TinyECC: Konfigurabilna programska knjižnica za uporabo kriptografije eliptičnih krivulj v brezžičnih senzorskih omrežjih

Mate Beštek

2010-03-01

Abstract

Ambientalna inteligenca je širok pojem, ki v osnovi združuje koncepte naprednih uporabniških vmesnikov, novodobne vseprisotne sisteme, ki so sposobni zaznavati okolje v katerem se uporabnik nahaja in nato tudi prilagajanje odločitev sistema glede na okolje ter sporočanje le teh nazaj k uporabniku. Med napredne uporabniške vmesnike štejemo tudi brezžična senzorska omrežja, ki predstavljajo množico strojnih naprav, ki vsaka zase zaznava delček okolice v kateri se uporabnik nahaja. Takšna omrežja se bodo v bodoče vedno bolj uporabljala tudi v zdravstvu saj so osnova za izvedbo koncepta domače oskrbe kjer takšna omrežja omogočajo nadzorovanje zdravstvenega stanja pacienta kar v njegovem domačem okolju. Ker pa takšna omrežja v osnovi komunicirajo na brezžičen način, bo potrebno zagotoviti ustrezno varnost za podatke, ki se bodo prenašali. To zaščito bo potrebno implementirati na način, ki bo upošteval majhno število virov, ki jih takšna strojna oprema ima na voljo - predvsem v smislu RAM, ROM in CPE moči. Ena od zelo zanimivih rešitev, ki bi lahko ustrezala tem zahtevam je tudi knjižnica TinyECC, ki temelji na uporabi eliptičnih krivulj.

1 Uvod

V [8] je omenjeno, da so senzorska omrežja postala priljubljena zaradi univerze UC Berkley, ki je razvila tudi serijo senzorskih vozlišč imenovano vozlišča mica ter tudi skupek odprtokodne programske opreme kot so operacijski sistem TinyOS ali podatkovna baza TinyDB. Seveda je bila ideja majhnih med seboj sodelujočih naprav znana že veliko prej. Raziskovalno področje vseprisotni sistemi (ubiquitous computing) je vpeljalo nekatere ideje, ki so povzročile razcvet senzorskih omrežij. Kot je omenjeno tudi v [11, 2] se brezžična senzorska omrežja(WSN - Wireless Sensor Networks) danes vedno bolj uveljavljajo in uporabljajo saj so v zadnjem desetletju pridobila prevladujoč pomen na področju tehnologije. Raznolikost področij na katerih se WSN uporabljajo kažejo, da za novo paradigmo obstaja precej svetla prihodnost.Številna podjetja se že ukvarjajo z razvojem aplikativnih sistemov, ki temeljijo na paradigmi WSN. Hkrati so ta podjetja predstavila številne probleme s katerimi se potem ukvarjajo univerze po celotnem svetu oziroma raziskovalci zaposleni na univerzah.

V osnovi so WSN sestavljena iz sto ali tisoč majhnih senzorjev, ki so v osnovi lahko zelo omejeni z vidika virov(velikost pomnilnika, procesorska moč in kot najbolj pomemben tudi vzdržljivost baterije oziroma električnega napajanja), ki jih imajo na voljo. Zelo pogosto takšne naprave nimajo neposredenga dostopa do obnovljivega vira energije. Pomembno je podariti tudi ekonomski vidik WSN saj je ena od osnovnih pogojev za uspeh te tehnologije tudi relativno nizka cena takšnih omrežnih tehnologij. Najbolj pogosti področji na katerih se aplikacije WSN uporabljajo glede na [11] sta zavedanje na vojaških bojiščih in prometni nadzorni sistemi. Omenil pa bi tudi še eno področje, ki bo v bližji in daljni prihodnosti postalo enako ali celo bolj pomembno kot zgornji dve. Kot je omenjeno tudi v [2] gre seveda za področje zdravstva. Človeško prebivalstvo se stara kar posledično pomeni, da s časom obstoječi zdravstveni sistemi ne bodo mogli preživeti, če se ne zgodijo inovativne in napredne spremembe oziroma koncepti. En od takih konceptov sicer ni nov je pa vedno bolj verjeten. Gre za koncept domače oskrbe (tele-health) seveda s podporo tehnologije, med drugim tudi WSN, ki se tu pojavljajo v obliki omrežij, ki delujejo v bližini človekovega telesa (BAN - Body Area Networks). Primer uporabe tehnologije WSN je opisan v [10]. Takšne tehnologije bi zmanjšale potrebe po novih bolnišnicah saj bi se bolni ljudje lahko zdravili v domačem okolju v katerem se gotovo tudi najbolje počutijo seveda pa sama lokacija bolnika ne bi bila več pomembna. Pomembna bi postala tehnologija, ki bi tako bolnikom kot tudi zdravstvenemu osebju olajšala vsakdan. Zdravniku tako ne bi bilo potrebno vsak dan ostajai v službi in delati nadure ampak bi lahko odšel domov in bi bil priklican samo v primeru, ko bi sistem, ki nadzira zdravstvena stanja pacientov ugotovil, da je pri kakšne pacientu kaj narobe. Sistem bi seveda lahko upošteval več parametrov pri ugotavljanju katerega zdravstvenega delavca bo priklical na pomoč. Seveda za samo izvedbo takšnih omrežij potrebujemo tudi standardizirane protokole, ki omogočajo komunikacijo med posameznimi vozlišči omrežij. V primeru WSN omrežij je 802.15.4 tisti protokol, ki ga potrebujemo. Protokol opisuje samo dve plasti OSI standarda. Gre za fizično plast in pa plast dostopa do prenosnega medija. Iz tega standarda so potem izvedeni standardi, ki opisujejo višje nivoje po OSI. Tako poznamo danes protokole ZigBee, WirelessHART in MiWi, ki definirajo še preostale nivoje po OSI in tako definirajo celotno omrežno rešitev. Kot je nakazano v [4] potrebuje omenjeni protokol malo energije za svoje delovanje prav tako pa so majhna tudi sporočila, ki protokol sestavljajo. V [4] je izvedena tudi simulacija, ki ugotavlja največje število vozlišč WSN omrežja pri katerem so še zagotovljeni QOS standardi rešitev s področja zdravstva. To je samo ena od mnogih podobnih analiz katerih rezultati omogočajo zanseljive in učinkovite aplikacije WSN omrežij. Vojska, promet in zdravstvo so področja pri katerih ima varovanje informacij še poseben pomen oziroma se daje na varnost velik poudarek. Zato je smiselno oziroma nujno potrebno izvajati tudi analize varnosti takšnih omrežij saj je s tem pogojena varnosti kasnejših aplikacij. V poglavju 2 bi si pogledali kako je z varnostjo v WSN omrežjih v smislu varnostnih pomanjklivosti, prav tako bi si pogledali kako te pomanjklivosti navadno rešujemo. Ugotovili bomo, da je za WSN omrežja smiselno uporabljati znanja s področja kriptografije, ki se ukvarjajo z eliptičnimi krivuljami saj so implementacije rešitev, ki le te uporabljajo, veliko bolj učinkovite in veliko manj potratne kot nekatere klasične rešitve s področja kriptografije in računalniške varnosti. Tako si bomo v 3 pogledali kaj so eliptične krivulje, kakšne so matematične osnove le teh in pa kako jih

