osobne podatke. Slično

vrijedi i za organizaciju koja koristi takvu aplikaciju, uz opasnost prekida rada tvrtke kako bi se mogle popraviti štete. Russel i sur. (1991, str. 20) tvrde da su informacije kojima se koristimo prilikom posla nužne za naše poslovanje i da svaka krađa i kompromis velika šteta bez obzira na područje djelovanja povezane organizacije.

Osim rečenoga, Russel i sur. (1991, str. 19) ističe i državne zahtjeve kao jedan od glavnih razloga implementacije sigurnosti u sustav. Državni zahtjevi su jedan od razloga implementacije sigurnosti u tehnologiju iz razloga što države za nove tehnologije navode sigurnosne zahtjeve uz operativne zahtjeve koje one trebaju ispunjavati kako bi se mogle staviti na tržište. Za to postoje i brojni standardi (Russel i sur., 1991, str. 19).

Računalna i informacijska sigurnost se može postići raznim metodama, no za naš rad koristit ćemo se kriptiranjem i šifriranjem podataka kao način postizanja sigurnosti rada. Bitni podaci će se prenositi putem NFC tehnologije i oni će morati biti enkriptirani tijekom prijenosa, dekriptiani i šifrirani kada stignu na odredište.

4.2. Kriptiranje podataka

Kriptografija omogućuje „maskiranje“ podataka od pošiljatelja do primatelja kako neautorizirana strana ne bi mogla dobiti informacije od podataka pri čemu primatelj može vidjeti dobivenu poruku u originalnom obliku (Kurose i Ross, 2010, str. 625). Pošiljatelj šalje poruku koju primatelj dobiva te procesom dekriptiranja mora biti jednaka. U slučaju da postoji napadač koji u prijenosu nekako pročita podatke, njemu ne smiju biti jasni podaci. Prema Russelu i sur. (1991, str. 65), kriptiranje omogućuje transformaciju originalne informacije u naizgled promijenjenu informaciju koja uobičajeno izgleda kao skup nasumičnih znakova.

Poruka u originalnom obliku se naziva plaintext, dok se šifrirana poruka naziva cyphertext (Kurose i Ross, 2010, str. 625). Plaintext se kriptira određenim algoritmom enkripcije as nekim ključem kako bi se dobio *cyphertext*. *Cyphertext* je šifrirana poruka koja je nečitljiva i stoga ne nosi nikakvu informaciju. Kako bi primatelj poruke originalnu poruku, ona se mora dekriptirati. Dekriptiranje je suprotan proces od kriptiranja te i on zahtjeva ključ za algoritam dekriptiranja (Kurose i Ross, 2010, str. 625).

Ključ je znakovni niz korišten u algoritmu kriptiranja i dekriptiranja zbog činjenice da je svaka kriptografska tehnika danas standardizirana te bi bez nekog ulaza u algoritam svaki prolaz iste poruke kroz algoritam rezultirao u istom izlazu (Kurose i Ross, str. 625). Postoje dvije vrste ključa korištenih u kriptiranju i to su simetrični i asimetrični ključevi koji tvore dvije različite vrste kriptiranja.

Aplikacija na računalu je izrađena u .NET Frameworku i koristi .NET model kriptografije (*.NET cryptography model*). U njemu su definirane i implementirane klase i metode putem kojih se mogu izvoditi kriptografske operacije.

Razlikujemo dva načina kriptiranja: simetrično i asimetrično. Simetrično kriptiranje korisno je za šifriranje velikih količina podataka, a asimetrično se izvodi na malom broju bajtova te se zbog toga koristi samo za male količine podataka (Microsoft, 2022).

4.2.1. Simetrično kriptiranje

Simetrično kriptiranje (kriptiranje tajnim ključem) je vrsta kriptiranja koja za kriptiranje i dekriptiranje koristi isti tajni ključ (Kurose i Ross, 2010, str. 626). Ako osobe X i osobe Y žele komunicirati putem simetričnog kriptiranja, osoba X će kriptirati svoj plaintext ključem koji osoba Y mora znati kako bi šifrirana poruka bila dešifrirana. Kako bi to bilo moguće, sudjelovatelji trebaju dogovoriti zajednički ključ kojim će se poruka i kriptirati i dekriptirati (Kurose i Ross, 2010, str. 632). Kako ni jedna neautorizirana ne bi zloupotrijebila informacije razmijenjene prilikom komunikacije, taj ključ treba biti tajan.

Osoba X poruku kriptira nekim od simetričnih algoritma kriptiranja tajnim ključem te poruku šalje osobi Y, koji taj isti tajni ključ koristi kako bi dekriptirao sadržaj poslane poruke. Neki od poznatijih simetričnih algoritma kriptiranja su DES, AES i 3DES (Kurose i Ross, 2010, str. 630). Svaki od tih algoritama temelji se na ulančavanju šifriranih blokova (Cypher Block Chaining). Ono funkcionira na način da se uzima nasumičan string koji se naziva inicijalizacijski vektor (IV) uz prvi blok poruke koja se kriptira, te svaki sljedeći blok koji se računa koristi prijašnji blok za daljnje kriptiranje (Kurose i Ross, 2010, str. 631).

4.2.2. Asimetrično kriptiranje

Asimetrično kriptiranje je način kriptiranja i dekriptiranja koja koristi dva različita ključa, jedan javni i jedan privatni. Diffie i Hellman su 1976. godine demonstrirali algoritam zvan Diffie-Hellman razmjena ključeva (*Diffie-Hellman Key Exchange*) koji je temeljen na parovima javnog i privatnog ključa (Kurose i Ross, 2010, str. 632).

U asimetričnom kriptiranju, osoba X i osoba Y poruke ne koriste jedan zajednički tajni ključ, već svaki od njih imas svoj javni i privatni ključ koji su međusobno matematički povezani, ali iz jednog ključa nije moguće dobiti drugi (Kurose i Ross, 2010, str. 633). Javni ključ je dostupan svima, a privatni je dostupan samo vlasniku ključa (Kurose i Ross, str. 633).

