NFC Lås

Fredrik Einarsson
Robin Gabre
Jonas Hemlin
Sebastian Karlsson
Daniel Moreau
Kristofer Tapper

2012-05-20

Förord

Denna rapport ingår i ett kandidatarbete utfört vid Institutionen för Data- och informationsteknik, Chalmers tekniska högskola våren 2013. Rapporten beskriver utvecklingen av ett NFC-baserat låssystem bestående av mjukvara till en Androidbaserad mobiltelefon samt en mikrokontroller av Arduino-typ.

Projektgruppen önskar tacka Lars Svensson för den tid han har lagt ner för att handleda projektet samt (någon mer för något annat?).

Abstract

In english motherfucker.

Sammanfattning

Mobiltelefonen utökas idag ständigt till att klara av att utföra fler och fler uppgifter åt dess användare. De saker vars funktioner nu erbjuds genom dagens mobiltelefoner behövs alltså inte längre tas med. Det här arbetet handlar om att ersätta nyckelknippan genom att ge mobiltelefonen funktionen att låsa upp en dörr, vilket ger att de otympliga nycklarna inte längre behöver tas med.

För att kunna erbjuda den önskade nya funktionalliteten har en applikation för Android utvecklats, vilken användaren agerar mot för att kunna låsa eller låsa upp dörren. Vidare har en låsenhet utvecklats kring Arduino-plattformen vilken kommunicerar mot mobilapplikationen via NFC (Near Field Communication) och har möjligheten att styra en elektrisk låskolv.

Säkerheten i systemet består i att användaren verifierar sig mot applikationen med förvald PIN-kod och kommunikationen är krypterad med krypteringsalgoritmen RSA(fotnot 1) med 512 bitar.

fotnot 1: efter upphovsmännen Ron Rivest, Adi Shamir och Len Adleman

Ordlista/Terminologi

Aktivitet En Java-klass som representerar en enstaka fokuserad uppgift. Ses vanligen som en skärm i Android.

Användare En person som använder mobilapplikationen.

Android Ett öppet mobilt operativsystem för främst smartphones och pekplattor.

Android beam En teknologi för att sända data mellan två Androidenheter via NFC.

API

APDU Application protocol data unit, generell benämning på datapaket.

Bluetooth

Börkrav Krav som bör ha uppfyllt för en bra produkt

DEP Data exchange protocol.

DSAP Destination service access point, adress som utgör mål för NFC kommunikation.

LLCP Locical link control protocol, det protokoll som dikterar hur NFC kommunikation sköts på länknivå.

Låsenhet Den del utav den utvecklade prototypen vilken är integrerad i låset.

Manifest Är en fil i ett Androidproject som specificerar globala inställningar för applikationen.

Material De fysiska delarna vilken prototypen utgörs av.

Mobil plånbok En ansatts till att ersätta plånboken med en applikation i en smart telefon.

NFC-Forum En organisation med målet att sprida och utveckla teknologin NFC.

NDEF Ett dataformat som specificerar hur data skall skickas och vad det är för data som skickas.

OSI-modellen

RSA Algoritm som används vid asymmetrisk kryptering.

SDP Service discovery protocol, är ett protkoll som används när avsändare inte känner DSAP hos målenhet.

Server Är ett datorsystem som har till uppgift att serva andra system.

Skallkrav Krav som skall vara uppfyllda i prototypen.

Smart telefon En mobil enhet som kan användas både som avancerad mobiltelefon och som handdator.

SNEP Simple NDEF exchange protocol. Detta protokoll beskriver utbytet av data vid användning av NFC.

SSAP Source service access point. Är den adress som brukande applikation tilldelats vid användandet av NFC.

Target Målenhet för NFC kommunikation

Verktyg Källkodsbibliotek, utvecklingmiljöer och specifikationer.

Wifi

Innehåll

1 Inledning **8**
- 1.1 Syfte . . . 9
- 1.2 Utmaningar . . . 9
- 1.3 Omfattning . . . 10

2 Metod **11**

3 Teori **11**
- 3.1 NFC . . . 12
  - 3.1.1 NFC Stack . . . 13
    - 3.1.1.1 NDEF . . . 14
    - 3.1.1.2 SNEP . . . 15
    - 3.1.1.3 LLCP . . . 17
    - 3.1.1.4 RF-protokoll . . . 22
    - 3.1.1.5 Ett typiskt kommunikationsförlopp . . . 23
  - 3.1.2 Säkerhet . . . 24
    - 3.1.2.1 Tjuvlyssning . . . 24
    - 3.1.2.2 Datamodifiering . . . 26
    - 3.1.2.3 Man in the middleättack . . . 26
- 3.2 Android . . . 26
  - 3.2.1 NFC för Android . . . 27
- 3.3 Arduino . . . 28
  - 3.3.1 Arduino-specifika källkodsbibliotek . . . 28
  - 3.3.2 NFC för Arduino mha PN532 NFC/RFID Shield . . . 28
    - 3.3.2.1 Generella Kommandon till PN532 NFC/RFID Shield . . . 28
    - 3.3.2.2 Kommandon till PN532 NFC/RFID Shield då enheten agerar målenhet . . . 28

4 Kravanalys och design **29**
- 4.1 Produktkrav . . . 29
- 4.2 Kommunikationen mellan enheterna . . . 29
  - 4.2.1 Kommunikationsförlopp . . . 29
  - 4.2.2 Meddelandetyper . . . 29
  - 4.2.3 Säker kommunikation . . . 29

5 Material och verktyg **29**
5.1 Mikrokontroller . . . 29
5.1.1 Utvecklingsmiljö och programspråk . . . 29
5.1.2 Implementerade och intressant protokoll samt specifikationer . . . 29
5.1.3 Val av källkodsbibliotek för implementation av NFC-stacken . . . 29
5.1.4 Val av källkodbibliotek för implementation av kommunikationssäkerhet . . . 29
5.2 Mobilapplikation . . . 29
5.2.1 Utvecklingsmiljö och programspråk . . . 29
5.2.2 Använda API:er . . . 29

6 Genomförande **29**
6.1 Utveckling av låsenheten . . . 29
6.1.1 RF-protokoll-lagret . . . 29
6.1.1.1 Kontroll av RF-protokollens funktioner . . . 29
6.1.2 Kommunikation mellan Arduino och NFC-sköld . . . 30
6.1.3 LLCP-lagret . . . 30
6.1.4 NPP- eller SNEP-lagret . . . 30
6.1.5 NDEF-lagret . . . 30
6.1.6 Applikationen, den som snurrar i loop() . . . 30
6.2 Utveckling av mobilapplikationen . . . 30
6.3 Utveckling av en prototyp bestående av båda enhterna . . . 30

7 Resultat **30**
7.1 Produkten som helhet . . . 30
7.2 Mobilapplikationen . . . 30
7.3 Låsenheten . . . 31

8 Diskussion **31**
8.1 Produkten . . . 31
8.2 Arduino . . . 31
8.3 Mobilapplikationen . . . 31

9 Slutsats **31**

1 Inledning

När mobiltelefonen först lanserades var den stor, otymplig och dyr, med kommunikation som enda syfte. Sedan dess har utveckligen eskalerat kraftigt och allt fler funktioner har integrerats, medan själva mobiltelefonen har gjorts mindre, smidigare och mer tillgänglig för allmänheten. Idag äger i princip alla en mobiltelefon och byter dessutom ofta, då det är både relativt billigt att köpa en ny samt att tekniken snabbt framskrider. På grund av detta är branchen hårdare än någonsin och för att överleva måste produkterna hela tiden utvecklas och innehålla nya, smarta funktioner för att bli konkurrenskraftiga på den snabbt skiftande marknaden.