lahko uporabimo tudi v PKI (Public Key Infrastructure) oziroma v kriptografiji na podlagi javnih ključev. Ker se tu ukvarjamo z WSN nas bo seveda zanimalo, kako eliptične krivulje uporabiti v takšnih omrežjih za katere vemo, da so zelo omejeni z viri. V ta namen si bomo v 4.2 podrobneje pogledali kaj nam omogoča programska knjižnica TinyECC, ki je v osnovi implementacija kriptografskih prijemov pri katerih se uporabljajo eliptične krivulje. Predpogoj za uporabo TinyECC je seveda operacijski sistem TinyOS, ki ga namestimo na dejanske senzorje. Osnovne lastnosti TinyOS operacijskega sistema bodo podane v 4.1. Za tem pa si bomo v 4 pogledali tudi kako vzpostaviti razvojno testno okolje, ki bi nam omogočalo razvoj novih programov, ki bi jih lahko uporabili na resničnih aplikacijah na resničnih senzorjih. V 5 bom podal še zaključek, kjer bom omenil težave do katerih sem prišel pri svojem delu in pa kako se bo projekt razvijal v prihodnje. Podal bom tudi lastno oceno o učni krivulji, ki je potrebna za to, da se nekdo začne ukvarjati z razvojem naprednih in inovativnih aplikacij, ki v osnovi delujejo nad brezžičnimi senzorskimi omrežji.

2 Varnost brezžičnih senzorskih omrežij (WSN)

V [8] je omenjen prepad oziroma razlikovanje med raziskovalnim delom na področju kriptografije in raziskovalnim delo na področju senzorskih omrežij saj varnostnih vprašanj ne moremo na preprost način prenesti iz kriptografije na senzorska omrežja in obratno. Varnost seveda izhaja iz robustnosti algoritmov in ne iz skrivanja algoritmov. V osnovi torej poznamo dva načina kriptiranja podatkov in sicer simetrični in asimetrični način pri čemer simetrični način uporabi isti ključ za kriptiranje in dekriptiranje podatkov. Prednosti simetričnega načina kriptiranja so dokaj preprosta implementacija algoritmov, ni potrebno veliko virov za računanje hkrati pa so nekateri načini dovolj varni pred razbitjem tudi če se kriptanalize loti velika računska moč. Slabost simetričnega načina pa je v tem, da se morajo vsi sodelujoči dogovoriti o skupnem ključi preden se izmenjavanje podatkov začne. Pri senzorjih bi seveda lahko predhodno namestili ključe na vsak senzor posebej vendar potem vedno obstaja možnost, da nekdo ukrade en senzor in iz pomnilnika pridobi ključ. Rešitev problema pri simetričnem načinu ponuja asimetrični način zaščite podatkov kjer imamo par ključev. Javni ključ je javen in je na voljo vsem, zasebni ključ pa je potrebno skrivati. Zgolj informativno naj omenim še to, da v primeru ko je lastnik para ključev človek, se zasebni ključ še dodatno zaščiti z uporabo simetričnega ključa. To pomeni da četudi napadalec ukrade prenosni medij ali vdre v sistem, še vedno ne more ugotoviti kakšen je zasebni ključ saj ne pozna simetričnega ključa. Vidimo torej, da je simetričen način kriptiranja v tem primeru edini primeren, saj je človek edini, ki kriptira in dekriptira vrednost zasebnega ključa. Osnovna lastnost asimetričnega načina je torej ta, da kar se zakriptira z javnim ključem se lahko dekriptira samo z zasebnim ključem. To pomeni, da predhodna izmenjava ključev ni potrebna in prav tako ni potreben več varni kanal. Seveda tudi pri tem načinu obstaja varnostna težava v obliki pasivnega poslušalca (naj mu bo ime Janez). Le ta lahko generira lasten par ključev in javni ključ objavi v imenik pod nekim drugim imenom (naj bo Peter). Ko bo neki Marko v prosto dostopni baz javnih ključev želel pridobiti Petrov javni ključ, bo seveda sedaj

dejansko dobil ponarejeni javni ključ, ki ga je generiral Janez. Res je, da v senzorskih omrežjih takšnih problemov ne bo saj eno senzorsko omrežje pripada eni domeni, ki pa ji lahko prednastavimo javne ključe. V realnem svetu pa takšne težave obstajajo in se lahko rešujejo tako, da vpeljemo agencijo ki zagotavlja, da je vsako osebo autenticirala in autorizirala. Lahko vpeljemo tudi koncept mreže zaupanja kjer poiščemo prijatelje osebe, ki ji želimo nekaj poslati in potem sklepamo tako, da če prijatelji zaupajo tej osebi in jaz poznam kakšnega od prijateljev, potem zaupam tudi tretji osebi. Kljub temu pa še vedno ostane dvom o tem komu zaupati saj še vedno nimamo zagotovila, da je nekdo res tisti za kogar se predstavlja. Rešitev problema je seveda ta, da dovolimo podpisovanje tudi z zasebnim ključem. Seveda lahko potem kdorkoli objavi čistopis a pomembno je to, da je kriptopis lahko generiral samo lastnik zasebnega ključa. Podobno je pri dokumentu na katerega se ročno podpišemo in to pomeni, da nihče drug ni mogel dokumenta podpisati. V že omenjenem konceptu mreži zaupanja bi digitalni podpis pomenil zagotovilo, da sta se dve osebi medsebojno autenticirali oziroma predstavili na elektronski način. To bi storili tako, da bi naša oseba Marko izračunal neko zgoščeno vrednost vsebine pisma v katerem bi zagotavljal poznavanje Petra. Nato bi s svojim zasebnim ključem podpisal samo zgoščeno vrednost. Ko bi Peter prejel elektronski dokument, bi uporabil javni ključ zato da dešifrira zgoščeno vrednost. Nato bi še sam izračunal zgoščeno vrednost pisma in jo primerjal s tisto, ki jo je izračunal Marko. V primeru ujemanja bo vedel, da pisma na prenosni poti nihše ni spreminjal hkrati pa ve tudi to, da je pismo res poslal Marko. V senzorkih omrežjih je podpisovanje z zasebnim ključem zelo pomembno saj le tako lahko preprečimo vdiralcu vnos škodljivih podatkov v senzorsko omrežje. Pakete pred podpisa tako lahko identificiramo in zavržemo.

Homomorfizem zasebnosti Če kot primer vzamemo hierarhično senzorsko omrežje, ki si ga lahko predstavljamo kot drevo. Zahtevamo tudi, da morajo biti vsi podatki kriptirani. To seveda ni problem za liste v našem drevesu saj le ti delajo samo nad svojimi podatki. Problem se pojavi na poti do korena drevesa saj se lahko nabere veliko podatkov in zgodi se lahko poplavljanje višje ležečih vozlišč oziroma korena vozlišč. Seveda obstajajo tehnike združevanja podatkov v vmesnih vozliščih, ki takšne poplave preprečujejo. Problem je tudi ta, da vmesna vozlišča dekriptirajo podatke vseh svojih podrejenih vozlišč oziroma otrok. To je seveda lahko velika obremenitev za drugače precej nepomembna vmesna vozlišča saj bi le ta morala varčevati z energijo saj izpad enega takšnih vozlišč pomeni izpad celotnega poddrevesa. Torej bi potrebovali način, ki omogoča procesiranje podatkov brez predhodnega dešifriranja in hkrati ohranaj vsebino. Vozlišče, ki izvaja združevanje podatkov ne bo nujno tudi interpretiralo podatke ampak mora imeti možnost dela s podatki. Koncept, ki zgornje zahteve izpolnjuje se imenuje homomorfizem zasebnosti in je znan že od leta 1978. Homomorfizem zasebnosti je enkripcijska funkcija, ki dovoljuje operacije kot so seštevanje in množenje. Result bo podoben rezultati, ki bi ga dobili, če bi operacijo najprej izvedli na čistopisu in potem kriptirali. Primer kjer se uporabi simetričen ključ tako za kriptiranje kot za dekriptiranje v senzorskih omrežjih je podan v [7].

