2015•2016
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: elektronica-ICT

Masterproef

Impact van publieke sleutel cryptografie op draadloze sensornetwerken

Promotor :
Prof. dr. ir. Nele MENTENS

Promotor :
ing RUBEN SMEETS

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

Umit Gultekin

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

2015•2016

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterproef

Impact van publieke sleutel cryptografie op draadloze sensornetwerken

Promotor :
Prof. dr. ir. Nele MENTENS

Promotor :
ing RUBEN SMEETS

Umit Gultekin

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

universiteit hasselt | KU LEUVEN

Woord vooraf

Als laatstejaarsstudent aan de Universiteit Hasselt (in samenwerking met de KU Leuven) heb ik de mogelijkheid gekregen om mijn masterproef, die bij mijn studie van Master in industriële wetenschappen hoort, te realiseren bij ES&S. De keuze omtrent het onderwerp cryptografie, binnen deze thesis, is ontstaan uit mijn interesse in het versleutelen van belangrijke gegevens. Enerzijds wil ik mensen helpen om hun gegevens strikt geheim te kunnen verzenden, anderzijds wens ik mij ook te richten op het onderzoek naar een *lightweight* en *low-power node*, voor gebruik in de cryptografie.

De thesis is bedoeld voor informatici, elektronici, onderzoeksgroepen en andere actoren die geïnteresseerd zijn in de cryptografie en de toepassing ervan in draadloze sensornetwerken.

Daarnaast wil ik iedereen bedanken die mij geholpen heeft bij het realiseren van deze masterproef.

In de eerste plaats mijn oprechte dank aan mijn interne promotor, Prof. dr. ir. Nele Mentens. Door haar jarenlange ervaring kon zij mij altijd goede raad geven als er problemen optraden. Ondanks haar drukke tijdsschema stond zij toch steeds paraat om mij te helpen. Ik mag zeker ook niet nalaten om mijn externe promotor, ing. Ruben Smeets, te bedanken. Hij volgde deze masterproef op de voet en gaf raad indien nodig.

Als laatste ben ik zeer dankbaar voor mijn ouders en vrienden voor hun voortdurende steun, liefde en begeleiding tijdens mijn studiejaren. Ik wil dit werk graag opdragen aan hen.

Ümit Gültekin
Master in de industriële wetenschappen: elektronica-ICT

Universiteit Hasselt
Campus Diepenbeek
Faculteit industriële ingenieurswetenschappen
Agoralaan – Gebouw D
3590 Diepenbeek
Tel.: 011 26 81 11
Fax.: 011 26 81 99

Inhoudstabel

Lijst van tabellen

Lijst van figuren

Lijst van woorden

ACLK	Auxillary clock
AES	Advanced Encryption Standard
CA	Certificate authority
CPU	Central Processing Unit
DH	Diffie-Hellman key exchange
DLP	Discrete logarithm problem
DoS	Denial-of-Service
DSA	Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Helmann
ECDSA	Eliptic Curve Digital Signature
ECMQV	Elliptische Curve Menezes-Qu-Vanstone
ES&S	Embedded Systems & Security
FIFO	First In First Out
KU Leuven	Katholieke Universiteit Leuven
MAC	Message authentication code
MCLK	Main clock
MIC	Message integrity code
NIST	National Institute of Standards and Technology
OS	Operating System
RAM	Random-access memory
RISC	Reduced instruction set computer
ROM	Read-only memory
RSA	Rivest, Shamir and Adleman
RSSI	Recieved Signal Strength Indicator
SECG	Stands for Efficient Cryptography Group
STS	Post-To-Station protocol
USB	Universal Serial Bus
WSN	Wireless Sensor Network

Abstract

Door het frequent gebruik van het internet voor allerhande applicaties, zijn draadloze sensornetwerken een belangrijk onderdeel geworden van verschillende toepassingen. Door de toegenomen afhankelijkheid van draadloze sensornetwerken en hun verbinding met het internet, zijn strenge databeveiligingsalgoritmes onmisbaar geworden. Deze netwerken zijn vooral batterij-gevoed en beschikken over beperkte rekenkracht en geheugen, waardoor het invoeren van beveiliging moeilijk en kostelijk wordt. In deze masterthesis zal onderzoek gedaan worden naar de implementatie van publieke sleutelcryptografie in *sensor nodes* van draadloze sensornetwerken. Hieromtrent wordt er ook een analyse uitgevoerd naar de impact op geheugen, energie en rekenkracht.

In deze thesis worden een aantal publieke sleutelalgoritmes geanalyseerd die geïmporteerd kunnen worden op de besturingssystemen. Om de haalbaarheid van publieke sleutelcryptografie te onderzoeken, worden verschillende cryptografische bibliotheken bestudeerd. Uit deze bibliotheken zijn TinyECC, ContikiECC en Relic gekozen voor een meer uitgebreide analyse. Na deze analyse worden de bibliotheken gelinkt aan de geschikte besturingssysteem voor evaluatie.

De resultaten geven aan dat publieke sleutelcryptografie, zonder veel moeite, mogelijk is in de onderzochte besturingssystemen. Belangrijke factoren, zoals snelheid van de processor en RAM-geheugen, leiden tot betere resultaten bij dergelijke implementaties, met gebruik van wiskundige optimalisaties.

Summary

Due to the increasing use of the internet for a multitude of applications, digital sensor networks have become an important part of several applications. This increased dependence on wireless sensor networks has made strong data security algorithms indispensable. These networks are mostly battery-powered and have limited processing power and memory capacity, making the introduction of security difficult and costly. The Embedded Systems & Security (ES&S) research group, part of the KU Leuven, requested a study to be done into the implementation of public key cryptography in wireless sensor network sensor nodes and its impact on memory, power consumption and processing power.

Several public key cryptography algorithms that can be imported on the operating systems are considered in this thesis. To study the feasibility of public key cryptography, several cryptographic libraries are studied. From these libraries, TinyECC, ContikiECC and Relic are investigated further. After investigation, the libraries are ported to a number of operating systems for evaluation.

The results indicate that public key cryptography is possible, without much efforts, on wireless sensor network sensor nodes. Important factors such as processor speed and RAM size, lead to better results in these implementations, with the use of mathematical optimizations.

Hoofdstuk 1

Introductie

1.1 Situering

Draadloze sensornetwerken worden steeds meer gebruikt in ons dagelijks leven. Ze worden gebruikt in bijna elk gebied van elektronica en computerwetenschappen. De oneindige vakgebieden in de wetenschap en technologie passen zich aan om draadloze sensornetwerken te implementeren voor de analyse van de gegevens. Dit geldt zowel voor het gebruik ervan in grote omgevingen, als in kleine nabijheden en voor het *real-time* verzamelen van gegevens. Door deze ontwikkelingen in het gebruik van draadloze sensornetwerken, gaan steeds meer kritische en gevoelige verrichtingen, zowel industrieel als militair, een deel uitmaken van draadloze sensornetwerken. Met een dergelijke hoge mate van afhankelijkheid van grote draadloze sensornetwerken wordt veiligheid een grote zorg. Als een aanvaller toegang heeft tot de gegevens en/of de gegevens kan manipuleren, kan dit voor ernstige ongewenste resultaten zorgen. Het onderzoek binnen ES&S [1] spitst zich dan ook toe op de ontwikkeling van ingebedde systemen met een focus op efficiënte implementaties voor digitale databeveiliging.

Beveiliging is een belangrijk aandachtspunt. In alle moderne communicatietechnologieën komt gegevensbeveiliging naar voren als een grote bezorgdheid. Ingebedde systemen vereisen speciaal ontworpen en geschreven besturingssystemen vanwege hun beperkt programma- en datageheugen. Er zijn een aantal beroemde besturingssystemen ontwikkeld, die draadloze sensornetwerken voor ogen hebben. Dat zijn Contiki Operating System (OS) [2], MANTIS OS [3] en TinyOS [4]. Contiki OS is het meest nieuwe van deze besturingssystemen. Tijdens de ontwikkelingsjaren van Contiki OS, is het door de bijdrage van een grote groep ontwikkelaars uitgegroeid tot het tweede belangrijkste besturingssysteem voor ingebedde systemen. Door het te ontwikkelen in de programmeertaal C zijn nu veel geavanceerde en *'light-weight'* onderdelen een deel van het besturingssysteem geworden, zoals onder andere de µIPv4 [5] en µIPv6 [6] netwerk stacks.

1.2 Beschrijving van het probleem

Hoewel de snelheid van ontwikkeling in Contiki OS zeer gunstig en up-to-date is, ontbreekt er nog steeds het gebruik van beveiligingsalgoritmes voor de veilige *'end-to-end'* communicatie tussen *sensor nodes*. Er is wel een beveiligingsarchitectuur in de vorm van ContikiSec [7] voorgesteld. ContikiSec is het resultaat van een diepgaande analyse over het gebruik van symmetrische sleutelcryptografie en het gevolg ervan op de beperkte middelen van de *sensor nodes*. Het *framework* van ContikiSec wordt gebruikt door Contiki OS om veilig te communiceren met gebruik van verschillende primitieve veiligheidseisen.

Voor een aantal databeveiligingsproblemen is het niet voldoende om enkel symmetrische sleutelcryptografie te gebruiken. Een aantal van deze problemen kunnen door het implementeren van *key management* en authenticatie opgelost worden. Bepaalde *features* kunnen evenwel alleen maar geïmplementeerd worden m.b.v. publieke sleutelcryptografie. De belangrijkste reden voor het

niet promoten van publieke sleutelcryptografie in Contiki tot nu toe, is het grote gebruik van programma- en datageheugen en het hoge energieverbruik. Vanuit ES&S kwam de vraag om publieke sleutelalgoritmen te implementeren in Contiki en ze te analyseren volgens geheugengebruik, energieverbruik en rekenkracht.

1.3 Doel

Het belangrijkste doel van deze thesis is om de haalbaarheid van een publieke sleutelcryptografie-implementatie in Contiki OS te exploreren. Dit omvat de integratie van publieke sleutelcryptografie in Contiki OS. De implementatie moet zorgen voor extra beveiligingsmogelijkheden in Contiki OS. Achteraf moet er ook een publieke sleutelcryptografie-implementatie in TinyOS geëxploreerd worden. Dit wordt gedaan met het oog op een vergelijkende studie.

De beoogde oplossing moet in staat zijn om details en suggesties, over de toevoeging van asymmetrische cryptografie te bieden aan de ontwikkelaars en beheerders. Het is noodzakelijk om de resulterende oplossingen te testen op een aantal sensornetwerken om bruikbaar te zijn in toekomstige implementaties.

1.4 Methode

Om deze masterproef te realiseren wordt het werk gepland in verschillende stappen. Het begint met een theoretische studie van Contiki OS, TinyOS en cryptografische algoritmes. Een onderzoek naar geschikte cryptografische bibliotheken, die bruikbaar zijn in de *open source* gemeenschap, moet gelijktijdig ook beginnen. Veel dingen zijn nieuw en in ontwikkeling op gebied van Contiki OS en TinyOS. Daardoor wordt de beschrijving van het probleem voortdurend herzien. De methodes voor de verificatie en beoordeling van de resultaten kunnen wel telkens opnieuw gebruikt worden.

Een alternatief voor het gebruik van *open source* bibliotheken, in het geval dat ze niet geschikt zijn voor implementatie op de *sensor nodes*, is het zelf ontwerpen van de code. Eenmaal men zeker is dat de code optimaal is, wordt ze gebruikt om verschillende testen uit te voeren voor detailanalyse. In de laatste fase van deze masterproef worden de resultaten geanalyseerd en vergeleken zodat aanbevelingen kunnen gedaan worden voor integratie in Contiki OS en TinyOS.

De implementatiefase bestaat uit de volgende stappen:

- het bestuderen van publieke sleutelcryptografie om Elliptic Curve Cryptography (ECC) te kunnen situeren;
- het bestuderen van basisbewerkingen van ECC;
- het begrijpen en analyseren van bestaande bibliotheken voor publieke sleutelcryptografie;
- het implementeren van de basisbewerkingen in ECC, zowel met als zonder besturingssysteem;
- het uitvoeren van *profiling* op de verschillende onderdelen van de code om een grondige analyse te kunnen uitvoeren op de geïmplementeerde onderdelen;
- het nagaan van de mogelijkheid tot uitbreiding van de implementatie.

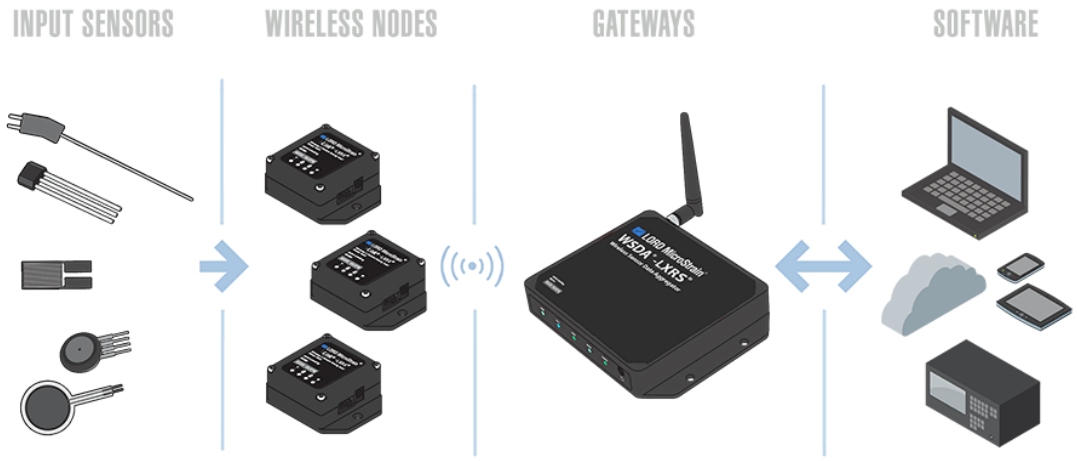
Hoofdstuk 2

Achtergrond

2.1 Draadloze sensornetwerken

Draadloze sensornetwerken bestaan uit kleine sensoren, die in de vakjargon ook *motes* genoemd worden. Deze sensoren meten fysische- of milieuomstandigheden in hun omgeving, zoals temperatuur, geluid, trillingen, bewegingen, procesgegevens en communiceren via draadloze verbindingen [8] met elkaar. Deze draadloze sensornetwerken worden de laatste jaren veel gebruikt en zijn erg populair in onderzoeken van ingebedde systemen in ons dagelijks leven. Draadloze sensornetwerken zijn gebruikt in toepassingen met betrekking tot het beheren, het volgen, of het organiseren, zoals leefomgeving monitoring, robotachtige machines, verkeersbeheer, object volgen en controle van nucleaire reactor.

Een draadloos sensornetwerk (WSN), bestaat gewoonlijk uit een *gateway*, routers, sensoren, actuatoren, *relays*, enz. Belangrijkste uit deze onderdelen van draadloos sensornetwerk is de *gateway*. De *gateway* heeft als functie om de sensoren te verbinden met een netwerk buiten het draadloos sensor netwerk. Het heeft ook bijkomende functies zoals het *routen* van pakketten binnen één draadloos sensornetwerk. De pakketten binnen één draadloos sensornetwerk kunnen *geroute* worden via verschillende *sensor nodes*, doormiddel van een routing-protocol om zo tot hun bestemming te bekomen [9]. In draadloos sensornetwerk gebruikte *sensor nodes* hebben beperkingen met betrekking tot hun *resources*. Een duidelijke overzicht hiervan wordt in Figuur 1 weergegeven.



Figuur 1: Draadloos sensornetwerk overzicht [10]

2.2 Veiligheid principes voor draadloze sensornetwerken

Er zijn verschillende fundamentele veiligheid primitieven voor het beveiligen van draadloze sensornetwerken. Deze zijn hieronder vermeld [7].

2.2.1 Vertrouwelijkheid van de data

Vertrouwelijkheid van de gegevens is in draadloze sensornetwerken zeer van belang. Dit omwille dat het enkel geautoriseerde *sensor nodes* toelaat voor het verkrijgen van de gegevens. Het belangrijkste doel van vertrouwelijkheid is het overdragen van de gegevens, van ene naar de andere *node*, dit met het oog op dat ze niet te begrijpen zijn vanuit elke tussenliggende *node* of ongeautoriseerde partijen. Voor het bereiken van confidentialiteit wordt er typisch gebruikgemaakt van symmetrische sleutelcryptografie, door de beperkte beschikbaarheid van *resources* in de *sensor nodes*. De afzender en de ontvanger gebruiken een vooraf gedefinieerde geheime sleutel of onderhandelen ze rechtstreeks via een gedeelde geheime sleutel. De data wordt vervolgens gecodeerd of gedecodeerd met gebruik van dergelijke sleutel.