Stryk den gamla inledningen här under, ett förslag. Allt sedan mobiltelefonen först lanserades har utvecklingen gått mot att integrera allt större funktionalitet i denna. Anledningen till detta är att mobilen oftast finns nära till hands och har därmed möjligheten att på sikt nästan ersätta alla de saker vi bär med oss. Anteckningar, miniräknare, alarmklocka, kalender och musikspelare är bara några av de funktioner som har integrerats och tillverkarna av mobiltelefoner ser det som en konkurrensfördel att föra in ytterligare funktioner i deras nya produkter.

På senare tid har övergången till så kallade smarta telefoner ytterligare bidragit till att öka funktionaliteten hos mobiltelefonen. Numera är den ej enbart ett redskap för kommunikation, utan en enhet med lika många eller till och med fler funktioner än en fullskalig persondator. En av anledningarna till denna utveckling är framstegen inom integrerade kretsar och dess tillverkningsteknik. Dessa faktorer har bidragit till att fler transistorer kan placeras på en mindre yta och därmed göra elektronikkomponenter både mindre, strömsnålare, snabbare och mer funktionella. Den direkta följden av detta är att mobiltelefonen nu kan inrymma både funktionaliteten av fler kringenheter, samt inneha kraft nog att hantera dem alla.

Wifi och Bluetooth är exempel på tekniker för kommunikation som i stort sett alltid finns inbyggt i en mobiltelefon och nu börjar även en ny teknik få allt större utbredning, NFC. Detta är en teknik för tillförlitlig kommunikation över korta avstånd som bland annat ligger till grund för en senare ansats att försöka ersätta något vi alltid bär med oss, plånboken. Tanken med denna produkt som i många fall kallas den mobila plånboken, vilket Timalsina (2012) beskriver, är att göra anspråk på att ersätta våra kontanter, rabatt- och kreditkort med programvara i mobiltelefonen.

Idén bakom detta arbete baseras på samma tankegångar som de som ligger bakom utvecklingen av trådlös betalning. Kandidatarbetet åsyftar till att utveckla en prototyp av ett låssystem som tillåter användaren att på ett

snabbt och bekvämt sätt låsa upp dörren till sitt hem med sin mobiltelefon. Till denna funktionalitet planeras användningen av samma kommunikationsteknik som ligger till grund för den mobila plånboken, NFC.

Resultatet av kandidatarbetet kan vara en mycket åtråvärd produkt för privatpersoner som är trötta på att leta efter sina hemnycklar samt önskar en flexiblare lösning än den klassiska nyckeln. Målet är att ytterligare följa trenden med en allt mer funktionell mobiltelefon genom att också använda den som en nyckel.

Visionen för en färdig produkt av detta slag, och alltså inte för resultatet av detta kandidatarbete, är att den funktionellt ska kunna användas i större företag där det finns hundratals dörrar som utrustats med det utvecklade låset samt tusentals anställda som alla använder den dedikerade mobilapplikationen för att kontrollera låsen. Många accessnivåer kan finnas där en specifik användare kan ingå i valfritt antal. Produkten kan underhållas av en central enhet som snabbt, smidigt och säkert kan konfigurera om systemet efter givna instruktioner.

Det finns också andra visioner för en färdig produkt av detta slag. Låssystemet behöver nödvändigtvis inte vara ett stort, relativt komplext och centraladministrerat system utan kan i stället vara litet, simpelt och mobilt så som ett cykellås eller ett hänglås. Då en låsprodukt av projektets slag kan utformas som sådant har en framtida produkt potential att ersätta även dem. Vem skulle inte önska sig ett hänglås där borttappandet av nyckeln inte innebär ett behov att klippa låset?

1.1 Syfte

Syftet med rapporten är att redogöra för hur en prototyp av ett låssystem designas och implementeras. Låssystemet består av en mobilapplikation och en mikrokontroller där mikrokontrollern är integrerad i låset samt exekverar egenutvecklad mjukvara. Skulle det visa sig att produkten inte uppfyller kravspecifikationen ska en analys genomföras i vilken en fullständig slutprodukt beskrivs teoretiskt.

1.2 Utmaningar

Tre huvudsakliga utmaningar identifierades.

- Stora delar av NFC:s kommunikationsstack måste implementeras för mikrokontrollern. Kommunikationsstacken måste vidare utformas efter hur Androids NFC-stack är uppbyggd. Att implementera kommunikationsstacken medför en större utmaning då den är relativt komplex,

innehåller ett flertal olika nivåer, och dokumentation kring Androids implementation är bristfällig.

- Att implementera en lättanvänd Androidapplikation vilken ska innehålla funktioner för att kommunicera mot mikrokontrollern via NFC. Androidutveckling medför ytterligare svårigheter jämfört med vanlig mjukvaruutveckling då hänsyn måste tas till Androids egenheter. En förståelse för de bibliotek vilka styr NFC-kommunikationen måste också byggas upp.

- Att implementera säkerhet vid NFC-kommunikation. Då en mikrokontroller generellt inte har mycket minne eller hög klockfrekvens kan dessa egenskaper skapa utmaningar då säkerhet i form av kryptering ofta kräver omfattande beräkningskapacitet samt minnesutrymme.

1.3 Omfattning

Rapporten beskriver endast konstruktionen av en prototyp utav en produkt av det beskrivna slaget. Vidare beskriver rapporten de nödvändiga förkunskaperna som krävs för en konstruktion av en sådan prototyp. Prototypen skall kortfattat utnyttja kommunikation via NFC med applicerad säkerhet och därför är förkunskaper inom dessa ämnen nödvändiga.

Mobilapplikationen begränsas till endast Android eftersom det är lättillgängligt, välkänt och lämpligt för användningsområdet. Denna avgränsning kommer också naturligt av att det bara finns stöd för NFC hos ett fåtal telefoner där merparten av dem är basereade på Android.

Mikrokontrollern med ansvar för låset baseras på Arduino-plattformen. Arduino valdes för dess enkla och tydliga utvecklingsmiljö samt för att många källkodsbibliotek finns att tillgå. Vidare har Arduino stöd för NFC i form av ett färdigutvecklat påstickskort, en så kallad sköld(fotnot 1), med tillhörande källkodsbibliotek.

Fokus ligger på att bygga en fungerande prototyp innehållande en låsenhet samt tillhörande mobilapplikation för till exempel privatpersoner, alltså inte för ett större företag. Då flera låsenheter eller flera användare av mobilapplikationen läggs till ökar komplexiteten för hela projektet och andra krav kommer att ställas på lösningen.

(fotnot 1) från engelskans shield

2 Metod

I den initiala fasen genomförs en analys från användarens perspektiv där de krav användaren ställer på prototypen listas. Dessa krav utformas sedan från låssystemets perspektiv och sammanställs till en kravspecifikation. Utifrån kravspecifikationen väljs de material, det vill säga de fysiska delarna vilken prototypen utgörs av, vilka är mest lämpade för utformningen av prototypen.