Kako bi osoba X osobi Y u poslao poruku koja je nečitljiva u prijenosu, on mora saznati javni kluč osobe Y. Osoba X kriptira poruku koristeći dobiveni javni ključ te je šalje osobi Y. Proces kriptiranja se izvodi pomoću algoritmi za kriptiranje. Osoba Y dobiva poruku koju je

osoba X poslala te ju za dekriptira uz svoj privatni ključ i algoritam za dekriptiranje kako bi dobio originalni tekst (Kurose i Ross, 2010, str. 633). Osoba X pritom ne koristi ni jedan od svojih ključeva, već samo ključeve osobe Y.

Najpoznatiji algoritam za asimetrično kriptiranje je RSA algoritam, koji za kriptiranje, dekriptiranje i generiranje ključeva koristi aritmetičke operacije uz module (Kurose i Ross, 2010, str. 634). U radu će se za prijenos podataka koristiti RSA algoritam koji je implementiran u .NET-u u klasi RSA.

4.2.2.1. Opis generiranja RSA ključeva

Za sigurnost je važno znati kako stvari rade u pozadini. Prije nego li prikažemo programski kod za geneiranje ključeva, prvo ćemo prikazati matematički postupak (Kurose i Ross, 2010, str. 635).

Prvo se navode dva prosta broja prostiBroj1 i prostiBroj2

1. Zatim računamo $n = prostiBroj1 * prostiBroj2$ i $z = (prostiBroj1 - 1) * (prostiBroj2 - 1)$
2. Odabire se broj e koji je manji od n i koji je relativno prost sa z
3. Nalazi se broj d takav da $e * d - 1$ ima rezultat operacije koji podijeljen sa ranije izračunatim z ima ostatak dijeljenja 0

Time smo dobili javni i privatni ključ. Javni ključ se sastoji od varijabli n i e, a privatni od varijabli n i d.

4.2.2.2. Implementacija asimetričnog kriptiranja

Kada radimo u .NET okruženju, za implementaciju asimetričnog kriptiranja koristimo RSA klasu. Pozivom metode RSA.Create() stvaraju se par privatnog i javnog ključa. U slučaju da bi im htjeli pristupiti, možemo stvoriti varijablu tipa RSAParameters koja će dobiti informacije o ključevima putem metode ExportParameters(bool ukljuciPrivatniKljuc) RSA klase. U slučaju da stavimo za parametar ukljuciPrivatniKljuc vrijednost true, onda će se u RSAParameters varijablu spremiti vrijednost privatnoga i javnoga ključa. Postavljanjem te vrijednosti na false vraćaju nam se samo vrijedosti javnog ključa. Drugi način dobivanja informacija je u XML obliku. Unutar RSA klase postoji metoda ToXMLString(bool ukljuciPrivatniKljuc) koja stvara i vraća vrijednosti ključeva u XML obliku. Isto kao i kod ExportParameters metode, ako ovisno o vrijednosti koju stavimo u paramtera, dobiti ćemo van i privatni i javni ključ ili samo javni.

Za proces kriptiranja i dekriptiranja koristimo metode RSA klase Encrypt(byte[] data, RSAEncryptionPadding padding) i Decrypt(byte[] data, RSAEncryptionPadding padding)). Podatke koje želimo kriptirati moramo prvo spremiti u polje bajtova ako već nisu.

RSAEncryptionPadding parametar traži jedan način „ispunjavanja“ podataka za veću sigurnost.

Sljedeći kod će stvoriti ključ, ispisati javni ključ u XML formatu, kriptirati i dekriptirati nekakav tekst pri čemu će ispisati plaintext, ciphertext i na kraju dekriptirani plaintext. Kod je napisan u C# sa .NET Frameworkom.

```
static void Main(string[] args)
        {
            RSA rsa = RSA.Create();
            string pKey=rsa.ToXmlString(false);
            Console.WriteLine(pKey + "\n ------------------------------
\n");
            string plaintext = "Ovo je tekst";
            Console.WriteLine(plaintext + "\n ----------------------------
- \n");
            byte[] plaintextInBytes = Encoding.ASCII.GetBytes(plaintext);
            byte[] cipherText = rsa.Encrypt(plaintextInBytes,
RSAEncryptionPadding.Pkcs1);
            Console.WriteLine(Encoding.ASCII.GetString(cipherText) + "\n --
---------------------------------- \n");
            byte[] podatak=rsa.Decrypt(cipherText,
RSAEncryptionPadding.Pkcs1);
            Console.WriteLine(Encoding.ASCII.GetString(podatak));
            Console.ReadLine();
        }
```

4.3. Šifriranje

Rijetko se u baze podataka ili lokalno spremaju čisti podaci, već se spremaju šifrirani podaci. Šifrirani podaci su „maskirani“ podaci (Maleković i Rabuzin, 2016, str. 153). *Hash* funkcije kao što su SHA256 jednosmjerno maskiraju podatke i uvijek daju isti rezultat ako je unesena ista vrijednost za maskirati. Sljedeći programski kod stvara instancu željenog algoritma, šifrira tekst te ga i ispisuje takvoga u konzolu:

```
static void Main(string[] args)
        {
SHA256 instancaAlgortima = SHA256.Create();
            byte[] passInBytes =
instancaAlgortima.ComputeHash(Encoding.UTF8.GetBytes(“Nekakav uneseni
tekst”));
            StringBuilder builder = new StringBuilder();
            for (int i = 0; i < passInBytes.Length; i++)
                builder.Append(passInBytes[i].ToString("x2"));
            Console.WriteLine(builder.ToString());
            Console.ReadLine();
        }
```

5. Implementacija dvorazinske autentifikacije

Tijekom rada je opisana svaka potreban komponenta kako bi se mogla isplanirati i napraviti implementacija heterogene dvorazinske autentifikacije. U ovom poglavlju ću opisati ideju implementacije putem dijagrama i crteža, moguće korisničke interakcije te definirane klase i metode.

5.1. Ideja implementacije dvorazinske autentifikacije

Ideja rada je implementirati dvorazinsku autentifikaciju. Kako bi to napravili potrebna su nam dva dokaza. U slučaju ovog rada se koriste lozinka i mobilni uređaj kao dokazi identiteta. U ovom poglavlju ćemo proći ideju prijave, registracije i zaboravljene lozinke.

