Prototyp för Bluetooth-baserat lås

Examensarbete D, 20 p.
Institutionen för datavetenskap
Umeå Universitet
2004-04-20

Författare:
Tobias Marklund
dva99tmd@cs.umu.se

Handledare:
Jerry Eriksson

Sammanfattning

Detta är ett examensarbete på 20 poäng under utbildningen Datavetenskapliga programmet vid Umeå Universitet. Det är utfört på uppdrag av Explizit AB i Skellefteå.

Under hösten 2003 utformade Explizit ett projekt med mål att undersöka möjligheterna med att skapa ett system för dörrpassering och nyckelhantering med mobiltelefon. Systemet skulle utnyttja Bluetooth för kommunikation mellan de olika delarna av systemet. Dessutom skulle systemet fungera på så många mobiltelefoner som möjligt som redan finns ute på marknaden.

Fördelarna med ett sådant system skulle vara att man slipper den fysiska hanteringen av nycklar och de problem som uppstår då en nyckel tappas bort. Istället för att vara tvungen att byta ut låsen samt dela ut nya nycklar kan man på ett enkelt sätt spärra nycklar och ge ut nya via t.ex. SMS.

Syftet med det här arbetet var att ta reda på en lämplig metod för att möjliggöra detta system, samt att undersöka eventuella säkerhetsproblem som kan uppstå.

Rapporten innehåller en genomgång av Bluetooth-teknologin, samt en redovisning av kända säkerhetsproblem. Även en utvärdering av användbarheten samt säkerheten i den färdiga prototypen har genomförts.

En analys av det slutliga arbetet visade att lösningen troligtvis inte lämpar sig för dagligt bruk som dörrlås, då tiden det tar att låsa upp en dörr är relativt lång, och handhavandet är alldeles för komplicerat. Dock kan det finnas ett nischområde där t.ex. engångsnycklar kan användas vid bokningar av lokaler eller dylikt. Där finns fördelen att det är lättare att distribuera ut nycklar till kunden än att en fysisk nyckel skall lämnas ut och kvitteras.

Abstract

This project is a Masters Thesis at the Computer Science-program at Umeå University, Sweden. The goal of the project was to investigate how a passage control system that utilises Bleutooth in mobile phones could be realised, and what security problems that exists with such a solution. The final prototype is supposed to work on the majority of mobile phones currently available on the market, and not only with a few select "smart phones". The door units should be wireless, except for power supply, to take full advantage of the Bluetooth technology.

The system uses a decentralized approach with a main server where all rights administration is handled. The server is responsible to contact the door units when a change to the access rights has been made. There are one or several door units which take care of the communication with the users mobile phone and via an electrical signal controls the lock on the door. The door units contain all the information needed to decide what users are authorized to unlock the door. The actual "door key" consist of a vCard that the user sends from their address book to the door unit.

The result is a working prototype that unfortunately is not suitable for every-day use because of the complex usage. Several security issues have also been recognized that could prevent this solution to be of commercial use.

Innehållsförteckning

1 Inledning 1
1.1 Bakgrund 1
1.2 Problem 1
1.3 Metod 1
1.4 Mål 1
1.5 Krav och begränsningar 2

2 Bluetooth 3
2.1 Historia 3
2.2 Radio 4
2.3 Arkitektur 6
2.4 Protokoll 8
2.5 Säkerhet 9

3 Säkerhetsrisker i Bluetooth 15
3.1 Avlyssning 15
3.2 Svaga lösenord 15
3.3 Krypteringens styrka 15
3.4 Bluejacking 16
3.5 Bluesnarfing 16
3.6 Backdoor attack 17

4 Utrustning 19
4.1 Hårdvara 19
4.2 Mjukvara 20

5 Implementation 21
5.1 Designval 21
5.2 Nyckeln 24
5.3 Telefonen 24
5.4 Låsenheten 24
5.5 Serverapplikationen 25
5.6 Användning 26
5.7 Prototypens begränsningar 28

6 Analys 29
6.1 Fördelar och nackdelar 29
6.2 Användarvänlighet 29
6.3 Säkerhet 30

7 Framtida förbättringar 33
7.1 Minne 33
7.2 Bluetooth 1.2 33

7.3 Group keys 33
7.4 Applikationer 34

8 Sammanfattning 37
8.1 Resultat 37
8.2 Personliga reflektioner 37
8.3 Problem 38

9 Tack 39

10 Referenser 41

Förkortningar ..... Appendix A

Figurer

Figur 2-1: a) FDM, b) TDM 5
Figur 2-2: Frequency Hopping Spread Spectrum 6
Figur 2-3: Exempel på en Bluetooth-stacks uppbyggnad 7
Figur 2-4: Piconet och Scatternet 8
Figur 2-5: E_2-algoritmen för generering av länknycklar 10
Figur 2-6: E_1-algoritmen för autenticering av enhet 10
Figur 2-7: E_3-algoritmen för generering av krypteringsnyckel 12
Figur 2-8: E_0-algoritmen för (de)kryptering av data 12
Figur 4-1: WRAP THOR-modul från BlueGiga 19
Figur 5-1: Exempel på vCard 23
Figur 5-2: Systemets arkitektur 24
Figur 5-3: Blockdiagram över serverapplikationen 26
Figur 5-4: Öppning av dörr 27
Figur 7-1: Group keys 34

1 Inledning

Detta arbete är ett examensarbete på den Datavetenskapliga Institutionen vid Umeå Universitet [17].

Arbetet är utfört åt, och hos, Explizit AB i Skellefteå. Explizit är ett konsultföretag med inriktning på teknisk programmering och har erfarenhet av trådlös kommunikation, client/server samt intra-/Internet-lösningar. [18]

1.1 Bakgrund

I dagens samhälle har vi allt fler nycklar till olika dörrar att hålla reda på. Dessa nycklar är svåra att administrera, då en förlorad nyckel ofta betyder att man måste byta hela lås, och därmed distribuera ut helt nya nycklar till alla andra som har tillgång till dörren. Detta har till viss del åtgärdats genom uppkomsten av passerkortssystem. Dessa system innebär fortfarande att ett stort antal kort skall hanteras, och man har själv en stor andel kort att hålla reda på.

Nuförtiden har nästan var man en egen mobiltelefon, och med uppkomsten av bl.a. Bluetooth i dessa telefoner har användningsområdet utökats väsentligt.

1.2 Problem

Hur kan man utforma ett låssystem som använder sig av Bluetooth så att det fungerar att använda med dagens mobiltelefoner? Vilka säkerhetsrisker finns det med en sådan lösning och går det att göra det användarvänligt?

1.3 Metod

Arbetet är uppdelat i en teoretisk, en praktisk och en analysdel. I den teoretiska delen undersöks Bluetooth-teknologin och vilka säkerhetsrisker det finns. Därefter diskuteras vilka alternativ som finns och hur en sådan här lösning kan realiseras. I den praktiska delen implementeras en prototyp av ett system som använder sig av den Bluetoothlösning som är baserad på en modul utvecklad av Explizit. I analysdelen kommer lösningen att testas och utvärderas.

1.4 Mål

Målet med detta arbete var att undersöka möjligheten med, och implementera en prototyp av, ett lås som kan användas via Bluetooth. Tanken bakom detta är att användare som har en mobiltelefon som har Bluetooth-kapabilitet skall kunna använda den för att låsa upp dörren. Fördelarna med ett sådant system skulle vara att det är lätt att distribuera nycklar samt att en borttappad nyckel inte innebär att man måste byta ut låskolven samt ett stort antal nycklar. Istället drar man bara in rättigheterna för den specifika nyckeln och skickar ut en ny nyckel till användaren.

1.5 Krav och begränsningar

Ett av huvudkraven som det här projektet hade var att lösningen inte endast skall fungera på ett fåtal dyra telefoner eller så kallade ”smartphones”, utan det bör kunna användas med alla tillgängliga telefoner som har Bluetooth inbyggt utan att behöva ändra i mobiltelefonens grundläggande operativsystem och funktioner.

Lösningen bör tillåta att nya nycklar går att skicka till en användare på distans, till exempel via SMS. Detta behöver dock inte implementeras i prototypen.

Lösningen skall vara helt trådlös, förutom eventuella strömförsörjningar till de olika delarna.

2 Bluetooth

Detta kapitel innehåller en beskrivning av Bluetooths historia, en genomgång av hur teknologin fungerar samt en beskrivning av de olika profilerna som används i detta projekt.

2.1 Historia

Bluetooth är en standard för trådlös kortdistanskommunikation samt trådlösa ad-hoc nätverk. Standarden var ursprungligen föreslagen och utvecklad av Ericsson, som påbörjade sina studier och utvärderingar redan 1994. Till slut föll namnvalet på Bluetooth, efter den danske vikingakungen Harald Blåtand som fredligt enade de två skandinaviska kungadömena Danmark och Norge. Motsvarande skulle Bluetooth ena telekommunikation och datoranvändande på ett smärtfritt sätt. [2]

Standarden blev snart accepterad av flera stora intressenter. Nokia, IBM, Toshiba och Intel bildade slutligen i februari 1998 Bluetooth SIG (Special Interest Group). Denna grupp kontrollerar utvecklingen av standarden, samt utser vilka aktörer som skall få rätt att testa och godkänna produkter som vill få ett officiellt godkännande. SIG släppte den första färdiga specifikationen på standarden i juli 1999.

SIG är uppdelat i tre olika medlemsklasser:

- *Promoter members*
  Detta är de företag som är med och leder SIG. För närvarande är dessa 3Com, Ericsson, IBM, Intel, Lucent Technologies, Microsoft, Motorola, Nokia och Toshiba.
- *Associates*
  Dessa medlemmar har tillgång till information om nya utvecklingar av standarden i ett tidigt skede. De kan även bli medlemmar i de olika arbetsgrupperna inom SIG.
- *Adopters*
  Har tillgång till en Bluetooth-licens som innefattar tillgång till patent som de kan använda utan royalties. De har också tillåtelse att använda Bluetooths symbol och namn på sina produkter.

Ett av de stora målen med SIG var att uppnå interoperabilitet mellan olika tillverkares enheter, så länge de använder samma profil när de kommunicerar. De olika profilerna är en del av Bluetooth-standarden och varje enhet måste testas mot en eller flera profiler för att bli certifierad.

Bluetooth arbetar i det licensfria frekvensspektrumet på 2.4 GHz kallat ISM-bandet (Industrial, Scientific, Medical). Eftersom detta spektrum har släppts fritt arbetar många andra teknologier i det, däribland bl.a. mikrovågsugnar samt b och g-varianterna av den största standarden för trådlösa nätverk: IEEE802.11. Detta har ställt stora krav på utformningen av standarden för att undvika att störa ut dessa andra

teknologier, samt att själv slippa bli utstörd.

Bluetooth har alltid inriktats mot att bli en standard som skall möjliggöra att små, strömsnåla enheter skall kunna utvecklas utan att prispremien skall bli alltför hög. Ett mål som har drivits länge är att få ner kostnaden för att lägga till Bluetooth i en produkt till under 5$/enhet. Detta mål trodde man att man skulle uppnå rätt snabbt, men det är först nu som det verkligen kan sägas att det har lyckats.

Bluetooth har nu börjat bli allt vanligare i mobiltelefoner och PDA:er, och har många användningsområden. De vanligaste är trådlösa headsets till mobiltelefoner samt för att synkronisera sin kalender/adressbok i sin PDA med den i sin stationära dator.

2.2 Radio

Bluetooth arbetar som sagt inom det licensfria ISM-bandet som ligger i frekvensområdet 2400 MHz till 2483.5 MHz. Inom detta område återfinns vattenmolekylens naturliga svängningsfrekvens på 2450 MHz. Mikrovågsugnar arbetar på just den frekvensen för att få vattenmolekylerna i mat att svänga snabbare och därmed alstra värme. Många är därmed oroliga att eftersom Bluetooth använder samma frekvens så är det risk att man ”kokar” användaren. Viss strålning tas också upp av huden, men eftersom sändningsstyrkan endast är ca 1/1000000 av en mikrovågsugns effekt och inte koncentrerad på en enda punkt så kan man inte upptäcka någon mätbar upphettning av huden. Vid maximal sändningsstyrka (100 mW) kan vågorna nå 1.5 cm in i kroppen. Som jämförelse kan nämnas att en GSM-telefon som arbetar på 900 MHz-bandet har en penetrationsförmåga på 2.5 cm och kan ge en mätbar temperaturhöjning på 0.1 C°. [7]

Eftersom Bluetooth är designat för att vara en kortdistanslösning är sändarstyrkan relativt låg jämfört med t.ex. mobiltelefoner. Det finns definierat tre olika sändningsstyrkor:

- Klass 1: 100 mW (20 dBm)
  Klarar upp till 100 m.