2.2.2 Authenticatie van de data

In draadloze sensornetwerken is het zeer belangrijk dat de data afkomstig is van de juiste bron, zodat een aanvaller geen valse berichten of data kan toevoegen in het netwerk. Publieke sleutelcryptografie heeft de capaciteit om handtekening te maken, die enkel met gebruik van een aantal speciale informatie gegenereerd kunnen worden bij een zender. Om deze eigenschap te verwezenlijken worden de handtekeningen bij de zender bereidt en gebruikt. De ontvanger controleert de handtekening, om te kunnen bevestigen dat de data vanuit de geautoriseerde *node* is verzonden.

2.2.3 Integriteit van de data

De data kan altijd aangepast worden onderweg, tijdens zijn transmissie van een punt naar een andere punt, door onbevoegden. Echter zit er verschil in de identificatie van de aanpassing. Men kan dus achterhalen dat de data is aangepast. Dit kan men realiseren met gebruik van berichtintegriteit code (MIC), of door een controlesom toe te voegen aan elke pakket. MIC kan bericht wijziging opsporen die door toevallige transmissiefouten voorkomen en evenals kwaadaardige wijzigingen opsporen. Met een controlesom of *checksum* kan men ook toevallige transmissiefouten en opzettelijke fouten kunnen detecteren. Maar door de data op een juiste manier aan te passen gaat de *checksum* toch nog juist uitkomen (de aanpassing wordt dus niet gedetecteerd).

2.2.4 Freshness van de data

De verzonden berichten moeten niet herhaald worden aan de kant van de ontvanger, evenmin moet een *node* niet twee identieke berichten, in een bepaalde tijdperiode, ontvangen. Het belangrijkste doel van dit aspect is ervoor te zorgen dat de data recent en fris is. Een andere belangrijke doel is dat het niet gebruikt is door een aanvaller, die de symmetrische sleutel van de communicatiekanaal wilt kennen. Er zijn twee soorten van *freshness* technieken. Eerste techniek, dat zwakke *freshness* wordt

genoemd, wordt bereikt door het gebruik van gedeeltelijk bericht ordening, zonder delay informatie. Terwijl de tweede techniek, dat sterke *freshness* wordt genoemd, bereikt wordt door het gebruik van de volledige ordening en delay informatie.

2.2.5 Beschikbaarheid

Het gehele systeem of een enkele *sensor node* moet in draadloos sensornetwerk aanwezig zijn en in staat zijn om zijn diensten te verlenen, wanneer deze nodig zijn. De beschikbaarheid van een *sensor node* wordt grotendeels beïnvloedt, wanneer in een draadloos sensornetwerk aanwijzingen in gebied van veiligheid worden toegepast. Met andere woorden gezegd, wanneer veiligheidseigenschappen worden toegepast in een draadloos sensornetwerk, is het belangrijk dat de beschikbaarheid van het gehele netwerk bewaard wordt.

2.3 Bedreigingen van de veiligheid in draadloze sensornetwerken

Klassieke beveiligingstechnieken kunnen in draadloze sensornetwerken niet direct toegepast worden. Draadloze sensornetwerken groeien, dag na dag, zeer snel in vele configuraties, hierdoor moet men de beveiligingsmechanisme voor draadloze sensornetwerken ook constant bijwerken. Deze opereren random in de lucht waar ze onbewaakt zijn. Dit betekent dat de veiligheid in de vroege stadia, in ontwerpfase, voorzien moet worden [11]. Veiligheid in draadloze sensornetwerken kan worden onderverdeeld in twee delen, namelijk operationele- en informatieveiligheid. Operationele veiligheid betekent dat het netwerk geen geheime informatie mag openbaren en het moet eveneens de integriteit en authenticiteit van de berichten garanderen. Veiligheid is een belangrijke voorwaarde voor draadloze sensornetwerken, d.w.z. dat de gegevens en de *sensor node* beschermd moeten worden tegen aanvallen, zoals afluisteren, *tampering*, denial-of-service (DoS) aanval [12], enz. Bij het ontwerpen van een veiligheidsmodel voor draadloze sensornetwerken, moeten al deze types aanvallen worden beschouwd en de veiligheid moet op verschillende lagen worden gedefinieerd voor het garanderen van hoge mate van betrouwbaarheid.

2.4 Draadloos sensornetwerk aanvaller

Een aanvaller is een persoon of een andere entiteit die probeert schade aan het netwerk te brengen. Dit kan variëren van een *denial-of-service* aanval tot grotere mate van ongeautoriseerde toegang tot het draadloos sensornetwerk, wat resulteert in verlies van vele veiligheidseigenschappen gerelateerd tot belangrijke data [13].

Er zijn twee algemene soorten van aanvaller aanvallen op netwerken, namelijk passieve en actieve aanvaller techniek. Een passieve aanvaller techniek is beperkt tot luisteren en analyseren van de elk stukje informatie dat passeert. Dergelijke aanvaller techniek is gemakkelijker om het te realiseren (het volstaat met enkel een geschikte ontvanger te hebben) en het is moeilijk te detecteren. Bij deze type aanvaller techniek kan de aanvaller geen wijziging aanbrengen op uitgewisselde informatie. De bedoeling van de aanvaller is om confidentiële informatie of belangrijke *sensor nodes* in het netwerk te weten, door het analyseren van routeringstabel. Dit doet het om achteraf een actieve aanvaller te voorbereiden. Een actieve aanvaller techniek omvat de manipulatie van de data, terwijl het onder proces is in het netwerk. Dit omvat schrapping, verandering en aanvulling van de data. Dit resulteert

tot schending van de fundamentele veiligheidsprimitieven, zoals integriteit, vertrouwelijkheid en authenticatie van de data.

In draadloze sensornetwerken worden de aanvaller classificaties uitgebreid naar vier klassen [14]. Deze klassen zijn *mote-class* aanvaller, *laptop-class* aanvaller, *outsider* en *insider*. *Mote-class* aanvaller heeft toegang tot aantal *sensor nodes* waarvan de mogelijkheden vergelijkbaar zijn met die van de *sensor nodes* geïmplementeerd in het netwerk. Een *laptop-class* aanvaller heeft toegang tot meer krachtige apparaten, zoals laptops. Dit zal de aanvaller een voordeel geven ten opzichte van de draadloos sensornetwerk, omdat het voor meer ernstige aanvallen kan zorgen. Hiermee kan hij dus grotere set van technieken gebruiken om zijn doel te bereiken. *Outsider* heeft geen speciale toegang tot de draadloos sensornetwerk, zoals bij passieve aanvaller (luisteren). Bij *insider* is de aanvaller in sommige gevallen in staat om enkele interne *sensor nodes* in gevaar te brengen of *capture* een aantal interne *sensor nodes*. Hierdoor kan hij gemakkelijk een deel van het netwerk worden. Zodra het klaar is met zijn werk, kan hij het hele netwerk voor zijn eigen doeleinden laten functioneren.

2.5 Differente types van aanvallen

Er zijn in ieder *layer* van een netwerk verschillende types van aanvallen. Deze aanvallen worden hieronder vermeld volgens in welk *layer* ze voorkomen:

- fysieke *layer*;
    - *jamming*: door het gebruik van interferentie worden de radiofrequenties van de draadloos sensornetwerk verstoord. Dit kan zorgen voor het veranderen van belangrijke radio parameters zoals *collision rate*, slechte *frame rate* en Recieved Signal Strength Indicator (RSSI) niveau [15];
    - *tampering*: is het fysisch aanvallen van de *node* doormiddel van *side-channel attacks* of letterlijk openbreken of stukmaken van de sensor *node*. Dit wordt vooral gedaan door *capturing* van een *node* uit het draadloos sensornetwerkveld. De aanvaller kan alle gegevens van de *node* verzamelen en proberen om de nuttige gegevens te recupereren. Een vooruitstrevende aanvaller kan in het hele draadloos sensornetwerkveld recupereren, reprogrammeren en implementeren om zo het hele draadloos sensornetwerk aan te vallen [16].
- *data-link layer;*
    - *collision;* de pakketten kunnen worden verstoord door het veranderen van de transmissie octet. Dit resulteert in *checksum mismatch* of *back-off* in sommige MAC-protocollen. Bij deze soort aanval luistert de aanvaller naar de communicatiekanaal en probeert de verwachtte tijd van berichttransmissie te raden. Zodra de aanvaller de verwachtte tijd heeft, kan de aanvaller op hetzelfde moment wanneer een juiste bericht wordt gestuurd, bericht sturen. Hierdoor botsen deze twee berichten in de *wireless* omgeving en resulteert in een onjuiste bericht voor de ontvanger;
    - *exhaustion;* de batterijen van de *sensor nodes* kunnen uitgeteld worden als *collision* voorkomt in de draadloos sensornetwerk. Dit leidt tot verslechtering van beschikbaarheid op grote in de draadloos sensornetwerk.

- Netwerk *layer;*
    - selectieve *forwarding*; bepaalde schadelijke *sensor nodes* kunnen weigeren om berichten te sturen en laten deze vallen. Dit zorgt voor vertraging en bandbreedte verslechtering in draadloos sensornetwerk;
    - *sinkhole*; in dergelijke aanval kan de routingsinformatie gewijzigd worden in verschillende *sensor nodes*. Een aanvaller maakt verbinding met een *node* voor het doorsturen van verkeerde routingsinformatie. De *node* neemt de routingsinformatie aan en vermoedt dat deze informatie de efficiënte *routing* is naar de *sink node*. Maar in de praktijk zorgen meestal deze aanvallers voor het verwijderen of veranderen van de ontvangen berichten [17].
    - *sybil* aanval; een *node* creëert zijn eigen meervoudige identiteiten en biedt deze aan andere *sensor nodes* in het netwerk. Dit zal resulteren in verwijdering van alle originele buren uit de tabel van actieve *sensor nodes* in de routeringstabel. Indien de transmissiekwaliteit van de besmette *node* goed is, kan het ook zorgen voor verwijdering van de originele *sink* node uit de routeringstabel [18].
    - *hello flood*; een *laptop-class* aanvaller zendt berichten met krachtige signalen in het netwerk. Hierdoor denken de *sensor nodes* in het netwerk dat de aanvaller geen aanvaller is en sturen opstart pakketten die gebruikt worden in *routing* protocollen [17].
- transport *layer;*
    - *flooding*; door de *node* constant verbinding *establishment* verzoeken te sturen kan de aanvaller belangrijke *resources*, zoals batterij, van de *node* uitputten. Dit zal resulteren in DoS-aanval;
    - de-synchronisatie; Verzoek voor re-transmissie van gemiste *frames* kunnen worden gemaakt door herhaaldelijk dwingen van berichten in het netwerk, die sequentienummers op één of beide eindpunten dragen.

2.6 Geïntegreerde cryptografische bouwblokken in software

Tot enkele jaren geleden werd beveiliging niet beschouwd als een van de belangrijkste aspecten in draadloze sensornetwerken. Wanneer draadloze sensornetwerken, dag na dag, steeds meer voorkwamen in belangrijke toepassingen, is er meer aandacht besteed om ze te beveiligen op alle mogelijke manieren, zodat de gegevens niet verder kunnen worden benut. Door een belangrijke reden kunnen traditionele beveiligingstechnieken niet worden toegepast in draadloze sensornetwerken. De belangrijke reden is dat *resources*, zoals energie, ROM en RAM-geheugen, zeer beperkt zijn in geval bij draadloze sensornetwerken. Er zijn tot nu toe slechts enkele geïntegreerde cryptografische bouwblokken in *software* aanwezig. Enkele hiervan worden hieronder gepresenteerd.

2.6.1 TinySec

TinySec [19] is de eerste volledig functionele *link layer security* architectuur die gepresenteerd is voor draadloze sensornetwerken. Deze geïntegreerde cryptografische bouwblok in *software* is ook een deel van TinyOS geweest. Het grootste doel om *link layer security* architectuur te ontwikkelen was om meteen ongeautoriseerde pakketten te detecteren wanneer ze werden geïnjecteerd in een netwerk. Dit was om energie en bandbreedte te besparen. De beveiligingsaspecten die door het gebruik van TinySec worden bereikt zijn vertrouwelijkheid, berichtintegriteit en toegangscontrole.

2.6.2 ContikiSec

ContikiSec [7] biedt een aantal noodzakelijke beveiligingsprimitieven op de *link layer* van Contiki OS. Draadloze sensornetwerken ondersteunen veel verschillende soorten toepassingen. Hierdoor is ContikiSec ontworpen om flexibeler te zijn. Deze geïntegreerde cryptografische bouwblok in *software* biedt drie verschillende beveiligingsniveaus, namelijk vertrouwelijkheid, authenticatie en authenticatie met encryptie. Dit voorziet een selectiekeuze, van de drie beveiligingsniveaus, aan de programmeur die afhankelijk van de toepassing of behoeften van de draadloos sensornetwerk kan selecteren.

ContikiSec is ontworpen om energieverbruik en veiligheid te balanceren, terwijl die voldoen aan een kleine geheugen *footprint* [7]. ContikiSec gebruikt een symmetrische sleutel encryptie algoritmen. Volgens de ontwikkelaar van de ContikiSec, is AES de meest geschikte *block codering* voor de draadloze sensornetwerken, rekening houdend met het geheugengebruik (ROM en RAM), de tijd, het energieverbruik en het overwegen *trade-off* tussen veiligheid en *resource* verbruik.

In draadloze sensornetwerken is de typische verkeer *patern* de *many-to-one* communicatie *patern*. Om te voorkomen dat een aanvaller een routeringstabel injecteert, die voor verspilling van energie en bandbreedte zorgt, is de veiligheid in ContikiSec op de *link layer* geïmplementeerd.

2.7 Contiki OS

Het Contiki OS besturingssysteem is ontwikkeld door het Zweedse Instituut voor computerwetenschappen [20]. Het is een lichtgewicht, *open source*, zeer draagbare en *multitasking* besturingssysteem voor ingebedde systemen, die zeer geheugenefficiënt zijn. Het geheugengebruik van Contiki OS is ongeveer 40kB ROM- en 2kB RAM-geheugen [2].

De eerste versie van Contiki OS werd uitgebracht in 2004 en de laatste versie, 2.7, is uitgebracht in november 2013. Contiki OS is in programmeertaal C ontwikkeld en wordt momenteel gebruikt in veel microcontrollers, zoals, MSP430, HC 12, Z80 en AVR. Het grootste voordeel van Contiki OS is dat het dynamisch *loading* en *unloading* van applicaties en services *event* gestuurd is. Dit levert een verbetering van veel *resource* benutting in sensornetwerken en de *kernel*. Contiki OS voorziet ook *multithreading* en het is in een afzonderlijk bibliotheek geïmplementeerd, wat door een toepassing gebruikt kan worden wanneer het nodig is. Door de combinatie van *multithreading* en *event* gestuurd *kernel*, is Contiki OS een geschikte kandidaat voor als besturingssysteem voor draadloze sensornetwerken om publieke sleutelcryptografie te implementeren.

2.8 TinyOS

TinyOS [4] besturingssysteem is, net als Contiki OS, een andere *open source* OS voor ingebedde systemen. Het is ontwikkeld door de samenwerking tussen de Universiteit van California en Intel Research in 1999, en de eerste versie werd uitgebracht in 2000 met versie nummer 1.0. TinyOS nieuwste versie is 2.1.2, die werd uitgebracht in 2012 en telt met honderden nieuwe *features* sinds de eerste versie.

De belangrijkste voordelen zijn een geheel nieuw besturingssysteem die speciaal ontworpen is voor draadloos sensornetwerk, die beschikt over een *event* gestuurd *kernel* met *non-preemptive multitasking* en volledig *non-blocking.* In feite is elke taak in FIFO (First In First Out) orde uitgevoerd en programma's zijn opgebouwd uit softwarecomponenten die reeds bestaan of worden gedefinieerd door de programmeur.

TinyOS gebruikt statisch gelinkt code en beide toepassingen en bibliotheken zijn geschreven met behulp van een taal die zeer vergelijkbaar is met C, genaamd nesC [21].

2.9 Zolertia Z1

De Z1 [22] van Zolertia, is een *low-power* draadloze module die voldoet aan de IEEE 802.15.4 en Zigbee protocollen, die bestemd is om ontwikkelaars van draadloze sensornetwerken te helpen om hun eigen applicaties en prototypes te testen en te implementeren, met de beste *trade-off* tussen de tijd van de ontwikkeling en *hardware* flexibiliteit.

Het wordt geleverd met ondersteuning voor de meest gebruikte *open source* besturingssystemen, zoals Contiki OS en TinyOS.