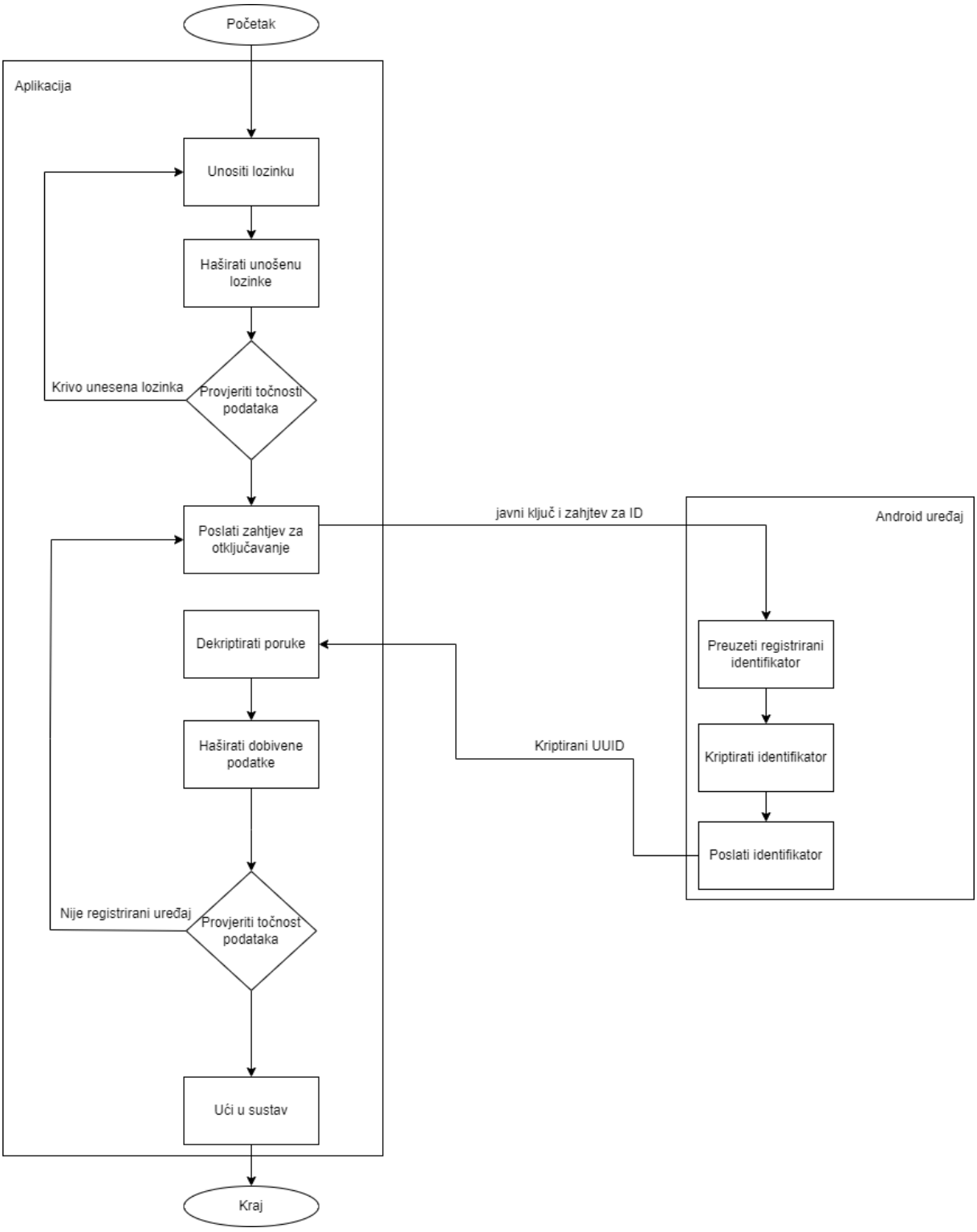
Rad služi kako bi se prikazala implementacija dvorazinske autentifikacije što znači da bi se implementacija mogla koristiti kao ulaz u sustave/programe uz određene promjene. Registracija uključuje unos lozinke i prislanjanje mobilnog uređaja koji će se registrirati na računalo. Ona se u teoriji vrši samo jednom. Prijava služi kako bi se osoba prijavila u neki sustav. Prvi dokaz za autentifikaciju je lozinka koja se unosi i provjerava se sa spremljenom lozinkom te se potom provjerava mobilni uređaj koji se prislanja na NFC čitač. Ako obje razine prođu, korisnici imaju pristup sustavu.

Ideja zaboravljene lozinke postoji zbog ljudskih grešaka ili promjena. Ako osoba zaboravi svoju lozinku ili izgubi mobilni uređaj, on više nema pristup svojem računu odnosno aplikaciji. U tu svrhu se postavlja sistem kojim se može zamijeniti lozinka ili potrebni android mobilni uređaj za ulaz u sustav. Taj sustav je zaključan svojom vlastitom lozinkom koji je odgovor na pitanje postavljen tijekom registracije. U slučaju da osoba točno odgovori na pitanje na isti način kao što je i prije, dobiva mogućnost ponovne registracije svojih podataka. U sljedećem poglavlju prolazimo kroz svaki slučaj korištenja i što se događa u aplikaciji.

5.2. Slučaji korištenja i implementacija

Prikaz slučaja korištenja je istovremeno i prikaz implementacije rada. Svaki slučaj je detaljno raspisan te putem dijagrama tokova možemo vizualno vidjeti kako aplikacija radi. Proći ćemo kroz 3 slučaja korištenja koji su prijava, registracija i zaboravljena lozinka.

Prvi slučaj koji ćemo analizirati je prijava. Kada se prijavljujemo u sustav, prvo unosimo lozinku te se ona šifrira i provjera jednakost sa postojećim podacima. U slučaju da je krivo unesena lozinka, korisnika se obavijesti o krivo unesenoj lozinki. U slučaju da je točno unesena lozinka, mora se provjeriti drugi dokaz koji je točni mobilni uređaj. Prislanjanjem mobilnog uređaja na čitač, šalje se zahtjev za otključavanje koji sadrži javni ključ i zahtjev za identifikator mobitelu te mobitel vraća kriptiranu poruku u kojoj se nalazi identifikator. Potom aplikacija dekriptira poruku, šifrira dobivenu vrijednost i uspoređuje ju sa postojećim podacima. U slučaju da su podaci isti, ulaz u sustav je dopušten. U slučaju da nije točan identifikator, ulaz u sustav nije dopušten. Scenarij je prikazan dijagramom toka.