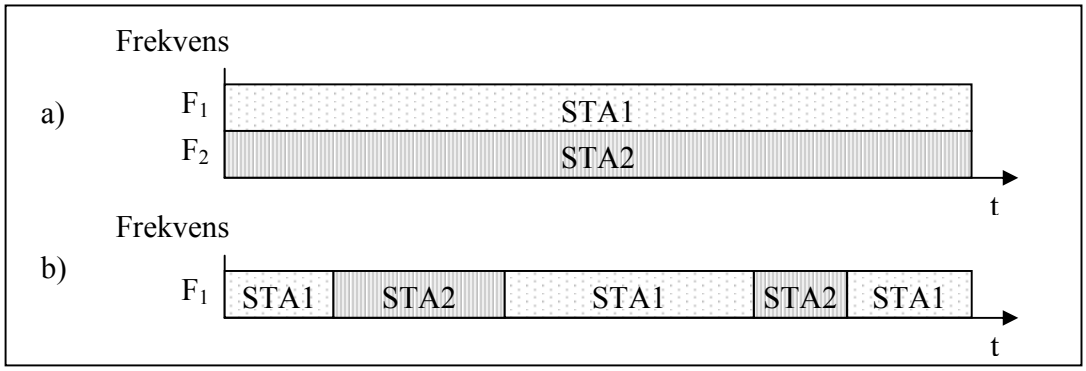
- Klass 2: 2.5 mW (4 dBm)
  Klarar upp till 25 m

- Klass 3: 1 mW (0 dBm)
  Klarar upp till 10 m

De vanligaste sändningsstyrkorna är i dagsläget klass 2 och 3, men klass 1-enheter börjar komma allt mer, särskilt som ”dongles” till PC.

Eftersom radiovågor hindras av såväl människokroppen samt av väggar är de verkliga uppnådda avstånden något kortare än de maximala som anges. Under optimala förhållanden kan man dock uppnå ännu längre avstånd än de specificerade.

Andra faktorer som kan påverka en enhets räckvidd är hur många störkällor som befinner sig inom samma område. Detta inkluderar bl.a. mikrovågsugnar, och trådlösa nätverk (IEEE 802.11b/g).



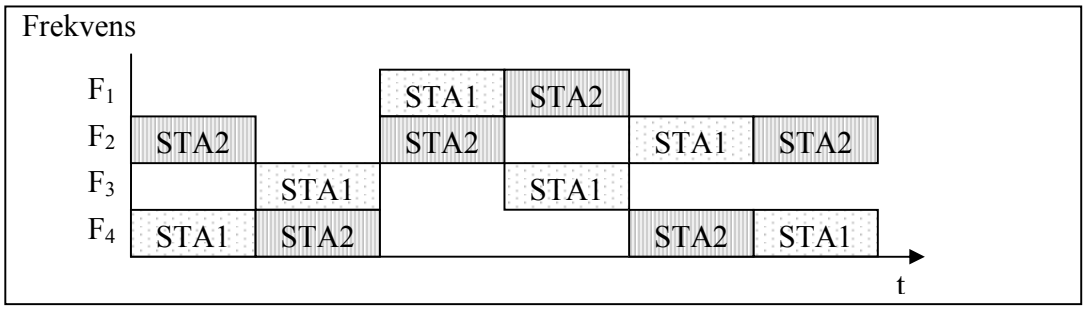
Figur 2-1: a) FDM, b) TDM

ISM-bandet är uppdelat i 79 kanaler (i Frankrike tillåts dock endast 23 kanaler) där bandbredden på varje kanal är 1 MHz. Många av de system som använder sig av ISM-bandet använder så kallad Frequency Division Multiplexing (FDM) i kombination med Time Division Multiplexing (TDM). FDM kan jämföras med hur radio- och TV-sändningar fungerar där en kanal har fullständig tillgång till en frekvens. I detta fall har ett system egen tillgång till en kanal. Detta ställs in vid installation av systemet så att det inte stör något annat system. TDM innebär att kommunikationen delas upp i tidsluckor där en enhet som vill sända först måste reservera en tidslucka för att sända. Detta kan liknas vid en modererad diskussion där varje talare får en viss tid på sig att tala. Sänder flera stationer samtidigt kommer dessa sändningar att krocka och i värsta fall kommer inte någon att nå sina tänkta mottagare. Detta är ett problem i områden med många olika system som skall kommunicera samtidigt. Om antalet kanaler som behövs blir för stort kommer vissa helt enkelt inte kunna få plats i etern. Om två eller flera system försöker använda samma kanal kommer kollisioner troligtvis att ske väldigt ofta. De enheter som använder sig av denna teknik ändrar väldigt sällan vilken kanal de arbetar på. Kanalerna i ISM-bandet är dock sällan maximalt utnyttjade, och det kan stundtals gå långa stunder där ingen sändning sker.

För att förhindra att bli utstörd av andra apparater använder Bluetooth sig av en teknik som kallas Frequency Hopping Spread Spectrum (FHSS). Detta är en teknologi som bl.a. skådespelerskan Hedy Lamarr utvecklade under andra världskriget. Deras mål med frekvenshoppning var att skapa ett hemligt kommunikationssystem för att kontrollera torpeder med hjälp av radio. [9]

Till skillnad från FDM där all kommunikation endast sker på en kanal fungerar FHSS på så sätt att en enhet sänder på en kanal under en kort tidsrymd och ”hoppar” sedan vidare till en annan kanal där den sänder en kort tid och så vidare. Hoppen sker 1600 gånger per sekund och använder sig av en pseudo-slumpmässig hoppsekvens som baseras på den ena kommunicerande enhetens interna klocka samt dess adress. [9]

Denna metod utnyttjar det faktum att kanalerna sällan är fullt utnyttjade, vilket gör att kollisioner sällan kommer att ske. Om en kollision sker är sannolikheten att det ska ske ännu en krock på nästa kanal i hoppsekvensen ännu mindre.



Figur 2-2: Frequency Hopping Spread Spectrum

Fastän FHSS kan minska antalet krockar i etern så finns det ändå en möjlighet att ett antal kanaler är så utstörda av andra enheter att det blir en krock varje gång dessa kanaler används. Detta skulle kunna resultera i många onödiga omsändningar och överföringshastigheten skulle bli lidande. I version 1.2 av Bluetooth-standarden har man därför tagit fram en metod för att få enheterna att anpassa sig efter rådande sändningsförhållanden. Detta kallas för Adaptive Frequency Hopping. Enkelt beskrivet så innebär metoden att om dåliga sändningsförhållanden upptäcks på en viss kanal så utesluter enheterna denna kanal från den hoppsekvens de använder när de kommunicerar.

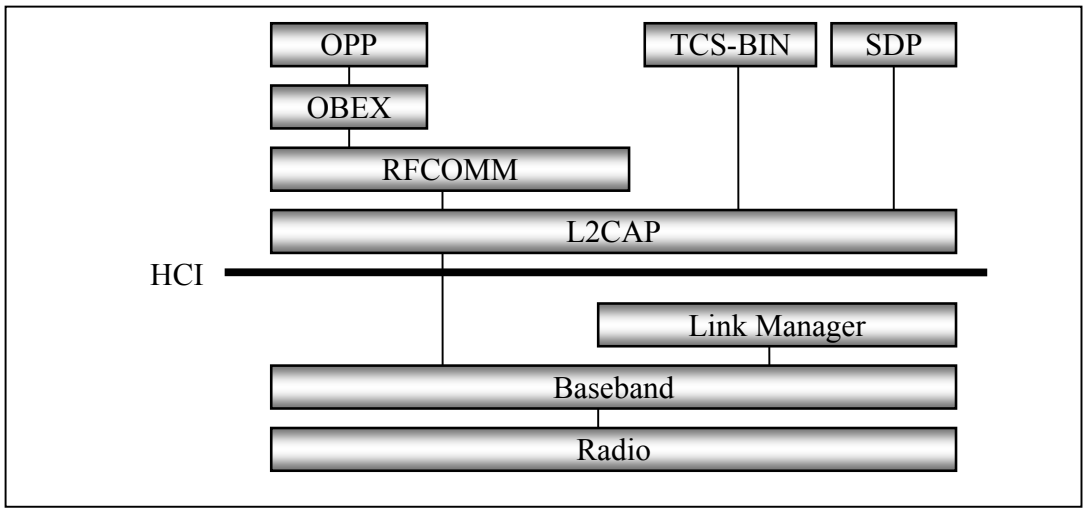
2.3 Arkitektur

För att Bluetooth skall kunna användas mellan enheter som har olika tillverkare innehåller inte specifikationen endast riktlinjer för radiodelen, utan även för en mjukvarustack som tillåter användare att söka efter enheter, se vilka tjänster de erbjuder samt använda dessa tjänster. Bluetooth-stacken består av flera olika lager (*Figur 2-3*) där varje lager är beroende av lagret under.

- Radiolagret modulerar och demodulerar data när det sänds genom luften.
- Basbandet sköter frekvenshoppningen och uppdelningen av data i paket inför sändning.
- Link controllern hanterar de uppkopplingar man gjort mellan enheter.
- HCI-lagret (Host Controller Interface) sköter kommunikationen mellan en Bluetooth-enhet och t.ex. en separat värddator.
- L2CAP (Logical Link Control and Adaptation Protocol) multiplexar data till högre lager, samt konverterar mellan olika storlekar på datapaket.
- Högre lager tillhandahåller funktionalitet till användaren, t.ex. serieportsemulering via RFCOMM.

Man behöver inte ha hela stacken på en och samma enhet, utan man kan använda sig av en så kallad HCI-split. I denna lösning innehåller enheten endast en stack som

sträcker sig upp till HCI-lagret. Man har då en del av HCI-lagret på enheten och en del på en kontrollerande enhet, t.ex. en extern mikroprocessor. Mellan dessa två enheter skickas då HCI-kommandon över t.ex. USB eller UART.

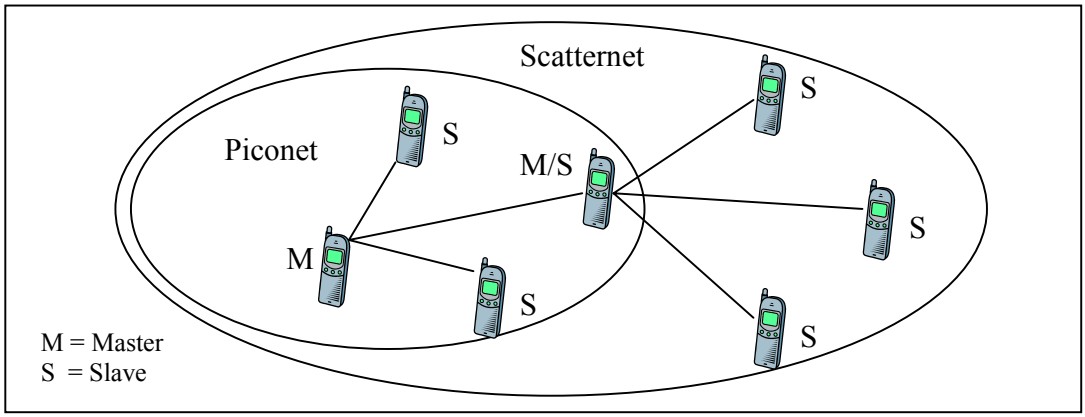


Figur 2-3: Exempel på en Bluetooth-stacks uppbyggnad

Varje enhet har en unik 48 bitar lång adress kallad BD_ADDR. De har även en intern klocka som används för timing vid frekvenshoppning och sändningar.

Vid kommunikation mellan två enheter är alltid en enhet ”master” och den andra ”slave”. Vem som är vad är oftast rätt ointressant, och de kan faktiskt byta roll när de vill. De enda privilegier en master har är att den är ansvarig för synkroniseringen av FHSS-kommunikationen. Den bestämmer vilken hoppsekvens som skall användas baserat på sin BD_ADDR samt sin interna klocka. För det mesta är det enheten som önskar göra en uppkoppling som tilldelas att bli master. [2]

Då en enhet som agerar som master är uppkopplad mot flera slavar samtidigt säger man att dom bildar ett ”piconet” (*Figur 2-4*). I ett piconet är det mastern som bestämmer när de andra slavarna får sända. Om två slavar vill skicka information mellan varandra måste de skicka den via mastern, alternativt att de kopplar ihop sig mot varandra och därmed skapar ett eget piconet. Alla slavar måste synkronisera sig mot masterns interna klocka och kommer att följa den hoppsekvens som mastern bestämmer. Då en enhet är medlem i två eller fler olika piconet samtidigt säger man att man har skapat ett ”scatternet” (*Figur 2-4*). En enhet kan då antingen vara slav i bägge näten, eller slav i ena och master i det andra. För att kommunicera mellan enheter i ett sådant nät kan det behövas tekniker för att hitta var rätt mottagare finns, så kallad routing. [2]



Figur 2-4: Piconet och Scatternet

En enhet kan som mest ha sju aktiva uppkopplingar mot andra enheter samtidigt.