Under designfasen av projektet väljs hur prototypens funktionalitet distribueras ut i låssystemet och ett kommunikationsprotokoll för systemets delar samanställs. All funktionalitet som låssystemet innehåller delas upp på prototypens två enheter. Kommunikationsprotokollet beskriver hur den data som skickas mellan enheterna är uppbyggd, hur kommunikationsförloppet mellan enheterna ser ut samt hur säkerhet appliceras till kommunikationen.

Projektet övergår nu till en iterativ arbetsgång där ytterligare funktionalitet, utifrån kravspecifikationen, försöker implementeras för varje iteration. För att kunna utföra en iteration undersöks vilka eller vilket verktyg såsom källkodsbibliotek, utvecklingsmiljöer och information som krävs för att implementera funktionaliteten. Verktygen utvärderas och de som bedöms kunna bidra till, eller underlätta för implementering av den sökta funktionaliteten, väljs ut. Verktyget används sedan för fortskridandet av iterationen. Den nytillagda funktionaliteten utvärderas och om den lever upp till kravspecifikationen påbörjas nästa iteration, men skulle den nytillagda funktionalitet inte uppfylla de krav som ställs undersöks vad som är möjligt att ta med utav den nytillagda funktionaliteten till nästa iteration eller om resultatet av iterationen måste kasseras.

3 Teori

I detta kapitel beskrivs den teori som är nödvändig att förstå vid konstruktionen av prototypen. Teorin bidrar till en ökad förståelse men bidrar framförallt till det direkta framskridandet av projektet.

Eftersom prototypen använder NFC-teknik som kommunikationsmetod är kunskaper inom NFC relevant. Kunskaper inom NFC, och inte bara inom dess användande, är främst nödvändig för utvecklingen av mjukvaran till mikrokontrollern då kommunikationsstacken behöver implementeras från grunden. Detta står i kontrast till utvecklandet av Android-applikationen där kunskaper om användandet av Androids API för NFC är nödvändiga och kunskaper inom dess implementation är mindre viktigt.

Vidare är kunskaper inom säker kommunikation relevant eftersom NFC

inte har någon inbyggt säkerhet. En applikation som nyttjar NFC bör därför själv implementera säkerhet om behov finns och eftersom ett låssystem skall konstrueras är säkerhet ett krav. Säker kommunikation kan uppnås på många olika sätt och därför är kunskap för att både välja rätt metod och kunskap om den valda metodens implementering av stor vikt.

Utvecklingen av Android-applikationen kräver förutom kunskap i Java-programmering även kunskap kring applikationsutveckling för Android vilket tas upp i detta kapitel. Bland annat har applikationer för Android en speciell livscykel som är nödvändig att känna till. Det är även nödvändigt att känna till de Android-specifika API som finns att tillgå. Särskilt bör vetskap om de Android-specifika API för kommunikation via NFC besittas.

För att utveckla en applikation till Arduino-plattformen krävs förutom kunskaper inom C/C++-programmering även kunskap om Arduino vilket behandlas i detta kapitel. En Arduino har inte något operativsystem vilket gör att en Arduino-applikation kör direkt på den underliggande hårdvaran. Det betyder att kunskaper kring hur kretskorten ser ut måste beskrivas. Vidare behöver kännedom om hur kretskorten kommunicerar med varandra redogöras. Slutligen bör kunskaper angående de nödvändiga Arduino-specifika bibliotek samt hur Arduino kommunicerar via NFC beskrivas.

3.1 NFC

NFC (Near Field Communication) är en trådlös kommunikationsteknik över avstånd upp till ungefär 10 cm (Timalsina 2012) och standardiserades år 2006, tekniken är alltså relativt ny. NFC baseras på RFID (Radio Frequency Identification) vilket är en väletablerad trådlös kommunikationsteknik som funnits sedan 1983. Tekniken bygger vidare på RFID genom att erbjuda tvåvägskommunikation mellan två enheter med inbyggt stöd för NFC(Timalsina 2012).

Tekniken bygger på att två mikrochip med stöd för NFC-teknik kommunicerar med varandra genom magnetisk induktion, och mer specifikt genom att modifiera det andra mikrochipsets magnetfält(Timalsina 2012). NFC-tekniken opererar på frekvensen 13.56 Mhz och data kan skickas i hastigheter från 106 Kbps upp till 424 Kbps.

Enheter kan vara antingen passiva eller aktiva. Aktiva enheter har strömförsörjning och återfinns till exempel i mobiltelefoner medan passiva enheter saknar strömförsörjning och återfinns till exempel som taggar på posters eller i busskort.(Timalsina 2012)

Två aktiva enheter kan kommunicera med varandra. Aktiva enheter kan även kommunicera med passiva enheter genom att den aktiva enheten ger

strömförsörjning till den passiva enheten med hjälp av magnetisk induktion. Endast halv-duplex kommunikation stödjs, dvs kommunikation kan endast ske i en riktning i taget. Två passiva enheter kan inte kommunicera eftersom inget magnetfält uppstår utan tillförd energi.(Timalsina 2012)

NFC-Forum(2013) standardiserade NFC-tekniken år 2006 och är det organ som har ansvar för att ta fram de specifikationer som gör kommunikation via NFC möjlig mellan olika typer av enheter. De har också ansvar för att sprida och uppmuntra användandet av NFC-teknik samt utbilda företag i mål om att företagen ska följa de officiella specifikationerna så att tekniken kan fungera sömlöst mellan enheter från olika tillverkare.

I de följande teoriavsnitt som tar upp NFC beskrivs de protokoll NFC-Forum har specifiserat. Dessa protokoll utgör den NFC-stack som behövs då kommunikation mellan två aktiva enheter önskas. Vidare täcker protokollen hela OSI modellen förrutom applikations-skiktet och de protokollen är således allt som behövs för att implementera NFC-stacken.

3.1.1 NFC Stack

NFC-stacken är uppbyggd av ett antal protokoll som sträcker sig i stort sett över hela OSI-modellen, då endast applikationslagret med tillhörande säkerhet ej ingår i NFC-stacken. Detta är en kort översikt till de protokoll som ingår och NFC-stackens motsvarande OSI-skikt gås igenom uppifrån och ned.

- Data som sänds mellan två aktiva NFC-enheter kappslas in i ett meddelande av typen NDEF(NFC Data Exchange Format). Informationen ligger som ett eller flera fält i NDEF-meddelandet som så kallade NDEF-records.

- Vidare används protokollet NPP(NDEF Push Protocol) eller SNEP (Simple NDEF Exchange Protocol) för att utbyta NDEF-meddelanden mellan två aktiva NFC-enheter. Då NPP har ersatts av SNEP kommer bara SNEP att beskrivas.

- För att upprätta och hantera en länk mellan två aktiva NFC-enheter används LLCP(Logical Link Control Protocol). Protokollet hanterar också dataöverföringen mellan enheterna och ser till att den görs utan förluster.

- Den fysiska överföringen av bitarna hanteras av RF-protokollen där bland annat hur och i vilka hastigheter data kan skickas specifieras.