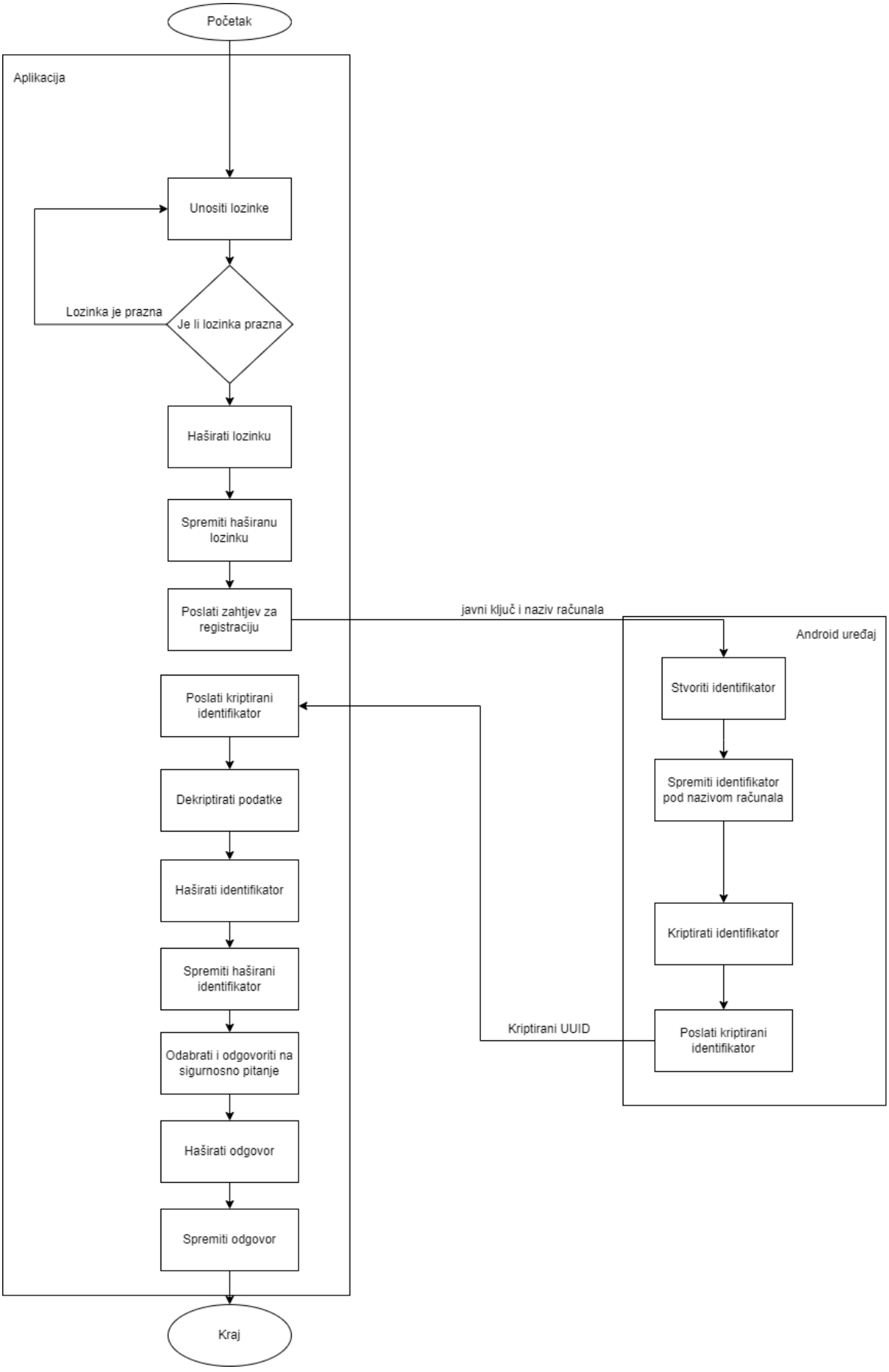


Slika 3: Dijagram tijeka prijave u sustav

Drugi slučaj jest registracija. Tijekom registracija moramo prvo unijeti lozinku. Jedino pravilo jest da lozinka ne smije biti prazna. Lozinka se šifrira i sprema se na računalo. Dalje se registrira mobilni uređaj. Tijekom registracije aplikacija šalje javni ključ i naziv računala. Na mobilnom uređaju se stvara se univerzali jedinstveni identifikator (UUID) i njegova vrijednost