2.4 Protokoll

Här beskrivs översiktligt de viktigaste protokollen som används av prototypen i denna rapport.

2.4.1 SDP

SDP står för Service Discovery Protocol. Detta protokoll används för att leta efter en viss tjänst på en annan enhet, och för att få information om hur man kopplar upp sig mot just den tjänsten. I ett trådbundet nätverk kan motsvarande information vara fast inställt, t.ex. att man skall använda en skrivare med en viss adress. I Bluetooth kan infrastrukturen ändras väldigt fort, och om man skulle vilja skriva ut något så måste man först göra en sökning där man undersöker vilka enheter som stödjer utskrifter. SDP kan då tillhandahålla en lista över vilka tjänster en enhet stödjer.

2.4.2 RFCOMM

Då Bluetooth var tänkt att ersätta kablar så behövdes ett protokoll som fungerade på samma sätt som serieportarna på en vanlig dator. RFCOMM är ett seriellt protokoll som utvecklades för att tillhandahålla kompabilitet med applikationer som tidigare använde serieporten för att kommunicera. Namnet på protokollet kommer från att det är ett trådlöst (RF, Radio Frequency) protokoll som är kopplat till en virtuell COM-port. Protokollet baseras på en standard från GSM-teknologin kallad TS 07.10.

RFCOMM utgör basen för många andra protokoll och profiler, just därför att man kan utveckla ett protokoll som använder RFCOMM och relativt lätt kunna konvertera det till att använda en vanlig kabelbaserad serieport.

2.4.3 OBEX

OBEX står för OBject EXchange och är en standard som ursprungligen kommer från IrDA-standarden för infraröd kommunikation. Protokollet används, som namnet antyder, för att utbyta objekt såsom bilder, telefonnummer osv. Att man valt att använda samma protokoll som det befintliga för IrDA är för att man lätt ska kunna

konvertera applikationer över till Bluetooth och på så sätt lättare skapa stöd för standarden.

Profiler som bygger på OBEX inkluderar bl.a. Object Push Profile (OPP) som används för att skicka objekt (t.ex. kontakter, bilder m.m.), samt Synchronization Profile som används för att synkronisera informationen i t.ex. adressboken och/eller kalendern med informationen på t.ex. sin stationära dator.

2.5 Säkerhet

Säkerheten är väldigt viktig i Bluetooth, såsom hos alla teknologier som baseras på radiokommunikation. I princip kan vem som helst som t.ex. äger en telefon med Bluetooth-kapabilitet koppla upp sig mot en enhet och ta del av den personliga information som finns där om den inte är ordentligt säkrad.

Vid designen av Bluetooth lade man en hel del vikt på säkerheten i protokollet samtidigt som protokollets lageruppbyggnad möjliggör för användaren att välja på vilken nivå han vill lägga säkerheten. Den grundläggande funktionaliteten möjliggör autenticering, auktorisering och kryptering på stacknivå.

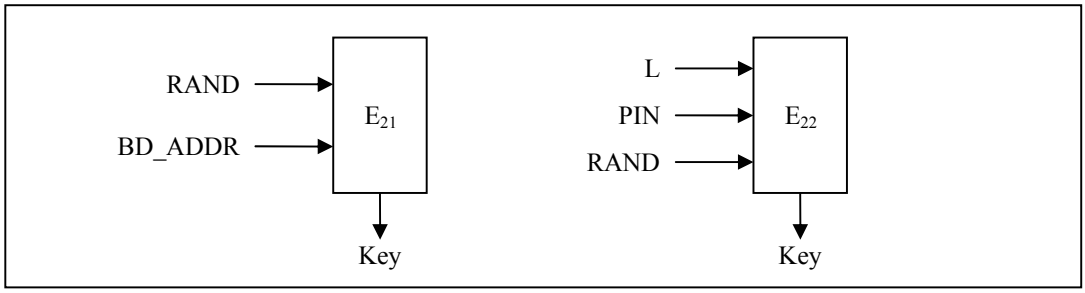
2.5.1 Autenticering

Autenticering används i de fall där man vill vara säker på att den enhet man kommunicerar med verkligen är den enhet som man avser att prata med. Att titta på den adress som enheten anger är en väldigt simpel form av autenticering, men även en väldigt lätt sak att förfalska. I de flesta fall krävs något utöver detta för att nå en acceptabel säkerhet.

Bluetooth använder sig av en metod där man ”parar ihop” två enheter för att etablera ett förtroende mellan dem. Två enheter som är ihopparade kommer att innehålla en s.k. länknyckel som används för att autenticera varandra. Med mobiltelefoner går parning till på så sätt att en användare startar en sökning efter de enheter som finns i närheten och väljer sedan den enhet som man vill para ihop, baserat på namnet på enheten. En uppkoppling kommer då att ske där en PIN-kod kommer att efterfrågas för att parningen skall kunna genomföras. Användaren slår in sin PIN-kod, och om koden överensstämmer med den kod som den andra enheten har angett (antingen som en fast PIN-kod eller via användarinteraktion) så kommer enheten att läggas till som godkänd enhet och länknyckeln kommer att sparas för framtida kommunikation.

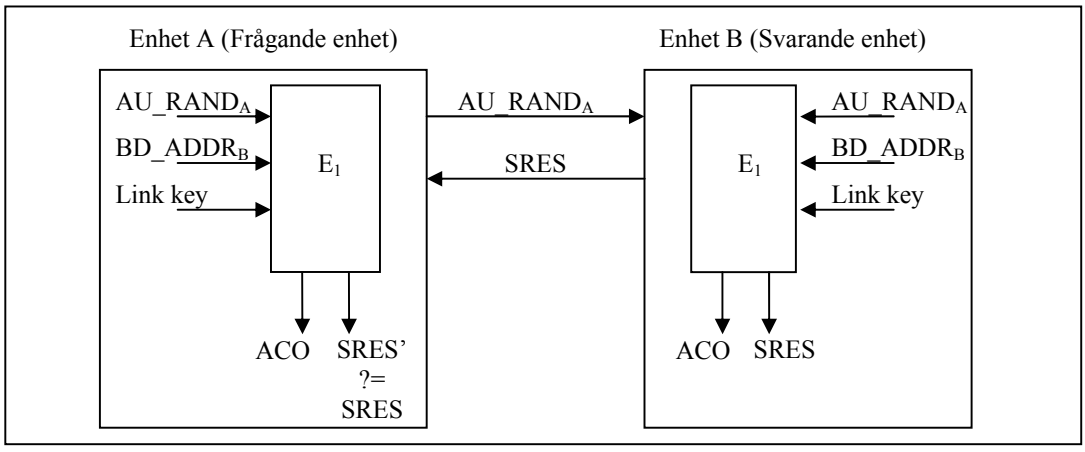
Mer ingående så skapas vid parningen mellan två enheter först en initialiseringsnyckel som används för att kryptera informationen som används för att skapa länknyckeln:

> En av enheterna (A) slumpar fram ett tal, IN_RAND, som den skickar över i klartext till den andra enheten. Bägge enheterna genererar sedan initialiseringsnyckeln K_{init} genom funktionen E_{22} (*Figur 2-5*) som använder PIN-koden, längden L av PIN-koden (i byte), adressen till enhet B som tog emot det slumpade talet, samt det slumpade talet. [2]



Figur 2-5: E_2-algoritmen för generering av länknycklar

En tvåvägsautenticering sker sedan för att kontrollera att bägge enheterna har kommit fram till samma nyckel genom att enhet A först genererar ett slumptal, AU_RAND_A, som med den andra enhetens adress och initialiseringsnyckeln krypteras i funktionen E_1 (*Figur 2-6*). Det slumpade talet skickas samtidigt över till den andra enheten som applicerar samma funktion på talet med sin egen adress och initialiseringsnyckeln. Resultatet SRES skickas sedan tillbaka till den första enheten som jämför resultatet med sitt eget beräknade resultat SRES'. Om dessa tal överensstämmer har autenticeringen lyckats. Den andra enheten genomför sedan samma procedur för att verifiera den första enheten. E_1 genererar dessutom ett tal kallat ACO (Authenticated Ciphering Offset) som används vid skapandet av en krypteringsnyckel. Detta beskrivs utförligare senare i rapporten. [2]



Figur 2-6: E_1-algoritmen för autenticering av enhet

Nästa steg är att skapa en kombinationsnyckel K_{AB} som används för framtida autenticeringar mellan enheterna. Detta går till så att varje enhet slumpar fram var sitt slumptal, LK_RAND_A och LK_RAND_B. De applicerar sedan algoritmen E_{21} på detta tal tillsammans med sin egen BD_ADDR. Detta genererar varje enhets tillskott till kombinationsnyckeln. Nästa steg är att enheterna på ett säkert sätt utbyter LK_RAND_A och LK_RAND_B genom att applicera en bitvis XOR-

operation på talet och den aktuella länknyckeln K. Sker detta under en parning är $K=K_{INIT}$. De båda enheterna upprepar därefter E_{21}-algoritmen med den andra enhetens adress och slumpade tal. Det resulterande talet kombineras sedan med det första talet genom en XOR-operation för att skapa den resulterande länknyckeln K_{AB}. Därefter upprepas autenticeringsproceduren som beskrevs vid skapandet av K_{INIT} för att säkerställa att bägge enheterna har kommit fram till samma tal. [2]

Vid framtida uppkopplingar mellan två hopparade enheter kommer autenticeringen att ske direkt med hjälp av länknyckeln K_{AB} utan att någon PIN-kod behöver anges.

2.5.2 Auktorisering

Auktorisering handlar om att ta reda på vem som har tillgång till en viss tjänst. Detta är ofta tätt sammanbundet med autenticering, men kan även användas utan det. I vissa applikationer kan det hända att man har flera olika tjänster på en enhet, men endast vissa enheter har tillåtelse att koppla upp sig mot en viss tjänst. Man kan då ha en lista över godkända enheter för varje tjänst. Om en enhet inte finns med i listan kommer den att nekas tillträde till den aktuella tjänsten.

2.5.3 Kryptering

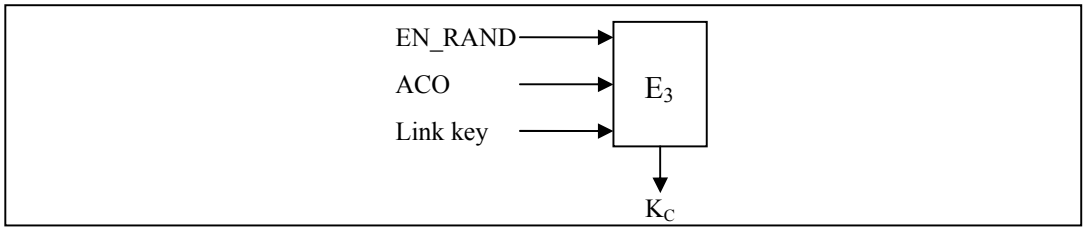
Eftersom trådlösa nätverk är möjliga att avlyssna utan att parterna som kommunicerar är medvetna om det, så ställer det krav på att känslig information som skickas inte skall vara möjlig att komma över på ett lätt sätt. Därför kan man vid behov kryptera trafiken.