3.1.1.1 NDEF *(NFC Data Exchange Format)*[källa] är särskilt utformat för att kunna kapsla in applikationsdata på ett effektivt sätt med minimal overhead. NDEF meddelanden byggs upp av `records`, vilket kan liknas vid sidor i en bok, där boken utgör NDEF meddelandet. Meddelandet kan bestå av ett obegränsat antal records som alla kan bära på olika sorters data.

När en applikation önskar sända data över en NFC-länk, kapslas datan in i ett NDEF meddelande. Beroende på datans storlek och struktur skapas varierande mängd records som tillsammans bildar ett NDEF meddelande. Meddelandet överförs sedan till målenheten som tar emot och behandlar det. NDEF garanterar inte att leverans av data är tillförlitlig och specificerar således inte vad målenheten skall göra vid mottagandet av ett felaktigt utformat NDEF meddelande. Detta lämnas till brukande applikationer att komplettera med datagarantier och felhantering som gemensamt brukar kallas QOS (Quality Of Service).

Varje record utformas efter en fördefinerad struktur och består av ett antal fält. Det första fältet är en header, som sedan följs av type length, payload length, ID length, type, ID och till sist payload.

- **Header.** En record header är en byte lång där varje bit har olika betydelser.
  - Den första biten är MB som är en förkortning av message begin vilken indikerar att detta är den första record i NDEF meddelandet. Denna bit är ettställd i första record, nollställd i övriga.
  - Den andra biten är ME som är en förkortning av message end vilken indikerar att detta är sista record i NDEF-meddelandet. Denna bit är endast ettställd i sista record, nollställd i övriga.
  - Den tredje biten är CF som är en förkortning av chunk flag vilken och indikerar att NDEF-meddelandet är chunked. Chunkmekanismen tillåter att ett stort datapaket delas upp över flertalet record. Denna mekanism kommer inte att behandlas i denna text då den utvecklade mjukvaran inte nyttjar detta.
  - Den fjärde biten är SR som är en förkortning av short record vilken används då en mindre mängd data skall överföras. När SR biten är ettställd minskas mängden payload length fält från 4 till 1.
  - Den femte biten är IL vilken indikerar huruvida ID length fältet existerar i record.

  - De sista bitarna, bit 6 till 8, bildar TNF vilken är en förkortning av type name format och definerar strukturen av type-fältet i record.

- **Type Length.** Definerar hur många bytes som utgör Type-fältet.

- **Payload length 0-3.** Varje delfält i payload är en byte stor vilket ger payload length fältet en total storlek på 4 bytes. Payload length indikerar storleken på payload fältet som huserar applikationsdatan. I en record med SR nollställd kan payload- fältet vara $8 * 2^8$ bytes stort, det vill säga 4,29 gigabytes. Med SR ettställd kvarstår enbart ett payload lenght-fält vilket ger payload-fältet en maximal storlek på 255 bytes.

- **Type.** Detta fält definerar vilken typ av data som payload-fältet innehåller. Kraven vad gäller struktur och kodning som specificerats av TNF fältet i header måste följas. Värdet på detta fält kan underlätta behandlingen av applikationsdatan hos målenheten, dock specificeras det inte hur behandlingen sker utan det är upp till applikationen att implementera.

- **ID.** Värdet på detta fält är unikt och utgör en unik identiferare kallad URI. Records kan således skiljas åt genom att studera värdet av URI och NDEF garanterar att detta nummer är unikt genom att tillhandahålla en generator.

- **Payload.** Detta fält huserar applikationsdata.

Nedan ges ett exempel på ett NDEF-meddelande som kommer att följa med till andra avsnitt som behandlar NFC-stacken.

(Använd referens istället för nedan!!!!!!!!!!!!!!!!!!!!!!!!)

3.1.1.2 SNEP (Simpel NDEF exchange protocol)[källa] är det protokoll i NFC-stacken som hanterar utbytet av NDEF-meddelanden. SNEP är ett så kallat request/response protokoll, en klient skickar en SNEP-förfrågan till en server som, beroende på förfrågans innehåll, returnerar ett passande svar om det erfodras.

Förfrågningar SNEP har specificerat förfrågningens utseende som följer.

- **Version.** Detta fält definerar vilken version av SNEP som förfrågan bygger på. Versionsfältet är indelat i två delfält kallat Major och Minor om 4-bit vardera där Major huserar heltalet av versionsnummret och Minor huserar decimaldelen av versionsnummret. För att en kunna genomföra en SNEP session måste klient och server förstå varandra och deras förmåga att förstå varandra avgörs av versionsfältet. Vid mottagande av en SNEP förfrågan kontrollerar servern versionsnummret. Stämmer numret helt överens med den version som servern brukar kan SNEP sessionen fullbordas. Stämmer Major delen men inte Minor är det möjligt att slutföra sessionen om servern anpassar sig efter klienten. Skiljer sig Major delen kan sessionen inte genomföras och servern ger avslag.

- **Request.** Request - fältet är 8 bitar stort och anger vilken typ av förfrågan det rör sig om. Fältet innehåller koder som alla är förfrågningsrelaterade. Olika förfrågan erfodrar olika svar från servern.

- **Length.** Detta fält anger hur många bytes som utgör informationsfältet. Length-fältet är 4 bytes långt vilket resulterar i att informationsfältet som störst får vara 4.29 gigabytes.

- **Information** Innehållet i detta fält beror på innehållet i Request-fältet dvs vilket sorts förfrågan det är. I regel är det ett NDEF meddelande som huseras i information-fältet.

De förfrågningar en klient kan begära från en server är Continue, Get, Put samt Reject. De två vanligaste förfrågninarna och de vilka är av relevans för projektet är Put och Get vars betydelse och användning förklaras nedan:

Put. En Put-förfrågan har hexadecimal kod 0x02. Klienten ber servern att ta emot det inkapslade NDEF meddelandet och inget svar erfodras av servern. Put-förfrågan följer den generella utformningen av ett SNEP förfrågan:

Get. En Get-förfrågan har hexadecimal kod 0x01. Klienten ber servern att returnera information inkapslat i ett NDEF meddelande. Vilken information som klienten vill få från servern specificeras i NDEF-meddelandet som huseras i Get-förfrågans information-fält.

Get-förfrågan skiljer sig från den generella förfrågningsstrukturen. Ytterligare ett fält förekommer, Acceptable Length, vars betydelse är att det anger hur stort NDEF-meddelandet i svarsmeddeleandet maximalt får vara. Acceptable length-fältet är 4 bytes vilket gör Get-förfrågan större än en generell förfrågan.

Svarsmeddelanden SNEP har också specificerat svarets utseende:

Fälten i svaret fyller samma funktion som i förfrågningen men med mindre skillnader.

Response. Fyller samma funktion som Request-fältet. Dock innehåller denna andra koder som alla är av svarstyp.

SNEP svar följer ganska sällan det fördefinerade utseendet. Det är enbart svar på Get-förfrågningar som fullt ut följer det generella utseendet hos ett svar. Vanligt förekommande svar är följande.