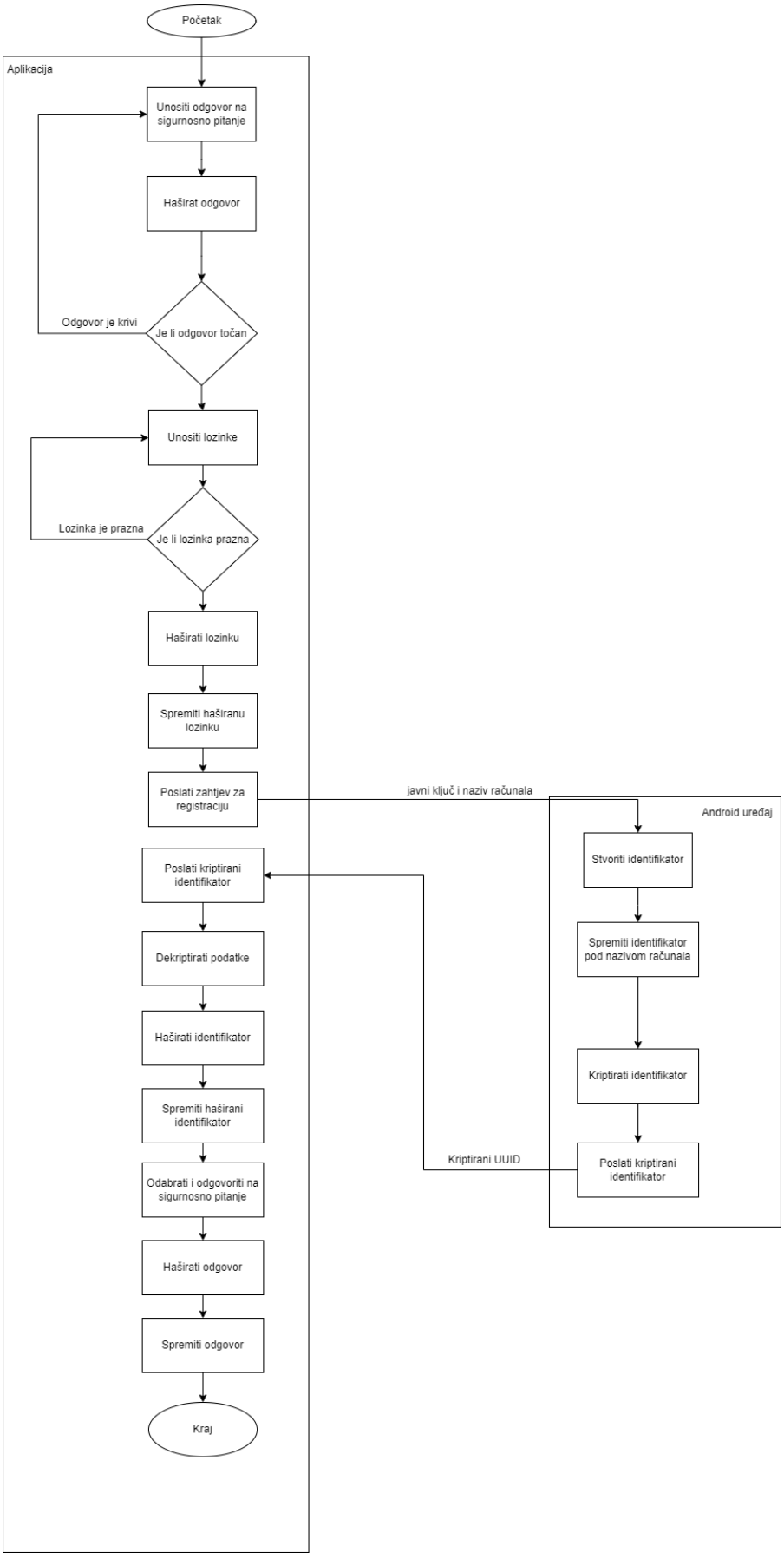
se sprema u sustav u SharedPreference sustavu, specifično EncryptedSharedPrefrence koji je kriptiran pomoću AES256 algoritma. UUID je 128 btni pseudo slučajan broj kojem su šanse da postoji duplikat dovoljno blizu broju 0 (H2, n.d.). Vrijednost UUID-a se kriptira putem RSA algoritma i javnog ključa koji je poslan uređaju te se šalje računalu kao APDU odgovor. Računalo preuzima kriptirani UUID, dekriptira ga, šifrira ga i sprema ga na računalo. Zadnji korak jest postavljanje sigurnosnog pitanja u slučaju izgubljenog uređaja ili zaboravljene lozinke. Odabire se jedna od ponuđenih pitanja, odgovara se na pitanje te se odgovor šifrira i sprema na uređaj. Time je proces registracije završen i korisnik se može prijaviti u sustav.

Napomena koja se mora naglasiti. Postoji propust u spremanju u EncryptedSharedPreference. Kako bi mogli konzistentno doći do UUID koji je spremljen u mobitelu, moramo postaviti ime ključa gdje se spremaju podaci. U slučaju da hardkodiramo ime onda se pri sljedećoj registraciji piše preko staroga ključa. U tom slučaju bi mobilni uređaj mogli jednom registrirati na samo jedan uređaj i potencijalno rješenje jest da se pošalje naziv računala u APDU i da se ključ nazove prema računalu. Potencijalni problem s time jest ako se dva računala isto zovu, nailazimo na isti problem. To rješenje čini se pogodno za izradu ovoga rada.



Slika 4: Dijagram tijeka registracije

Treći slučaj se koristi u slučaju da je potrebno zamijeniti mobilni uređaj ili da je lozinka za ulaz u sustav zaboravljen. Postavlja se odabir sigurnosnog pitanja te se traži i odgovor na to pitanje. Odgovor se šifrira i uspoređuje se sa odgovorom koji se nalazi računalu. U slučaju da su jednake, otvara se mogućnost ponovne registracije. Z



Slika 5: Dijagram tijeka procesa vraćanja računa

5.3. Klase i metode

Implementacija uključuje programiranje za dva različita uređaja i dva različita programska jezika. Programiranje na strani čitača i aplikacije se vrši u C# programskom jeziku koristeći .NET Framework i WinForms za sučelje aplikacije i Visual Studio IDE. Za programiranje na strani Android sustava koristimo Java programski jezik i Android Studio IDE. Prvo ćemo proći klase i metode implementirane na strani aplikacije te nakon toga na strani mobitela.

5.3.1. Klase i metode na strani aplikacije

Na strani aplikacije su implementirane dvije klase, dvije forme i tri korisničke kontrole. Dvije klase implementirane su NFCKomande i RSAImplement. Unutar NFCKomande su implementirane sve metode potrebne za interakciju sa NFC uređajem. Unutar RSAImplement se nalaze metode i instanca klase RSA i služi za dekriptiranje podataka dobivenih od mobitela i preuzimanje javnog ključa za enkripciju. Forme i korisničke kontrole ćemo prikazati pomoću slika sučelja. Kako niti jedna metoda nema parametre, zapisat ćemo samo povratne tipove.