Bluetooth använder sig av en krypteringsalgoritm kallad E_0. Det är en stream-baserad algoritm som är baserad på den befintliga algoritmen SAFER+ (Secure And Fast Encryption Routine). Den utvecklades av Cylink Corporation och var en av kandidaterna till USA:s Advanced Encryption Standard (AES), den standard som alla amerikanska myndigheter kommer att använda för att skydda sin data. Algoritmen blev dock inte vald och släpptes senare ut till allmänheten och är numera helt öppen för inspektion. [7]

E_0 är en symmetrisk algoritm, vilket innebär att samma nyckel används av bägge parter för att både kryptera och dekryptera data. Standarden har hittills definierat stöd för krypteringsnycklar med storlekar mellan 8 och 128 bitar. Vilken storlek som skall användas måste förhandlas fram mellan de kommunicerande parterna. Säkerheten i denna algoritm sägs i dagsläget vara bra för nyckelstorlekar på 128 bitar. [2]

För att kunna kryptera en uppkoppling måste först en autenticering ha skett mellan de kommunicerande enheterna. Som tidigare nämnts så fick man som en restprodukt av E_1-algoritmen ut värdet på ACO. Detta värde används vid genereringen av en krypteringsnyckel, $K_{C,}$ med hjälp av algoritmen E_3 (*Figur 2-7*).[1] [2]

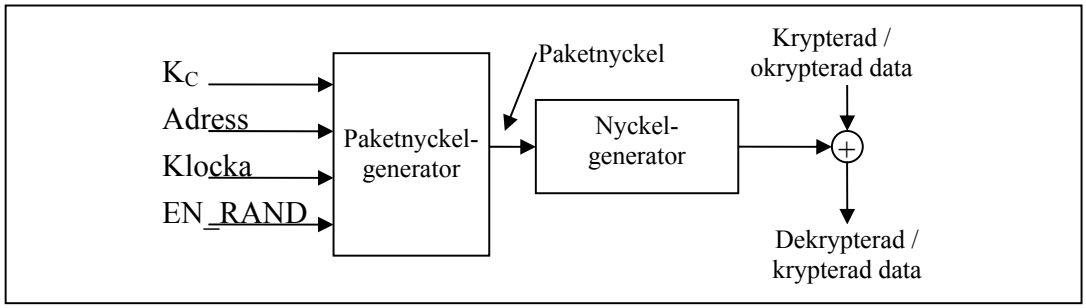
[1] Detta exempel är endast i de fall två enheter talar direkt med varandra. I de fall kryptering och kommunikation sker i ett piconet kommer E_3 att användas på ett lite annorlunda sätt.



Figur 2-7: E_3-algoritmen för generering av krypteringsnyckel

Nyckeln K_C är den nyckeln som kommer användas vid kryptering och dekryptering av data som skickas mellan enheterna. Då denna nyckel är 128 bitar lång från början kommer den att kortas av till rätt bitlängd om enheterna har kommit överens att använda en kortare bitlängd.

Algoritmen fungerar så att för varje paket som skall skickas genereras en paketnyckel baserad på K_C, det slumpade talet EN_RAND, master-enhetens adress och klocka. Denna paketnyckel används för att generera en nyckelström som används i en bit-operation för att kryptera datat i paketet. Dekryptering fungerar på samma sätt, fast där nyckelströmmen används för att dekryptera det krypterade datat.



Figur 2-8: E_0-algoritmen för (de)kryptering av data

2.5.4 Säkerhetsnivåer

För att kunna anpassa säkerheten efter sin applikation tillhandahåller Bluetooth tre säkerhetsnivåer.

- Säkerhetsnivå 1 är en osäker nivå. I detta läge kommer ingen autenticering, auktorisering eller kryptering äga rum.
- Säkerhetsnivå 2 ger säkerhet på tjänstnivå. Här kan programmeraren tilldela olika säkerhetskrav till olika tjänster. Inga säkerhetsåtgärder kommer att vidtas förrän en L2CAP-uppkoppling skett. Därefter kommer enheten att bestämma om den behöver autenticering, auktorisering eller kryptering och gå igenom dessa steg om nödvändigt.
- Säkerhetsnivå 3 ger säkerhet på länk-nivå. En enhet kommer därmed att vidta säkerhetsåtgärder när en uppkoppling sker. Om säkerhetskraven inte kan uppfyllas av den andra enheten kommer uppkopplingen inte att tillåtas. I detta

läge kan man få en enhet att endast tillåta uppkopplingar med tidigare ihopparade enheter.

För att ytterligare skydda en enhet kan den t.ex. sättas i ett läge där ingen annan enhet kan koppla upp sig mot den när den inte används. Man kan även sätta den i icke sökbart läge, där enheten fortfarande accepterar uppkopplingar från enheter som känner till dess adress, men den svarar inte på sökningar från andra enheter. [7]

Har man ytterligare säkerhetskrav på sin applikation än de som tillhandahålls av stacken så är det fullt möjligt att lägga in sina egna lösningar på en högre nivå.

3 Säkerhetsrisker i Bluetooth

Bluetooth har länge förespråkats som en säker lösning för kortdistanskommunikation. Under senare tid har dock mer eller mindre allvarliga säkerhetsproblem uppdagats. Här tas en del av dessa upp.

3.1 Avlyssning

Genom att utnyttja frekvenshoppningstekniken är Bluetooth ganska väl skyddat för så kallad passiv avlyssning. Om en attackerare inte känner till hoppsekvensen måste hela frekvensspektrat avlyssnas och analyseras, vilket är ett relativt orimligt scenario. Om en attackerare är närvarande då en uppkoppling och synkroniseringen mellan enheterna sker är det dock förhållandevis lätt att avlyssna. Ett exempel på en sådan utrustning är protokollanalysatorn Merlin från CATC [21], som kan avlyssna all trafik i ett piconet. Ett sätt att förhindra detta är att använda sig av kryptering, vilket gör att analysatorn inte kan räkna ut hoppsekvensen. Om länknyckeln däremot är känd kan analysatorn även avlyssna krypterade meddelanden.

Vid trådlös kommunikation är avlyssning mycket svårt att skydda sig mot eftersom det är svårt att begränsa radiovågornas utbredning. Dessutom är det omöjligt att se om någon avlyssnar kommunikationen eftersom en attackerare inte behöver sända något.

3.2 Svaga lösenord

Detta är inte ett specifikt problem för just Bluetooth, utan för alla applikationer där ett lösenord skall skydda känslig information. I Bluetooth är det dock ett mera påtagligt problem, då lösenordet oftast matas in på en enhet med ett begränsat användargränssnitt. På en mobiltelefon har man vanligtvis endast siffertangenterna till hjälp vid inmatning av en PIN-kod. Allt som oftast är dessutom koden relativt kort, endast 4 siffror. Detta ger endast 10 000 olika kombinationer, vilket är en munsbit för dagens datorer att knäcka.

3.3 Krypteringens styrka

Krypteringens styrka är i högsta grad beroende av bitlängden på krypteringsnyckeln. Fastän Bluetooth stöder 128 bitars kryptering så är inte detta ett krav. I själva verket stöder många enheter bara 56 bitars kryptering, och vissa klarar inte ens det. Vilken styrka som används vid kommunikation mellan två enheter förhandlas fram vid uppkopplingsfasen. Eftersom nyckeln får vara allt mellan 1-16 bytes lång (8-128 bitar) så måste man sätta en gräns någonstans för att få en acceptabel säkerhet.

En 56-bitars nyckel är i dagsläget inte klassat som säkert och går relativt snabbt att knäcka med endast vanliga hemdatorer med hjälp av brute force.

Undersökningar av E_0-algoritmen har dessutom visat att en divide-and-conquer-attack skulle kunna vara möjlig vilket skulle reducera problemet till en komplexitet på $O(2^{64})$

för en 128-bitars nyckel under vissa förutsättningar. Detta har man dock omöjliggjort i designen av Bluetooth, då det sker omsynkroniseringar i nyckelgenereringen för varje paket [1]. Vissa forskare vidhåller dock att den verkliga säkerheten på krypteringen inte kan överstiga den för en 100-bitars nyckel [16].

3.4 Bluejacking

I och med att allt fler telefoner nu har Bluetoothkapacitet får en företeelse som kallas ”bluejacking” fler och fler anhängare. Ordet bluejacking kommer från en sammanslagning av ”bluetooth” och ”hijacking” (kapning). Man kan därmed få intrycket av att detta fenomen innebär att ens mobiltelefon kan bli ”kapad” av någon annan. Egentligen är det i grunden en standardfunktion i OBEX-protokollet för Bluetooth som innebär att man kan skicka visitkort från sin telefon till en annan telefon även om man inte har parat ihop dem. I detta hänseende skickar man dock oftast inte ett vanligt visitkort med sina personliga uppgifter, utan istället har man skrivit in ett meddelande i visitkortet till mottagaren, som oftast är ovetande om vem avsändaren är. Ingen information skickas dock från mottagarens telefon, och mottagaren kan välja att läsa det, spara det eller kasta det direkt. [8]

I princip går det till så att man i sin dator/PDA/telefon skapar ett visitkort där man istället för namnet skriver in ett meddelande till mottagaren, går till ett ställe där man kan tänka sig att det finns telefoner som har Bluetooth påslaget, t.ex. ett café, söker efter telefoner i närheten och skickar iväg sitt meddelande samtidigt som man diskret ser sig om för att se om man kan se vem det är man skickat till. Man skulle kunna tänka sig att personen som fått meddelande ser konfunderad och förvånad ut. [8]

Detta fenomen är egentligen inte något nytt påfund, eftersom exakt samma sak går att göra mellan t.ex. Palm-handdatorer med IR-port som använder samma OBEX-protokoll. Det har dock blivit betydligt lättare eftersom man inte längre behöver fri sikt, vilket IR kräver, och allt fler telefoner börjar ha Bluetooth inbyggt.

Då ovanstående metod inte innebär en säkerhetsrisk i sig själv så finns det en variant av denna attack som kan visa sig vara betydligt allvarligare för en icke van användare. Varianten använder sig av det faktum att en enhet kan ges ett namn ända upp till 248 tecken. Detta tillåter en attackerare att starta en parningsprocedur där han istället för namnet på mobilen anger ett meddelande till mottagaren där denne uppmanas att till exempel trycka ”yes” för att vinna ett pris. För många människor som är ovana med mobiltelefoner och dylikt kan det vara möjligt att man tror att meddelandet är genuint. När personen väl har tryckt ”yes” paras telefonerna ihop och attackeraren har därmed tillgång till informationen på den drabbades telefon. För att skydda sig från eventuella risker vid en sådan bluejacking-attack är det bara att trycka ”no” om man får ett oväntat meddelande på sin telefon. [14]

3.5 Bluesnarfing

Under arbetet med detta projekt började det även komma fram rapporter och rykten på nätet om så kallad ”bluesnarfing”. Detta är en mer allvarlig attack mot telefonen vilket

skulle kunna tillåta attackeraren att få tillgång till känslig information. Tillverkarna av mobiltelefoner förnekade till en början alla rykten och menade att utförlig testning visade att det inte var möjligt. [14]

I början av februari 2004 medgav dock både Ericsson och Nokia att några av deras mobiltelefoner var sårbara för bluesnarfing-attacker. På dessa modeller skulle en attackerare kunna läsa, modifiera och kopiera all data i telefonens adressbok och kalender utan att telefonens ägare skulle märka det. De båda tillverkarna har vid skrivande stund inte föreslagit några andra sätt att säkra telefonen än att slå av Bluetooth helt. [12] [13]

3.6 Backdoor attack

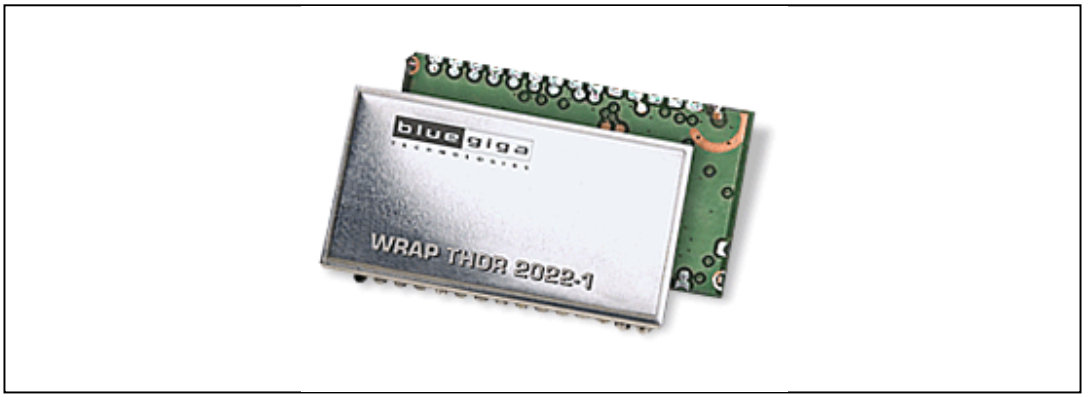
Ännu en allvarlig attack har upptäckts nyligen. Denna attack kräver dock fysisk tillgång till telefonen som attackeras för en kort stund. Telefonen är sedan vidöppen för attacker utan att ägaren upptäcker det. [14]

Attacken utnyttjar parningsmekanismen i Bluetooth för att öppna en ”bakdörr” in i telefonen. Att en hopparad och betrodd enhet har tillgång till telefonen är inte något konstigt i sig, detta är ju hur det ska funka. Det allvarliga i denna attack är dock att efter enheten har parats ihop och sedan raderats från listan över ”tillåtna” enheter, så är enheten **fortfarande** tillåten att koppla upp sig mot telefonen och använda de tillåtna funktionerna. Dessa funktioner kan vara allt ifrån att titta i och ändra adressboken, skicka SMS, eller få tillgång till Internet via GPRS. Det är inte säkert att en person som fått sin telefon hackad på detta sätt behöver märka något, om inte personen i fråga tittar på telefonens skärm i de ögonblick som intrången sker. [14]

4 Utrustning

4.1 Hårdvara

Låsenheten är baserad på WRAP THOR-modulen från BlueGiga [22]. I denna modul finns ett BlueCore2-chip från Cambridge Silicon Radio (CSR) [23].