- **Success.** Detta svar indikerar att servern klarat av att genomföra klientens Get eller Put förfrågan. Utformningen på Success-svaret skiljer sig beroende på om det ges som svar på en Get eller Put förfrågan. I fallet Get följer svaret den generella utformningen med response satt till 0x81 vilket indikerar success. I information-fältet huseras NDEF-meddelandet som ges som svar på Get-förfrågan. I fallet Put sätts response till 0x81, men informations-fältet närvarar ej.

- **Not found.** Detta svar ges om servern inte klarar av att returnera önskat svar till klienten. Svarskoden är 0xC0 och inget information-fält förekommer.

- **Bad request.** Ges om förfrågningen inte följer fördefinerat utseende. Svarskoden är 0xC2 och inget information-fält förekommer.

- **Not implemented.** Detta svar ges om servern inte stödjer den funktionaliet som klienten ber om i förfrågan. Detta svar lämpar sig väl om felaktig request-kod angivits i förfrågan. Inget information-fält förekommer.

- **Unsupported version** Ges då klientens SNEP version inte stämmer överens med vad servern kan hantera. Major-delen av version-fältet skiljer sig åt alternativt klarar servern inte av att överbygga de skilnader som olika Minor-värden medför. Inget information-fält förekommer.

Alla NFC-enheter har som krav att ha en SNEP-server process körandes. På så vis garanteras att all SNEP kommunikation behandlas. Denna garanti förenklar för både ovanliggande applikationslager som underliggande LLC-lager.

3.1.1.3 LLCP

LLCP(Logical Link Control Protocol)[källa] är protokollet som sköter samtliga datalänkar med andra NFC enheter. För att

öka överskådligheten hos protokollet har det delats upp i fyra centrala delar: Media access mappning, länkhanteraren, förbindelselös transport och förbindelseorienterad transport. NFC-stacken ser med dessa delar ut som figuren nedan visar.

Media access mappning Knyter samman ovanliggande protokoll med de RF-protokoll, vilka styr tekniken som står för den faktiska överföringen av data mellan två NFC enheter.

Länkhanteraren Detta är en process som styr all NFC-kommunikation och körs på samtliga enheter som stödjer NFC. Länkhanteraren serialiserar all kommunikation till och från NFC-enheter samt kan hantera flera parallella datalänkar och sköta övervakningen av deras aktuella tillstånd.

Ytterligare en viktig funktion som länkhanteraren fyller är att genomföra ett så kallat symmetriförfarande, vilket gör datakanalerna asynkront balanserade. Eftersom NFC är halvt duplex, dvs kommunikation kan ske i båda riktningar men endast en part i kommunikationen får sända samtidigt, måste länkhanteraren hålla reda på vem som har rätten att sända. Under ett normalt utbyte åstadkommes en balans i kommunikationen genom att parterna sänder en LLCP-ram var. Om endera part har rätten att sända, men inte har något att sända, genomför länkhanteraren symmetriförfarandet genom att skicka en speciell LLCP-ram (SYMM) och rätten att sända överlämnas. Symmetriförfarandet gör det möjligt att upptäcka länkförluster som uppkommer av att t.ex kommunikationsavståndet blivit för stort. Om en part i kommunikationen förs utom räckhåll upptäcks detta genom att all form av svar uteblir, dvs varken en normal ram eller en SYMM-ram når målet.

LLCP erbjuder två alternativa transporttjänster till det ovanliggande applikationslagret, förbindelseorienterad transport och förbindelselös transport.

Förbindelselös transport Förbindelselös transport skickar onumrerade datapaket och ger inga garantier till sändande applikation att data tas emot av mottagaren. Mindre kontroll av datatrafiken leder till minskad overhead och därigenom till en snabbare transport. Denna typ av transport lämpar sig således för applikationer som lägger stor vikt på snabbhet och som inte kräver några transportgarantier alternativt implementerar dessa själv. En datalänk av denna sort kallas för logisk datalänk och karaktäriseras av addresserna som den sändande (SSAP) och motta applikationen (DSAP) använder.

Förbindelseorienterad transport Förbindelseorienterad transport skickar numrerade datapaket och garanterar applikationer säker leverans av data men till en kostnad av ökad overhead. Förbindelseorienterad transport lämpar sig för applikationer som ställer högre krav på säkerhet än snabbhet. Två för NFC centrala protokoll, SNEP samt NPP, använder denna transportmetod.

En datalänk av denna typ kallas för uppkoppling. En uppkoppling karaktäriseras av SSAP och DSAP men också ett virtuellt tillstånd som bestäms av fyra stycken numeriska variabler vars olika värden definerar tillståndet. Samtliga variabler kan anta ett värde 0-15 (modulo 16) och syftar till att garantera säker transport av data. Denna mekanism kallas också för ACK från engelskans aknowledgement.

- **VS** Första variabeln, sändvariabeln. VS står för hur många numrerade paket, även kallat ramar, som har skickats från enheten över aktuell uppkoppling.
- **VSA** Andra variabeln, sändbekräftelsevariabeln. VSA anger senast mottagna numrearad ram som målenheten tagit emot.
- **VR** Tredje variabeln, mottagningsvariabeln. VR indikerar vilket nummer som nästa numrearad ram länkhanteraren väntar på att ta emot på aktuell uppkoppling.
- **VRA**Fjärde och sista variabeln. VRA står för senast sända VR i en numrerad ram.

Utöver dessa variabler håller också länkhanteraren reda på lokalt mottagningsfönster RWL och mottagningsfönstret i målenheten RWR som båda står för hur många numrerade ramar som får skickas mellan enheterna utan att VSA uppdateras. Att VSA inte behöver uppdateras efter varje sändning syftar till att minska behovet av att skicka bekfäftelseramar och overheaden på uppkopplingen minskas.

Ack mekanismen går till på så vis att VS och VR skickas med i varje numrerad ram. VRA sätts till det senast sända VR numret. Vid mottagande av en numrerad ram kontrolleras medföljande VR, VSA sätts till VR -1 då VR implicit indikerar att alla tidigare paket tagits emot korrekt. Vid avikelser, eg om VR i en mottagen ram inte stämmer överens med VSA har ett fel uppstått. Uppkopplingen stängs då ner.

vad händer om fel? FMRMRMFMRM ram skickas med massa satans felkoder. Ta med?

LLCP-meddelanden Eftersom NFC körs på flertalet olika enheter som skiljer sig åt vad avser mjuk- och hårdvara måste LLCP vara flexibelt nog för att kunna köras oberoende av plattform. Denna flexibilitet erhålles genom att specifika parametrar för en datakanal kan förhandlas fram mellan de två kommunicerande LLC-processerna. Viktiga parametrar som är förhandlingsbara är MIU och LTO.

MIU är en förkortning av Maximum Information Unit och indikerar hur stor ett datapaket maximalt får vara som skall skickas på datalänken. Detta tillåter enheter med begränsade databuffertar att bruka LLCP. Om denna parameter utelämnas i förhandlingen antags ett standardvärde på 128 byte.

LTO står för Link time Out och indikerar hur länge en länk kan vara inaktiv innan den betraktas som förbrukad och stängs ner. Genom att anpassa LTO kan enheter med låg beräkningskapacitet använda LLCP och kommunicera via NFC, vilket är fallet för ATMEL baserade Arduino-plattformar. Denna parameter utbytes efter att en datalänk etablerats alternativt innan om NFC-enheterna stödjer detta.