De kern architectuur is gebaseerd op de MSP430 en CC2240 familie van microcontrollers en radio-ontvangers van Texas Instruments, die het overstemmend maakt met *motes* die gebaseerd zijn op dezelfde architectuur. De MSP430F26127 microcontroller wordt gebruikt in Zolertia Z1 waarin een 16bits RISC-architectuur wordt gebruikt op een kloksnelheid 16Mhz. Daarnaast beschikt de Zolertia Z1 over 92kB *flash* ROM en 8kB RAM-geheugen. Omvat ook de bekende CC2420 ontvanger, die werkt op 2,4GHz met een effectieve datasnelheid van 250kbps. Zolertia Z1 *hardware* aanbod garandeert de maximale efficiëntie en robuustheid met lage energiekosten. Deze heeft evenwel ingebouwde digitale sensoren, zoals digitale accelerometer (ADXL345) en een digitale temperatuursensor (TMP102), op zijn moederbord. Het omvat ook twee Phidgets *sensor ports* voor uitbreiden van aantal sensoren op de moederbord. Een andere eigenschap van deze microcontroller is dat het geen externe *hardware* nodig heeft om te kunnen programmeren. Dit komt omdat ingebouwde USB-functie zorgt voor snel ontwikkelen van draadloze sensornetwerktoepassingen en snelle integratie met meerdere systemen.

Figuur 2: Zolertia Z1

Hoofdstuk 3

Asymmetrische cryptografie

3.1 Cryptografie

Cryptografie is een wiskundig onderwerp dat wordt toegepast in de computerwetenschappen voor het garanderen van de databeveiliging. De term cryptografie is afgeleid van twee Griekse woorden, *kryptos* wat “verborgen” betekent en *grafo* wat “schrijven” betekent. Cryptografie wordt gebruikt voor verschillende doeleinden, zoals encryptie, berichtintegriteit, authenticatie, asymmetrische encryptie en digitale handtekeningen. Er zijn twee soorten cryptografische technieken, namelijk symmetrische en asymmetrische cryptografie. In deze thesis wordt enkel asymmetrische cryptografie in detail bekeken.

3.2 Asymmetrische cryptografie

De communicatie van gegevens kan beveiligd worden door middel van encryptie. Een gedeelde geheime sleutel wordt bij de zender en ontvanger gebruikt voor het vercijferen en ontcijferen van de gegevens. Een dergelijke encryptie noemt men symmetrische sleutelcryptografie. Het grootste probleem, bij gebruik van symmetrische sleutelcryptografie, is dat de sleutel afgesproken moet worden over een publiek netwerk. Er zijn ook scenario’s waarbij de sleutels op voorhand geïmplementeerd zijn op de *sensor nodes*. M.b.v. asymmetrische sleutelcryptografie kunnen symmetrische sleutels worden afgesproken over een publiek kanaal. Verder kunnen publieke sleutelencryptie en digitale handtekeningen gerealiseerd worden met asymmetrische sleutelcryptografie. Er worden dan twee sleutels gegenereerd, namelijk een private sleutel en een publieke sleutel. De private sleutel wordt geheim gehouden, terwijl de publieke sleutel voor iedereen toegankelijk is. Bij publieke sleutelencryptie wordt het bericht door de zender versleuteld met de publieke sleutel en gebruikt de ontvanger de private sleutel om het bericht te ontcijferen. Asymmetrische cryptografie wordt ook gebruikt voor digitale handtekeningen, waarbij de private sleutel aangewend wordt om het bericht te ondertekenen, terwijl de publieke sleutel wordt gebruikt om de handtekening te controleren. Het grootste nadeel van asymmetrische cryptografie is dat de uitvoeringstijd doorgaans veel trager is dan symmetrische cryptografie. Dit komt omdat er meer wiskundige bewerkingen moeten worden uitgevoerd in vergelijking met symmetrische cryptografie.

3.2.1 Asymmetrische cryptografische algoritmes

In de afgelopen jaren was het een grote uitdaging voor onderzoekers om op het gebied van draadloze sensornetwerken de rekenkundige complexiteit te reduceren en het geheugengebruik te verminderen van de traditionele asymmetrische cryptografische algoritmes, zoals DH, RSA en ECC. Onder al deze algoritmes wordt ECC beschouwd als het meest geschikt algoritme voor draadloze sensornetwerken. Dit komt omdat bij ECC het gebruik van geheugen en *resources* minimaal is [23].

In Tabel 1, wordt de aanbevolen minimale grootte van de sleutels voor elk behandeld algoritme volgens de National Institute of Standards and Technology (NIST) [24] weergegeven. De aanbevolen sleutellengte voor een veiligheid van 80bits (symmetrisch) is momenteel nog steeds aanvaardbaar. In de toekomst zal 80bits evenwel niet meer veilig zijn.

Tabel 1: Aanbevolen minimum grootte van de sleutels

Symmetrische sleutel sterkte (in bits)	DH	RSA	ECC
80	1024	1024	160
112	2048	2048	224
128	3072	3072	256

3.2.1.1 RSA

Rivest, Shamir en Adleman (RSA), is een asymmetrische sleutelcryptografie algoritme. Het is een van de eerste algoritmen (gepresenteerd in 1977) die geschikt zijn voor zowel digitale handtekeningen als publieke sleutelencryptie. Het wordt als zeer veilig beschouwd en wordt nog steeds op grote schaal gebruikt. De veiligheid van RSA steunt op het feit dat het moeilijk is om het product van twee grote priemgetallen te ontbinden in factoren

De encryptie gebeurt bij een *node*, bijvoorbeeld *node A*, die twee willekeurige priemgetallen p en q kiest, waarin $p \neq q$, en de modulus n als volgt berekent: $n = p \cdot q$. Bovendien berekent *node A* de waarde $\varphi = (p-1) \cdot (q-1)$ en kiest de publieke exponent e, zodanig dat e groter is dan 1 maar kleiner is dan φ. Daarna berekent de *node* een private exponent d, die uniek is: $d = e^{-1} \bmod \varphi$. De publieke sleutel (K_{pub}) van de *node A* bestaat uit {e, n} en de private sleutel (K_{pri}) bestaat uit $\{d, n\}$, waarbij de publieke sleutel over het netwerk wordt verzonden en de private sleutel wordt opgeslagen en geheim gehouden. Na het genereren van de sleutels worden de priemgetallen p en q nooit meer gebruikt.

Bovendien kan de *klaartekst m* worden vercijferd met de publieke sleutel van de *node {e, n}* en de *cijfertekst c* worden berekend door $c = m^e \bmod n$, zo kan de *cijfertekst* veilig via het netwerk verzonden worden. Aan de andere kant kan het vercijferde bericht alleen worden ontcijferd met behulp van de private sleutel die is opgeslagen door het berekenen van $m = c^d \bmod n$. Hieruit kan men concluderen dat de aanvaller de vercijferde berichten enkel kan ontcijferen indien hij in bezit is van de private sleutel van ontvangende *node* of hij erin slaagt de priemgetallen p en q te verkrijgen, door *factoring* van de modulus n. Hieruit kan men bepalen dat het RSA-algoritme steeds veiliger wordt indien de sleutelgrootte steeds groter wordt. Momenteel is de aanvaardbare lengte van de

sleutel tenminste 1024bits en de aanbevolen lengte is 2048bits. Ondanks het gebruik van een grote sleutellengte, rapporteert [25] dat het RSA-algoritme realiseerbaar is in draadloze sensornetwerken.

Het RSA-algoritme kan ook gebruikt worden voor verificatie van de communicerende *nodes*. Na het genereren van de private en publieke sleutel, kan elk *node* een bericht met zijn private sleutel encrypteren om een digitale handtekening aan te maken. De digitale handtekening wordt bij de data (het oorspronkelijke bericht - *klaartekst*) gevoegd en de gegevens worden over het netwerk verzonden. Anderzijds scheidt de ontvangende *node* de *cijfertekst* (handtekening) van de *klaartekst* (oorspronkelijk bericht) en gebruikt de publieke sleutel van de zender, die publiek bekend is, om de vercijferde tekst te ontcijferen en om te vergelijken met de ontvangen vercijferde tekst. Deze handeling is bekend als de verificatie van de handtekening. Op dit punt moet men opmerken dat het bericht wordt vercijferd met de private sleutel van de zender, eerst voor verificatie, en vervolgens vercijferd met de publieke sleutel van de ontvanger om veilig over te dragen, zoals eerder vermeld. De ontvangende *node* ontcijfert eerst de vercijferde data met zijn private sleutel en controleert vervolgens met de publieke sleutel van de zender de authenticiteit van de bericht, door de handtekening te vergelijken met de *klaartekst*.

Het bovenstaand digitale handtekenings-algoritme, dat RSA-sleutels gebruikt, verifieert niet alleen de authenticiteit van de zender, maar zorgt ook voor de integriteit van het verzonden bericht. De handtekening wordt gemaakt met de private sleutel van de opdrachtgever. Dus als een aanvaller het bericht probeert te wijzigen zal de handtekening niet meer geldig zijn. Dit omdat het na ontcijfering niet meer overeen zal komen met het originele bericht en daarmee zal de ontvangende *node* het bericht verwerpen.

In de praktijk wordt het digitale handtekenings-algoritme niet precies gebruikt zoals hierboven beschreven. Dit komt omdat de handtekening niet dezelfde lengte heeft als het oorspronkelijke bericht. Dit komt vooral omdat:

- in veel gevallen heeft het oorspronkelijke bericht zeer grote afmetingen;
- de encryptie en decryptie met publieke sleutelmethoden zijn trage processen en consumeren klokcycli van de processor en computervermogen. Het is noodzakelijk om het verbruik van te veel energie te vermijden in *low-power* ingebedde systemen;
- als een aanvaller toegang krijgt tot meerdere *klaarteksten* en hun bijhorende *cijferteksten*, kan hij door het gebruik van cryptanalyse de coderingssleutel raden.

Aldus, de meest gebruikelijke aanpak is om de digitale handtekening te creëren door het versleutelen van de *hash digest* van het bericht. De zender is het met de ontvanger eens over het gebruik van een bericht *digest* functie en vercijfert de *digest* van het bericht om zijn handtekening te maken. Vervolgens hecht de zender de handtekening aan de *klaartekst* en stuurt het resulterende bericht over het netwerk. De ontvanger gebruikt dezelfde bericht *digest* functie om de *digest* van de *klaartekst* te nemen, decodeert de handtekening met de publieke sleutel van de opdrachtgever en controleert of de twee *digests* passend zijn. Hieruit kan men afleiden dat de handtekening beduidend kleiner is dan de *klaartekst*.

Hieruit kan men concluderen dat het RSA-algoritme, ondanks de grote omvang van de sleutels, een toepasbare oplossing is voor sleuteluitwisseling en authenticatie, in draadloze sensornetwerken.

Maar, in vergelijking met andere publieke sleutel technieken (ECC) [23], is er iets meer rekentijd nodig, vooral voor de decryptie. RSA verbruikt ook zeer veel energie in tegenstelling tot ECC en is dus niet de voorkeursimplementatie in draadloze sensornetwerken.

3.2.1.2 Diffie-Hellman sleuteluitwisseling

Het Diffie-Hellman (DH) algoritme is gebaseerd op het discrete logaritme probleem (DLP) en maakt gebruik van publieke sleutelcryptografie, niet voor het distribueren van de gedeelde symmetrische sleutel, maar voor het creëren ervan. Dus de communicatie voor sleutel *establishment* wordt uitgevoerd door het sturen van *klaartekst* berichten over het netwerk.

Allereerst zijn de twee *nodes* akkoord over het gebruik van een groot priemgetal *p* en een generator *g*, die met zorg zijn gekozen, zodat voor elk positief getal *n* kleiner dan *p*, er een macht *k* is die voldoet aan $n = g^k \bmod p$. Nadien creëren ze allebei hun private sleutels, die hoe dan ook geheim blijven en creëren hun publieke sleutels die zijn afgeleid van de private sleutels. De private sleutel van *node A* is de willekeurige waarde *x* en de publieke sleutel wordt berekend als $g^x \bmod p$. Op dezelfde manier is de willekeurige waarde *y* de private sleutel van *node B* en de publieke sleutel is $g^y \bmod p$. Op dit punt, sturen ze allebei hun publieke sleutels via het netwerk (niet gecodeerd) om de bestemming aan de andere kant te bereiken. Na het ontvangen van elkaars publieke sleutel combineren ze het met hun private sleutel om een gemeenschappelijk gedeelde geheime sleutel te creëren. *Node A* berekent vervolgens de geheime sleutel door $(G^y)^x \bmod p$ en *node B* door $(G^x)^y \bmod p$, waarbij:

$$\left((g^y) \bmod p\right)^x \bmod p \;=\; (g^y)^x \bmod p \;=\; g^{xy} \bmod p \;=\; (g^x)^y \bmod p$$
$$= \left((g^x) \bmod p\right)^y \bmod p$$

Daarom, zonder elkaars private sleutel te weten slagen ze erin om een geheime waarde vast te stellen als hun gedeelde geheime symmetrische sleutel voor veilige communicatie. Deze sleutel kan worden gebruikt bij een symmetrische sleutelcryptografie algoritme, zoals AES. Het Diffie-Hellman-algoritme is gebaseerd op de DLP, waarbij het vanuit de waarde $(g^a \bmod b)$ zeer moeilijk is om de waarde *a* (private sleutel) te raden, zelfs indien de waarden *g* en *b* algemeen bekend zijn. Bijgevolg is het algoritme veilig zolang er geen snelle manier is om het discrete logaritme probleem op te lossen. Momenteel is de aanbevolen grootte van de modulus *p* 2048bits en het voorziene beveiligingsniveau is vergelijkbaar met het RSA-algoritme met een 2048bits private sleutel.

De *nodes* die deelnemen aan het sleutel *agreement* proces met het Diffie-Hellman-algoritme kunnen niet onderling worden authentiseert, als ze geen andere verificatiemethode gebruiken. Aldus, het DH-algoritme is kwetsbaar voor de *man-in-the-middle* aanval. Het Station-To-Station (STS) protocol [26] is een geverifieerd sleutel *agreement* protocol met sleutelverificatie, op basis van DH en digitale handtekeningen, waardoor het eerder genoemde probleem geëlimineerd wordt. Elke deelnemende *node* creëert niet enkel een private en publieke sleutel voor DH-uitwisseling (DH sleutels), maar heeft ook een sleutelpaar van asymmetrische sleutels (private en publieke sleutels) zoals in het RSA-algoritme, waarin de private sleutel wordt gebruikt om een handtekening te genereren en de publieke sleutel voor verificatie.

3.2.1.3 Elliptic Curve Cryptography

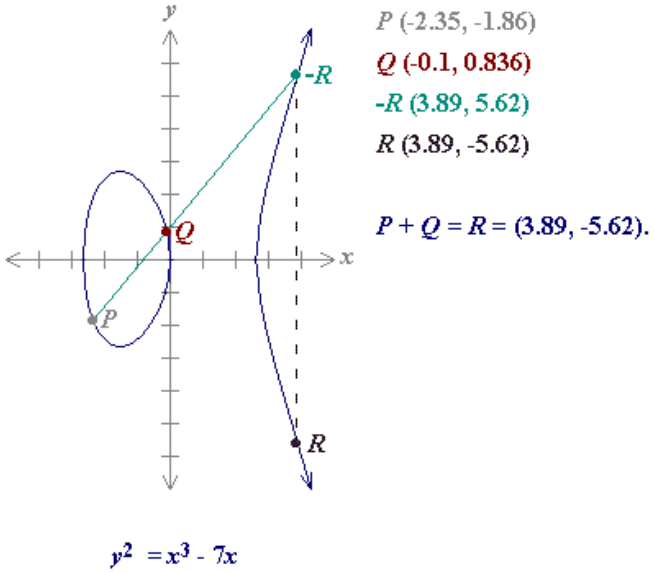
Elliptic Curve Cryptography (ECC), is een publieke sleutelcryptografie algoritme voor gebruik in *resource-constraint* omgevingen, zoals draadloze sensornetwerken. Rekening houdend met dat het voorziene geheugen in ingebedde systemen zeer beperkt is en met dat het energieverbruik moet worden geminimaliseerd, lijkt ECC een geschikte oplossing. De reden is dat de vereiste sleutelgrootte voor ECC zeer klein is in vergelijking met de grootte van de sleutels in andere publieke sleutelcryptografie algoritmes.

De meeste elliptische krommen (over de *prime field*) worden beschreven door de vereenvoudigde Weierstrass vergelijking [27]:

$$y^2 = x^3 + ax + b$$

Een grafische weergave van de voorgestelde Weierstrass vergelijking wordt getoond in Figuur 3. Het is belangrijk dat de verzameling van punten op een elliptische kromme een additieve groep is. Dit betekent dat het resultaat van de optelling van twee elementen op de kromme en de verdubbeling van een element op de kromme eveneens voldoet aan de vergelijking van de kromme. Door het combineren van puntoptellingen en puntverdubbelingen, zoals later wordt uitgelegd, wordt de vermenigvuldiging van een punt op de curve met een *scalar* bereikt.