Figur 4-1: WRAP THOR-modul från BlueGiga

Chipet innehåller förutom logiken för basbandet och radiodelen även en 16-bitars RISC processor. Denna processor gör att chipet kan användas som en komplett enchipslösning utan behov av externa processorer. Chipet använder stacken BlueStack som är en komplett Bluetooth-stack utvecklat av Cambridge Consulting Limited (CCL) som är ett dotterbolag till CSR. I det inbyggda 8 Mb stora flashminnet kan man antingen lägga in en HCI-stack, vilket tillåter HCI-split-lösningar med externa processorer, eller en kombinerad stack och applikation i de fall man vill använda chipet som en självständig enhet.

Modulen har många möjligheter att kommunicera med externa komponenter. Bland annat finns det en USB-port och en UART. Utöver det finns även åtta datapinnar som går styra från programmet. Av dessa är dock två stycken reserverade för styrning av radioförstärkardelen i de applikationer där enheten fungerar som en Klass 1-modul. Två av datapinnarna kan även användas som ett I^2C-interface för att koppla in t.ex. ett externt EEPROM.

För att kommunicera med modulen kan man antingen använda USB eller UART. Vid kommunikation över UART kan man bl.a. använda det s.k. H4-protokollet som beskrivs i Bluetooth-standarden, eller BCSP (BlueCore Serial Protocol) som är en egen standard från CSR. Denna standard har bättre felkontroll än H4 och verkar kunna bli en de facto-standard, om den inte till och med kommer in i Bluetooth-protokollet.

Det beständiga minnet i chipet som är tillgängligt för egna applikationer är uppdelat i så kallade ”Persistent keys” eller beständiga nycklar. Det finns 50 stycken tillgängliga nycklar, och den rekommenderade maximala totala storleken på dessa är 1 KB. Då

dessa nycklar ligger i flash-minne som inte bara kan skrivas över hur som helst, utan måste raderas sektorvis, så kan man bara skriva till dem ett visst antal gånger innan enheten måste startas om. Vid omstarten defragmenteras och optimeras minnet och kan sedan fortsätta att användas.

Låset är ett motordrivet lås från Abloy som har en egen kontrollenhet. Det finns möjlighet att styra detta lås via en enkel insignal i denna enhet.

4.2 Mjukvara

Utvecklingen till denna modul sker med hjälp av en SDK kallad BlueLab. Programmeringsspråket som används är C. Kompileringsmiljön är gjord för att köras under Cygwin i Windows. Programmering av flashminnet sker via en adapter som kopplar samman datorns parallell-port med chipets SPI-port. I BlueLab följer det med flera varianter av Bluetooth-stacken för BlueCore-chipet. Bland annat finns det en HCI-stack, där en extern processor, eller dator, krävs för att kunna använda chipet. Det finns även en RFCOMM-stack där alla lager upp till RFCOMM samt SDP finns implementerat. Denna stack stödjer användarapplikationer som körs på chipet.

Stackarna har i standardutförande endast 56 bitars kryptering på grund av exportrestriktioner av krypteringsalgoritmer från England. En 128-bitars version gick att tillgå efter samtal med CSR.

Det följer även med en emulator för BlueCore-chipet som är utvecklad i Java. Den visade sig vara ett bra hjälpmedel vid debugging av program, samtidigt som den inte verkade vara alltför stabil. Ett minnesläckage i emulatorn gjorde att internminnet så småningom tog slut och man var då tvungen att starta om den igen. Ibland hängde den sig så pass att man var tvungen att stänga av både programmet och Javas runtime. Dessutom verkar inte emulatorn stödja alla funktioner som finns i BlueLabs SDK. Enligt inlägg i CSR:s nyhetsgrupper har utvecklingen av emulatorn stannat av, och eventuellt skall det komma en ny, men i dagsläget finns det inget rakt besked om detta. Emulatorn kan köras självständigt så länge man inte använder sig av radiodelen eller någon del av stacken. I dessa fall måste man koppla in en enhet via serieporten till datorn då dessa delar körs på chippet.

Serverprogramvaran är utvecklad i Microsoft Visual C++ 6.0 och använder till största del Microsoft Foundation Classes (MFC). Utvecklingen har skett i Windows XP. Bluetooth-stacken som användes i servern är BlueStack från Mezoe, som är ett systerbolag till CSR. Denna stack är utvecklad för att fungera med BlueCore-chip, och har nästan exakt samma gränssnitt som BlueLab vilket gör att man inte behövder lära sig två olika stackar. Den version av stacken som användes var en utvärderingsversion kallad ProtoDeveloper [11].

5 Implementation

5.1 Designval

Det finns många möjliga sätt som man skulle kunna tänka sig att systemet skulle kunna fungera på. Här diskuteras de viktigaste valen, samt deras för och nackdelar.

5.1.1 Centraliserad eller Decentraliserad?

Distribuerade system som detta kan byggas upp på flera olika sätt. De mest intressanta i detta fall är om man skall göra lösningen centraliserad, dvs. att låsenheterna inte har någon direkt information lagrad i sig, utan de är istället kopplade till en central server som hanterar alla förfrågningar. Alternativt kan man göra lösningen decentraliserad, dvs. låsenheterna arbetar som egna självständiga enheter som har all information lagrad i sig och tar sina egna beslut vid förfrågningar.

I en centraliserad lösning finns låsenheterna till endast för att fungera som en kontaktpunkt för användarna, samt att det sköter själva kontakten med låset. Enheten måste alltid kunna vara i kontakt med servern, antingen via en ständig uppkopplad länk eller att den kontaktar servern när den behöver få ett svar på en förfrågan. Att låta enheten koppla upp sig varje gång är inte en realistisk lösning, eftersom det då skulle minska responsiviteten i systemet med flera sekunder. Att ha en länk öppen hela tiden är ett bättre alternativ, men även här uppkommer problem. Som bekant kan Bluetooth endast hantera sju samtidiga uppkopplingar, och detta skulle begränsa systemets kapacitet nämnvärt. Det finns även en ”single point of failure” med denna lösning. Om servern skulle krascha, eller om en länk skulle brytas av någon anledning och inte går att upprätta igen, är låset obrukbart, och dörren förblir låst.

Fördelarna med en centraliserad lösning ligger i att det är väldigt lätt att administrera användare, samt att intrångsförsök snabbt kan resultera i åtgärder i hela systemet. Det är dessutom möjligt att logga och övervaka alla in- och utpasseringar på ett enkelt sätt.

I en decentraliserad lösning finns problem med övervakningen av intrångsförsök. Eftersom enheterna inte är direkt uppkopplade mot servern så kommer eventuella intrångsförsök mot en dörr inte att rapporteras lika snabbt, och attacken kan under tiden fortsätta mot ett annat lås. Samma problem uppstår om man önskar övervakning på vilka som har öppnat låset och vid vilka tidpunkter. Sådana loggar måste då laddas ner med jämna mellanrum från de olika enheterna till den centrala servern. Det krävs även mer minne och mer resurser i enheterna i denna lösning, eftersom man är tvungen att spara alla användarlistor samt sköta all kommunikation med mobiltelefonerna. Ännu mera minne krävs ifall man vill ha loggar.

En fördel med en decentraliserad lösning är att man kan placera ut enheter som kan fungera utan någon som helst infrastruktur. De kommer därmed antagligen att vara driftssäkrare då det är färre ställen som saker kan gå fel på.

Lösningen valdes att bli decentraliserad. Detta bland annat eftersom det vid undersökningen av vad THOR-modulen är kapabel till fanns att även fastän dess mjukvara uppgavs stödja tre stycken RFCOMM-anslutningar samtidigt, så hade uppkopplingshanteraren i SDK:n inte stöd för mer än en samtidig uppkoppling. Efter undersökningar i deras officiella nyhetsgrupper och forum hittades endast svar i stil med ”Ja, det är möjligt att skriva en egen uppkopplingshanterare som stödjer fler uppkopplingar, men det är inte lätt och vi kommer inte att hjälpa er.”. Det fanns därmed inte möjlighet att överhuvudtaget kunna göra en centraliserad lösning.

5.1.2 Telefon kontaktar låset eller låset kontaktar telefonen?

En i första åtanken tänkbar lösning på detta problem vore att det när man närmar sig dörren så får man en förfrågan om man vill öppna dörren eller inte. Alternativt att telefonen visar en utökad meny där man kan välja vilken dörr man vill öppna. Problemet med detta är just räckvidden hos Bluetooth. Om man skulle tänka sig att systemet skulle jobba med den lägsta specificerade sändningsstyrkan skulle kontaktradien fortfarande vara ca 10 m. Tänk tanken att dörren skulle befinna sig i mitten i, eller i närheten av ett kontorslandskap. Möjligheten finns då att det finns ett stort antal telefoner samt datorer som har Bluetooth påslaget i närheten. Om låset skulle kontakta alla dessa telefoner så överskrider man eventuellt de sju enheter som kan vara aktiva på en gång. Kommer det då ännu en person med en telefon inom räckvidd så kan inte låset kunna kontakta denna, även om personen i fråga kanske är den enda i närheten som har för avsikt att öppna dörren.

Ännu ett problem med denna variant är att många telefoner endast är sökbar under korta perioder. Till exempel Ericssons T68i är bara sökbar i tre minuter, sedan stängs Bluetooth av igen. Detta är antagligen för att spara batterierna i telefonen.

Om man istället väljer alternativet att telefonen kontaktar låset, så kommer endast uppkopplingar att ske mot de telefoner som verkligen vill öppna låset. Detta kräver dock någon form av användarinteraktion. Telefonerna söker nämligen inte själva efter enheter i närheten, eftersom detta skulle dra alldeles för mycket ström.

Med ovanstående argument som stöd valdes att låta telefonen kontakta låset.

5.1.3 Vilken profil?

För att svara på denna fråga behöver man ta reda på vad mobiltelefonerna klarar av. Det behövdes en profil som så många telefoner som möjligt klarade av, samtidigt som det fanns något sätt att styra denna profil från användarens sida.

De profiler som bör finnas implementerade på de flesta mobiltelefoner är Telephony Control System, Synchronization Profile och Object Push Profile.

Telephony Control System
TCS används av bland annat handsfree-utrustning för att styra telefonen. Bland annat kan det användas för att svara eller lägga på samtal, eller för att ringa upp någon. Denna profil innehåller även en funktion som kallas för AT-kommandon. Dessa

kommandon härstammar från modemvärlden och namnet kommer från att varje kommando börjar med texten ”AT” (Attention). Vilka av dessa kommandon som en telefon skall stödja finns inte specificerat i Bluetooth-standarden utan är upp till varje enskild tillverkare. Ericsson har i sina telefoner t.ex. valt att stödja ett par AT-kommandon som kan visa en meny på mobiltelefonen vilka kan användas för att styra specifika funktioner på deras headset. Dessa kommandon har utnyttjats för att skapa applikationer där man t.ex. kan styra Winamp på sin dator för att byta låt eller höja och sänka volymen. De stöds dock inte av andra tillverkare.

Synchronization Profile
Denna profil tillhandahåller en standard för att synkronisera information mellan två enheter. Till exempel kan man få sin mobiltelefon att uppdatera dess adresslista eller kalender från sin stationära dator. Denna synkronisering kan man få att ske automatiskt när de två enheterna har kommit inom varandras räckvidd. Detta kan vara praktiskt för att t.ex. automatiskt uppdatera telefonen då man kommer in i kontoret.