Tablica 2: metode klase NFCKomande

Naziv	Povratni tip	Opis
SpajanjeNaKarticu	bool	Metoda pokreće petlju čiji je uvjet uspješno izvđenje funkcije SCardConnect
PokreniContext	bool	Metoda pokreće funkciju SCardEstablishContext i vraća rezultat true ili false ovisno o rezultatu
IspisCitaca	bool	Metoda pokreće funkciju SCardListReadersA i postavlja vrijednost čitača u atribut klase. Vraća vrijednost true ako je uspješno funkcije izvedena, inače false
SpajanjeNaAplikaciju	bool	Metoda koja koristi SCardTransmit funkciju kako bi poslala SELECT AID APDU mobilnom uređaju. AID je kodiran u samu funkciju. U slučaju

		uspješne transakcije, vraća se true i možemo dalje izvoditi druge NFC operacije. U slučaju da je odgovor dobiven 0x6A i 0x82, onda znamo da potrebna aplikacije nije instalirana na mobilni uređaj i vraća false. Ako se dogodila greška u prijenosu, SCardTransmit će imati povratni odgovor različit od SCARD_S_SUCCESS i metoda vraća false
RegistrirajUredaj	byte[]	Metoda šalje na mobilni uređaj APDU [0xAA, 0xAC, 0x01, 0x02, 0x00, (javni ključ)]. U slučaju uspjeha će vratiti novi dekriptirani UUID, inače vraća null
VratiUUID	byte[]	Metoda šalje na mobilni uređaj APDU [0xAA, 0xAD, 0x01, 0x02, 0x00, (javni ključ)]. U slučaju uspjeha će vratiti dekriptirani UUID, inače vraća null
PosaljiJavniKljuc	byte[]	Metoda šalje na mobilni uređaj APDU [0xAA, 0xAB, 0x01, 0x02, 0x00, (javni ključ)]. Svrha ove metode je da mobilni uređaj pošalje bilo kakav odgovor. U slučaju uspjeha će vratiti dekriptirana poruka, inače vraća null

Tablica 3: atributi klase NFCKomande

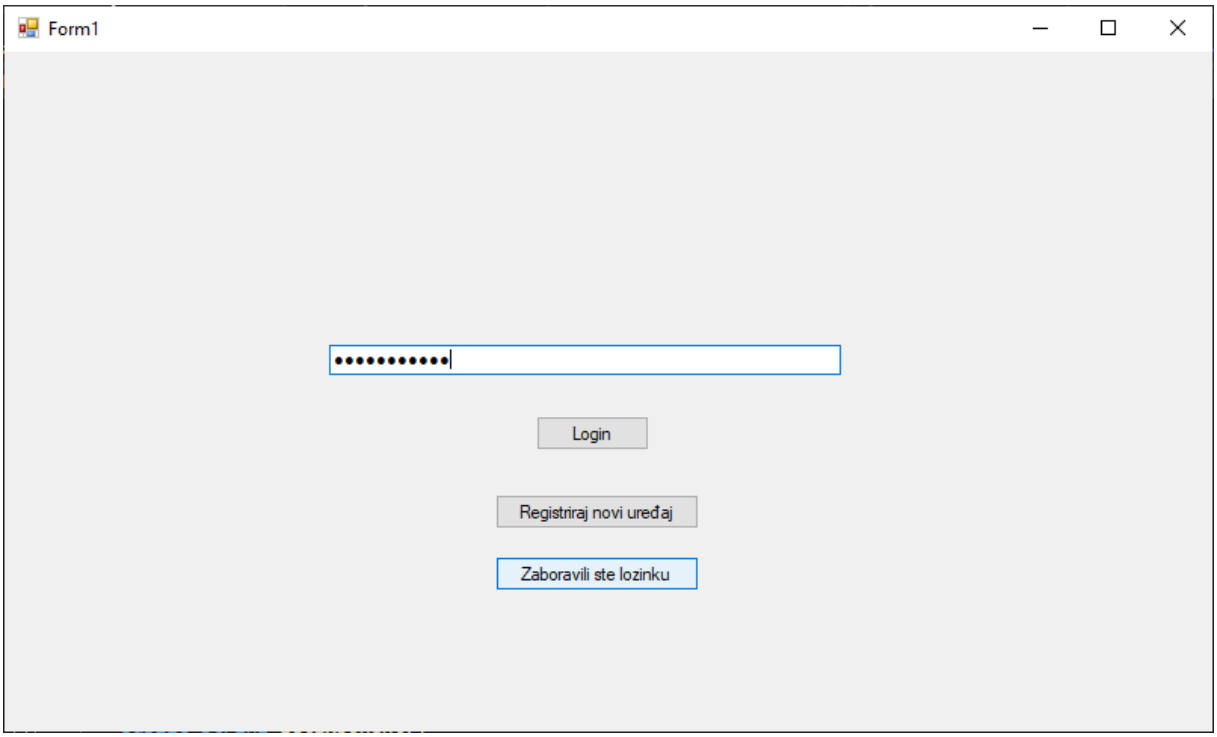
Naziv	Tip podatka
ContextHandle	IntPtr
CardHandle	IntPtr
AktivniProtokol	IntPtr
OdabraniCitac	string
PovratniKod	int
kljuc	string
RSAImplementacija	RSAImplement

Tablica 4: metode klase RSAImplement

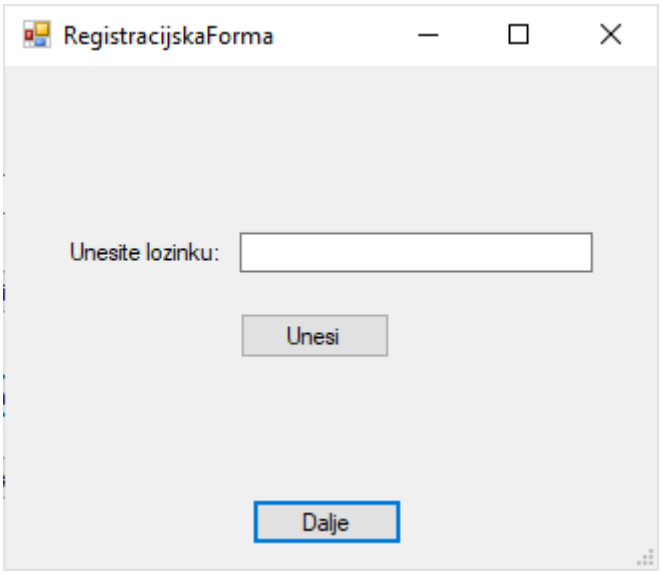
Naziv	Parametri	Povratni tip	Opis
getPublicKey	-	string	Metoda koja iz atributa rsa vraća javni dio ključa u obliku XML stringa
getModulus	-	string	Metoda koja iz atributa rsa vraća modulus dio javnog ključa u obliku stringa
getExponent	-	string	Metoda koja iz atributa rsa vraća eksponent dio javnog ključa
DecryptPoziv	cipherText	byte[]	Metoda koja dekriptira dolazeću poruku te ju vraća