LLCP datapaket för förbindelseorienterad transport:

- **SAP:** Förkortning av Service Accsess Point och används som addresser vid kommunikation mellan två LLC processer. SAP kan liknas vid en brevlåda i vilken applikationer lämnar meddelanden (data) som den önskar att överföra till en applikation som kör på målenheten. Till varje SAP hör exakt en applikation.

- **DSAP:** Detta fält utgör adressen som addministreras av mottagar LLC vilken sändaren önskar att upprätta en kommunikation-session med. Skall användas som SSAP när ett eventuellt svar ges på det mottagna datapaketet.

- **SSAP:** Anger vilken SAP som applikationen blivit tilldelad skickar från. Skall användas som DSAP när ett eventuellt svar ges på det mottagna datapaketet.

- **PTYPE:** Detta fält anger vilken sorts datapaket det är och formen på den. Alla datapaket är inte utformade på exakt samma sätt då de är avsedda för olika ändamål.

- **Sequence:** Detta fält är indelat i två stycken underliggande fält. N(S) är här samma nummer som VS. N(S) innehåller således en siffra 0-15 som visar vilket sändnummer ramen har den lokala LLC processen. N(R) är samma som VR. N(R) innehåller alltså en siffra 0-15 som

visar vilket nummer som den lokala LLC processen väntar på att ta emot. Sequence-fältet finns enbart med i de ramar som är numrerade. Sekvensnumret används i LLC protokollets ACK mekanism.

- **Information:** Innehåller data från ovanliggande lager, t.ex SNEP eller NPP som i sin tur kapslar in applikationsdata. Alternativt kan informationsfälltet innehålla LLC specifika parametrar. Vad informationsfältet innehåller kan utläsas av PTYPE-fältet.

Nedan följer ett urval av de typer av datapaket vilka anses vara relevanta för projektet.

Connect

Denna ram används när en NFC-enhet vill etablera en uppkoppling till en annan enhet. DSAP sätts till en välkänd SAP, som t.ex 0x04 vilket indikerar att sändaren vill etablera en uppkoppling med SNEP-servern som kör på målenheten. Alternativ kan SSAP sättas till 0x01 vilken indikerar att målenheten vid mottagande av denna connect-ram skall köra service discovery protocol (SDP) för att hitta en lämplig SAP. SSAP sätts till det värde applikationen som vill utväxla data över NFC blivit tilldelad. Observanta läsare lägger märke till att sequence-fältet inte finns i en Connect-ram. Detta beror på att en Connect-ram är onumrerad, vilket är fallet för många LLCP-ramar. Informationsfältet specificeras till att innehålla datalänkspecifika parametrar. Genom att undersöka PTYPE-fältet kan mottagaren utläsa vilka fält som skall närvara och vad dess innehåll betyder.

Connection Complete

Denna ram skall ges som svar på en Connect-ram om en uppkoppling kan upprättas. DSAP sätts till Connect-ramens SSAP värde. SSAP sätts antingen till det värde som SDP returnerat, eller till det fördefinerade värdet som sändaren angav i DSAP. Precis som Connect-ramen är denna ram onumrerad och dess informationsfällt innehåller datalänks-specifika parametrar.

DM

Denna ram ges som svar på en Connect-ram när en uppkoppling inte kan etableras. Ramens betydelse är att LLC processen som skickar den inte längre lyssnar på uppkopplingen, den är logiskt avslutad. Anledningen till att uppkopplingen blivit avslutad ges i informations-fälltet. Ramen används också vid avslutande av uppkopplingen och skickas då som svar på en DISC-ram.

Information

När utbytet av Connect-ramen och CC-ramen genomförts är uppkopplingen etablerad. Utbytet av applikationsdata kan inledas. Informationsramen är ramen som är ämnad för att överföra data mellan två NFC-enheter.

För att kunna upptäcka eventuell dataförlust är denna ram numrerad. Informationsfältet innehåller i detta fall data från ovanliggande applikationer.

Symmetry

Denna ram används i den tidigare beskrivna symmetri-proceduren för att undvika länk-timeout under pågående kommunikation. När en NFC-enhet har exklusiv rätt att skicka data men saknar något att skicka och önskar bibehålla datalänken genomförs symmetri-proceduren under vilken denna ram spelar en central roll.

Denna ram används för att indikera att föregående frame togs emot korrekt när ingen nyttig data kan skickas. Detta är således LLC protokollets dedikerade ACK-ram.

DISC

Används när en av parterna vill stänga ner datalänken. När en LLC process tar emot en ram av denna typ kommer den inte att skicka mer data över denna uppkoppling. Kanalen stängs ned först när sändande part mottager ett svar i form av en DM-ram.

3.1.1.4 RF-protokoll De RF-protokoll vilka det utvalda NFC-chippet PN532 från NXP Semiconductors implementerar är ISO/IEC 18092 NFCIP-1(Near Field Communication – Interface And Protocol)[7] samt ISO/IEC 14443 där det förstnämnda används i samband med kommunikation mot en aktiv NFC-enhet. NFCIP-1 är således det protokoll detta projekt nyttjar hos NFC-chippet och detta specifiserar bland annat moduleringstekniker, kodningsanvisningar, överföringshastigheter, datakollisionskontroll och ramformat. Vidare definierar NFCIP-1 initieringsförfarandet, datautbytesförfarandet (DEP) och deaktiveringsförfarandet vilket används vid aktiv kommunikation.

Initieringsförfarandet I initieringsförfarandet sänds en `ATR_REQ` från den initierande enheten till målenheten, vilken sänder tillbaka en ATR_RES som svar. En ATR_REQ och en ATR_RES är snarlikt uppbyggda och beskrivs samtidigt i följande stycke.

De två första bytesen utav en `ATR_REQ` eller `ATR_RES`(se figur R) är $0xD4$ och $0x00$, sedan följer NFCID3 vilket är ett slumpartat identifikationsnummer vilket ska vara detsamma under en och samma kommunikations-session. Följande fem bytes i en ATR_RES respektive fyra bytes i en ATR_REQ (TO uteblir), är parametrar vilka sköts av NFCIP-1. Slutligen följer så kallade generella bytes vilka inleds med de magiska LLCP nummren och efter

dem radas de TLV parametrar(se 3.1.1.3) upp som önskar utbytas i initieringsförfarandet.

Datautbytesförfarande DEP(Data Exchange Protocol) är det protokoll som följs vid utbytandet av data mellan två aktiva NFC enheter på RF-protokolls-nivå och är ett blockorienterat transportprotokoll med inbyggd felhantering och kedjemekanism. Denna mekanism används då den data vilken ska överföras inte får plats i en ram.

Deaktiveringsförfarande RELEASE/deaktivering

3.1.1.5 Ett typiskt kommunikationsförlopp

Uppkopplingsupprättande En applikation som körs på en NFC-enhet önskar att skicka data till en annan applikation som körs på en målenhet. För att kunna sända datan måste LLC processen på den sändande enheten först upprätta en uppkoppling genom att skicka en Connect-ram. PTYPE sätts till 0100 vilket indikerar en Connect-ram. Connect-ramens SSAP sätts till den address som aktuell applikation blivit tilldelad. På samtliga NFC-enheter körs en SNEP-server. Denna server använder en välkänd SAP (0x04) och det är denna SAP som används som DSAP i Connect-ramen om applikationsdatan skall utväxlas med SNEP vilket ofta är fallet.