Object Push Profile
Denna profil stöds av i princip alla telefoner som har Bluetooth. Detta för att man skall kunna skicka kontakter mellan olika telefoner och synkronisera adresslistan och/eller kalendern i mobilen med datorn. Då man skickar kontakter från sin adressbok överförs de till mottagaren som ett vCard. Detta är en standard utvecklad av Internet Mail Consortium (IMC) [18] och kan även användas i bl.a. vanliga email som elektroniska visitkort.

```
BEGIN:VCARD
VERSION:2.1
N:Marklund;Tobias
FN:Tobias Marklund
TEL;HOME;VOICE:0123-123456
TEL;CELL;VOICE:070-1234567
EMAIL;PREF;INTERNET:dva99tmd@cs.umu.se
END:VCARD
```

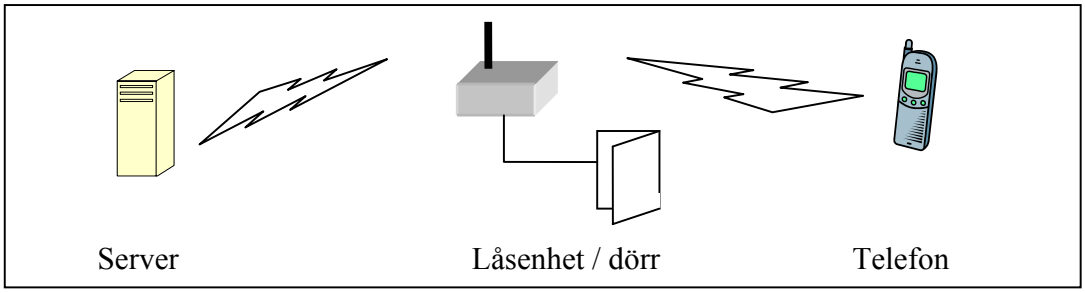
Figur 5-1: Exempel på vCard

Ett vCard består av rader av text som beskriver olika fält i visitkortet. Bland annat kan ett vCard innehålla information om namn, telefonnummer och email, men det står olika tillverkare fritt att lägga till egna fält, så länge dessa fält föregås av prefixet ”X-” (t.ex. X-BILMODELL: Volvo).

Object Push Profile verkar vara den vettigaste profilen att använda, då den tillåter användaren att sända något till en annan enhet på kommando. Menyer via AT-kommandon hade kunnat vara ett alternativ ifall de hade haft ett större stöd i mobiltelefonerna.

5.1.4 Slutligt val

Lösningen kommer att bestå av tre huvuddelar. Dels en server som sköter registrering och hantering av användare och låsenheter. Eftersom detta är en decentraliserad lösning kommer den endast att skicka ut uppdateringar till låsen då en förändring har skett. Dels själva låsenheterna som är kopplade till dörrlåset. Dels användarnas telefoner som används för att öppna dörren. Telefonerna kommer att använda sig av Object Push-funktionaliteten för att kommunicera med låsen, då detta verkar vara det enda realistiska alternativet som en användare kunde använda för att göra en aktiv uppkoppling mot låsenheten. Låsen kommer att vara kopplade till en av datapinnarna på låsenheten.



Figur 5-2: Systemets arkitektur

5.2 Nyckeln

Den nyckel som används i denna lösning består av ett vCard som skickas ut till användaren via Bluetooth, eller via SMS (ej implementerat i denna lösning). Detta vCard lägger sig i mottagarens telefonlista som en vanlig kontakt.

Enligt specifikationen för vCard skall alla implementationer åtminstone stödja fälten:

- Namn
- Fullständigt namn
- Telefonnummer
- Email

Utöver detta finns en mängd fält som är definierade, men som inte är säkert att de stöds av alla telefoner. I vår prototyp finns en beskrivande text i namnfältet, samt den aktuella nyckeln som en nummerserie i telefonnummerfältet.

5.3 Telefonen

Telefonen behöver som sagt inte innehålla speciella mjukvaror eller funktioner, utan de enda krav som ställs är att telefonen klarar Bluetooth samt OBEX-protokollet.

5.4 Låsenheten

Servern innehåller två stycken tjänster. En administrationstjänst, och en tjänst som hanterar användarnas uppkopplingar.

Administrationstjänsten bygger på RFCOMM-protokollet. Denna tjänst är endast tillgänglig för den enhet som är registrerad som server till låsenheten. Ett enkelt request-responseprotokoll har implementerats för att serverapplikationen skall kunna se vilka användare som enheten har registrerat samt för att skicka ut nya användarlistor till låsen.

Låsenheterna bygger på en RFCOMM-mjukvara som inte har stöd för OBEX-protokollet. Detta innebar att en egen implementation var nödvändig. Denna har utgått från OBEX-specifikationerna från IrDA och stöder endast grundläggande funktionalitet i Object Push-profilen.

Användarinformationen lagras i de persistent keys som finns tillgängliga, och innehåller information om användarens Bluetooth-adress, PIN-kod, länknyckel och nyckel.

5.5 Serverapplikationen

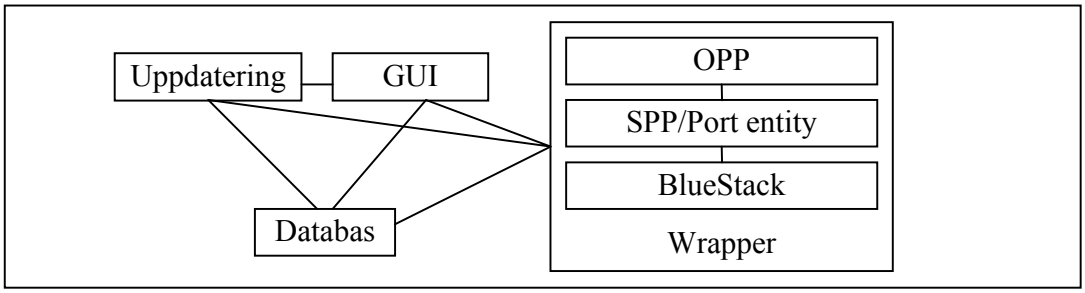
Serverapplikationen består av fyra huvuddelar: användargränssnittet, Bluetooth-stacken, låsuppdateringen och en användardatabas (*Figur 5-3*).

Användargränssnittet som används för att administrera systemet är byggt i MFC. Det innehåller funktioner för att lägga till/ta bort låsenheter och användare, samt möjlighet att ändra användares tillgång till olika lås.

Då Bluetooth-stacken inte är synkron, d.v.s. det är inte säkert att svaren till ett antal frågor kommer tillbaka i rätt ordning, så är en ”wrapper” implementerad som hanterar alla stackanrop och skickar svaret till rätt del av användargränssnittet. För att inte användargränssnittet skall stanna av i väntan på ett svar från Bluetooth-stacken kör wrappern i en egen tråd. Den stacken som användes hade inte heller i Windows något stöd för Object Push Profile, som används för att skicka nyckeln till användarens telefon, eller för Serial Port Profile, som används för att uppdatera låsenheterna, så därför implementerades enkla versioner av dessa profiler. SPP fanns beskrivet i användarhandboken till stacken och var ganska lätt att implementera, men OBEX-protokollet blev även här inte komplett, utan endast de delar som var nödvändiga för våran lösning implementerades.

En separat tråd sköter låsuppdateringen för att inte användargränssnittet skall låsa sig.

Det finns en databas där all information om vilka användare som finns i systemet och vilken tillgång de har till de olika delarna lagras. Denna databas innehåller även information om parade enheter. Databasen använder inte någon separat databas utan sparas i ett eget filformat.



Figur 5-3: Blockdiagram över serverapplikationen

5.6 Användning

5.6.1 Lägga till låsenheter

Att lägga till låsenheter i systemet måste kunna ske på ett säkert sätt, och en enhet skall inte gå att para ihop med en attackerande enhet på ett enkelt sätt. Eftersom låsenheterna inte har något användargränssnitt har varje låsenhet en fast och lång PIN-kod lagrad i minnet. Denna bör vara unik för varje enhet och skall programmeras in då mjukvaran läggs in på chippet. För att starta parningsproceduren krävs att en knapp på enheten trycks ner, vilket innebär att man måste ha fysisk tillgång till enheten. När knappen tryckts ner kommer enheten att vara tillgänglig för parning i 30 sekunder. En lyckad parning innebär att låsenheten sparar adressen till servern och länknyckeln i det beständiga minnet. Efter en lyckad parning kommer enheten heller inte att svara på fler parningsförsök förrän knappen är nedtryckt ännu en gång. Detta omöjliggör en brute force-attack på låsenheterna. Trots att PIN-koden är lång och en ny parning omöjlig utan att ha fysisk tillgång till enheten bör parningen med enheten ske i en kontrollerad miljö, så långt det är möjligt. Är parningen avlyssnad finns det nämligen en möjlighet att en off-line brute force-attack är möjlig för att räkna fram rätt länknyckel.

5.6.2 Lägga till/ta bort användare

För att lägga till en användare söker man först efter användarens telefon med hjälp av servern. Sedan väljer man att para ihop dessa enheter. Detta sker genom att man kommer överens om en PIN-kod som matas in på både servern samt telefonen. Denna PIN-kod kommer också att användas då användaren skall para ihop sin telefon med låsenheterna. Efter att telefonerna har parats ihop genereras ett vCard som skickas över till användarens telefon. Registreringen är därmed fullbordad.

Därefter måste användaren få tillgång att använda någon av låsenheterna. Detta väljs i en lista där alla tillgängliga låsenheter finns listade. Man kan alltså låta en användare använda endast en del av dörrarna i systemet.

Borttagning av användare ur systemet är så enkelt som att markera användaren och välja ”Delete”.

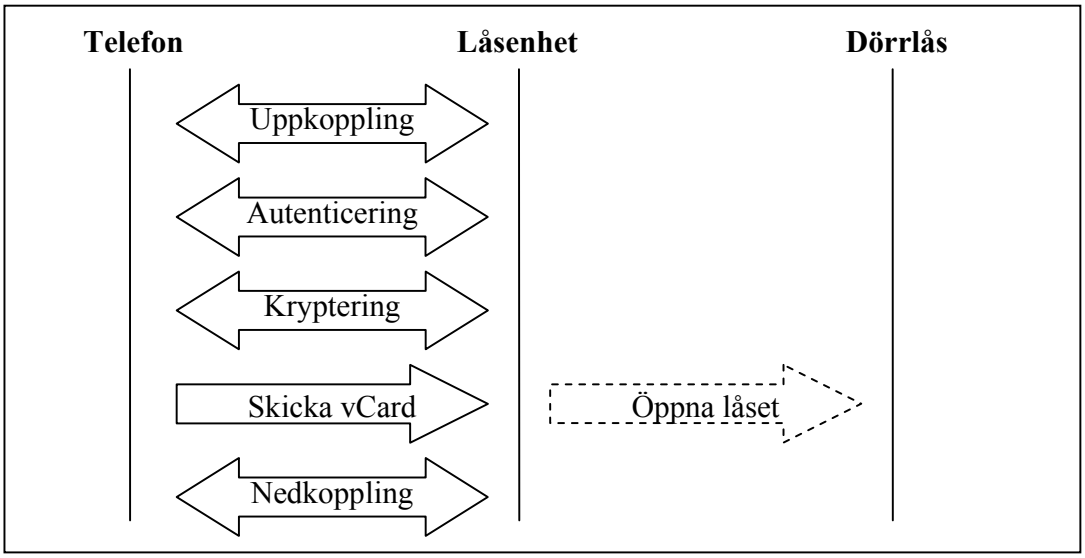
För att en användare skall kunna använda ett lås i systemet krävs att dess telefon registreras mot det låset. Detta sker genom en vanlig parningsprocedur där användaren söker efter låset och parar ihop telefonen med det med hjälp av PIN-koden som registrerades då användaren lades in i systemet. Detta måste göras mot varje lås i systemet som användaren kommer att använda.

5.6.3 Uppdatering av systemet

När en förändring av användarbehörigheterna har skett måste dessa ändringar skickas ut till alla berörda enheter. Detta sker genom att servern i tur och ordning kontaktar alla låsenheter och skickar över informationen om varje användare som har tillgång till den enheten. Eventuella länknycklar som sparats i enheten behålls för att inte andra användare som inte berörs av uppdateringen skall behöva para ihop sin telefon mot låset igen. Uppdateringen sker i en egen tråd för att inte användargränssnittet skall blockeras.

5.6.4 Öppna en dörr

För att öppna en dörr väljer användaren först rätt kontakt i sin adressbok och väljer att skicka den över Bluetooth. Därefter görs en sökning efter närliggande enheter. När rätt låsenhet har hittats väljer användaren den enheten och en uppkoppling sker. Efter att låsenheten har bekräftat användarens identitet och krypterat länken skickas nyckeln över. Därefter sker en nerkoppling av kommunikationen mellan enheterna. Låsenheten kommer att analysera nyckeln och bedöma om den är rätt för användaren i fråga. Om så är fallet kommer en utgång på modulen att läggas på hög spänning och därmed öppna låset. Hur länge låset är öppet ställs in i själva dörrlåsets kontrollcentral.