Tablica 5: atributi klase RSAImplement

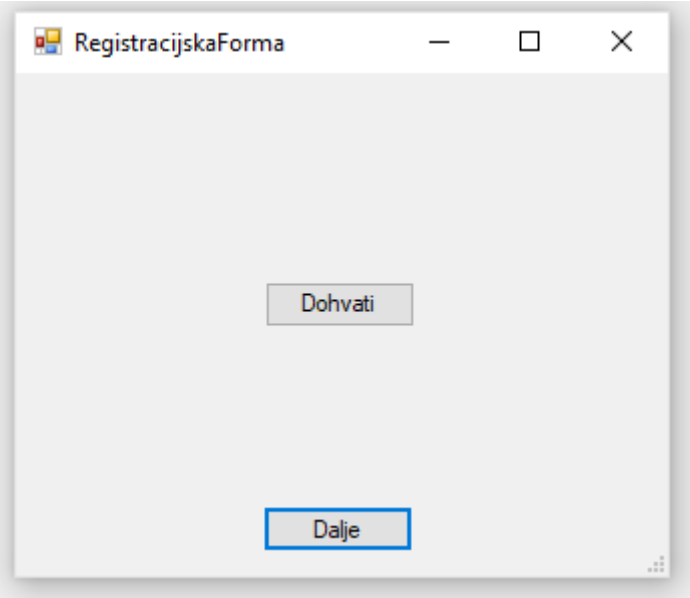
Naziv	Tip podatka
rsa	RSA



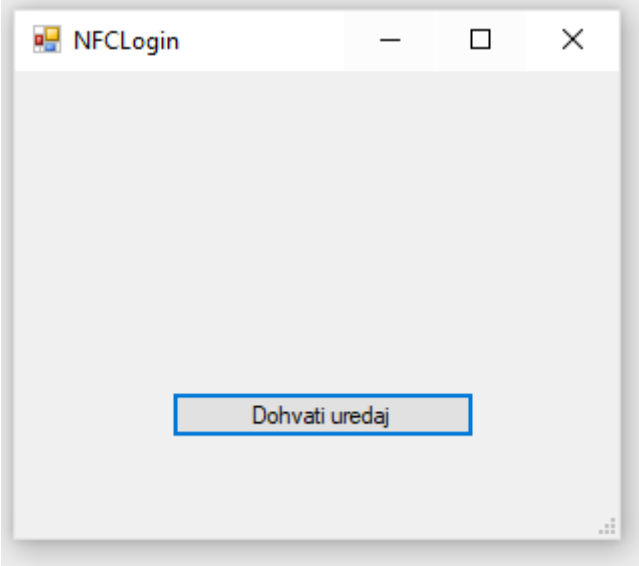
Slika 6: Početna forma aplikacije



Slika 7: forrma za registaciju lozinke



Slika 8: forma za registraciju mobitela



Slika 9: forma za prijavu mobitela

5.3.2. Klase i metode na strani android uređaja

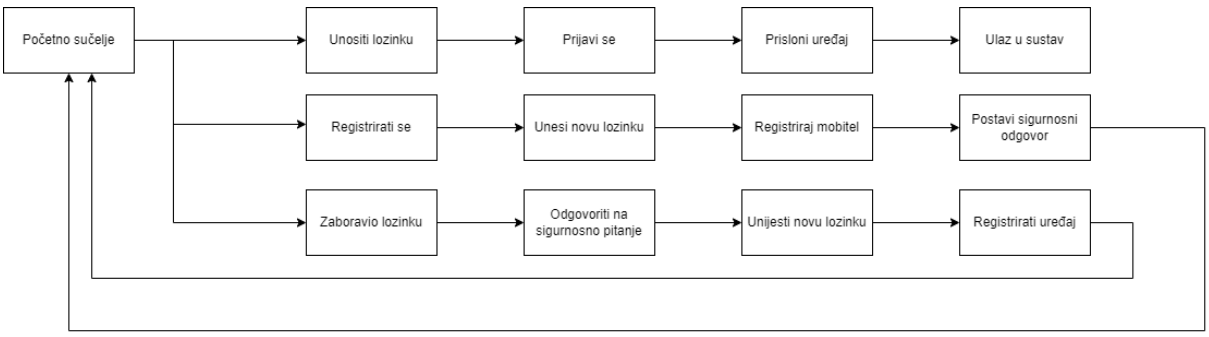
Pri programiranju na strani android uređaja koristimo samo jednu klasu, a to je servis HCEImplement koja proširuje klasu `HostApduService`. Aplikaciji smo definirali i dopustili korist HCE servisa i dali joj AID `F0010203040506`. Ona sučelje nema već ima definirane odgovore ovisno o pristigloj APDU poruci. APDU sekvencu ćemo ovdje zapisati kao metode iako su dio `processCommandApdu` metode. Jedina obaveza jest da mobitel mora biti otključan kako bi NFC mogao raditi. Neke od ovih funkcija ne sudjeluju u samom procesu aplikacije već su se koristile pri testiranju.

Tablica 6: Metode klase HCEImplement

Naziv	Opis
[0xAA, 0xAA, 0xAA, 0xAA]	Ispisuje pristiglu poruku unutar APDU poruke
[0xAA, 0xAB, 0x01,0x02]	Preuzimanje javnog ključa iz poruke i kriptiranje teksta „test“
[0xAA, 0xAC, 0x01, 0x02]	Stvaranje, spremanje i slanje kriptiranog UUID-a koji identificira uređaj
[0xAA, 0xAD, 0x01, 0x02]	Prikupljanje UUID-a iz sustava i slanje kriptiranog UUID-a računalu
[0x00, 0xA4, 0x04, 0x00]	Šalje se tekst „mystery to me“ natrag računalu
VratiKljuc	Metoda VratiKljuc vraća tip podatka RSAPublicKeySpec iz dobivenog XML zapisanog javnog ključa
KriptirajPodatke	Metoda uzima parametre tipa RSAPublicKeySpec generiranog od metode VratiKljuc i string kojega želimo kriptirani i vraća polje bajtova unutar kojeg se nalazi kriptirani tekst
dohvatiEnkriptiraniSharedPreference	Metoda koja stvara i dohvaća enkriptiranu shared preference datoteku