Varnost protokola 802.15.4 Brezžična senzorska omrežja temeljijo na protokolu 802.15.4, ki opisuje samo fizično plast (PHY) in pa plast dostopa do

prenosnega medija (MAC). Kot je opisano v [2] povezovalna plast zagotavlja štiri osnovne varnostne storitve. Gre za kontrolo dostopa, integriteto sporočil, zaupnost sporočil in zaščita pred ponovitvijo prenosa. Če aplikacija ne zahteva varnosti in ne nastavi določenih parametrov, potem varnosti ne bo. Kontrola dostopa in integriteta sporočil nam zagotavljata, da protokol ne bo dovolil sodelovanja v omrežju tistim, ki za to nimajo pravic. Za zagotavljanje autentikacije in integritete sporočil se uporablja autentikacijska koda sporočila (Message Authentication Code - MAC), ki se pripne na vsako sporočilo. Za izračun MAC vrednosti morajo oddajniki in sprejemniki sporočil imeti nek skupen kriptografski ključ. V tabeli 1 so podane vse varnostne rešitve, ki so definirane v standardu 802.15.4.

Name	Description
Null	No Security
AES-CTR	Encryption only, CTR Mode
AES-CBC-MAC-128	128 bit MAC
AES-CBC-MAC-64	64 bit MAC
AES-CBC-MAC-32	32 bit MAC
AES-CCM-128	Encryption & 128 bit MAC
AES-CCM-64	Encryption & 64 bit MAC
AES-CCM-32	Encryption & 32 bit MAC

Figure 1: Varnostne rešitve standarda 802.15.4

Tabela 2 prikazuje tudi primerjavo arhitektur varnostnih rešitev. Opazimo lahko, da se je trend rešitev za autentikacijo v WSN omrežjih premikal od mehanizmov s predhodnim nameščanjem ključev, do simetričnih dogovorov o ključih (SKA) in nato do kriptografije z eliptičnimi krivuljami, kar je natančneje opisano v [2].

Protocol	Encryption	Freshness (CTR)	Overhead	MAC Used	Key Agreement	Release Year
SPINS	Yes	Yes	8 Bytes	Yes	Symmetric Delayed	2002
LEAP	Yes	No	Variable	Yes	Pre-Deployed Variable	2003
TINYSEC	Yes	No	4 Bytes	Yes	Any	2004
ZigBee (Commercial Mode)	Yes	Yes	4, 8 or 16 Bytes	Yes	Trust Center	2005
SM	Yes	No	Variable	Yes	EC-MQV Initial Trust	2006

Figure 2: Varnostne rešitve standarda 802.15.4

Če se sedaj dvignemo malo višje in pogledamo na varnost v WSN z višjega zornega kota potem lahko ugotovimo, da obstajajo različna varnostna vprašanja oziroma slabosti v WSN. Avtorji v [18] navajajo sledeče varnostne pomanjkljivosti:

- Oddaljena lokacija
- Veliko število komponent pri namestitvah

- Aplikacije pri katerih je cena zelo pomembna
- Specifičnost za aplikacije
- Vabljive tarče
- Nenadzorovan dostop
- Storitve vmesne plasti (Middleware services)

Avtorji navajajo, da klasični kriptografski in stenografski prijemi niso možni na napravah, ki imajo zelo malo virov na voljo. Dodatno izpostavljajo tudi upravljanje s ključi(ustvarjanje, izmenjevanje, preklic, obnavljanje). Za zagotavljanje varnosti na fizičnem nivoju navajajo uporabnost rešitve preskakovanja frekvenc. Avtorji potem navajajo tudi načine zaznavanja napadov, ki so bili identificirani tudi že v drugi literaturi [17, 1]. Vse to je seveda potrebno upoštevati pri načrtovanju varnih brezžičnih senzorskih omrežij. Pri tem je verjetno smiselno uporabiti tudi neko ogrodje kot je definirano v [16] in omogoča definicijo varnostnih razredov v WSN, kar posledično pomeni, da lahko tako pridobljena znanja uporabimo pri definicijah morebitnih novih protokolov, ki bi imela ta znanja vgrajena v sam protokol in bi tako zagotavljali višji nivo varnosti v WSN. Eliptične krivulje so se torej izkazale za uporabne v WSN, veliko raziskovalcev po svetu se ukvarja z raznimi analizami v katerih poskušajo ugotavljati uporabnost kriptografije eliptičnih krivulj v WSN. To je seveda razumljivo saj v osnovi pri ECC potrebujemo precej manjše ključe kot pri recimo klasičnem RSA. V [12] je naveden podatek da je pri ECC uporaba ključev dolžine 132 bitov ekvivalentna ključem dolžine 952 bitov pri RSA in DSA. Dodatni razlogi v prid uporabe PKC na podlagi ECC v WSN so podani tudi v [13]. Varnostne storitve kot sta autentikacija in upravljanje s ključi nosijo ključno vlogo pri komunikacijski varnosti v WSN kot tudi pri varnosti aplikacij senzorskih omrežij. V tradicionalnih omrežjih kot je internet je PKC osnova za številne varnostne storitve in protokole (npr. SSL in IPSec). Zgodba pa je pri WSN drugačna in se PKC ni uveljavila ravno zaradi omejenih virov predvsem bi izpostavil omejeno energijo, ki jo vsebujejo baterije, ki napajajo senzorje. Zato se je veliko raziskovalnega dela usmerilo v razvoj tehnik, ki ne temeljijo na PKC. Kot primer lahko omenimo tehniko naključnega pred-razdeljevanja ključev in pa tehniko broadcast autentikacije. Problem pri teh tehnikah pa je, da ne ponujajo dovolj visokega nivoja varnosti kot PKC. Kot primer zopet lahko uporabimo tehniko naključnega predhodnega razdeljevanja ključev, ki ni ne zagotavlja hkrati izmenjave ključev med dvema vozliščema in toleranco komprimiranosti poljubnega vozlišča. V PKC seveda to ni problem saj lahko uporabimo Diffie-Hellman protokol za izmenjevanje ključev in pri tem nimamo težav, če nam kdo ukrade poljubno vozlišče v omrežju. Prav tako lahko PKC zagotovi broadcast autentikacijo z uporabo sheme za digitalne podpise ECDSA. Zato je torej smiselno raziskovati možnosti uporabe PKC v WSN.

3 Eliptične krivulje

Kriptografija z eliptičnimi krivuljami (ECC) je pristop k PKC (Public-Key Cryptography), ki temelji na algebraični stukturi eliptičnih krivulj nad končnimi obsegi [9]. Eliptične krivulje, ki se jih uporablja v kriptografiji tipično definiramo nad dvemi tipi končnih obsegov:

- praštevilski obseg F_p kjer je p veliko praštevilo
- obsegi binarnih razširitev F_{2^m}.