Figur 5-4: Öppning av dörr

5.7 Prototypens begränsningar

På grund av det begränsade minnet och de begränsade antal gånger man kan skriva till minnet innan enheten måste startas om så kunde inte en funktion för att förhindra brute force-attacker implementeras. Bluetooth-standarden föreslår att vid ett misslyckat uppkopplingsförsök så skall enheten förhindras att försöka igen under en kort tidsperiod. Skulle ännu ett misslyckat försök upptäckas efter det fördubblas tiden, och så vidare.

Prototypen innehåller inte heller någon möjlighet att logga in- och utpasseringar. Detta beror också på det begränsade minnet, samt att de inte innehåller någon klocka. Man kan inte heller få låset att vara öppet under vissa tider på dygnet pga. avsaknaden av klocka.

Endast 49 användare per enhet kan lagras eftersom det endast finns 50 stycken tillgängliga data-areor, och en area används för att lagra vilken enhet som agerar server.

OBEX-implementationen är inte komplett, vilket kan betyda att meddelanden som är felaktigt utformade skulle kunna ge oväntade konsekvenser. Troligtvis skulle ett sådant meddelande kunna krascha enheten, som då skulle starta om automatiskt pga. den inbyggda felhanteringen.

Ingen användarfeedback finns tillgänglig i de fall en nyckel är fel, eller om användaren inte får använda en specifik dörr. Detta skulle man dock kunna lösa genom att använda någon av utgångarna på modulen för att driva ett par lysdioder eller t.o.m. en display.

6 Analys

6.1 Fördelar och nackdelar

Som redan nämnts har detta system stora fördelar när det gäller installation av systemet. Ett kortlåssystem kräver omfattande kabeldragning som kan ställa till problem i vissa fall. Detta system kan enkelt sättas på vilken vägg som helst intill en dörr. Det enda kravet är att det finns en strömförsörjning i närheten, vilken oftast är ett väldigt litet problem. Då räckvidden är ända upp till 100 m så kan även stora områden täckas förutsatt att det inte finns för många objekt i vägen.

Kortlåsen och vanliga nycklar har en fördel i att ett borttappat kort eller nyckel kan ersättas relativt fort, medan en borttappad telefon kan vara en ganska kostsam affär för användaren. Antagligen köper därför inte användaren en ny telefon samma dag heller och kommer då vara utan möjlighet att använda låssystemet under tiden tills en ny köps in.

6.2 Användarvänlighet

Användarvänligheten i ett sådant här system kan, tyvärr, med nuvarande tekniska möjligheter inte anses som speciellt god. Det största problemet är det faktumet att det tar väldigt lång tid att använda systemet. Detta är till stor del en brist i Bluetooth-standarden, och är svår att påverka från ett applikationsperspektiv. Då en uppkoppling, inklusive sökning, kan ta upp till ett tiotal sekunder att utföra, anser jag det som troligt att många skulle föredra att använda andra teknologier eller rent av bara använda den gamla hederliga nyckeln. I version 1.2 av standarden har förändringar gjorts för att minska tiden det tar att söka och koppla upp enheter mot varandra, och detta kan göra att den aspekten inte längre är relevant. Detta kommer att visa sig då det finns enheter som stödjer standarden.

Under detta arbete så ansåg några personer att det aldrig skulle fungera att ha kryptering påslaget, eftersom det skulle ta fruktansvärd lång tid att koppla upp sig. Tester med såväl 128 bitars kryptering samt utan kryptering visade dock ingen direkt skillnad i tidsåtgång vid uppkopplingen.

Själva användargränssnittet lämnar förstås även det mycket att önska. På grund av telefonernas begränsade möjligheter är alla lösningar med Java och dylikt i dagsläget helt uteslutna. Många ovana användare skulle antagligen tycka att metoden att skicka över en kontakt från sin telefonbok till en annan enhet för att låsa upp ett lås är ganska svår att greppa. Detta är dock i dagsläget det enda gångbara alternativet för att åstadkomma en någorlunda säker funktion i låset som stöds av i princip alla telefoner på marknaden, och inte endast av ett fåtal dyra modeller.

Ett stort problem är också att en användare som skall använda ett lås först måste para ihop sin telefon med det låset innan han kan använda det. I ett system med många

dörrar kan detta vara mycket irriterande för användarna, även om det bara sker en gång. Telefonerna har även ett begränsat minne när det gäller antalet enheter som kan vara ihopparade med dem. Hur stort detta minne är beror på modell och tillverkare, och vissa modeller skulle säkert kunna få problem med ett system som innehåller många lås. Dessa modeller skulle då inte fungera särskilt bra, eftersom om man inte hade ett visst lås i sin lista då man ville öppna det skulle man behöva ta bort en enhet ur listan och sedan lägga till låset man vill använda genom att utföra parningsproceduren igen.

6.3 Säkerhet

Hur säker är då denna lösning? Här analyseras olika säkerhetsproblem som kan uppstå.

6.3.1 Den mänskliga faktorn

Ett stort problem med alla system som har höga krav på säkerhet är att inget system är säkrare än de som använder det. Det finns många sätt att ta reda på information som t.ex. PIN-kod från en användare. Ett exempel är då folk skriver upp sin kod på ett papper som någon kanske får syn på, eller att man använder en alldeles för lätt PIN-kod som har någon anknytning med personens privatliv, t.ex. en del av ens telefonnummer.

6.3.2 Nyckelns säkerhet

Man kan diskutera hur mycket påverkan innehållet i det vCard man skickar över till låset verkligen har på säkerheten i systemet. Den verkliga säkerheten ligger egentligen på själva parningsmekanismen i Bluetooth, men som bekant så kan denna kringgås genom att endast känna till PIN-koden för en viss användare. I det fall som en attackerare skulle ha kommit över PIN-koden så krävs ändå att han har tillgång till koden som finns i nyckeln.

Dock så är det relativt lätt att kopiera nyckeln eftersom koden endast är ett fält i ett vanligt vCard. Det betyder att man kan se koden i klartext bara genom att öppna det i telefonens adresslista. Eftersom vCard har en specifik struktur är det lätt att skapa en egen nyckel. Nyckeln har således en viss påverkan på säkerheten, men den är ändå relativt marginell.

6.3.3 Avlyssning

I de fall en avlyssning skulle lyckas så är systemet direkt sårbart eftersom all information som behövs för att öppna låset skickas genom luften. Att all information skickas krypterad försvårar givetvis en eventuell avlyssning, men det är fortfarande möjligt att göra det. En avlyssning förutsätter dock oftast att attackeraren har lyckats avlyssna själva parningsförfarandet och därigenom lyckats knäcka PIN-koden och den resulterande länknyckeln. När den är känd finns det inget som hindrar attackeraren från att avlyssna trafiken.

Ett annat alternativ är att attackeraren lyckas knäcka krypteringsnyckeln, vilket kan vara fullt möjligt vid svaga nycklar. Vid tester av systemet gick det till exempel inte att få Ericssons T68i att använda starkare kryptering än 48 bitar, vilket inte anses säkert. Tillåter man sådana enheter att använda systemet så har man således en potentiell säkerhetsrisk på halsen.

En av nackdelarna med att ha enheter som kan sända upp till 100 meter är att en eventuell attackerare också kan befinna sig inom 100 meters radie och avlyssna trafiken utan att någon märker honom.

6.3.4 Man-in-the-middle

Istället för att bara passivt avlyssna kommunikationen mellan en användare och ett lås kan en attackerare göra en så kallad man-in-the-middle-attack, där attackeraren utger sig för att vara en annan enhet än vad den är. Mot användaren kan den utge sig för att vara en låsenhet, och mot låsenheten utger den sig för att vara användaren. Då bägge enheter tror att de talar med rätt motpart kommer ingen att märka att det sitter en person mellan dom och avlyssnar all information.

Denna attack kräver att attackeraren har kommit över antingen PIN-koden eller länknyckeln. Om länknyckeln är känd kommer användaren inte att märka av något, då den tror att den talar med rätt enhet. Om PIN-koden är känd kan attackeraren koppla upp sig mot låsenheten, men det krävs även att användarens mobiltelefon paras ihop mot attackerarens enhet.

6.3.5 Stöld

Eftersom Bluetooth endast kan autenticera enheter, och inte användare, och eftersom systemet inte innehåller något högre lager av säkerhet, så betyder en stöld av en mobiltelefon att tjuven kommer att ha direkt tillgång till låset utan att behöva ange någon PIN-kod eller annan information. Detta är i och för sig exakt samma problem som med vanliga nycklar. Skillnaden här är att en borttappad eller stulen telefon lätt kan spärras från systemet utan att någon annan användare kommer att märka något istället för att alla nycklar och låset måste bytas ut. Användaren kan sedan få en ny nyckel utskickad till en annan telefon och måste sedan registrera den telefonen mot låsenheterna.

7 Framtida förbättringar

Här diskuteras på vilket sätt ett sådant här system skulle förbättras eller kunna dra nytta av framtida teknologier och tekniker.

7.1 Minne

Det stora problemet i det här systemet är just minnestillgången. För att kunna lägga in saker som passeringsloggar eller stöd för fler användare måste man lägga till ett extra minne. Detta bör inte vara ett alltför stort problem då det finns stöd för EEPROM med I^2C-interface. Ett externt minne skulle dessutom eliminera problemen med att man endast kan skriva till minnet ett visst antal gånger innan enheten måste startas om.

7.2 Bluetooth 1.2

Ett av de stora problemen med användningen av ett Bluetoothbaserat låssystem är svarstiden på systemet. En användare vill inte stå och vänta ett tiotal sekunder för att en dörr skall öppna sig. Då skulle man antagligen lika gärna kunna ta fram sin nyckel från fickan och låsa upp, vilket kanske tar 3-5 sekunder.

Den nya revisionen av Bluetoothstandarden, version 1.2, har nyligen (hösten 2003) färdigställts och blivit godkänd av SIG. I denna standard har man designat om uppkopplingsförfarandet på basbandsnivå så att två enheter som är anpassade för Bluetooth 1.2 teoretiskt sett skall kunna upprätta en uppkoppling sinsemellan på kortare tid än enheter som bara stödjer version 1.1.

Enheter som stöder Bluetooth 1.2 kommer att vara bakåtkompatibla med enheter som bara stöder tidigare versioner. De kommer dock inte att dra nytta av den förkortade uppkopplingstiden.

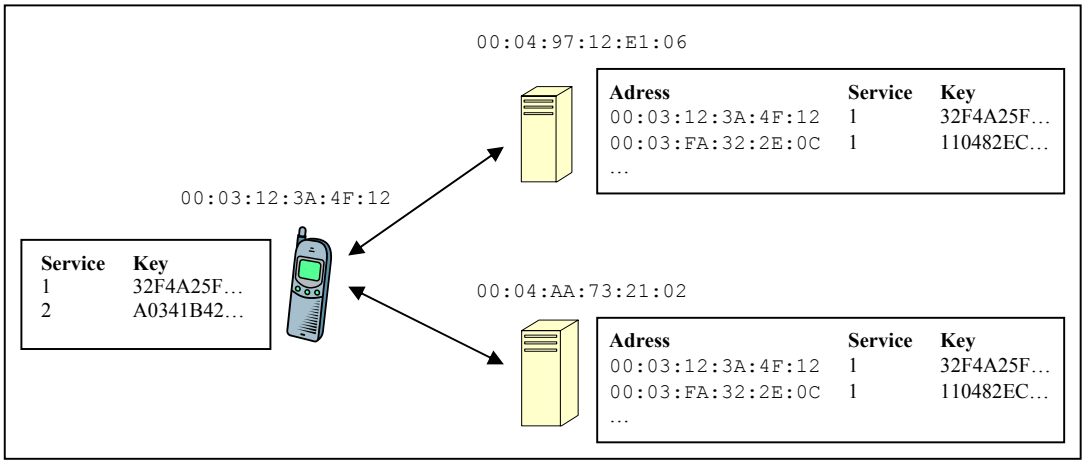
7.3 Group keys

I den nuvarande implementationen av Bluetooth finns det som sagt två stycken metoder att autenticera en enhet: unit keys och combination keys. Unit keys anses inte vara säker att använda och har i och med specifikationen för Bluetooth 1.2 helt tagits bort. Combination keys kan bara användas för att autenticera en enhet mot en annan enhet. Om man i ett system vill kunna kommunicera säkert med ett stort antal enheter kommer detta snabbt att bli opraktiskt, då man måste genomgå parningsförfarandet mot varje enskild enhet. Detta är inte heller praktiskt ur säkerhetssynpunkt eftersom det är möjligt att avlyssna parningsförfarandet och räkna ut PIN-koden som används. [1]

En framtida utveckling av Bluetooth kan kanske erbjuda en ny sorts autenticeringsmekanism som föreslås i ”Bluetooth Security White Paper” [1] som de kallar ”Group Keys” (*Figur 7-1*). Denna mekanism är baserad på samma sorts nyckelgenereringssystem och parningsförfarande som för vanlig autenticering, med

den skillnaden att en genererad nyckel inte bara är giltig för kommunikation mellan två distinkta enheter. Istället kommer en nyckel att genereras som är giltig för den enheten vid kommunikation med en viss tjänst. Detta innebär att parningsförfarandet kan ske i en kontrollerad miljö samt att den resulterande nyckeln kan distribueras ut (förhoppningsvis) säkert till alla enheter inom systemet.