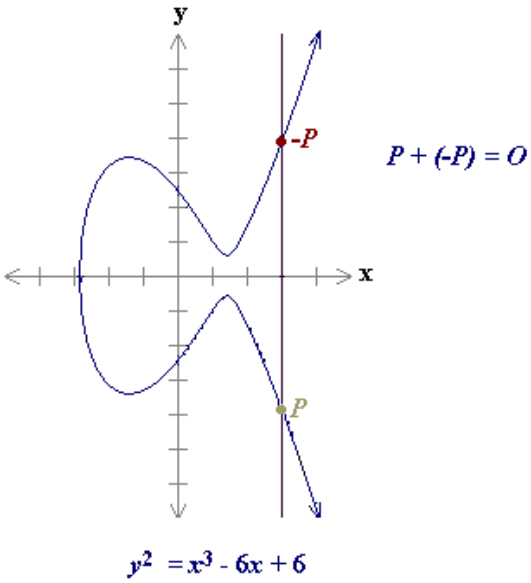
Aangezien de groep additief is, kunnen elke twee verschillende punten op een elliptische curve *E*, *P* *(x_P, y_P)* en *Q* *(x_Q, y_Q)*, waarbij het punt $P \neq -Q$, worden opgeteld *(P + Q = R)* om een nieuw punt R *(x_R, y_R)* te creëren, dat ook voldoet aan de krommevergelijking. De puntoptelling wordt grafisch voorgesteld in Figuur 3.



Figuur 3: Puntoptelling op een elliptische kromme, *P + Q = R* [28]

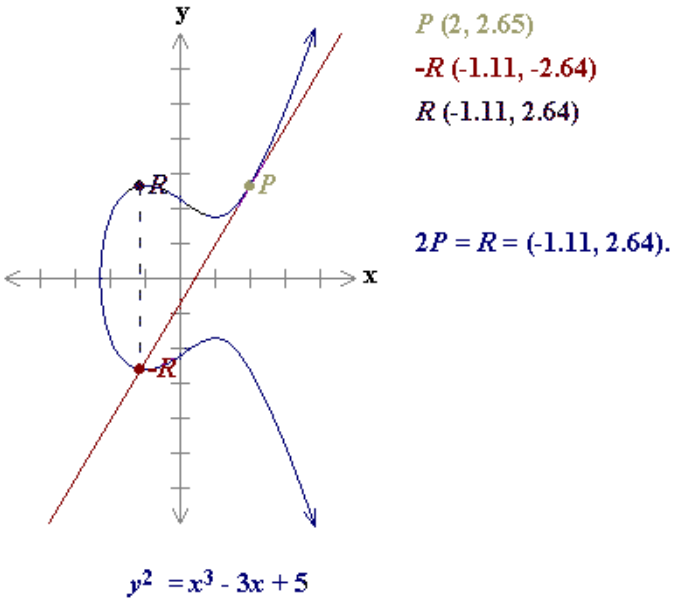
De optelling wordt bekomen door een de twee punten met elkaar te verbinden met een rechte. De rechte snijdt de curve *E* een derde keer in het punt *-R (xR, -yR)*, dat de reflectie van het punt *R (xR,*

yR) is. Bovendien, als het punt *P (xP, yP)* opgesteld wordt bij het punt *Q (xQ, yQ)*, waarbij *Q* gelijk is aan het tegengestelde punt *-P (xP, -yP)*, zoals *Q (xQ, yQ) = - P (xP, -yP)*, is het resultaat gelijk aan punt *O*. Hier snijdt de rechte tussen *P* en *–P* nooit een derde keer de elliptische kromme. Dit ziet men terug in Figuur 4.



Figuur 4: Puntoptelling op elliptische kromme, *P + (-P) = O* [28]

Ook moet men nagaan of *P (xP, yP) = Q (xQ, yQ)*. Het uitvoeren van een puntoptelling van deze punten is een verdubbeling van het punt *P* met als resultaat *R = P + Q = P + P = 2P*, zie Figuur 5.



Figuur 5: Puntverdubbeling op elliptische kromme, *R = 2P* [29]

Zoals boven vermeld, is men in staat om een punt *R* op de elliptische kromme te vinden, dat gelijk is aan het dubbele van het punt *P*, *R = 2P*. Terugkerend naar de geometrie aanpak, kan dit verkregen worden door de raaklijn te tekenen aan de elliptische curve in het punt *P*. Indien y_P niet gelijk is aan

nul *($y_P \neq 0$)*, snijdt de raaklijn de curve precies één keer in het punt *-R*, gelijk aan de spiegeling van het punt *R* over de *x-as*.

Door het herhaald uitvoeren van de bovengenoemde bewerkingen is men in staat om een punt *P* te vermenigvuldigen met een *scalar*, afhankelijk van de volgorde van de uitgevoerde bewerkingen. Bijvoorbeeld, als men het punt *P* wilt vermenigvuldigen met vier, berekent men de raaklijn aan het punt *P* om zo het punt *R = 2P* te vinden en dan verdubbelt men het punt *R* door het berekenen van zijn raaklijn (tangens). Het punt *R'*, waar de lijn de kromme snijdt, is gelijk aan *2R* en bijgevolg gelijk aan *4P*. Bovendien, door het toevoegen van punten *R'* en *P*, is het resulterende punt gelijk aan *5P* en als het punt *P* opnieuw wordt toegevoegd, is het resultaat *6P*.

In vergelijking met alle andere algoritmes, biedt ECC meer veiligheid en gelijkwaardige beveiliging met kleinere sleutelgroottes, wat resulteert in een snellere berekening, lager energieverbruik, minder geheugen en bandbreedte [30]. Hierdoor wordt ECC beschouwd als zeer nuttig voor zijn gebruik in mobiele apparaten.

3.2.1.4 ECDH

Het Elliptic Curve Diffie-Helmann (ECDH) algoritme is een publieke sleutelcryptografie algoritme voor sleuteluitwisseling tussen twee *nodes*. Het is gebaseerd op het Diffie-Hellman-algoritme maar het verschil met de klassieke DH is dat de private en de publieke sleutels worden gemaakt met elliptische kromme cryptografie.

In eerste instantie gaan de twee partijen akkoord over de gebruik van een specifieke elliptische kromme over een eindig veld, door het definiëren van de parameters *a* en *b* van de kromme, het priemgetal *p*, de orde van *n* en een puntgenerator *G*. Vervolgens zal elk deel zijn eigen private sleutel *d* kiezen. Dit wordt gekozen zodanig dat *d* een willekeurig gekozen getal is en kleiner is dan de orde *n* van de *prime* modulus. Daarna neemt *node A* het gegenereerde punt *P* van de kromme en voert een scalaire vermenigvuldiging uit van de punt, waarin zijn private sleutel wordt gebruikt als de *multiplier* *($Q_A = d_A P$)*. De vermenigvuldiging wordt bereikt door het uitvoeren van puntoptellingen en puntverdubbelingen. Het punt *Q* wordt gebruikt als publieke sleutel van de *node* en wordt zonder encryptie over het netwerk verzonden. Vanwege de ECDLP kan de private sleutel *d* binnen een redelijke tijd niet worden geëxtraheerd uit *Q*, dus hierdoor wordt het beschouwd als veilig. Dezelfde bewerking wordt uitgevoerd door *node B* om zijn eigen publieke sleutel Q_B te creëren, door zijn private sleutel d_B vermenigvuldigen met het punt *P*, $Q_B = d_B P$.

Wanneer *node B* de publieke sleutel Q_A van *node A* ontvangt, vermenigvuldigt *node B* de publieke sleutel van *node A* met zijn eigen private sleutel om de gedeelde geheime sleutel *s* te creëren, $s = d_B Q_A = d_B d_A P$. Deze vermenigvuldiging wordt uitgevoerd door het berekenen van puntoptellingen en puntverdubbelingen, soortgelijk aan de vermenigvuldiging van de publieke sleutelberekening. Evenzo creëert *node A* de gedeelde geheime sleutel door zijn private sleutel d_A met publieke sleutel Q_B van *node B* te vermenigvuldigen. Op dit punt houden ze dezelfde geheime sleutel voor symmetrische codering. Dit wordt als volgt gedaan:

$$d_B Q_A = d_B d_A P = d_A d_B P = d_A Q_B$$

De ECDH biedt geen authenticatie tussen de deelnemende *nodes*, zoals de klassieke Diffie-Hellman en RSA-sleuteluitwisselingsalgoritmes en dus is ook kwetsbaar voor de *man-in-the-middle* aanval. De aanvaller neemt dan de plaats in van één van de entiteiten. Daardoor kan de aanvaller berichten ontcijferen, de berichten wijzigen, opnieuw versleutelen met zijn eigen sleutels en zo doorsturen naar de ontvangers. Dit kan worden vermeden door het gebruik van het elliptische kromme digitale handtekening algoritme (ECDSA). ECDSA is vergelijkbaar met andere digitale handtekening algoritmes, maar het fundamentele verschil is dat de gebruikte sleutels zijn aangemaakt via een elliptische krommen punt vermenigvuldiging.

3.2.1.5 ECDSA

Elliptische kromme digitale handtekening (ECDSA) is een variant van Digital Signature Algorithm (DSA). Bij ECDSA is de sleutel slechts 160bits lang. Dit betekent dat ECDSA sneller is dan DSA.

Node A heeft het EC sleutelpaar (d_A, Q_A) en heeft ook de domeinparameters waarbij *a* en *b* de parameters van de elliptische kromme zijn, *q* is een priemgetal $(q = p)$ of een macht van twee $(q = 2^m)$. Bovendien is *G* het basispunt dat wordt gebruikt voor het creëren van de publieke sleutel en *n* is de orde van *G*. Om een handtekening op een bericht *m* te ondertekenen, genereert *node A* een willekeurig getal *k* en met scalaire vermenigvuldiging berekent het punt *R* met de coördinaten (x_R, y_R), zodanig dat $R = kG$. Vervolgens berekent het de inverse van k $(k^{-1} \bmod n)$ en voert het de SHA-1 hashing-functie op het bericht *m* om *H* te verkrijgen en berekent het $s = k^{-1} (H + d_A R) \bmod n$. De handtekening van *node A* bestaat uit (s, R) en wordt samen met het bericht *m* verstuurd naar *node B*.

Aan de andere kant, de *node B* krijgt een certificaat voor *node A* uit een Certificate Authority (CA) om in staat te zijn om handtekening van *node A* te verifiëren. Het certificaat is een origineel exemplaar van *node A* domeinparameters en zijn publieke sleutel. *Node B* berekent na ontvangst van het bericht *m* en handtekening van *node A* (s, R), de *hashing H* van het bericht en berekent de inverse van s $(w = s^{-1} \bmod n)$. Vervolgens creëert *node B* het scalaire product u_1 $(u_1 = Hw)$ en vermenigvuldigt het met het basispunt *G* om het punt P_1 op de curve $(P_1 = u_1 G)$ te krijgen. Daarna creëert *node B het product* u_2 $(u_2 = XRW)$ en door dat te vermenigvuldigen met de publieke sleutel Q_A van *node A*, verkrijgt *node B* een tweede punt op de elliptische curve P_2, $P_2 = u_2 Q_A$. Tenslotte voert *node B* een puntoptelling van de punten P_1 en P_2 uit en controleert of het verkregen punt R' dezelfde coördinaten als het ontvangen punt *R* heeft. Indien deze overeenkomen, is de ondertekening van *node A* geldig.

3.2.1.6 Samenvatting

Het klassieke Diffie-Hellman-algoritme is niet geschikt voor het gebruik in de draadloze sensornetwerken als het een grote sleutel vereist en het biedt geen verificatie. Om hierbij verificatie te bereiken, dient men in een later stadium een digitale handtekeningalgoritme in het systeem te importeren. Maar toch is de vereiste berekening en communicatie *overhead* zeer hoog, hetgeen voor een hoog energieverbruik zorgt.

Zowel RSA en ECC (sleuteluitwisseling met ECDH en ECDSA voor authenticatie) zijn geschikt op draadloze sensornetwerken. Het gebruik van ECC wordt aanbevolen in plaats van RSA vanwege de kleinere grootte van de vereiste sleutels, de rekentijd en het beperkte energieverbruik. Bovendien

verbruikt het ECC algoritme minder geheugen tijdens het berekeningsproces en geeft minder communicatie overhead ten opzichte van het RSA-algoritme.

Door al het bovenstaande heeft men voor deze masterproef gekozen voor de uitvoering van het ECDSA sleutel uitwisselingsalgoritme.

Hoofdstuk 4

Bibliotheken details

4.1 Selectienorm van de bibliotheken

Na het uitvoeren van de literatuurstudie was het de bedoeling om cryptografische bibliotheken te zoeken die gebruikt kunnen worden in draadloze sensornetwerken. Er zijn verschillende factoren die bepalen tot welk bibliotheek gekozen mag worden. Vanaf de start van het project is er veel belang gehecht aan het vinden van een *open source* bibliotheek, die eenvoudig implementeerbaar is en over een kleine geheugen *footprint* beschikt. Een cryptografische bibliotheek die ontworpen is voor het gebruik ervan in ingebedde systemen, zal meer geschikt zijn voor selectie. Een andere belangrijke selectienorm is de programmeertaal, die liefst in de taal C geschreven moet zijn. Deze norm moet op het oog gehouden worden omdat Contiki OS op C werkt.

Zoals hierboven is vermeld, is Contiki OS volledig geschreven in de C taal en vandaar kan enkel een cryptografische bibliotheek die in de C taal is geschreven gebruikt worden. Na het onderzoek is er aan het licht gekomen dat er meer cryptografische bibliotheken in C++ en Java geschreven zijn ten opzichte van in de C taal. Er zijn enkel drie *open source* bibliotheken specifiek voor ingebedde systemen gevonden. Bijna alle speciale cryptografische bibliotheken zijn niet *open source* en niet zomaar beschikbaar.

Om Contiki OS te vergelijken met TinyOS moest men een bibliotheek zoeken die geschikt was voor het gebruik onder TinyOS. TinyOS is een besturingssysteem dat onder nesC taal is geschreven. Na onderzoek is er gebleken dat onder deze taal er bijna geen cryptografische bibliotheken zijn die geschikt zijn voor het bieden van ECC.

Symmetrische cryptografie is in tegenstelling tot asymmetrische cryptografie eenvoudiger implementeerbaar. Het vergt geen enkel complexe bewerkingen voor het uitvoeren van zijn code. Maar bij asymmetrische cryptografie is dit niet het geval. Het vraagt veel grote wiskundige bewerkingen, die soms niet uitgevoerd kunnen worden door wiskundige functies van de gebruikte compilers. Vanwege deze hebben asymmetrische cryptografie nood aan wiskundige bibliotheken die nodig zijn voor het uitvoeren van de bewerkingen.

4.2 Cryptografische bibliotheken

4.2.1 Relic

Relic [31] is cryptografische *toolkit* die speciaal geschreven is voor het gebruik in draadloze sensornetwerken. Het is een zeer aanpasbare bibliotheek en voorziet in een groot aantal moderne cryptografische functies. Informatie over deze functies vindt men terug in [31]. Naast deze functies biedt Relic ook belangrijke cryptografische algoritmes. Belangrijke uit deze algoritmes zijn RSA, ECDSA, ECDH, ECMQV en ECIES. Er zijn geen documentatie *resources* beschikbaar van deze bibliotheek. Dit omwille van het feit dat het nieuw is en continue in ontwikkeling is. Dit zorgde voor moeilijkheden om het te begrijpen bij het begin, maar zodra een gebruiker alle details begrijpt, is het zeer makkelijk om ermee te werken.

4.2.2 ContikiECC

In deze thesis is ook gebruikgemaakt van ContikiECC [32] om de fundamentele elliptische kromme handelingen uit te voeren. De ContikiECC overzet de functies van de TinyECC bibliotheek, die in alinea hieronder gesproken wordt, naar het Contiki OS. De TinyECC bibliotheek is een implementatie van de ECC bewerkingen, die oorspronkelijk ontwikkeld is voor gebruik in TinyOS.

ContikiECC implementeert functies om zeer grote getallen als *multiple* van 8bits of 16bits woorden te hanteren en verschaft de basis numerieke bewerkingen op 8bits en 16bits microprocessor. Deze functies kunnen niet alleen worden gebruikt voor de elliptische kromme cryptografie, maar ook voor andere toepassing die bewerkingen op grote getallen nodig hebben. Bovendien biedt ContikiECC een aantal elliptische krommen van grootte 128, 160 en 192bits (specificeert de parameters *a*, *b*) als het basispunt van elke curve. De ongebruikte elliptische krommen worden niet opgesteld om onnodig geheugengebruik te vermijden. De programmeur kan eenvoudig kiezen om een andere kromme samen te stellen door het te definiëren in een *makefile*. De basis numerieke bewerkingen voor grote getallen worden gebruikt door de functies van ECC, om de fundamentele ECC operaties te implementeren. Deze operaties zijn het puntoptelling, puntverdubbeling en vermenigvuldiging met een scalair *multiplier*. ContikiECC maakt gebruik van de Sliding window methode in de implementaties van de basisfuncties, die meer geoptimaliseerde eigenschappen bieden in vergelijking met andere methoden zoals het later wordt aangegeven in hoofdstuk 6.