Slika 3 prikazuje primere končnih obsegov, ki se uporabljajo v kriptografiji z eliptičnimi krivuljami.

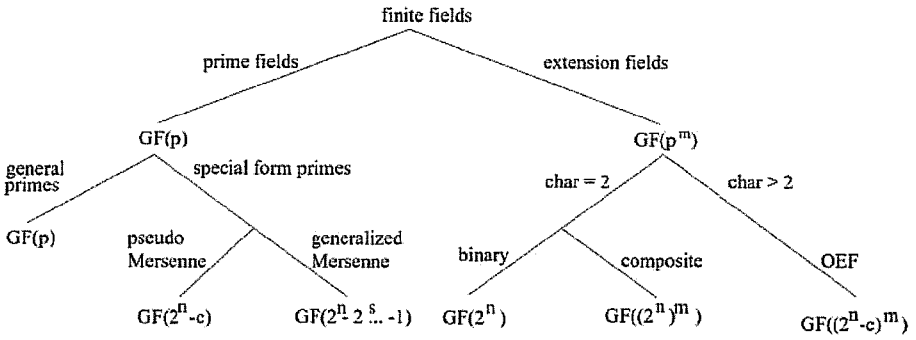

Figure 3: Končni obsegi

Ker pa se želimo ukvarjati z zagotavljanjem varnosti v brezžičnih senzorskih omrežjih, kjer so različni viri (npr. prostor) tipično zelo omejeni, pa se bomo ukvarjali samo z praštevilskimi obsegi Fp. Eliptična krivulja nad praštevilskim obsegom F_p je definirana kot kubična enačba:

$$y^2 = x^3 + ax + b \tag{1}$$

in za konstanti $a, b \in F_p$ velja:

$$4a^3 + 27b^3 \neq 0 \tag{2}$$

Enačba 1 je oblika Weierstrass enačbe, ki je v osnovi definirana kot

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{3}$$

Dve eliptični krivulji sta izomorfni, če lahko iz ene enačbe pridelamo drugo na podlagi zamenjave spremenljivk enačbe. Takšna transformacija se imenuje dopustna sprememba spremenljivk in je definirana kot 4:

$$(x, y) \rightarrow (u^2x + r, u^3y + u^2sx + t) \tag{4}$$

Če izbrano praštevilo p v F_p ni enako 2 ali 3, potem je transformacija 4 oblike, kot je prikazano na 5

$$(x, y) \rightarrow (\frac{x - 3{a_1}^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{{a_1}^3 + 4a_1a_2 - 12a_3}{24}) \tag{5}$$

V tem primeru se Weierstrass enačba 3 pretvori v izomorfno in precej preprostejšo enačbo, ki smo jo definirali v 1, pri tem sta $a, b \in F_p$. V tem primeru je diskriminanta $\Delta = -16(4a^3 + 27b^2)$. Pogoj $\Delta \neq 0$ zagotavlja zveznost eliptične krivulje, ki je seveda pomemben saj pove, da na krivulji ne obstaja točka, ki bi imela dve različni tangenti. Od tod tudi izpeljemo pogoj, ki smo ga definirali z 2.

Za eliptično krivuljo E definirano nad končnim praštevilskim obsegom F_p obstaja pravilo (chord-and-tangent rule), ki omogoča, da na podlagi vsote dveh točk krivulje E pridobimo tretjo točko krivulje E.

Pravilo seštevanja Pravilo seštevanja opišemo geometrijsko takole: naj bosta $P = (x_1, y_1)$ in $Q = (x_2, y_2)$ dve različni točki na eliptični krivulju E.Vsoto R teh dveh točk pridobimo tako, da narišemo premico skozi ti dve točki in premica bo sekala eliptično krivuljo v tretji točki. Vsota obeh točk je podana s točko, ki jo dobimo s preslikavo tretje točke preko osi x.

Pravilo podvajanja točk Pravilo podvajanja točke pa geometrijsko opišemo takole: Nariši tangento na eliptično krivuljo v podani točki $P(x, y)$. Dobljena tangenta bo eliptično krivuljo sekala v drugi točki. Nato naredimo preslikavo dobljene točke preko osi x in dobljena preslikana točka R je torej $2P(x, y)$.

Eliptična krivulja nad praštevilskim obsegom F_p je sestavljena iz vseh parov afinih koordinat (x, y) za $x, y \in F_p$, ki zadoščajo enačbi 2 in točki v neskončnosti Σ. Podane točke na krivulji tvorijo abelovo grupo kjer je Σ aditivna identiteta. Vsa pravila, ki za to grupo veljajo so navedena v nadaljevanju.

1. Identiteta $P + \infty = \infty + P$ za vsak $P \in E(F_p)$

2. Inverz Če $P = (x, y) \in E(F_p)$, potem $(x, y) + (x, -y) = \infty$. Točko $(x, -y)$ označimo z $-P$ in predstavlja inverz točke P. $-P$ je dejanska točka v $E(F_p)$. Velja tudi $-\infty = \infty$.

3. Seštevanje točk Naj bosta $P = (x_1, y_1) \in E(F_p)$ in $Q = (x_2, y_2) \in E(F_p)$ kjer $P \neq \pm Q$. Potem velja $P + Q = (x_3, y_3)$ kjer $x_3 = (\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2$ in $y_3 = (\frac{y_2 - y_1}{x_2 - x_1})(x_1 - x_3) - y_1$

4. Podvajanje točk Naj bo $P = (x_1, y_1) \in E(F_p)$ in $P \neq -P$. Potem je $2P = (x_3, y_3)$ kjer je $x_3 = (\frac{3x_1^2 + a}{2y_1})^2 - 2x_1$ in $y_3 = (\frac{3x_1^2 + a}{2y_1})(x_1 - x_3) - y_1$

Če je E eliptična krivulja definirana nad F_p potem številu točk v $E(F_P)$ označeno z $\#E(F_p)$ imenujemo red krivulje E nad F_p. Ker za Weierstrass enačbo 3 velja, da ima za vsak $x \in F_p$ največ dve rešitvi potem velja $\#E(F_p) \in [1, 2p + 1]$. Obstaja pa še Hasse-jev teorem, ki še bolj natančno opredeli meje za red krivulje in sicer z intervalom, ki mu pravimo Hasse-jev interval:

$$[p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}] \tag{6}$$

Alternativna razlaga Hasse-jevega teorema je: če je E definiran nad F_p, potem je $\#E(F_p) = p + 1 - t$ kjer je $|t| \leq 2\sqrt{p}$ in t imenujemo sled (ang. trace) krivulje E nad F_p. Ker pa je število $2\sqrt{p}$ majhno v primerjavi s številom p, lahko rečemo $\#E(F_p) \approx p$. Velja še, da če izbrano praštevilo v F_p deli sled t, potem pravimo, da je eliptična krivulja E supersingularna.