Fördelar med detta i denna tillämpning vore att man endast behöver registrera sin telefon en enda gång, nämligen då man registrerar den i systemet. Telefonen kommer då att använda den nyckeln vid kommunikation mot alla enheter som har tjänsten.



Figur 7-1: Group keys

I dagsläget finns inte denna form av autenticering specificerad någonstans i Bluetooth-standarden, men det går att uppnå samma sorts funktionalitet med specialiserad mjukvara i enheterna om man använder sig av säkerhetsnivå 2.

7.4 Applikationer

Sedan en tid tillbaka finns det stöd för Java-program i vissa modeller av mobiltelefoner. Dessa mobiler har dock ännu inte börjat stödja Bluetooth i sina implementationer, eftersom det ännu inte har funnits någon gemensam standard i MIDP 1.0 som de flesta telefoner använder. För något år sedan kom dock MIDP 2.0 som innehåller standarden JSR-82 för Bluetooth-kommunikation i Java. Hittills är det i princip endast de alldeles nysläppta SonyEricsson P900 samt Nokia 6600, alla respektive märkes dyraste och mest exklusiva telefoner, som stöder Bluetooth över Java, vilket gör utbudet minst sagt litet för närvarande.

Då tillverkarna börjar stödja Bluetooth på allvar genom t.ex. MIDP 2.0 även i deras "mainstream"-telefoner, och inte bara i deras premiummodeller kommer vi nog att få se en stor tillströmning av nya användningsområden för mobiltelefonen. I detta fall skulle dock en javaapplikation vara ett logiskt steg att ta, genom att man då via ett GUI snabbt skulle kunna få en överblick om vilka dörrar som finns i närheten och lätt

kunna öppna dom genom endast en knapptryckning. Detta skulle även öppna upp för säkrare krypterings- och autenticeringsmetoder på en högre nivå än Bluetooth-nivå.

Ett problem med Java är dock laddningstiderna. I dagens telefoner tar det ett par sekunder att ladda ett Java-program. Detta är dock inte ett akut problem i många tillämpningar, men om funktionen används flitigt kan detta bli ett störande moment. Förhoppningsvis blir telefonerna bättre och snabbare på detta i framtiden.

Att utveckla en applikation som baseras på operativsystemet Symbian är också ett tänkbart alternativ, då det har ungefär samma fördelar som Javaapplikationer. Denna lösning lider dock av samma problem som Java, dvs. endast de allra dyraste telefonerna stöder Symbian. Dessutom delar dessa telefoner inte samma användargränssnitt (t.ex. Nokia använder System 60 och Ericsson använder UIQ), vilket skulle betyda att man måste tillverka modellspecifika applikationer. Det senare är dock troligtvis inget direkt problem vid en större satsning.

8 Sammanfattning

8.1 Resultat

En fungerande prototyp av ett Bluetooth-baserat lås som kan användas av en majoritet (om inte alla) mobiltelefoner som stöder Bluetooth har utvecklats. Denna lösning är dock inte på något sätt lättanvänd eller snabb och är ur säkerhetssynpunkt inte helt pålitlig. Dock så anser jag att denna lösning är så nära man kan komma för att ha stöd i en majoritet av mobiltelefonerna på marknaden utan att göra förändringar i deras mjukvara.

8.2 Personliga reflektioner

När jag började arbeta med detta arbete insåg jag vilka möjligheter ett sådant här system skulle kunna ha. Mobiltelefonen är idag var mans egendom, och fastän andelen telefoner som har Bluetooth ännu är rätt liten, så är den helt klart växande. Om ett par år tror jag att de flesta nya mobiltelefonerna kommer ha Bluetooth inbyggt.

Efter ett tags forskande insåg jag dock att dagens mobiltelefoner är väldigt begränsade i sin funktionalitet av Bluetoothfunktionen. I princip är man begränsad till att skicka kontakter mellan varandra, synkronisera sin adresslista och kalender med sin dator och att använda ett trådlöst headset. Det sistnämnda är såklart väldigt användbart för en del av användarna, men om mobiltelefontillverkarna och SIG har för avsikt att få Bluetooth användbart till andra områden samt få en allmän acceptans behövs en ”killer application”. En öppnare utvecklingsmiljö skulle underlätta detta väsentligt, och detta kan man åstadkomma genom att stödja Bluetooth i Java för mobiltelefonerna. MIDP 2.0 är förhoppningsvis på gång nu i fler ”mid-range”-telefoner och detta kan förhoppningsvis sporra utvecklingen av applikationer. Varför det har tagit så här många år att åstadkomma kan jag inte riktigt förstå.

Att arbeta med det här projektet har varit väldigt intressant och roligt, men samtidigt rätt ofta ganska frustrerande. Utrustningen har varit allt annat än hjälpsam mot mig, vilket jag har börjat misstänka är rätt vanligt vid Bluetoothutveckling. Det är lätt att sätta enheter i lägen där de inte svarar, eller det inte går att koppla upp sig mot dem utan att man måste starta om dem. Samma problem fanns 2001 då jag gick en kurs i ”Mobilitet och trådlösa nätverk” och vi använde Ericssons moduler vid laborationerna.

Det är lite synd att det inte gick göra systemet enklare än vad det är, men då målet var att få funktionen att fungera på så många telefoner som möjligt så känner jag mig ändå rätt nöjd med det som jag åstadkommit. Jag anser att en lösning där en Java- eller Symbian-applikation används istället mycket väl kan fungera i kommersiella sammanhang, men inte förrän om ett par år då det finns fler telefoner som klarar av det.

8.3 Problem

BlueCore2-chipet var dessutom väldigt sporadiskt dokumenterat, och mycket var man tvungen att gissa sig fram till. Särskilt minnesmängd och minnesanvändning i enheterna var väldigt svårt att få fram. Specifikationerna visade att enheten skulle ha 32 kb ram, vilket skulle räcka gott och väl för denna applikation. Vid närmare undersökning hittade vi till slut i deras nyhetsgrupp att denna minnesmängd används av alla delar av chippet, inklusive stacken. Hur mycket minne som finns kvar för användarprogram är det ingen som kan eller vill svara på. Det enda som vi lyckades få fram var att man kunde allokera som mest 12 block av minne, och de *trodde* att det maximala som man skulle kunna allokera var 20 byte.

Dessutom var själva minneshanteringen i chippet ett problem. Eftersom det innehåller en 16-bitars processor har de valt att göra alla datatyper 16 bitar långa. En byte som i vanliga fall tar 8 bitar tar då istället upp 8 bitar plus 8 bitar som utfyllnad. Då alla operatorer, som t.ex. minneskopiering, arbetade med ett 16 bitar långt värde blev vi rätt förvånade då den data som kommer över en RFCOMM-uppkoppling visade sig vara 8 bitar långa värden. Ett värde som på sändarsidan var 16 bitar långt kom därför fram som två 8 bitars värden som man då via multiplikation och addition var tvungen att lägga ihop för att få det ursprungliga värdet.

Efter att ha arbetat på prototypen ett tag visade det sig att det inte var så fruktansvärt svårt att implementera en uppkopplingshanterare som klarade av mer än en uppkoppling. Dock så kanske det skulle vara svårt att stödja allt som den befintliga klarade av, men en för projektet lämplig version var fullt möjlig. Som tidigare nämnts så stödjer ändå inte mjukvaran i chipen fler än tre samtidiga uppkopplingar, så en sådan lösning skulle ändå vara väldigt begränsad. Då vi dessutom redan bestämt oss för den decentraliserade versionen och redan hade börjat på det spåret så ändrade vi inte utformningen av projektet.

9 Tack

Först och främst vill jag tacka Arne Viktorsson på Explizit för att han anförtrodde denna uppgift åt mig och gav mig chansen att göra ett examensarbete under vintern 2003/04.

Tack även till mina handledare Mattias Franck och Markus Nilsson på Explizit för deras hjälp och även alla andra på företaget för ett trevligt sällskap.

Min handledare Jerry Eriksson på Datavetenskapliga Institutionen för synpunkter på rapporten.

Patrik Wandin för korrekturläsning och åsikter på arbetet.

Jag vill även tacka Uminova Center för deras ekonomiska bidrag till arbetet.

10 Referenser

1. Bluetooth SIG. *Bluetooth Security White Paper*. http://www.bluetooth. com/upload/24Security_Paper.PDF (2003-11-17)
2. Bluetooth SIG. *Bluetooth Specification Version 1.2*. https://www.bluetooth.org/spec/ (2004-01-03)
3. Ericsson. *About Bluetooth*. http://www.ericsson.com/bluetooth/aboutbluet/ (2004-01-03)
4. Kammer, McNutt, Senese, Bray. *Bluetooth application developer's guide: The short range interconnect solution*. Syngress.
5. Dean A. Gratton. *Bluetooth Profiles: The definitive guide.* Prentice Hall.
6. Jochen Schiller. *Mobile Communications.* Addison Wesley
7. Jennifer Bray, Charles F Sturman. *Bluetooth 1.1: Connect Without Cables. Second Edition*. Prentice Hall.
8. *How to bluejack*, http://www.bluejackq.com/
9. Brent A. Miller, Chatschik Bisidikan. *Bluetooth revealed.* Prentice Hall.
10. Widcomm, http://www.widcomm.com
11. Mezoe, http://www.mezoe.com
12. ZDNet. *Sony Ericsson advises users to turn off Bluetooth.* http://news.zdnet.co.uk/communications/wireless/0,39020348,39146123,00.htm (2004-02-25)
13. ZDNet. *Nokia admits multiple Bluetooth security holes.* http://news.zdnet.co.uk/communications/wireless/0,39020348,39145886,00.htm (2004-02-25)
14. Bluestumbler, http://www.bluestumbler.org/ (2004-02-25)
15. Nikhil Anand (2001). *An overview of Bluetooth security*. http://www.giac.org/practical/gsec/Nikhil_Anand_GSEC.pdf (2004-02-25)
16. Jakobsson, Wetzel (2001). *Security weaknesses in Bluetooth*. http://www.rsasecurity.com/rsalabs/staff/bios/mjakobsson/bluetooth/bluetooth.pdf
17. Datavetenskapliga Institutionen, Umeå Universitet. http://www.cs.umu.se
18. Explizit AB. http://www.explizit.se
19. Internet Mail Consortium. http://www.imc.org
20. Internet Mail Consortium. *vCard 2.1 specification*, http://www.imc.org/pdi/vcard-21.rtf (2004-02-25)
21. Computer Access Technology Corporation. http://www.catc.com

22. BlueGiga. http://www.bluegiga.com

23. Cambridge Silicon Radio. http://www.csr.com

A. Förkortningar

ACL	Asynchronous Connection-Less
ACO	Authentication Ciphering Offset
AFH	Adaptive Frequency Hopping
AU_RAND	Authentication Random Number
BCSP	BlueCore Serial Protocol
CSR	Cambridge Silicon Radio
EN_RAND	Encryption Random Number
FHSS	Frequency Hopping Spread Spectrum
FDM	Frequency Division Multiplex
HCI	Host Controller Interface
IN_RAND	Initialisation Random Number
I^2C	Inter-IC
IrDA	Infrared Data Association
L2CAP	Logical Link Control and Adaptation Protocol
MFC	Microsoft Foundation Classes
OBEX	Object EXchange
OPP	Object Push Profile
RFCOMM	Radio Frequency Communication
SCO	Syncronous Connection Oriented
SDK	Software Development Kit
SDP	Service Discovery Protocol
SPP	Serial Port Profile
SRES	Signed Response
TCS	Telephony Control System
TDM	Time Division Multiplex
UART	Universal Asynchronous Reciever Transmitter
USB	Universal Serial Bus