När LLC processen på målenheten mottager en Connect-ram, och DSAP och SSAP är giltliga, bearbetas de uppkopplingsspecifika parametrana som eventuellt medföljde Connect-ramen. Avsedd applikation, i detta fall SNEP-servern, meddelas om att någon önskar utbyta data. SNEP-servern kontrollerar huruvida datautbyte är möjligt. Är datautbyte inte möjligt meddelas LLC processen som då sänder en DM-ram och uppkopplingen avslutas. Är utbyte av data möjligt meddelas LLC om detta som då skickar en CC-ram. I denna ram sätts DSAP till Connect-ramens SSAP och SSAP till Connect-ramens DSAP. VS, VR, VSA och VRA är samtliga satta till 0.

Vid mottagandet CC-ramen sätter den sändande LLC processen VS, VR, VSA och VRA är samtliga satta till 0. Uppkopplingen är etablerad och kommunikationen övergår i informationsutbytesfasen.

Informationsutbyte Den lokala LLC processen sänder iväg den första Informations-ramen som kapslar in applikationsdata. Eftersom det är den första informations-ramen sätts sekvensnummret N(S) och N(R) till 0. Om ytterligare informations-ramar skickas ökas successivt sekvensnummret VS

och N(S). Informationsfältet innehåller SNEP-ramen som i sin tur innehåller datan som skall överföras.

Vid mottagande av informations-ramen erhålls den inpackade SNEP-ramen som levereras till SNEP-servern. Om applikationen på målenheten önskar svara kapslar LLC processen på målenheten in svaret i en information-ram med sekvensnummret N(S) = 0 och N(R) = 1. Det första indikerar att detta är den första informations-ramen som målenheten skickar till sändarenheten, det senare indikerar att föregående informations-ram tagits emot korrekt. Om inget svar ges skickar LLC-processen på målenheten en RR-ram med sekvensnummer N(R) = 1 som fyller samma funktion som informationsramen-ramen men utan medföljande data. Datautbytet fortlöper med successivt ökande tillståndsvariabler tills antingen ett fel uppstår eller tills utbytet är färdigt. Båda resulterar i att kommunikationen övergår i uppkopplingsavslutningesfasen.

Uppkopplingsavslutning Applikationen signalerar den lokala LLC processen om att utbytet är färdigt. Den lokala LLC processen skickar då en DISC-ram till LLC-processen på målenheten. LLC processen på den sändande enheten slutar ta emot samtliga ramar förutom en DM-ram. Vid mottagande av denna DM-ram avslutas uppkopplingen. Vid mottagande av DISC-ramen ses uppkopplingen som avslutad av LLC processen på målenheten. Processen meddelar SNEP-servern om att uppkopplingen stängs ned och skickar en sista DM-ram över uppkopplingen.

Båda parterna ser nu uppkopplingen som avslutad.

3.1.2 Säkerhet

Detta avsnitt är baserat på artikeln Strengths and Weaknesses of Near Field Communication (NFC) Technology[14].

NFC har likt RFID i sitt grundutförande inga inbyggda säkerhetsfunktioner som förhindrar avlyssning, modifiering eller utstörning av kommunikationen. Eftersom kommunikationen är trådlös så kommer avlyssning alltid att vara möjligt och det är upp till sändare och mottagare att kryptera meddelanden om behovet finns. Följande sektioner beskriver hur dessa säkerhetsbrister kan åtgärdas.

3.1.2.1 Tjuvlyssning Eftersom NFC är trådlös kommunikation som skickar information via radiovågor kan icke avsedda enheter fånga upp signalerna. För att kunna fånga upp signalerna måste dessa icke-avsedda enheter befinna sig inom kommunikationsräckhåll. På grund av NFCs korta

kommunikationsavstånd, kring 10 centimeter, besitter NFC ett inneboende skydd mot avlyssning. Avlysstningsutrustningen måste föras väldigt nära de kommunicerande enheterna vilket underlättar upptäckten av avlyssning. Det finns dock ingen garanti för att signalerna kan fångas upp på betydlig längre avstånd eftersom flertalet parametrar spelar in i hur lång radiovågor kan färdas. Viktigast av dessa är:

- Antenn
- Effekt
- Moduleringsteknik
- Omgivning

Vilken antenn som sändare och avlyssnare använder spelar en avgörande roll. Dess dimensioner, utformning och omslutande material är några viktiga faktorer som spelar i hur väl en antenn kan sända och ta emot radiovågor. I t.ex telefoner används en relativt liten spole vilket starkt begränsar signalstyrkan. Vidare så är telefoner i dagsläget ofta innesluten i robust plast eller aluminium som också minskar signalsstyrkan.

Effekten som signalen sänds med avgör hur långt de kan färdas. En signals effekt är starkt sammankopplad med bärvågens amplitud. Ju större amplitud bärvågenvågen har dessto större effekt har signalen. En signal som sänds med hög effekt och följaktligen stor amplitud kan färdas längre än en signal som sänds med låg effekt och som har liten amplitud.

Moduleringstekniken, enkelt uttryckt hur informationen mönstras in i bärvågen, är också avgörande i sammanhanget. Olika moduleringstekniker medför olika mönster i bärvågen. Hur kraftig modulering som används uttrycks i procent och symboliserar hur tydligt mönster informationen resulterar i. NFC bygger på amplitudmodulering vilket innebär att informationen mönstras in som amplitudvariationer. I detta sammanhang betyder alltså kraftig modulering hur stora amplitudvariationer hos bärvågen som informationen kodas med. Kraftig modulering uppåt 100% innebär att bärvågen stundom helt släcks ut. Svagare modulering kring 10% innebär mindre variationer i amplitud hos bärvågen. Det följer att olika moduleringstekniker med olika kraftfull modulering resulterar i signaler med olika egenskaper varav en, hur långt signalen effektivt kan bära information, är relevant i detta sammanhang.

Omgivning spelar en avgörande roll i hur långt radiosignaler kan färdas. Inne i en byggnad finns väggar och andra föremål vilket kraftigt begränsar

signalens förmåga att breda ut sig. Bakgrundsbrus, andra radiovågor från telefoner, radioapparater och övrig elektronik, stör ut signalen och minskar signalens räckvidd.

Alla dessa faktorer gör att det inte är känt vilka avstånd avlyssning kan genomföras på, men experiment har genomförts med framgång på avstånd uppåt 10 meter.

3.1.2.2 Datamodifiering Datamodifiering går ut på att en attackerande enhet försöker störa kommunikationen mellan två NFC-enheter genom att modifiera informationen som skickas mellan enheterna. I sin enklaste form går datamodifiering ut på att störa ut informationsbärande signaler. Mer avancerad datamodifiering går ut på att modifiera signalerna och därigenom informationen som sänds mellan enheterna på ett sådant sätt informationen fortfarande uppfattas som korrekt av mottagaren.