Za vsako točko G na eliptični krivulji je množica $\Sigma, G, 2G, 3G, \ldots$ ciklična grupa. Izračun kG, kjer je k celo število se imenuje skalarno množenje. Skalarno množenje izvajamo z uporabo že omenjenih pravil za seštevanje in podvajanje točk saj gre pri skalarnem množenju kP k-kratno prištevanje točke P sami sebi. Matematični problem iskanja števila k pri podanih točkah kG in G se imenuje problem diskretnega logaritma eliptičnih krivulj (ECDLP). Ker rešitve tega problema ne moremo izračunati z uporabo procesorske moči v nekem normalnem času, nam to dejstvo torej omogoča uporabo različnih kriptografskih shem,

ki temeljijo na eliptičnih krivuljah. Programska knjižnica TinyECC ponuja tri različne kriptografske sheme:

- Diffie-Hellman shema za dogovor o ključih (ECDH-Elliptic Curve Diffie-Hellman key agreement protocol. Gre za različico Diffie-Hellman protokola kot je definiran v [3])
- Algoritem za digitalno podpisovanje (ECDSA-Elliptic Curve Digital Signature Algorithm. Gre za različico DSA algoritma)
- Shema za integrirano enkripcijo (ECIES-Elliptic Curve Integrated Encryption Scheme. Gre za kodirno shemo, ki ponuja semantično varnost pred napadalci, ki lahko napadajo z izbranimi čistopisi ali izbranimi kriptopisi kot je opisano v [9]).

Prednost kriptografskih shem, ki temeljijo na neizračunljivosti ECDLP problema je v tem, da je potrebnih manj bitov za zagotavljanje enakega nivoja zaščite kot klasični DSA in DH kriptografski shemi. Pri vsaki od omenjenih shem se mora vsak sodelujoči člen, ki bi sheme rad uporabil, sprejeti nekaj vsebinskih parametrov kot so uporabljena eliptična krivulja in točka G na krivulji, prav tako pa potrebuje tudi par ključev, ki je sestavljen iz zasebnega ključa d in javnega ključa $Q = dG$. Parametri eliptične krivulje definirane nad F_p so

$$T = (p, a, b, G, n, h) \tag{7}$$

p predstavlja praštevilski končni obseg F_p,
a, b predstavljata eliptično krivuljo definirano z enačbo 1,
$G = (x_G, y_G)$ je bazna točka na $E(F_p)$,
n je praštevilo in predstavlja red točke G,
h je celo število, ki predstavlja kofaktor $h = \frac{\#E(F_p)}{n}$

Generiranje parametrov Parametre domene eliptične krivulje definirane nad F_p generiramo tako:
Vhodni podatek: nivo varnosti v bitih podan kot celo število $t \in [56, 64, 80, 96, 112, 128, 192, 256]$
Izhodni podatek: $T = (p, a, b, G, n, h)$ za katere velja, da računanje logaritmov na dobljeni eliptični krivulji vzame približno 2^t operacij.
Algoritem:

1. Izberi praštevilo p da velja $\lceil log_2 p \rceil = 2t$, če $t \neq 256$ in vzami $\lceil log_2 p \rceil = 521$, če $t = 256$ in tako definiraj obseg F_p.
2. Izberi elemente a,b,G,n in h pri čemer upoštevaj sledeče omejitve:
   - $4a^3 + 27b^2 \neq 0 (\text{mod } p)$
   - $\#E(F_p) \neq p$
   - $p^B \neq 1 (\text{mod } n)$ za vsak $1 \leq B \leq 20$
   - $h \leq 4$
3. Na izhod zapiši $T = (p, a, b, G, n, h)$

Seveda lahko tako pridobimo poljubno kombinacijo parametrov vendar je zaradi zahtev po interoperabilnosti priporočljivo, da uporabimo vnaprej definirane nabore teh parametrov, ki so seveda tudi že standardizirani.

Par ključev Pri podanih parametrih domene eliptične krvulje $T = (p, a, b, G, n, h)$ z (d, Q) označujemo par ključev. Pri tem je d skriti oziroma zasebni ključ, ki je celo število z intervala $[1, n-1]$, javni ključ pa je $Q = (x_q, y_q)$, ki predstavlja točko $Q = dG$.

ECDH - Diffie-Hellman z uporabo eliptičnih krivulj Končni rezultat postopka je torej nova skrita vrednost s katero se potem kriptira sporočila pri prenosu skozi nezaščiten komunikacijski kanal pri komunikaciji med entiteto U in entiteto V. Postopek je sledeč:

Vhodni podatki:

- $T = (p, a, b, G, n, h)$
- privatni ključ d_U, ki pripada entiteti U
- javni ključ Q_V, ki pripada entiteti V

Izhodni podatki:Bodisi dobimo novo skrivnost z, ki si jo obe entiteti delita bodisi pa dobimo informacijo o napaki.
Algoritem:

1. Izračunaj točko na eliptični krivulji $P = (x_P, y_P) = d_U Q_V$ (v tem koraku bi lahko izvedli množenje oblike $hd_U Q_V$ in tako zaščitili ključe pred nekaterimi napadi kot so napadi malih podgrup.
2. Preveri da $P \neq 0$. Če $P = 0$, potem na izhod zapiši obvestilo o napaki.
3. Na izhod zapiši vrednost $z = x_P$ kot novo deljeno skrivnost med dvema entitetama U in V.

Seveda je zgoraj dobljena skupna skrivnost pri obeh entitetah enaka saj velja enačba 8.

$$d_U Q_V = d_U d_V G = d_V d_U G = d_V Q_U \tag{8}$$

4 Razvojno-testno okolje

Pri svojem spoznavanju pripomočkov in orodij za delo z brezžičnimi senzorskimi omrežji sem uporabljal pernosni računalnik s Core Duo procesorjem in 3GB pomnilnika. Nameščen sem imel operacijski sistem Ubuntu 8.10 z linux jedrom 2.6.27-17-generic. Seveda je pri delu na operacijskih sistemih Linux potrebno precej pogosto uporabljati konzolo, ker pa ima osnovna konzola, ki pride nameščena z distribucijo, nekaj pomanjkljivost (predvsem z vidika uporabniške izkušnje), sem uporabljal Terminator kot alternativno rešitev. Omogoča namreč poljubno deljenje ene konzole na več delov bodisi na vertikalen ali horizontalen način. Za pregledovanje in pisanje programske kode sem uporabljal odprtokoden IDE eclipse. Dodatno sem si seveda moral namestiti tudi nekaj dodatkov kot je tudi 4.3. Le ta mi je omogočal razvoj novih aplikacij za operacijski sistem TinyOS. Tudi TinyOS sem si namestil na računalnik. Več o tem operacijskem sistemu pa v 4.1. Namestil sem si tudi izvorno kodo knjižnice TinyECC o čemer bom povedal več v 4.2. Ker v času dela na tem projektu nisem imel na voljo strojne opreme na kateri bi lahko preizkusil aplikacije, sem seveda uporabil emulator

Aurora 4.4 kot alternativo. Dodatno sem se ukvarjal tudi z programoma WSim in WSNet saj omogočata simulacijo na višjem nivoju kot Aurora. To pomeni, da program, ki ga napiš razvijalec za TinyOS lahko simulacijsko izvajamo kar na WSim in torej ne potrebujemo nameščenega TinyOS sistema niti emulatorja Aurora. Če uporabimo tudi WNet pa lahko simuliramo celotno brezžično senzorsko omrežje saj nam le ta zagotavlja povezljivost.