In aanvulling op de basis elliptische kromme operaties, zoals ECDH en ECDSA-algoritme operaties zijn eveneens in ContikiECC beschikbaar voor het gebruik ervan in draadloze sensornetwerken.

4.2.3 TinyECC

TinyECC [33] is een elliptische kromme cryptografie gebaseerde publieke sleutelcryptografie softwarepakket, dat eenvoudig kan worden geconfigureerd en geïntegreerd in de toepassingen uit draadloze sensornetwerken. Het is specifiek ontworpen voor gebruik in TinyOs. TinyECC biedt speciale optimalisaties die in- of uitgeschakeld kunnen worden door de programmeur, naargelang zijn behoeftes. Door het configureren van verschillende optimalisaties kan uitvoeringstijd en *resource* verbruik worden gecontroleerd. Bovendien biedt TinyECC ook elliptische kromme parameters die aanbevolen zijn door SECG (Stands for Efficient Cryptography Group), zoals secp160k1, secp160r1 en secp160r2, zoals gedefinieerd in [34]. Naast al deze biedingen, zorgt TinyECC ondersteuning voor verschillende ECC systemen, zoals digitale handtekening algoritme (ECDSA), sleutel uitwisseling protocol (ECDH) en elliptische curve integreerde encryptie systeem (ECIES).

4.2.4 Samenvatting

Na onderzoek en gebruik van deze bibliotheken is er gebleken dat deze alle drie bibliotheken beschikken over ECDSA-algoritme, die voor beveiliging zorgt in draadloze sensornetwerken. Echter zijn niet alle optimalisaties beschikbaar in de bibliotheken. In onderstaande Tabel 2, wordt er gepresenteerd welke opties beschikbaar zijn in de bibliotheken. Er zijn een aantal optimalisaties die niet aanwezig zijn of niet verschaft in de bibliotheek. Dit kan voor gevolgen zorgen aan de resultaten, die later besproken worden.

Tabel 2: Beschikbare ECC optimalisaties in elke bibliotheek

Optimalisaties	Relic	ContikiECC	TinyECC
Simpele vermenigvuldiging	Beschikbaar	Beschikbaar	Beschikbaar
Sliding window vermenigvuldiging	Niet beschikbaar	Beschikbaar	Beschikbaar
Montgomery reductie	Beschikbaar	Niet beschikbaar	Niet beschikbaar
Barrett reductie	Niet beschikbaar	Niet beschikbaar	Beschikbaar
Affiene coördinatensysteem	Beschikbaar	Niet beschikbaar	Beschikbaar
Projectieve coördinatensysteem	Beschikbaar	Beschikbaar	Beschikbaar
Simpele verificatie	Beschikbaar	Beschikbaar	Beschikbaar
Verificatie van Shamir	Beschikbaar	Beschikbaar	Beschikbaar
SECG160 curve	Beschikbaar	Beschikbaar	Beschikbaar
SECG192 curve	beschikbaar	Beschikbaar	Beschikbaar

Hoofdstuk 5

Meetmethoden

5.1 Uitvoeringstijd

In de hedendaagse tijd worden er twee technieken gebruikt om uitvoeringstijd van *software* te meten, die het nodig heeft om verschillende bewerkingen uit te voeren. De meest nauwkeurige methode is om een oscilloscoop met een zeer hoge precisie te gebruiken om uitgangspin van de data te controleren. Naast zijn voordeel heeft deze ook nadelen. Deze lijkt complexer en vele externe factoren, zoals marge van afwijking van meetinstrument, moeten ook worden beschouwd. Dus voor het gebruik van deze methode verliest men veel inspanning om het uit te voeren. De tweede methode is om één van de *real-time* timer, die aanwezig zijn in Contiki OS en TinyOS, te gebruiken. Door gebruik van deze *real-time* timer kan men hogere nauwkeurigheid, dan *event* timer, in uitvoeringstijd bereiken.

5.1.1 Meten van uitvoeringstijd onder Contiki OS

Contiki OS ondersteunt twee soorten van timers, namelijk *real-time* timer en *event* timer. Een *real-time* timer stuurt geen *event* wanneer het bezig is, maar roept in plaats hiervan een functie op. Dit zorgt voor een betere controle over een verrichte functie in een programma, dan de *event* timer [35]. Hiermee wordt er duidelijk bedoelt dat *real-time* timer niet voor een vertraging zorgt, terwijl een *event* timer dat wel doet.

In Contiki OS is *real-time* timer afhankelijk van de *hardware*. Dit biedt een keuze tussen één van de twee klokken die aanwezig zijn in de *sensor node* te selecteren. Eerste klok is de main clock (MCLK). Deze biedt een maximum resolutie van 0,4069µsec als de processorcyclus op 2,4576MHz wordt uitgevoerd. Dit is zeer nauwkeurig maar de klok was niet beschikbaar voor de timer aan te sturen. De tweede klokoptie is het gebruik van auxillary clock (ACLK), die werkt door gebruik van een kristaloscillator met een frequentiesnelheid van 32768Hz. Deze configuratie is aangewezen en geselecteerd voor alle metingen in het project.

Onder Contiki OS kan gebruikmaken van rtimer voor het gebruik als *real-time* timer. De rtimer bibliotheek onder Contiki OS biedt *scheduling* en uitvoering van *real-time* operaties. Om hoge klokresolutie te bereiken gebruikt deze rtimer bibliotheek zijn eigen klok module voor *scheduling*. De functie *RTIMER_NOW()* is om huidige systeem in *ticks* te verkrijgen en door gebruik van *RTIMER_SECOND()* kan men bekijken hoeveel *ticks* een seconde bedraagt.

In Tabel 3, hieronder, vindt men voorbeeld code uit Relic bibliotheek gebaseerde ECDSA-programma terug. In deze stuk code wordt er uitvoeringstijd van een bewerking gemeten. Hier wordt er gebruikgemaakt van ECDSA-generatie bewerking, met *d* als private sleutel en *q* als publieke sleutel.

Tabel 3: Uitvoeringstijd meten onder Contiki OS

```
#Execution time
t1=RTIMER_NOW();     #Get the current time in ticks
cp_ecdsa_gen(d, q);  #Generation of ECDSA
t2=RTIMER_NOW();     #Get the current time in ticks
```

Dus om prestatie van de snelheid te meten moet men de initiële *ticks* voor en na de uitgevoerde bewerking opslaan. Verschil van deze twee waarden deelt men door *RTIMER_SECOND()*. De resultaat van deze bewerking geeft de prestatie van de snelheid in seconde terug.

5.1.2 Meten van uitvoeringstijd onder TinyOS

Zoals bij Contiki OS, is er bij TinyOS ook gebruikgemaakt *real-time* timer die gebaseerd is op 32768Hz kristaloscillator. Deze geeft dus ook een precisie tot 32768Hz. Hier wordt er gebruikgemaakt van *LocalTime* functie. Deze is een 32bits teller, die zorgt voor dat er geen *overflows* optreden.

In Tabel 4, hieronder, vindt men voorbeeld code uit TinyECC bibliotheek gebaseerde ECDSA - programma terug. Hierin ziet men *ECDSA.init* als ECDSA-initialisatie bewerking terug, met *PublicKey* als publieke sleutel.

Tabel 4: Uitvoeringstijd meten onder TinyOS

```
#Execution time
time_a = call LocalTime.get();   #Get the current time in ticks
call ECDSA.init(&PublicKey);     #Initialization of ECDSA
time_b = call LocalTime.get();   #Get the current time in ticks
```

Zoals bij Contiki OS wordt bij TinyOS ook verschil genomen van de *ticks* voor en na het gewenste bewerking, waarvan men uitvoeringstijd wilt weten. Deze verschil wordt dan gedeeld door 32768 om het resultaat in seconden te bekomen.

5.2 ROM

ROM is een zeer beperkte *resource* in draadloze sensornetwerken en hierdoor is het ook een belangrijke parameter die voldaan moet worden voor elke toepassing toepasbaar te maken op een *sensor node*. Indien men dit niet op het oog houdt kan men de toepassing niet *uploaden* op de *sensor node*. De meeste van de beschikbare cryptografische bibliotheken hebben met deze beperking geen rekening gehouden. Daarnaast zijn er weinig zomaar beschikbare bibliotheken gepresenteerd die ontworpen zijn voor het gebruik in ingebedde systemen. Indien de bibliotheek voor gebruik gekozen is, dan kan het niet als een heel pakket in Contiki OS of in TinyOS worden overgezet. Hiervoor is het noodzakelijk dat men het pakket eerst reduceert en een minimale toepassing creëert die past op een *sensor node*, met een geheugen van 15 tot 30bytes. Publieke sleutelcryptografie gebruikt zeer grote getallen voor berekeningen van de bewerkingen. Soortgelijke berekeningen kunnen niet worden uitgevoerd met eenvoudige wiskundige bewerkingen. Vanwege deze reden bevatten of ondersteunen alle publieke sleutel cryptografische bibliotheken enkele groot getallen bibliotheken. Dit maakt dus een extra belasting van de geheugen. Maar dit is een belangrijke vereiste en kan niet worden overgeslagen.

5.2.1 Meten van ROM onder Contiki OS

Om de ROM-geheugen vereiste van de code te bepalen is msp430-size hulpprogramma [36] gebruikt. In Tabel 5, hieronder, vindt men een voorbeeld van *console command* die men moet gebruiken om het uit te voeren. Hierin is *msp430-size command* om de hulpprogramma te roepen, met *–A* als formaat ervoor en met *ecdsa_program.z1* als object file, waarvan de ROM-geheugen gemeten moet worden.

Tabel 5: Gebruik van msp430-size om onder Contiki OS ROM te meten

$ msp430-size -A ecdsa_program.z1

Deze *console command* geeft een reeks van gegevens in een tabel terug. Uit deze tabel is de som van *.text* en *.data*, het gemeten ROM-geheugen van de gecompileerde bestand. Dit ziet men in Tabel 6, hieronder terug.

Tabel 6: Output van msp430-size hulpprogramma

ecdsa_program.z1 :		
section	size	addr
.text	41536	12544
.rodata	1566	54480
.data	180	4352
.bss	3720	4532
.noinit	2	8252
.vectors	64	65472
.debug_aranges	2316	0
.debug_info	41924	0
.debug_abbrev	15004	0
.debug_line	27435	0
.debug_frame	7076	0
.debug_str	8563	0
.debug_loc	25856	0
.debug_ranges	1288	0
Total	176530	

Eerst wordt er met deze hulpprogramma grootte van de code, die gebruikt wordt door Contiki OS, gemeten. Wanneer dit gebeurd is, wordt publieke sleutelcryptografie geïntegreerd in Contiki OS en wordt er dan weer het gecompileerde geheugen gemeten. Hierna neemt men het verschil van de twee waarden, wat resultaat als ROM-geheugen geeft. Met deze aanpak wordt de vereiste ROM-geheugen bij cryptografische functies gemeten. Deze aanpak kan wiskundig als volgt geschreven worden:

$$Vereiste\ ROM = publieke\ sleutelcryptografie\ bevattende\ gecompileerde\ geheugen \\ - gecompileerde\ geheugen\ zonder\ publieke\ sleutelcryptografie$$

5.2.2 Meten van ROM onder TinyOS

Onder TinyOS heeft men geen extra hulpprogramma nodig om gemeten ROM-geheugen te verkrijgen. Wanneer men onder TinyOS in *console* de code *build*, geeft de *console* het gebruikte ROM-geheugen terug. Dit ziet men terug in Tabel 7, hieronder.

Tabel 7: ROM grootte onder TinyOS

```
mkdir -p build/z1
    compiling ECDSA to a z1 binary
ncc -o build/z1/main.exe  -Os -DTOSH_MAX_TASKS_LOG2=8 -fnesc-separator=__ -Wall -Wshadow
-Wnesc-all -target=z1 -fnesc-cfile=build/z1/app.c -board= -DTOSH_DATA_LENGTH=102 -
DDEFINED_TOS_AM_GROUP=0x22 -DSECP160R1 -DBARRETT_REDUCTION   -DCURVE_OPT   -
DPROJECTIVE   -DSHAMIR_TRICK   -DIDENT_APPNAME=\"ECDSA\" -DIDENT_USERNAME=\"root\" -
DIDENT_HOSTNAME=\"ubuntu\" -DIDENT_USERHASH=0xa3473ba6L -
DIDENT_TIMESTAMP=0x568a497dL -DIDENT_UIDHASH=0x9342dd20L  ECDSA.nc -lm
    compiled ECDSA to build/z1/main.exe
         17046 bytes in ROM
          1602 bytes in RAM
```

In deze *output* van de *console* vindt men langs ROM-geheugen, ook RAM-geheugen terug.

5.3 RAM

RAM is de meest belangrijkste beperking in *sensor nodes*, die meeste attentie nodig heeft. Draadloze sensornetwerken zijn ontworpen in slechts een enkele kilobytes RAM-geheugen. Zolertia Z1 is bijvoorbeeld uitgerust met slechts 8kb RAM-geheugen. De *stack* heeft constant variabele grootte tijdens het procesuitvoering. Hierdoor is het zeer moeilijk om RAM-geheugen te schatten.

Onder TinyOS heeft men geen extra hulpprogramma nodig om dit te schatten. Zoals in punt 5.2.2, is weergegeven vindt men de RAM-geheugen terug wanneer men de code gaat compileren.

5.3.1 Schatten van RAM onder Contiki OS

Onder Contiki OS is er eerst gebruikgemaakt van msp430-size hulpprogramma om RAM-geheugen te schatten. Uit onderzoek is er gebleken dat sommatie van *.bss* en *.data* uit de Tabel 6, in punt 5.2.1, de vereiste RAM-geheugen is om het te kunnen *uploaden* op de *sensor node*. De waarden met deze hulpprogramma, voor RAM-geheugen, waren niet zo evident. Bij de tweede aanpak heeft men msp430-ram-usage gebruikt, dat men terug kan vinden in [37]. In Tabel 8, hieronder, vindt men de *console commands* terug die men in de *console* moet gebruiken, voor het verkrijgen van de RAM-geheugen.

Tabel 8: Gebruik van msp430-ram-usage.py

```
phyton  msp430-ram-usage.py ecdsa_program.z1
```

Uit de resultaten bleek er dat tweede aanpak iets minder waarde gaf voor RAM-geheugen. Hierdoor is de voorkeur gegaan naar het gebruik van de tweede aanpak, met gebruik van msp430-ram-usage *script*.

5.4 Energiemetingen

Draadloze sensornetwerken zijn uitgerust met een beperkte voedingsbron. Dit zorgt ervoor dat energie een zeer schaarse middel is in een draadloos sensornetwerk. Normale toepassingen verbruiken energiegebruik in het gebied van µjoules. Maar wanneer in de publieke sleutelcryptografie complexere en langere wiskundige vergelijkingen opgelost moeten worden, stijgt het verbruik van µjoules tot millijoules [33]. Men kan deze energieschattingen bepalen m.b.v. meettoestellen [38], maar zo'n dergelijk *hardware* gebaseerde meettoestel is niet nodig voor energiemetingen op Contiki OS en TinyOS.

5.4.1 Energiemetingen onder Contiki OS

Contiki OS biedt een ingebouwde *tool*, Energest, die kan gebruikt worden voor energieschattingen te maken. Energest is een *software* gebaseerde online energieschattingsmechanisme die het energieverbruik van een *sensor node* kan schatten [39].

Het is een mechanisme dat binnen Contiki OS is aangebracht om bij de *sensor node* gebruikt te worden om energieschattingen van alle componenten te verkrijgen. De meest belangrijke componenten zijn radiozender en radio-ontvanger, *low-power* modi en CPU. Het belangrijkste aan dit proces is om timers te gebruiken. Wanneer een onderdeel actief is, begint een teller het geschatte energieverbruik te meten. Wanneer de component wordt uitgeschakeld, dan wordt de huidige waarde aan de entrytabel van de component toegevoegd. Dit ziet men terug in Tabel 9, hieronder. Hier gaat men aantal *ticks* die ContikiECC gebaseerde *ecdsa_init*, met *pbkey_alice* als publieke sleutel, bewerking nodig heeft tellen.