5.4. Korisnički tijek

Kada korisnik dođe na aplikaciju, njemu će biti ponuđene 3 putanje koje može iskoristiti. Prva je prijava gdje se korisnik prijavljuje putem lozinke i zatim mobilnog uređaja. Druga je registracija u sustav koja prolazi kroz točke unosa lozinke i registracije mobilnog uređaja. Treća je putanja zaboravljenih informacija gdje zbog ili zaboravljene lozinke ili zamjene uređaja više ne možemo ući u naš sustav, možemo vratiti naše podatke za prijavu putem odgovora na sigurnosno pitanje. Odabrano sigurnosno pitanje će biti prikazano i ako korisnik točno odgovori, može ponovno registrirati podatke za prijavu. Korisnici ne moraju proći velik broj procesa kako bi ušli u sustave ili se registrirali. Time stvaramo jednostavnost prijave.



Slika 10: Korisnički tijek

6. Zaključak

Predmet rada je implementacija dvorazinske autentifikacije. Dvorazinska autentifikacija je autentifikacija putem dva dokaza. Oba moraju biti točna kako bi se potvrdio identitet korisnika. Dva dokaza korištena u radu su lozinka i Android mobilni uređaj koji je korišten kao hardverski token. Bilo je potrebno isprogramirati dvije aplikacije. Jedna je na računalu koja služi kao terminal za ulaz u program i kako bi se osposobio NFC čitač. Druga se nalazi na mobilnom uređaju i u njoj se definiraju akcije koje će mobitel postupiti ovisno o dobivenoj poruci i koju će poruku vratiti. Tu komunikaciju je bilo potrebno složiti sigurnom. Proces registracije i prijave je morao isto tako biti siguran. U tu svrhu smo detaljno prolazili kroz teme NFC-a, točnije programiranja NFC uređaja i sigurnosti.

Neke biblioteke .NET Frameworka kao i programskog jezika Jave za kriptografiju su olakšale posao jer nije bilo potrebno ručno implementirati algoritme RSA i šifriranja. Uz pomoć tih alata se napravio sustav putem kojeg se može sigurno prijavljivati na sustave pri čemu hakeri čak da i da dobiju pristup našim lozinkama, ne mogu dobiti drugi potrebni dokaz kako bi ušli u sustave. Samu aplikaciju može se nadograditi uz npr. korištenje drukčijeg spremišta podataka kao što je baza podataka.

Za rad se koristio Java programski jezik za implementaciju mobilne aplikacije koji je pisan u Android Studio programu. C# i .NET Framework su se koristile unutar programskog alata Visual Studio 2019.

Kao što se i povećava količina transkacija beskontaktnim karticama radi jednostavnosti, tako i ovaj rad opisuje mogućnost i nudi rješenje za koje nije potrebno čitati i pisati dobivene mobilne poruke za dvorazinske autentifikaciju, već je potrebno samo prisloniti mobitel na čitač i ući u sustav sigurno.

Popis literature

- Željko Panian (2005.) Informatički enciklopedijski rječnik. Zagreb: Europapress holding
- Ministarstvo unutarnjih poslova Republike Hrvatske (n.d.) *Osobna iskaznica (eOI)*. Preuzeto 23.08.2022. s https://mup.gov.hr/osobna-iskaznica-eoi/328
- Gilad David Maayan (n.d.) *5 User Authentication Methods that Can Prevent the Next Breach* Preuzeto 23.08.2022. s https://www.idrnd.ai/5-authentication-methods-that-can-prevent-the-next-breach
- Jozo Vrdoljak (2020.). *'Udio beskontaktnih transakcija u Hrvatskoj sa 80% veći je od europskog prosjeka'.* Preuzeto 23.08.2022. s https://novac.jutarnji.hr/novac/aktualno/udio-beskontaktnih-transakcija-u-hrvatskoj-sa-80-veci-je-od-europskog-prosjeka-15002870
- NFC Forum (n.d.) *Learn* Preuzto 23.08.2022. s https://nfc-forum.org/learn/nfc-technology
- Google (n.d.) *Near field communication overview.* Preuzeto 23.08.2022. s https://developer.android.com/guide/topics/connectivity/nfc
- Advanced Card Systems Ltd. [acs] (2020.) *ACR1252U NFC Forum Certified Reader Application Programming Interface V1.17*
- Tutorialspoint (n.d.) *Android – Overview.* Preuzeto 27.08.2022. s https://www.tutorialspoint.com/android/android_overview.htm
- Tutorialspoint (n.d.) *Android – Architecture*. Preuzeto 27.08.2022. s https://www.tutorialspoint.com/android/android_architecture.htm
- Google (n.d.) *Application Fundamentals*. Preuzeto 27.08.2022. s https://developer.android.com/guide/components/fundamentals
- Tutorialspoint (n.d.) *Android - Content Providers.* Preuzeto 27.08.2022. s https://www.tutorialspoint.com/android/android_content_providers.htm
- Google (n.d.) *Host-based card emulation overview* Preuzeto 28.07.2022. s https://developer.android.com/guide/topics/connectivity/nfc/hce
- Russell, D., Russell, D., Gangemi, G. T., Gangemi, S., & Gangemi Sr, G. T. (1991). *Computer security basics*. O'Reilly Media, Inc

- Kurose, J., & Ross, K. (2010). Computer networks*: A top down approach featuring the internet.*
- Microsoft. (2022). *Encrypting data*. Preuzeto 31.08.2022 s https://docs.microsoft.com/en-us/dotnet/standard/security/encrypting-data
- Microsoft. (n.d.) *winscard.h header.* Preuzeto 05.09.2022. s https://docs.microsoft.com/en-us/windows/win32/api/winscard/
- BusinessofApps (2022). *Android Statistics (2022)* Preuzeto 3.9.2022. s https://www.businessofapps.com/data/android-statistics
- H2 (n.d.) *Universally Unique Identifiers (UUID)* Preuzeto 3.9.2022. s http://www.h2database.com/html/advanced.html

Popis slika

Popis tablica

Prilozi (1, 2, …)

1. Računalna aplikacija i kod kao .NET Framework rješenje
2. Android aplikacija i kod za Android Studio