4.1 TinyOS

Je odprtokodni operacijski sistem, ki je namenjen brezžičnim vgrajenim senzorskim omrežjim. Arhitektura temelji na komponentah kar omogoča hitro implementacijo neke inovativne funkcionalnosti hkrati pa je količina programske kode minimalna saj so senzorska omrežja zelo omejena v smislu količine pomnilnika in ostalih virov. V knjižnici komponent tako najdemo omrežne protokole, porazdeljene storitve, gonilnike senzorjev in pa orodja za zajem podatkov. Vse komponente lahko seveda uporabimo takšne kot so ali pa jih še dodatno dpolnimo oziroma nadgradimo za potrebe naših aplikacij. TinyOS je bil prvotno razvit kot produkt raziskovalnega projekta na Univerzi Berkley v Kaliforniji a se je s časom razvil in pridobil mednarodno skupnost razvijalcev in uporabnikov.

4.2 TinyECC

Glavni cilj knjižnice TinyECC je ponuditi brezplačno odprtokodno programsko rešitev za ECC operacije, ki jih lahko zelo prilagajamo našim potrebam in integriramo v aplikacije nad brezžičnimi senzorskimi omrežji. TinyECC vsebuje številne optimizacijske možnosti, ki jih razvijalec lahko vklopi in jih uporabi ali pa ne seveda v odvisnosti od konkretne aplikacije. Različne kombinacije optimizacij imajo seveda različno porabo virov in čas izvajanja, kar seveda pomeni, da imajo razvijalci veliko fleksibilnosti pri integraciji knjižnice v svoje aplikacije. Knjižnica je napisana za uporabo na TinyOS in je napisana z uporabo nesC(network embedded systems C) programskega jezika, ki je bolj natančno opisan v [6].

Načrtovalski principi Pri načrtovanju TinyECC so se avtorji držali sledečih principov:

- Varnost - vgrajena je podpora za ECDH, ECDSA in ECIES. Upoštevana so tudi priporočila za parametre eliptičnih krivulj, ki jih definira SECG (Stands for Efficient Cryptography Group). Gre predvsem za secp160k1, secp160r1 in secp160r2.
- Prenosljivost - TinyECC naj bi bilo možno uporabiti na čim večjemu številu različnih platform. Zato so se odločili za TinyOS, ki že sam po sebi velja za defacto standard na področju WSN.
- Zavedanje virov in konfigurabilnost - implementacija je bila izvedena zelo previdno, da se viri ne bi po nepotrebnem uporabljali.
- Učinkovitost - Za povečanje učinkovitosti so uporabili tri ukrepe: ECC so implementirali nad F_p saj so artimetične operacije nad F_2^m slabo podprte v mikrokontrolerjih; implementirali so večino obstoječih optimizacij za

ECC; nekatere dele ECC operacij so implementirali z uporabo zbirnega jezika za najbolj pogosto uporabljene platforme WSN.

- Funkcionalnosti - TinyECC podpira najbolj pogosto uproabljene PKC funkcije. Gre za ECDS, ECDH in ECIES.

Uporabljene optimizacije

1. Optimizacija operacij nad veliki celimi števili: prva je Barett-ova redukcija - bolj učinkovit način od klasičnega deljenja postane takrat, ko veliko različnih števil delimo z enakim številom seveda z uporabo tabele s predhodno izračunami rezultati. V našem primeru se bodo deljenja izvajala s praštevilom p in je torej uporaba te optimizacije smiselna. Problem je seveda več kode za implementacijo kot pri navadnem deljenju in imamo torej večjo porabo ROM-a, hkrati pa se poveča tudi poraba RAM-a. Če ima mikrokontorler velikost besede w potem je potrebno predhodno izračunati število $\mu = \lfloor \frac{b^k}{p} \rfloor$ pri čemer je $b = 2^w$. To število se potem uporablja pri vseh modulo redukcijah. Druga optimizacija pa sta hibridno množenje in hibridno kvadriranje pa sta metodi, ki omogočata bolj učinkovito izrabo registrov mikroprocesorja in tako zmanjšata število dostopov do pomnilnika.

2. Optimizacije ECC operacij - a)projekcijski koordinatni sistem - gre za to, da točke na eliptični krivulji predstavimo v obliki (x, y, z). Na ta način se znebimo izračuna modularnih inverzov, ki se pojavijo pri osnovnih operacijah nad ECC (glej 3). Seveda je tudi tu večja poraba pomnilniškega prostora. Dodatni optimizaciji sta tudi seštevanje mešanih točk kjer seštevamo eno točko definirano z afinimi koordinatami in drugo točko definirano z projekcijskimi koordinatami, ter ponavljajoče podvajanje za skalarno množenje. Prva od obeh dodatno zmanjša število potrebnih modularnih množenj. Druga optimizacija pa omogoča hitrejši izračun saj pri m zaporednih podvajanjih točke ta algoritem namesto $m-1$ seštevanj in $m-1$ deljenj porabi samo 2 in še dodatno množenje za dva kvadrata.
b) plavajoče okno za skalarno množenje - gre za osnovno operacijo kP kjer je k celo število in P neka točka na eliptični krivulji. Navadno se računanje izvede tako, da se sprehodimo od najbolj pomembnega bita do najmanj pomembnega bita v k in pri vsakem bitu izvedemo eno operacijo podvajanja točke. Če je bit enak 1 pa je potrebno še sešteti dve točki. Pri tej optimizaciji se hkrati pregleduje w bitov. Za vsakih w bitov algoritem izračuna w podvajanj točke. Ker pa se vrednosti $2P, 3P...(2^w-1)P$ lahko izračunajo vnaprej, mora ta algoritem izračunati samo eno seštevanje točk vsakih w bitov namesto w seštevanj. Seveda potrebujemo za vnaprej izračunane vrednosti dodaten prostor v RAM, zaradi večje količine kode pa tudi več ROM-a.
c)Shamir-ov trik: uporablja se samo pri preverjanju pravilnosti ECDSA podpisa kjer je potrebno izračunati $aP + bQ$. Tu rabimo dve skalarni množenji in eno seštevanje točk. Shamir pa je pokazal, kako ta izračun narediti v času podobnem enemu skalarnemu množenju. Tudi pri tej optimizaciji se poveča prostor v ROM in v RAM.

d) Krivuljam specifična optimizacija - številne eliptične krivulje uporabljajo pseudo-Marsenn praštevila oblike $p = 2^n - c$. Redukcija po modulu takšnega praštevila se lahko izvede brez operacij deljenja kar zelo pospeši operacijo modularne redukcije.

3. Rezultati meritev uporabe različnih konfiguracij knjižnice so podrobno analizirane v [13] zato bralca pozivam, da si jih ogleda. Meritve kažejo na to, da je z različnimi možnimi konfiguracijami TinyECC resnično mogoče prilagoditi naši strojni opremi oziroma virom, ki so nam na voljo. Izpostaviti pa je potrebno še dodaten problem, ki se pojavi pri uporabi PKC in sicer možnost DOS napadov kjer bi napadalec lahko prepričal senzorje, ad izvajajo številne PKC operacije, kar bi posledično izčrpalo zalogo energije, ki jo baterije imajo. Za zaščito pred takšnimi napadi je smiselno razmisliti o uporabi dodatnih mehanizmov kot je definiran v [15].