Tabel 9: Gebruik van Energest tool

```
ENERGEST_ON ( ENERGEST_TYPE_CPU );
last_init.cpu = energest_type_time ( ENERGEST_TYPE_CPU );
ENERGEST_OFF ( ENERGEST_TYPE_CPU );
ENERGEST_ON ( ENERGEST_TYPE_CPU );
ecdsa_init ( &pbkey_alice );
diff_init.cpu = energest_type_time ( ENERGEST_TYPE_CPU ) - last_init.cpu;
```

Het verschil tussen de twee waarden is een resulterende dat vervolgens wordt vermenigvuldigd met de stroom van de componenten om het energieverbruik dat verbruikt wordt tijdens het uitvoeren van een bewerking, te bekomen. Deze procedure heet de-normalisatie procedure. Om de waarden van de tabel de-normaliseren dienen de specifieke kenmerken van het gebruikte apparatuur bekend te zijn. Onderstaande gegevens zijn de parameters die gebruikt worden voor de-normalisatie procedure.

Tabel 10: Gegevens van MSP430F2617 en CC2420

Component	Stroomverbruik (in mA)
CPU	4,5
LPM	0,000426
Radio TX	20,5
Radio RX	19,9

Zodra entrytabel van ieder component verkregen wordt, dan worden deze componenten ge-de-normaliseert en hierna worden deze vermenigvuldigd met de spanning van *sensor node* om energie te schatten. Tenslotte wordt het verdeeld door *RTIMER_SECOND* component van Contiki OS om energieschattingen voor tijd, die is gebruikt door het proces, te bieden. Volgende formule, wat men terugvindt in [39], kan gebruikt worden om energieverbruik te berekenen:

$$Energieverbruik = \frac{\big((4{,}5mA \times \Delta CPU) + (0{,}000426mA \times \Delta LPM) + (20{,}5mA \times \Delta TX) + (19{,}9mA \times \Delta RX)\big) \times 3V}{RTIMER\_SECOND}$$

VCC heeft een waarde van 3V als *sensor node* geconnecteerd is aan een USB-interface en als het op volvermogenstand werkt. *RTIMER_SECOND* is onder Contiki OS gelijk aan 32768, wat waarde is voor *ticks* per seconde. Omdat men in dit project enkel op één *sensor node* bekeek naar de resultaten - dus geen transfer tussen *sensor nodes* - had men geen energieverbruik voor radiozender, radio-ontvanger en LPM. Dus buiten CPU waren al de andere componenten gelijk aan nul voor energieverbruik. Dit kon men ook terugzien aan de bekomen resultaten.

5.4.2 Energiemetingen onder TinyOS

Onder TinyOS is er geen gebruikgemaakt van een hulpprogramma. Hier heeft men de uitvoeringstijden voor bewerkingen gebruikt voor de energiemetingen te bepalen. Deze uitvoeringstijden zijn vermenigvuldigd met de stroom van de componenten, wat de-normalisatie proces wordt genoemd. Achteraf is het resulterende waarde vermenigvuldigd met de spanning van de bron voor het bekomen van energieverbruik. Tenslotte wordt deze waarde gedeeld door *RTIMER_SECOND*, om energieverbruik voor uitgevoerde tijd te bekomen.

Hoofdstuk 6

Evaluatie

6.1 Optimalisaties voor elliptische kromme cryptografie

Om de uitvoeringstijd en het gebruik van *resources* te verminderen hebben onderzoekers gewerkt aan het optimaliseren van verschillende wiskundige bewerkingen. Er zijn verschillende papers waarin men korte beschrijving van deze optimalisaties terug kan vinden, die vooral succesvol gebruikt kunnen worden in draadloze sensornetwerken. In deze [33], [40] en [41] papers kan men een volledig geoptimaliseerd ECC *framework* voor *resources* terugvinden, met ook een efficiënte geoptimaliseerde implementatie door gebruik van beschikbare optimalisaties. In dit project worden niet alle beschikbare optimalisaties beschreven. Dit omdat de drie gebruikte bibliotheken niet alle optimalisaties bieden. De beschikbare optimalisaties, in de gebruikte bibliotheken, worden in detail bekeken. Dit zorgt voor de efficiënte implementatie van elliptische kromme cryptografie op Contiki OS en TinyOS, wat zeer noodzakelijk is in draadloze sensornetwerken. Zonder het gebruik van deze optimalisaties heeft men slechte prestaties in de draadloze sensornetwerken.

6.1.1 Projectieve coördinatensysteem

Een elliptische kromme bestaat uit het punt-op-oneindig *O* en een verzameling van punten in de affiene coördinaten (*x, y*) voor *x, y* ∈ *prime field*, die voldoen aan de vergelijking van de kromme. Een alternatief om een punt op een elliptische kromme te weer te geven is het projectieve coördinatensysteem [42]. Dit heeft drie coördinaten in plaats van twee, in de vorm van *(x, y, z)*.

Puntoptelling en puntverdubbeling zijn belangrijke bewerkingen in elliptische kromme cryptografie. Het zijn de fundamentele bouwstenen voor scalaire vermenigvuldigingen die nodig zijn voor alle elliptische kromme cryptografie algoritmes. Deze bewerkingen hebben in het affiene coördinatensysteem nood aan modulaire inversiebewerkingen, die meer *resources* en tijd vragen dan andere bewerkingen, zoals modulaire vermenigvuldigingen. Door gebruik van een projectieve coördinatensysteem [42] kunnen modulaire inversies worden vermeden door compensatie met een paar modulaire vermenigvuldigingen en kwadraten. Als gevolg hiervan zijn de uitvoeringstijden van puntoptelling en puntverdubbeling die gebaseerd zijn op een projectieve coördinatensysteem sneller dan die gebaseerd zijn op affiene coördinatensysteem [42].

Er worden twee extra optimaliseringen gebruikt wat de projectieve coördinatenrepresentatie efficiënter maakt. Deze zorgen voor meer vermindering van zowel uitvoeringstijd en het grootte van het programma. De eerste van twee extra optimaliseringen is de *mixed* puntoptelling [42], wat bestaat uit het optellen van een punt in projectieve coördinaten en tweede punt in affiene coördinaten. De tweede optimalisatie wordt *repeated* verdubbeling [42] genoemd, wat gebruikt wordt voor scalaire vermenigvuldigingen. Wanneer er achtereenvolgende puntverdubbelingen uitgevoerd moeten worden, kan men in plaats hiervan het *repeated* verdubbelingsalgoritme gebruiken. Dit zorgt voor snellere prestaties dan telkens herhaald gebruikmaken van de verdubbelingsformule. In *m* achtereenvolgende verdubbelingen behandelt dit algoritme *m-1* optellingen in het onderliggende eindig veld, *m*-1 delingen door twee en een verdubbeling [42].

Alhoewel de projectieve coördinaten representatie minder uitvoeringstijd nodig heeft, vraagt ze meer ROM-geheugen (voor het berekenen van complexe formules) en RAM-geheugen (voor het opslaan van extra variabelen) dan het affiene coördinatensysteem.

6.1.2 Sliding window

Scalaire vermenigvuldiging is een basisbewerking die gebruikt wordt in alle elliptische kromme cryptografie algoritmes. Het heeft een vorm van *kP*, waarbij *k* een geheel getal is met *P* een punt op een elliptische kromme. In de meest eenvoudige methode voor het berekenen van *kP*, wordt *k* gescand vanaf de meest significante bit naar minst significante bit. Wanneer het algoritme klaar is met het scannen van elke bit, moet men een puntverdubbeling berekenen. Als de gescande bit gelijk is aan "1", heeft het algoritme ook een puntoptelling nodig. De Sliding window methode [42] zorgt voor het versnellen van de scalaire vermenigvuldiging door het scannen van *w* aantal bits tegelijk. Telkens wanneer een *w* bit *window* wordt gescand moet het algoritme *w* puntverdubbelingen uitvoeren. Bij vooraf berekenen van 2*P*, 3*P* tot en met $(2^w-1)P$, moet de Sliding window slechts 1 puntoptelling uitvoeren. Dit zorgt dus voor minder rekentijd.

Het is eenvoudig om te begrijpen dat de Sliding window methode ervoor zorgt dat het verbruik van zowel de ROM-geheugen (voor aanvullende geheugen) en de RAM-geheugen (voor het opslaan van vooraf berekende punten) zal toenemen.

6.1.3 Shamir's Trick

Shamir's Trick [42] wordt alleen gebruikt voor de verificatie van ECDSA-handtekeningen. De verificatie van een ECDSA-handtekening vereist de berekening van de vorm *aP* + *bQ*, waarbij *a*, *b* gehele getalen zijn met *P*, *Q* twee punten op een elliptische kromme. Een eenvoudige implementatie vereist twee scalaire vermenigvuldigingen en een puntoptelling. Maar Shamir's Trick laat ons toe om bovenstaande waarde te berekenen aan een kost dichtbij één scalaire vermenigvuldiging. Concreet gezegd, met vooraf berekende *P* + *Q*, kan men dezelfde bits *a* en *b* scannen vanuit de meest significante naar minst significante. Voor elke bit moet men de tussenwaarde verdubbelen, die als het punt-op-oneindig wordt geïnitialiseerd. Als de gescande bitposities gelijk zijn aan *(ai= 0, bi= 1)*, *(ai= 1, bi= 0)* of *(ai= 1, bi= 1)*, dan moet men *P*, *Q* of *P* + *Q* toevoegen aan de tussenwaarde.
Net zoals de Sliding window methode, zal Shamir's Trick het verbruik van zowel de ROM-geheugen (voor aanvullende geheugen) en de RAM-geheugen (voor het opslaan van de vooraf berekende *P* + *Q*) doen verhogen.

Een aantal van de elliptische krommen die aangegeven zijn door NIST [24] en SECG [34] gebruiken pseudo-Mersenne priemgetallen. Een pseudo-Mersenne priemgetal is in de vorm $p = 2^n-c$, waarbij *c* kleiner is dan 2*n*. Reductie *modulo* een pseudo-Mersenne primegetal kan worden verkregen door een paar modulaire vermenigvuldigingen en optellingen, zonder enige deelbewerking. Hierdoor kan de tijd voor modulaire reductie aanzienlijk worden verminderd. Dus, met behulp van elliptische krommen over een pseudo-Mersenne priemgetal kan men extra prestatiewinst bereiken.

6.1.4 Montgomery reductie

Montgomery vermenigvuldiging [43], is een methode voor het uitvoeren van snelle modulaire vermenigvuldiging. Gegeven twee getallen *a*, *b* met *modulo* positieve integer *N*, berekent Montgomery vermenigvuldiging *ab* mod *N*. Montgomery vermenigvuldiging vereist het converteren van *a* en *b* in een speciale representatie. Door de *overhead* die betrokken is bij de omzetting, is het berekenen van een enkel product van Montgomery vermenigvuldiging trager dan het berekenen van het product van de gehele getallen en het uitvoeren van een modulaire reductie door deling of Barrett reductie. Echter, wanneer veel producten vereist zijn, zoals in *modulo* machtsverheffing, wordt de conversie naar Montgomery vorm een verwaarloosbare fractie van de tijd van de berekening, en het uitvoeren van de berekening van Montgomery vermenigvuldiging is dan sneller dan de beschikbare alternatieven. Barrett reductie vraagt naar meer ROM en RAM-geheugen in vergelijking met Montgomery reductie [33].

6.1.5 Barrett reductie

Een eenvoudige manier om voor een groot geheel getal modulaire reducties te verkrijgen, is het gebruikmaken van delingen [44]. Daarbij wordt de geheugen van de deling hergebruikt, hetgeen resulteert in een compactere geheugen.

Barrett is een alternatief voor modulaire reductie [45]. Het zet de reductie *modulo* een willekeurig geheel getal om naar twee vermenigvuldigingen en een paar reducties *modulo* gehele getallen van de vorm 2^n. Wanneer Barrett reductie gebruikt wordt voor het reduceren van één enkel getal, is Barrett reductie trager dan een normale deling. Echter, wanneer het gebruikt wordt voor het reduceren van verschillende getallen *modulo* hetzelfde getal een aantal keren op rij, door het vooraf berekenen van sommige waarden, kan Barrett reductie sneller zijn dan modulaire reducties verkregen door deling.

Barrett reductie zorgt voor een versnelling van de berekeningen. Dit omdat aangezien bijna alle modulaire bewerkingen *modulo* hetzelfde priemgetal zijn. Echter, dit vereist de implementatie van een apart reductie-algoritme, wat zorgt voor een grotere code op de *sensor node*. Bovendien verhoogt Barrett reductie ook RAM-geheugen. De Barrett reductie vereist de voorafgaande berekening van de volgende formule:

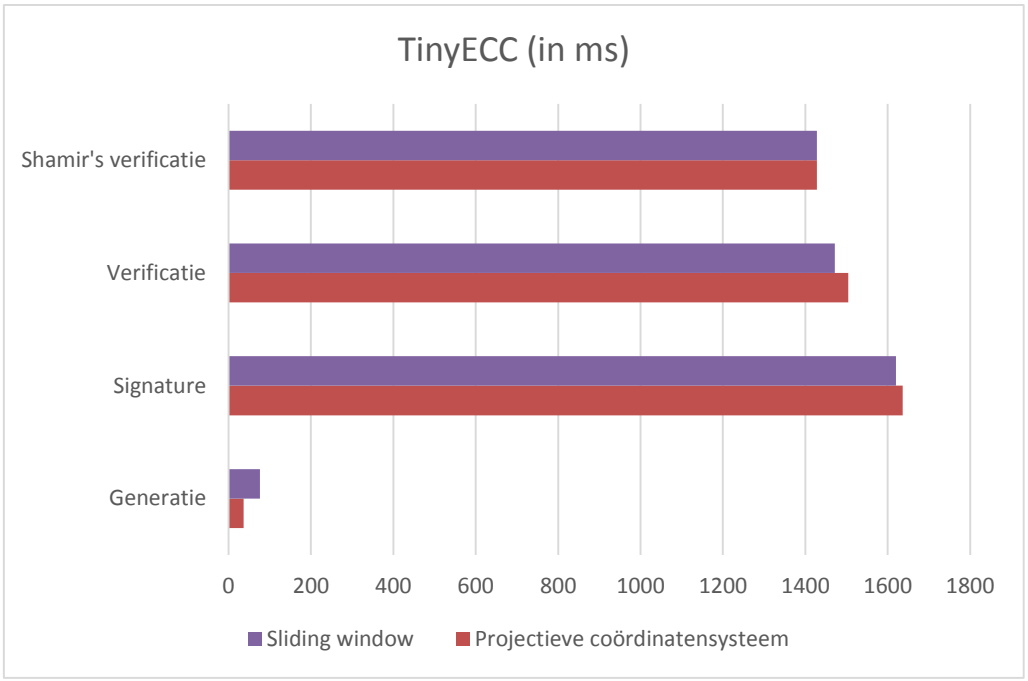
$$\mu = \left(\frac{b^k}{p}\right)$$

Waarin *b* is gelijk aan 2^w (met *w* als woordgrootte van de microcontroller) en *p* een *k*-woorden lang priemgetal. Het getal μ moet altijd opgeslagen worden en gebruikt worden door alle modulaire reducties. In tegenstelling tot snellere berekeningen, vraagt Barrett reductie meer ROM en RAM-geheugen dan de normale modulaire reductie.

6.2 Resultaten

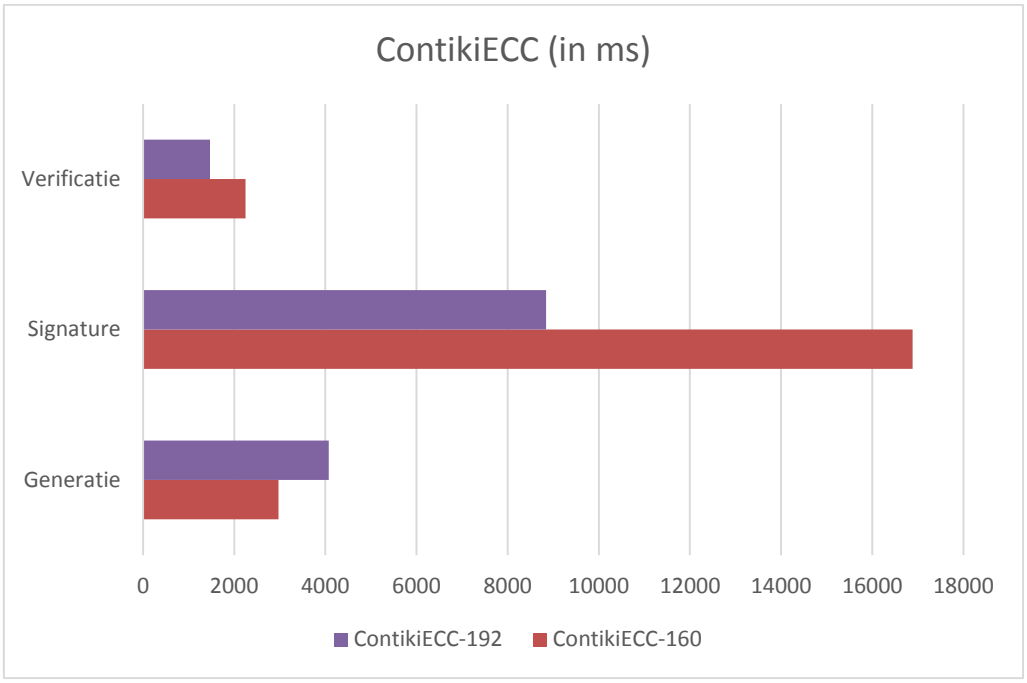
6.2.1 Tijd

Zoals het eerder vermeld is in punt 3.2.1.6, heeft men voor alle metingen gebruikgemaakt van ECDSA-algoritme. Voor meer informatie over de keuze ervan kan men terugkijken naar punt 3.2.1.6. ECDSA-algoritme heeft drie belangrijke delen die apart geanalyseerd worden voor tijd. Deze drie delen zijn namelijk generatie, *signature* en verificatie. Met gebruik van een publieke sleutel wordt er generatie van ECDSA-algoritme gecreëerd, terwijl de *signature* zorgt voor handtekening op het verzonden bericht en met verificatie aan de ontvanger kan men de bericht verifiëren. Een aantal prestaties van de optimalisaties per bibliotheek worden in Figuur 5, 6 en 7, hieronder, weergegeven.