3.1.2.3 Man in the middleättack En så kallad man in the middle attack går ut på att en illasinnad enhet agerar mellanhand mellan sändare och mottagare. Sändare och mottagare handlar i god tro om att de talar direkt till varandra. Mellanhanden får tillgång till känslig information och har dessutom möjligheten att ändra innehållet av datan. En attack av detta slag är i praktiken omöjligt att genomföra på en NFC länk oavsett om kommunikationen sker mellan två aktiva enheter eller en aktiv och en passiv enhet. För att kunna posera som sändare måste mellanhanden stundom aktivt störa ut data som parterna försöker utbyta med varandra och andra gånger försöka skapa en signal som perfekt överlappar befintlig signal. Minsta avvikelse i utstörningen eller överlappningen resulterar i att datan som skickas över länken blir förvrängd och om de två enheterna aktivt lyssnar på länken kan denna förvräning relativt lätt upptäckas.

3.2 Android

Enligt Google(2013) är Android världens mest använda operativsystem för mobila enheter. Android är baserat på Linuxkärnan och kan med hjälp av denna stödja många olika typer av hårdvara. Detta gör Android kompatibel med enheter från många olika tillverkare och det är en av anledningarna till att Android har blivit så dominerande. En annan anledning till dominansen är Androids öppna källkod och dess användande av världens mest använda programspråk, Java(Google, 2013).

För att komma igång och skriva applikationer för Android är det några saker en utvecklare behöver känna till. Förutom grundläggande kunskaper

inom Java-programmering och mjukvaruutveckling i allmänhet bör en utvecklare av Android-applikationer inneha kunskaper inom i huvudsak tre områden.

Först bör en utvecklare inneha kunskaper om den särskilda livscykel, se bilden nedan, en Android aktivitet har. Livscykeln beskriver de olika tillstånd en aktivitet(fotnot 1) kan befinna sig i samt de olika automatiska anrop som sker då aktiviteten tar sig mellan dem. Livscykeln ger ytterligare funktionalitet men också vissa svårigheter så som att viktiga data måste sparas och återställas när användaren till exempel vrider telefonen.

För det andra måste en utecklare av Android-applikationer ha kunskap om de Android-specifika API som Google tillhandhåller. Dessa API beskriver de funktioner som är unika för Android-enheter. Där beskrivs t.ex. hur ett användargränssnitt byggs upp, hur en lokal databas sätts upp och hur applikationen kommunicerar med andra applikationer och med omvärden. Särskilt användbart för projektet är det API som specificerar kommunikation över NFC.

Vidare bör en utecklare också känna till filstrukturen hos ett Androidprojekt, de specifika verktyg för Android-utveckling som finns samt vilka filer som definierar vad. Exempelvis bör kunskap finnas om vilket xml-dokument som definierar vilken vy och hur inställningar påverkar för såväl en aktivitet som för hela applikationen.

3.2.1 NFC för Android

Google (2013) skriver att Android innehåller funktionalitet för att kommunicera via NFC mot två huvudsakliga enheter. Dels finns det funktionalitet för att skriva och läsa från taggar och dels finns det funktionalitet för att kommunicera mot andra enheter vilka har Android som operativsystem. Eftersom detta arbete nyttjar kommunikation mellan två aktiva NFC-enheter är det den senare som är mest relevant. Detta eftersom det är en Androidenhet mikrokontrollern imiterar.

Vidare skriver Google(2013) att det är möjligt att skicka vilken typ av meddelande som helst via NFC men att de rekommenderar meddelanden av typen NDEF (NFC Data Exchange Format) vilken är den av NFC Forum definierade standarden. När ett NDEF-meddelande tas emot hanteras det av en process som är ansvarig för att den inkommande datan skickas vidare till rätt applikation. En applikation kan deklarera vilka typer av data den kan ta emot i dess manifest.

För den eftersökta kommunikationen mellan två Androidenheter används en teknik vilken kallas Android Beam(Google 2013). Denna teknik byg-

ger på utbyten av NDEF-meddelanden. Android Beam fungerar genom att två enheter förs mot varandra vilket förbereder skickandet av ett meddelande. Sändningen slutförs genom att användaren trycker på skärmen. Tekniken sker alltså utan att någon form av parning(fotnot 1) behövs så som är fallet i andra kommunikationsmetoder, exempelvis Bluetooth (`http://www.cs.utexas.edu/~shmat/shmat_fcs07.pdf`).

Google (2013) gör dess Android Beam teknik tillgänglig via ett antal API. Dessa definierar allt ifrån ihopsättande av ett meddelande till hur systemet automatisk förbereder sändande och mottagande. Första steget i användandet av kommunikation via NFC är att få tag på ett objekt som representerar NFC-hårdvaran. Den representeras som ett objekt av typen NFCAdapter. Objektrepresentationen fås via den statiska metoden getDefaultAdapter.

Vidare kan en aktivitet ställas in för sändning genom att en klass implementerar ett särskilt interface, CreateNdefMessageCallback. Implementationen ger klassen förmågan att dynamiskt skapa och skicka meddelanden via Android Beam. För att sedan möjliggöra sändningen från en aktivitet behöver denna välja den implementerade klassen genom att kalla på metoden setNdefPushMessageCallback. För att ta emot meddelandet behövs, förutom att det står angivit i manifestet, metoden onNewIntent skrivas över och i denna behöver meddelandet fångas upp för att sedan tolkas.

fotnot 1 parning eng Pairing

3.3 Arduino

3.3.1 Arduino-specifika källkodsbibliotek

3.3.2 NFC för Arduino mha PN532 NFC/RFID Shield

3.3.2.1 Generella Kommandon till PN532 NFC/RFID Shield

SAMConfiguration

3.3.2.2 Kommandon till PN532 NFC/RFID Shield då enheten agerar målenhet

TgInitAsTarget

TgSetData

TgGetData

4 Kravanalys och design

4.1 Produktkrav

4.2 Kommunikationen mellan enheterna

4.2.1 Kommunikationsförlopp

4.2.2 Meddelandetyper

4.2.3 Säker kommunikation

5 Material och verktyg

5.1 Mikrokontroller

5.1.1 Utvecklingsmiljö och programspråk

5.1.2 Implementerade och intressant protokoll samt specifikationer

5.1.3 Val av källkodsbibliotek för implementation av NFC-stacken

5.1.4 Val av källkodbibliotek för implementation av kommunikationssäkerhet

5.2 Mobilapplikation

5.2.1 Utvecklingsmiljö och programspråk

5.2.2 Använda API:er

6 Genomförande

6.1 Utveckling av låsenheten

6.1.1 RF-protokoll-lagret

6.1.1.1 Kontroll av RF-protokollens funktioner

6.1.2 Kommunikation mellan Arduino och NFC-sköld

6.1.3 LLCP-lagret

6.1.4 NPP- eller SNEP-lagret

6.1.5 NDEF-lagret

6.1.6 Applikationen, den som snurrar i loop()

6.2 Utveckling av mobilapplikationen

6.3 Utveckling av en prototyp bestående av båda enhterna

7 Resultat

7.1 Produkten som helhet

7.2 Mobilapplikationen

Övergripande design

Login

Main

Success och Denied

Settings

Password

Keys

7.3 Låsenheten

8 Diskussion

8.1 Produkten

8.2 Arduino

8.3 Mobilapplikationen

9 Slutsats

Referenser

Bilagor