4.3 Yeti2

Gre za modul, ki ga dodamo v Eclipse razvojno okolje in nam omogoča preprostejšo izdelavo novih aplikacij za TinyOS. Funkcionalnosti, ki jih modul podpira so sledeče:

- Zaznavanje napak - modul preveri datoteko z izvorno kodo tako, da razvijalca opozarja na potencialne probleme (primer: overflow problem pri implicitnemu cast-anju)
- Dopolnjevanje programske kode - pohitri razvoj saj nam modul ddopolni imena funkcij, spremenljivk in ostalega.
- Iskalnik - omogoča iskanje deklaracij, referenc, vmesnikov, modulov in funkcij.
- Povezave - omogočajo hitre prehode do izvornih deklaracij spremenljivk, konstant, funkcij in ostalega.
- NesC dokumentacija - če se z miškinim kazalcev ustavimo nad elementov kot je ime funkcije, se nam prikaže okno, ki poda opis funkcije kot je podan v uradni dokumentaciji za programski jezik NesC.
- Modul omogoča tudi pregled vseh elementov, ki so v NesC datoteki prisotni v obliki drevesa. Tudi če je deklaracije v kateri drugi datoteki, se drevo dopolni tudi s podatki iz te druge datoteke.
- Grafični prikaz omogoča pregled elementov izvorne datoteke v obliki grafa na katerem so vidne tudi odvisnosti med elementi.
- Predprocesor je prav tako vgrajen v modul in omogočen je tudi pregled izhoda predprocesiranja.

Modul je potrebno dodati v Eclipse in sicer preko posodobitvene strani http://tos-ide.ethz.ch/update/site.xml.

4.4 Aurora

Za potrebe spoznavanja knjižnice TinyECC smo uporabili emulator Avrora (http://docs.tinyos.net/index.php/Avrora), ki nam omogoča zagon programov, ki so napisani za operacijski sistem TinyOS. Težava je v tem, da strojne opreme v obliki nekih senzorjev na katere bi lahko namestili TinyOSS nismo imeli na voljo. Zato smo v ta namen TinyOSS namestili kar na prenosni računalnik z nameščenim Ubuntu Linux sistemom. Poleg tega smo kot že rečeno namestili tudi emulator operacijskega sistema TinyOSS imenovan Avrora.

Navodila za uporabo emulacijskega okolja Avrora se nahajajo na naslovu http://docs.tinyos.net/index.php/Avrora.

1. Namestiš TinyOSS (Glej navodila na http://docs.tinyos.net/index.php)
2. Poženeš ant /opt/tinyos-2.1.0/support/sdk/java –¿build.xml se zažene in prevede javansko kodo, ki se kasneje uporablja v aplikacijah
3. Nastaviš spremenljivko TOSROOT, ki naj kaže na TinyOSS namestitveno mapo (/opt/tinyos-2.1.0)
4. Premakneš se v mapo TOSROOT/apps/Blink
5. Zaženeš ukaz make micaz, ki prevede aplikacijo Blink
6. Izvedeš še ukaz mv build/micaz/main.exe Blink.elf
7. Nato z avrora emulatorjem zaženeš aplikacijo Blink: java avrora.Main -platform=micaz -simulation=sensor-network -seconds=3 -monitors=leds Blink.elf

Izpis izgleda takole:

```
Avrora [Beta 1.7.110] - (c) 2003-2007 UCLA Compilers Group

Loading /opt/tinyos-2.1.0/apps/Blink/Blink.elf...OK
=={ Simulation events }=========================================================
Node          Time   Event
--------------------------------------------------------------------------------
   0       8006682  off off on
   0       8006684  off on  on
   0       8006686  on  on  on
   0       8006688  on  on  off
   0       8006690  on  off off
   0       8006692  off off off
   0       9808089  off off on
   0      11608089  off off off
   0      11608562  off on  off
   0      13408089  off on  on
   0      15208089  off on  off
   0      15208562  off off off
   0      15209090  on  off off
   0      17008089  on  off on
   0      18808089  on  off off
```

```
   0       18808562  on  on  off
   0       20608089  on  on  on
=====
Simulated Time: 22118400 cycles
```

Podobno lahko preizkusimo tudi preostale aplikacije, ki so že nameščene na TinyOSS distribuciji.

4.5 WSim

WSim je simulator platforme. Omogoča popolno simulacijo platforme do cikla natančno in pri tem uporablja podatke o času izvrševanja posameznih uka- zov na mikrokontrolerjih, ki jih specificirajo proizvajalci le teh. Simulator omogoča popolno simulacijo strojnih dogodkov, ki se zgodijo na platformi in 9nato razvijalcu vrne natančno časovno analizo programske opreme, ki jo anal- izira. Za samo simulacijo lahko uporabimo programe, ki so napisani za siljno platformo in torej ni potrebno nekega ponovnega prevajanja programske kode. Simulator uporablja klasični GCC "cross-compiler toolchain" in sama simulacija ni omejena na nek določen programski jezik ali operacijski sistem. To pomeni, da je možno izvajati razhroščevanje in ocenjevati performance celotnega sistema na nivoju zbirnika. Simulacija pridobi natančne časovne podatke, porabi pomnilnika in energije. Operacijski sistem TinyOS je bil uspešno uporabljen za simulacije, prav tako pa tudi še dva podobna operacijska sistema Contiki in FreeRTOS. Simulator lahko uporabimo v standalone načinu za potrebe razhroščevanja, ko ne uporabimo fizičnih radijskih naprav (ozr. tega ne želimo testirati). Seveda pa je ena glavnih funkcionalnosti, ki jih WSim ponuja ravno ta, da vsebuje vmesnik do WSNet simulatorja kar omogoča simulacijo celotnega senzorskega omrežja. Navodila za namestitev WSim simulacijskega okolja so na voljo na spletnem naslovu http://wsim.gforge.inria.fr/installation.html. Po zaključeni konfiguraciji dobimo izpis podoben spodnjemu, ki nam pove katere komponente so bile omogočene, kar seveda lahko eksplicitno definiramo v ukazni vrstici ali pa enostavno sprejmemo privzete nastavitve.

```
  Configuration summary:
host : i686-pc-linux-gnu
host type : Linux
target(s) : all
wsim debug : yes
target debug : yes
Etrace : no
Elf loader : internal
GUI : yes X11
zlib : system
Ptty : yes
```

Programska koda je prosto dostopna in odprtokodna. Licenca se imenuje CeCill in je dosegljiva na naslovu http://wsim.gforge.inria.fr/license.html. Dodaten opis je na voljo tudi v [5]

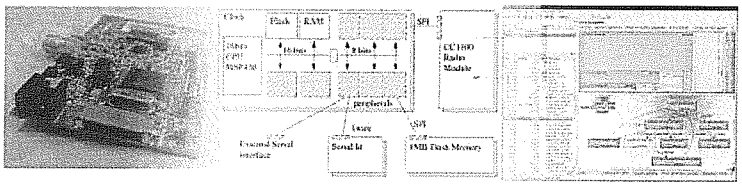

Figure 4: WSim

4.6 WSNet

Gre za program, ki omogo?ca simulacijo na podlagi dogodkov. Glavne funkcionalnosti programa so:

- Simulacija vozlišč
- Simulacija okolja
- Simulacija radijskega medija
- Razširljivost

Simulacija vozlišč Simulirana vozlišča so zgrajena kot poljubna množica blokov, ki lahko predstavljajo strojno opremo, programsko opremo, vir energije ali obnašanje vozlišča. Ni nikakršnih omejitev glede števila blokov ali glede relacij med bloki. Posledično lahko MIMO sistem modeliramo kot sisteme z več radijskimi vmesniki. Bloki lahko modelirajo sledeče komponente/obnašanje:

- mobilnost
- vir energije
- aplikacija
- usmerjevalni protokol
- mac protokol
- aplikacije
- radijski vmesnik
- antena

Primer arhitekture vozlišča prikazuje slika 5

Natančnejša dokumentacija je na voljo na spletnem naslovu http://wsnet.gforge.inria.fr/overview.html. Navodila za namestitev so na voljo na spletnem naslovu http://wsnet.gforge.inria.fr/installation.html.