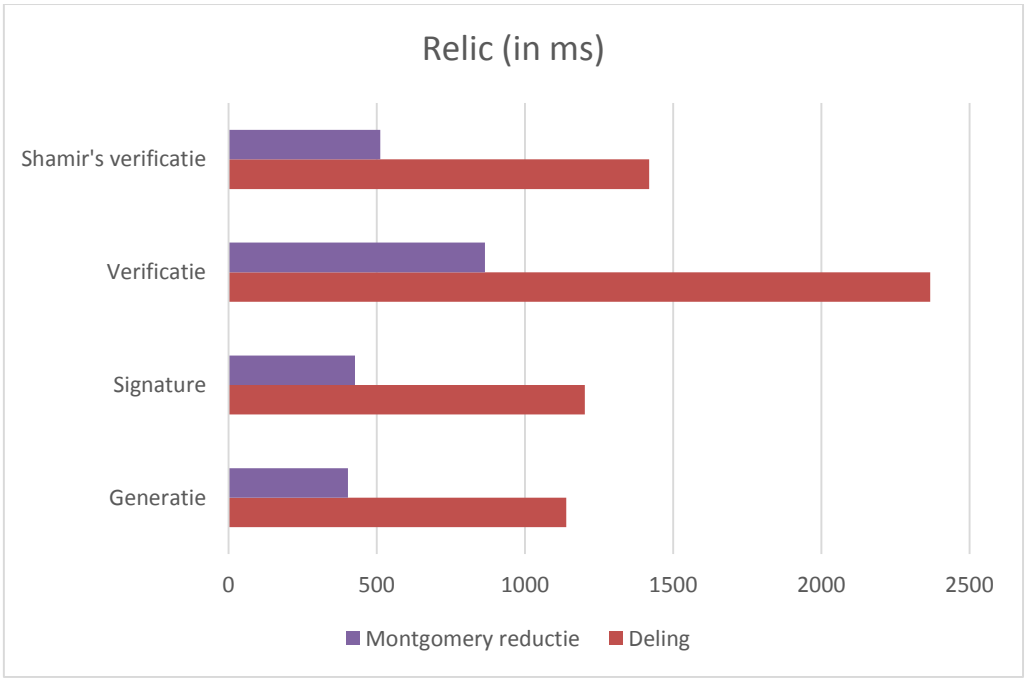
Figuur 6: Prestaties van TinyECC

In deze figuur wordt er vergelijking gemaakt tussen simpele vermenigvuldiging en Sliding window vermenigvuldiging. Bij beide metingen is er gebruikgemaakt van SECG160 curve. Uit de literatuurstudie die voor het beginnen aan de metingen doorgenomen is, blijkt er dat Sliding window algoritme zorgt voor minder tijd bij gebruik. Deze zorgt voor reductie van de scalaire vermenigvuldiging, door in het begin aantal *w windowen* te berekenen. Dit kunnen we duidelijk terugzien aan de resultaten.



Figuur 7: Prestaties van ContikiECC

In deze metingen, die hierboven weergegeven zijn, is er weer projectieve coördinatensysteem gebruikt. Enerzijds is er gebruikgemaakt van curve SECG160, anderzijds curve SECG192. Hieruit ziet men dat SECG192 voor reductie zorgt in de tijd. Uit deze figuur kan men duidelijk concluderen dat de metingen binnen de ContikiECC meer tijd hebben dan die van TinyECC, voor berekenen van de bewerkingen. Dit komt omdat ContikiECC bij het uitvoeren van de wiskundige bewerkingen traag is. Verder beschikt ContikiECC ook over Sliding window optimalisatie maar deze kan niet geüpload worden omdat Zolertia Z1 niet genoeg RAM-geheugen heeft.



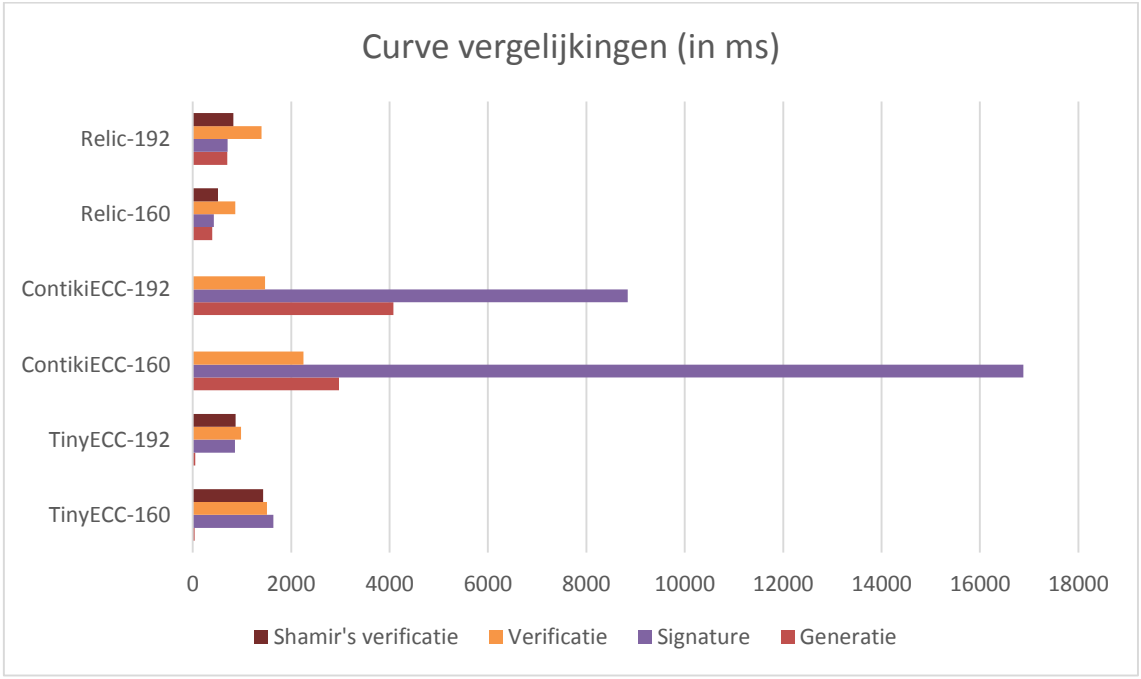
Figuur 8: Prestaties van Relic

Hier heeft men Montgomery reductie vergelijkt met de normale deling, die voor hetzelfde functie kunnen zorgen. Hieruit kan men zien dat generatie en *signature* bijna even lang duren. Dit komt omdat ze bijna dezelfde proces gebruiken. Bij handtekening worden er slechts enkele kleine wiskundige functies na scalaire vermenigvuldiging toegevoegd die voor zeer kleine verschil zorgen. Verificatie neemt meeste tijd in ECDSA-algoritme. Maar deze kan men verminderen door gebruik te maken van Shamir's algoritme. Indien men Shamir's algoritme gebruikt heeft men wel meer ROM en RAM-geheugen.

Uit deze bovenstaande figuren kan men duidelijk concluderen dat de bewerkingen binnen Relic, minder tijd nodig hebben dan de bewerkingen uit de andere bibliotheken. Dit komt omdat Relic bibliotheek gebruikmaakt van externe wiskundige bibliotheek. Er is een keuze mogelijkheid tussen twee wiskundige bibliotheken, die geschikt zijn op Relic. Deze wiskundige bibliotheken zijn, namelijk Relic-Easy en GnuMP).

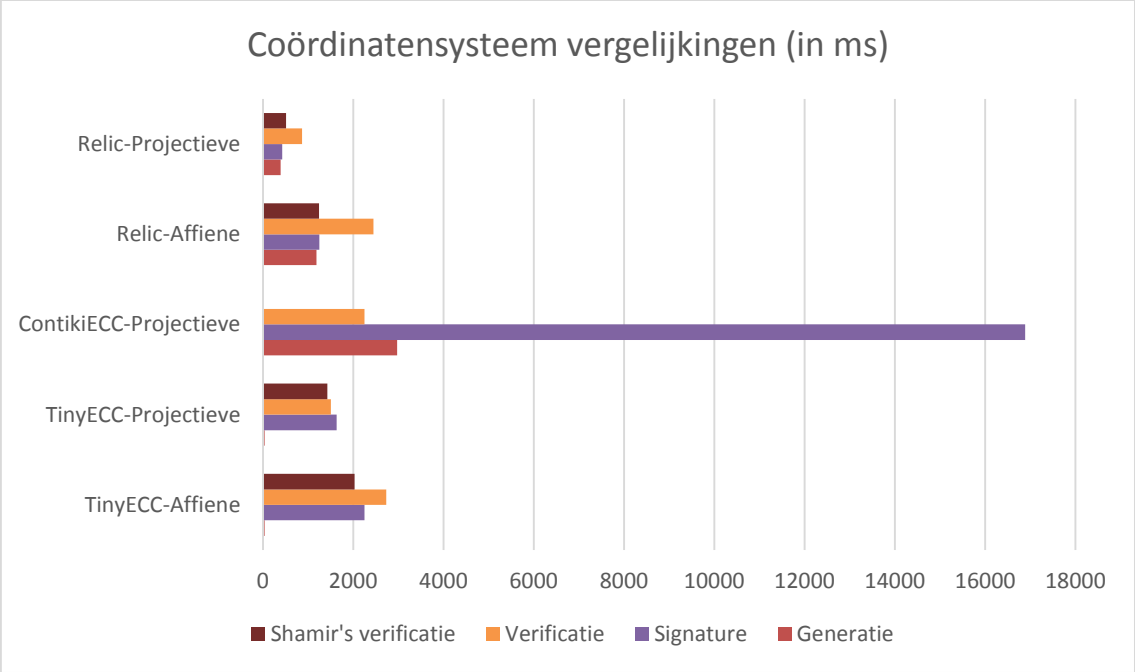
Contiki OS en TinyOS *runnen* op 8MHz bij *default*. Dit is een voordeel voor minder gebruik van tijd. Want bij kleinere *clock rates* heeft de bibliotheek meer tijd nodig om bewerkingen te maken. Dit zorgt voor meer uitvoeringstijd. Hogere *clock rates* zorgen wel voor meer gebruik van energie op *sensor nodes*.

Er zijn vele standaard curves die kunnen worden gebruikt voor ECDSA-algoritme functies. SECG160 en SECG192 zijn hier gebruikt voor alle metingen. Men kan uit Figuur 8, hieronder, zien dat SECG160 minder tijd neemt en minder handelingen nodig heeft in vergelijking met SECG192. Het belangrijkste punt dat iedere gebruiker zich moet concentreren, is het niveau van beveiliging die nodig is. De onderstaande gegevens van Relic bibliotheek tonen aan dat bij meer niveau van veiligheid, meer tijd nodig is. Dit geldt ook voor alle andere belangrijke factoren, zoals energie, ROM en RAM-geheugen. Echter, bij TinyECC en ContikiECC bibliotheken is dit het omgekeerde. Bij deze bibliotheken heeft men minder tijd nodig voor meer beveiliging.



Figuur 9: Curve vergelijkingen

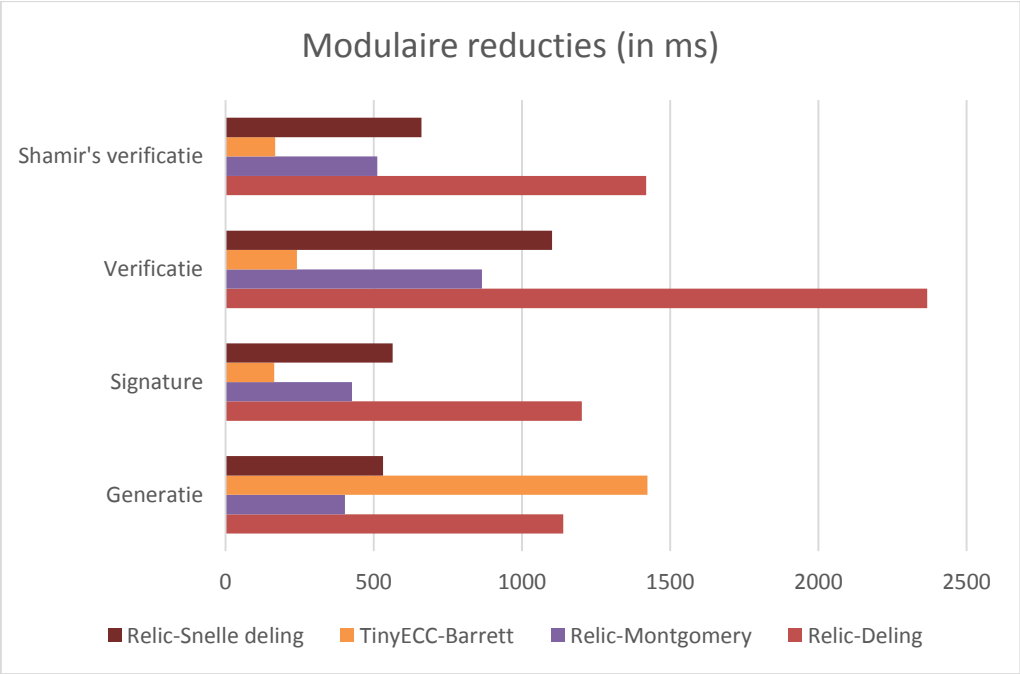
Er zijn twee typen coördinatensysteem die kunnen worden gebruikt in elke ECC gebaseerde algoritme voor het visualiseren van de punten. Deze zijn namelijk basiscoördinaten en projectieve coördinaten. Een vergelijking van deze coördinaten wordt in Figuur 9, hieronder, weergegeven om te visualiseren welke coördinatensysteem sneller voert en dus voldoet aan de eisen.



Figuur 10: Coördinatensysteem vergelijkingen

Bij deze metingen is er gebruikgemaakt van SECG160 curve. Het is hier duidelijk vastgesteld dat projectieve coördinatensysteem veel betere resultaten geeft in elk scenario vergeleken met basiscoördinaten. Dit ten koste van meer ROM en RAM-geheugen. In geval van ROM-geheugen heeft het bijna geen invloed op geheugen, maar RAM-geheugen gebruik is veel meer dan bij de affiene-coördinaten systeem. De Zolertia Z1 sensor heeft een RAM-geheugen capaciteit van 8kB en kan hierdoor makkelijk projectieve coördinatensysteem ondersteunen. Dus hierdoor is het is beter dat men projectieve coördinaten als voorkeur kiest in deze thesis.

Modulaire reductie optimalisaties zijn zeer belangrijk voor het leveren van efficiënte resultaten. Er zijn vier optimalisaties voor modulaire reducties. Deze zijn namelijk simpele deling, Montgomery reductie, Barrett reductie en snelle deling gebaseerde modulaire reductie. Deze worden gebruikt in *multi precision* gehele getallen om grote gehele getallen te reduceren in kleine getallen. Dit zorgt voor veel minder gebruik van *recources* tijdens het uitvoeren van grote wiskundige bewerkingen. Zo heeft men met gebruik van deze technieken minder uitvoeringstijd, ROM en RAM-geheugen en energieverbruik nodig. In Figuur 10, hieronder, worden de resultaten van de modulaire reductie optimalisaties weergegeven.