5 Zaključek in nadaljne delo

Med delom na tem projektu sem prebral precej literature s področja varnosti, kriptografije in brezžičnih senzorskih omrežij. Lotil sem se tudi praktičnega preizkusa pridobljenih znanj in si uspel namestiti programski del razvojnega

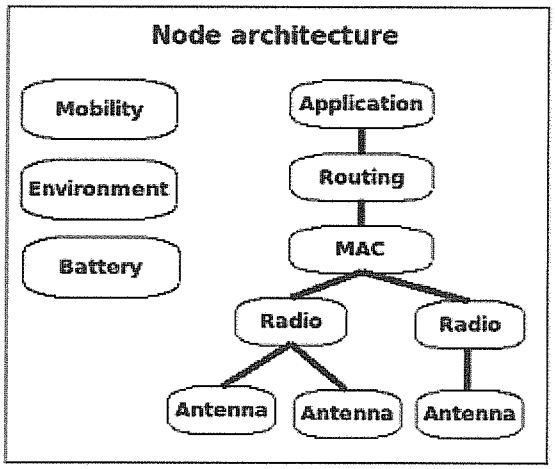

Figure 5: Primer arhitekture vozlišča

okolja, ki mi omogoča razvoj novih aplikacij za brezžična senzorska omrežja, njihovo testiranje na emulatorju naprav in tudi na emulatorju platform, prav tako sem si pripravil že vse potrebno za kasnejše dodajanje strojne opreme oziroma senzorjev. Sedaj ko imam postavljeno takšno okolje torej nimam več ovir pred tem, da bi se lotil kakšnih bolj natančnih analiz kriptografskih prijemov na področju WSN. Vprašanje, ki se mi poraja je sledeče: kakšne so minimalne zahteve glede virov na senzorjih, da lahko uporabimo kriptografijo eliptičnih krivulj. Zelo rad bi videl tudi orodje, ki bi omogočalo načrtovanje senzorskih omrežij z vidika protokolov, virov in pa varnosti, ki bi omogočala jasno upoštevanje vseh teh parametrov pri posameznih aplikacijah. Primer takšnega orodja je opisan v [14]. Vprašanj s področja WSN je še precej tudi na področju autentikacije in integritete sporočil. Protokoli, ki temeljijo na standardu 802.15.4 (ZigBee) definirajo dodatne varnostne ukrepe na višjih nivojih omrežnih plasti, ki pa seveda niso vsi enako dobri. Primer takšne rešitve je podan v [17]. Tudi na teh višjih nivojih je potrebno pridobiti boljše rešitve saj v trenutnih obstajajo prevelike nevarnosti za morebitne napade. Vsa ta vprašanja je smiselno odgovarjati na podlagi nekih empiričnih meritev na realnih senzorskih omrežjih. Zato mislim, da je bil projekt zelo koristen zame saj sem si uspel pripraviti precej dobro podlago za nadaljne delo na področju varnosti v vseprisotnih sistemih.

References

[1] a.S.K. Pathan.
Security in wireless sensor networks: issues and challenges.
2006 8th International Conference Advanced Communication Technology, pages 6 pp.–1048, 2006.

[2] D. Boyle and T. Newe.
Security Protocols for Use with Wireless Sensor Networks: A Survey of Security Architectures.

2007 Third International Conference on Wireless and Mobile Communications (ICWMC'07), pages 54–54, March 2007.

[3] W. Diffie and M. Hellman.
New directions in cryptography.
IEEE Transactions on Information Theory, 22(6):644–654, November 1976.

[4] Helena Fernandez-lopez, Pedro Macedo, and Jose A Afonso.
Performance Evaluation of a ZigBee-based Medical Sensor Network.
Performance Evaluation, pages 2007–2010, 2007.

[5] E. Fraboulet, A. And Chelius, G. And Fleury.
Worldsens: Development and Prototyping Tools for Application Specific Wireless Sensors Networks.
Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on, pages 176 –185, 2007.

[6] David Gay, Philip Levis, Robert Von Behren, Matt Welsh, Eric Brewer, and David Culler.
The nesC language: A holistic approach to networked embedded systems.
volume 38, pages 1–11, San Diego, California, USA, May 2003. ACM.

[7] J. Girao, D. Westhoff, and M. Schneider.
CDA: concealed data aggregation for reverse multicast traffic in wireless sensor networks.
IEEE International Conference on Communications, 2005. ICC 2005. 2005, 00(C):3044–3049, 2005.

[8] Thomas Haenselmann.
Sensornetworks.
Number April. 2006.

[9] Darrel Hankerson, Alfred Menezes, and Scott Vanstone.
Guide to Elliptic Curve Cryptography.
Springer-Verlag, 2004.

[10] Shuo-jen Hsu, Hsin-hsien Wu, Shih-wei Chen, Tsang-chi Liu, Wen-tzeng Huang, Jen Chang, Chin-hsing Chen, and You-yin Chen.
Development of Telemedicine and Telecare over Wireless Sensor Network.
Architecture, pages 597–604, 2008.

[11] H. Kumar, D. Sarma, and a. Kar.
Security threats in wireless sensor networks.
IEEE Aerospace and Electronic Systems Magazine, 23(6):39–45, June 2008.

[12] S. Kumar, M. Girimondo, A. Weimerskirch, C. Paar, A. Patel, and A.S. Wander.
Embedded end-to-end wireless security with ECDH key exchange.
pages 786–789, Cairo, Egypt, 2003. IEEE.

[13] An Liu.
TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks *.
Office, 2008.

[14] S. Moller, T. Newe, and S. Lochmann.
Review of platforms and security protocols suitable for wireless sensor networks.

2009 IEEE Sensors, pages 1000–1003, October 2009.

[15] Peng Ning, An Liu, and Wenliang Du.
Mitigating DoS attacks against broadcast authentication in wireless sensor networks.
ACM Trans. Sen. Netw., 4(1):1–35, January 2008.

[16] Ali Nur and Mohammad Noman.
A generic framework for defining security environments of Wireless Sensor Networks.
2008 International Conference on Electrical and Computer Engineering, 00(December):924–929, December 2008.

[17] Taeshik Shon, Bonhyun Koo, Hyohyun Choi, and Yongsuk Park.
Security Architecture for IEEE 802.15.4-based Wireless Sensor Network.
2009 4th International Symposium on Wireless Pervasive Computing, pages 1–5, February 2009.

[18] Hasan Tahir and Syed Asim Ali Shah.
Wireless sensor networks - a security perspective.
2008 IEEE International Multitopic Conference, pages 189–193, December 2008.