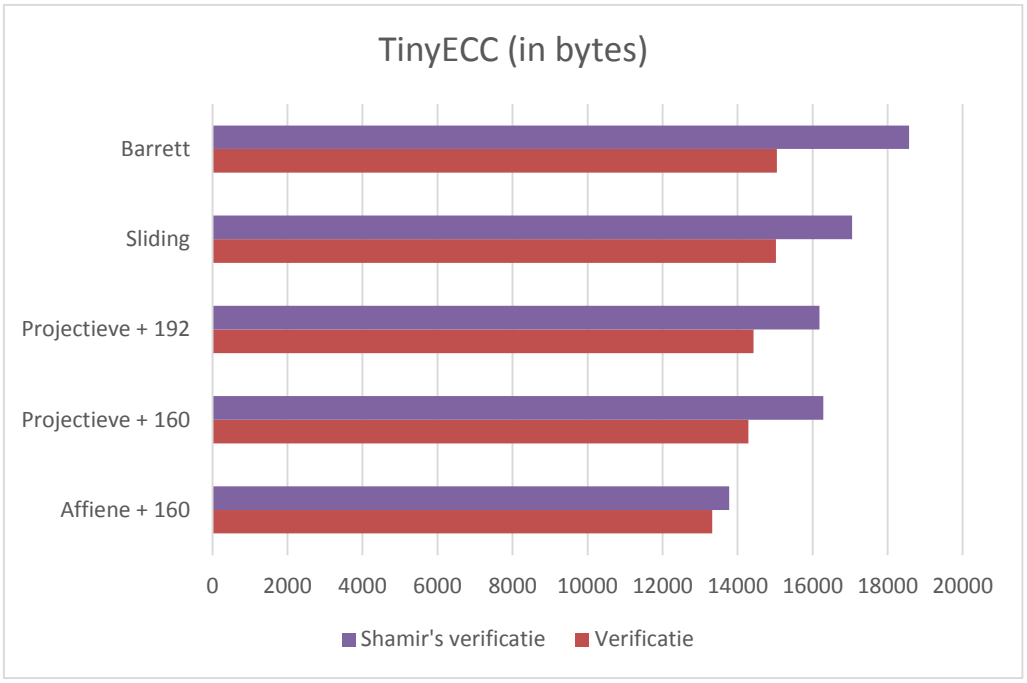
Figuur 11: Modulaire reducties

Uit deze grafiek kan men concluderen dat Barrett gebaseerde modulaire reductie minste tijd heeft om zijn functies te implementeren. Deze moet voorafgaand berekenen van waarde μ, wat eerder besproken is, en altijd opslaan om te gebruiken tijdens alle bewerkingen. Hierdoor heeft het in begin, bij generatie, meer tijd nodig dan de andere delen, zoals *signature* en verificatie. Deze ziet men duidelijk in de grafiek terug. Maar Barrett reductie is enkel geïmplementeerd in TinyECC bibliotheek.

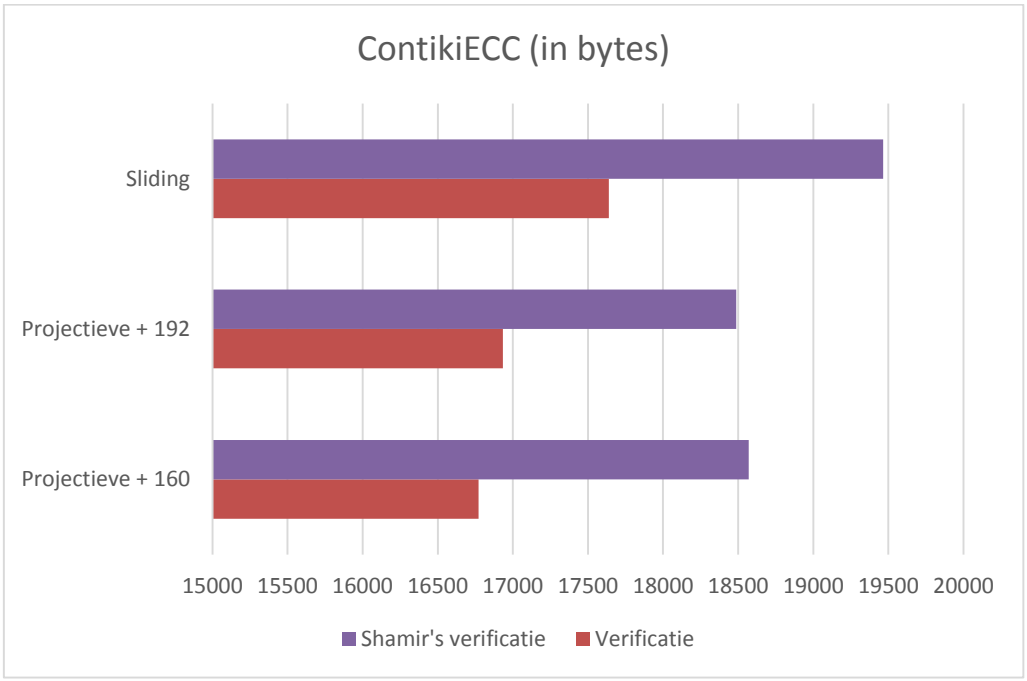
Voor Relic bibliotheek is het duidelijk uit de grafiek onderscheidbaar dat Montgomery reductie de rekentijd een grotendeels reduceert, te vergelijken met de andere modulaire reductie optimalisaties. Hierdoor is de voorkeur om Montgomery reductie te gebruiken tijdens het gebruik van Relic bibliotheek.

6.2.2 ROM

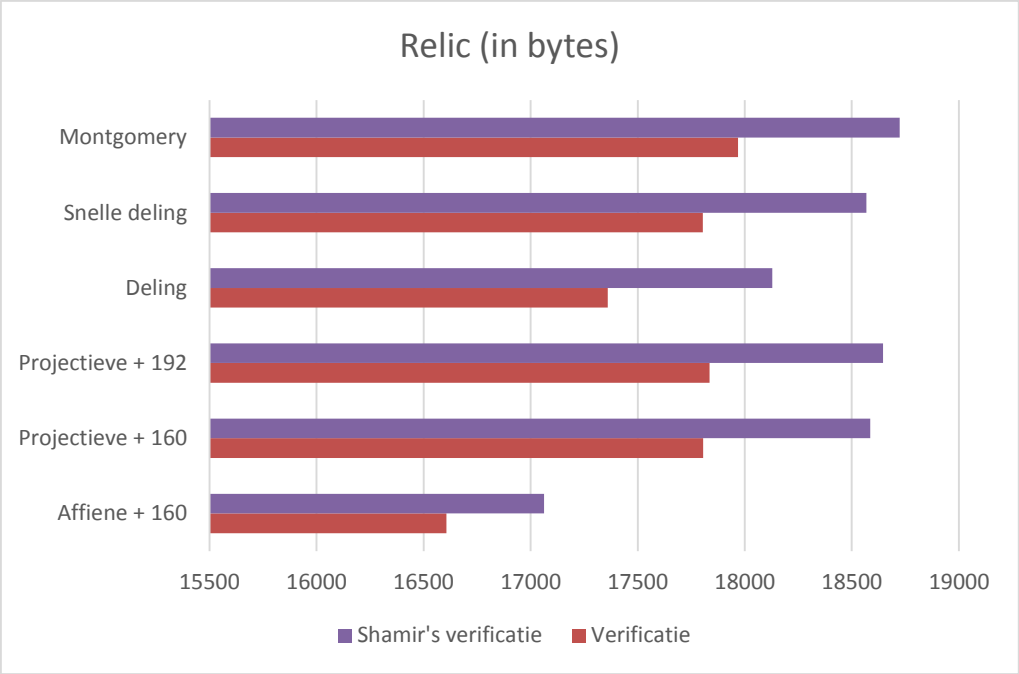
ROM is een belangrijke factor dat op het oog moet gehouden worden in draadloze sensornetwerken. Bij keuze van implementatie met gebruik van een optimalisatie moet men altijd bekijken of de microprocessor voldoende ROM-geheugen heeft, zodanig dat *uploaden* van de implementatie op de *sensor nodes* in draadloze sensornetwerken kan gebeuren. Indien men tekort is aan ROM-geheugen, kan men sommige gewenste codes gebruik van de optimalisaties niet *uploaden* op de *sensor nodes*, die aanwezig zijn in de draadloze sensornetwerken. In de drie komende figuren, namelijk Figuur 11, 12 en 13, wordt er voor ieder bibliotheek apart bekeken naar vereiste ROM-geheugen per gebruik van gekozen optimalisatie.



Figuur 12: ROM-geheugen van TinyECC



Figuur 13: ROM-geheugen van ContikiECC

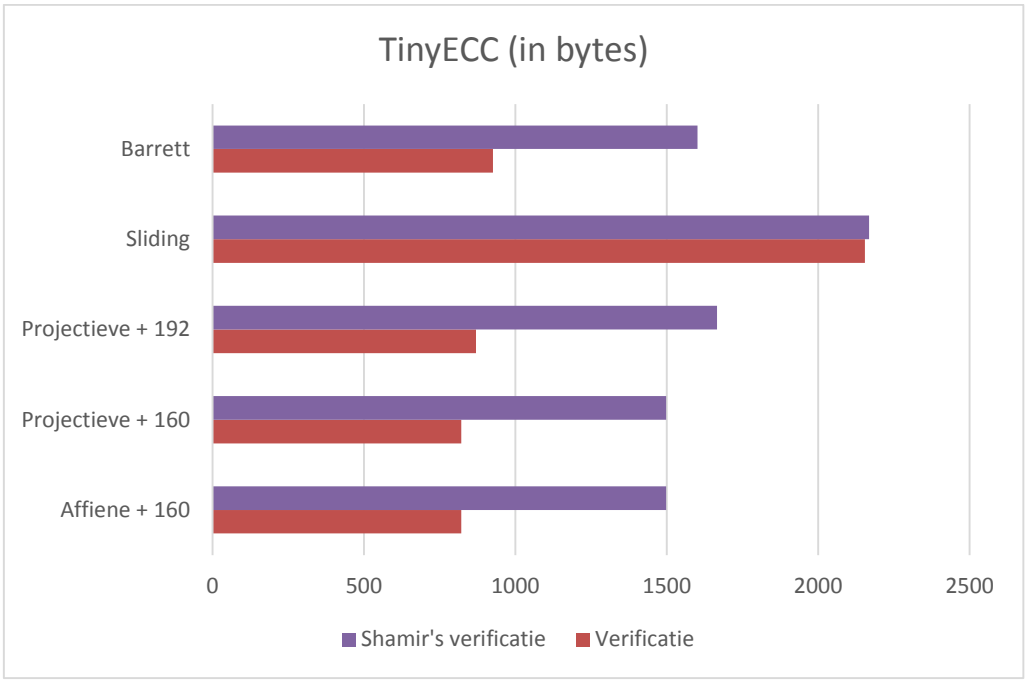


Figuur 14: ROM-geheugen van Relic

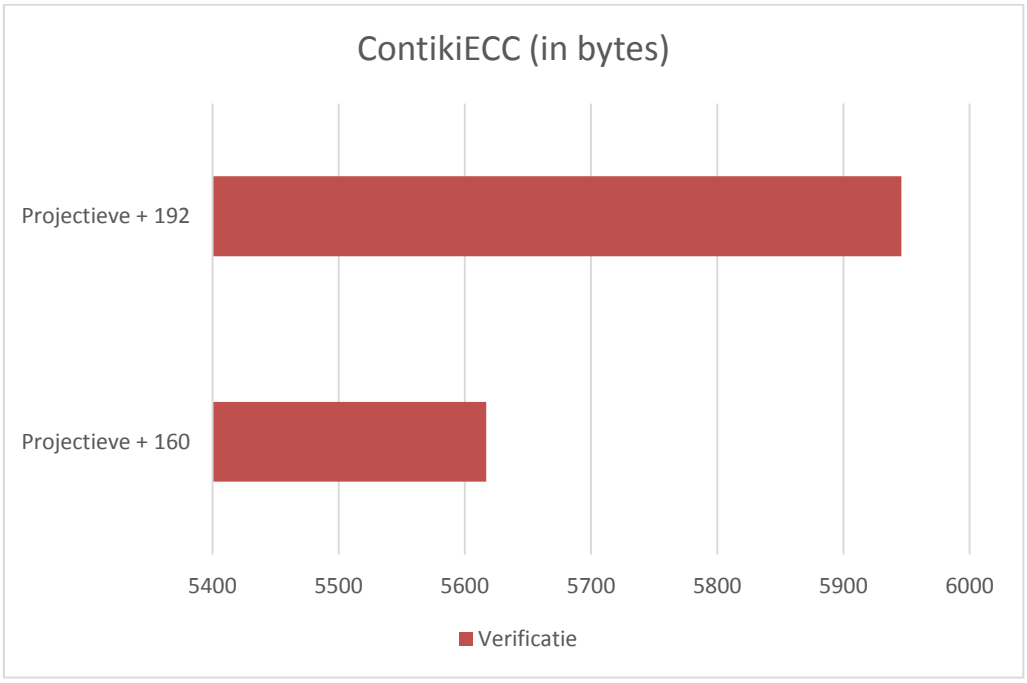
De tabellen tonen gedetailleerde vereiste geheugen voor het toevoegen van elliptische kromme cryptografie, met gekozen optimalisatie, in Contiki OS en TinyOS. De door hulpprogramma's gemeten geheugen voor het gebruik van elliptische kromme cryptografie in *sensor nodes* is ongeveer tussen 13 en 19kB groot. Uit deze tabellen is het duidelijk herkenbaar dat TinyECC minder ROM-geheugen vraagt, dan de andere bibliotheken. Hier is ook te zien dat de ROM-geheugen niet afhankelijk is van het gewenste beveiligingsniveau. Daarnaast blijkt dat vereiste geheugen voor SECG160 en SECG192 krommen grotendeels niet verandert. Dit betekent dat deze geen invloed hebben op ROM-geheugen. Verder ziet men ook dat Shamir's verificatie algoritme naar meer ROM-geheugen vraagt. Dus indien men snellere verificatie wilt hebben, moet men meer ROM-geheugen weggeven. Dit geldt ook voor indien men modulaire reductie optimalisaties neemt, voor reductie in de uitvoeringstijd.

6.2.3 RAM

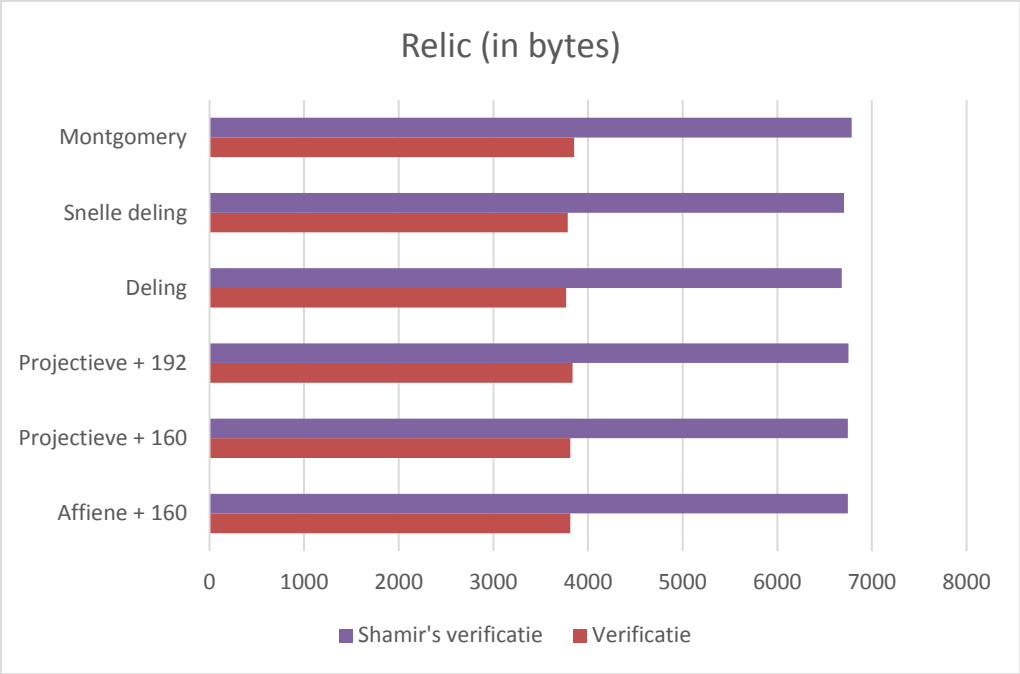
De RAM-geheugen is een strikt belangrijke factor dat naar veel aandacht vraagt in de *sensor nodes* die toepasbaar zijn in draadloze sensornetwerken. Dit komt omdat *sensor nodes* vaak niet genoeg over RAM-geheugen beschikken en hierdoor sommige optimalisaties niet op de *sensor nodes* toepasbaar zijn. Een duidelijke overzicht van RAM-geheugen voor ieder bibliotheek apart per optimalisatie, ziet men in Figuur 14, 15 en 16.



Figuur 15: RAM-geheugen van TinyECC



Figuur 16: RAM-geheugen van ContikiECC

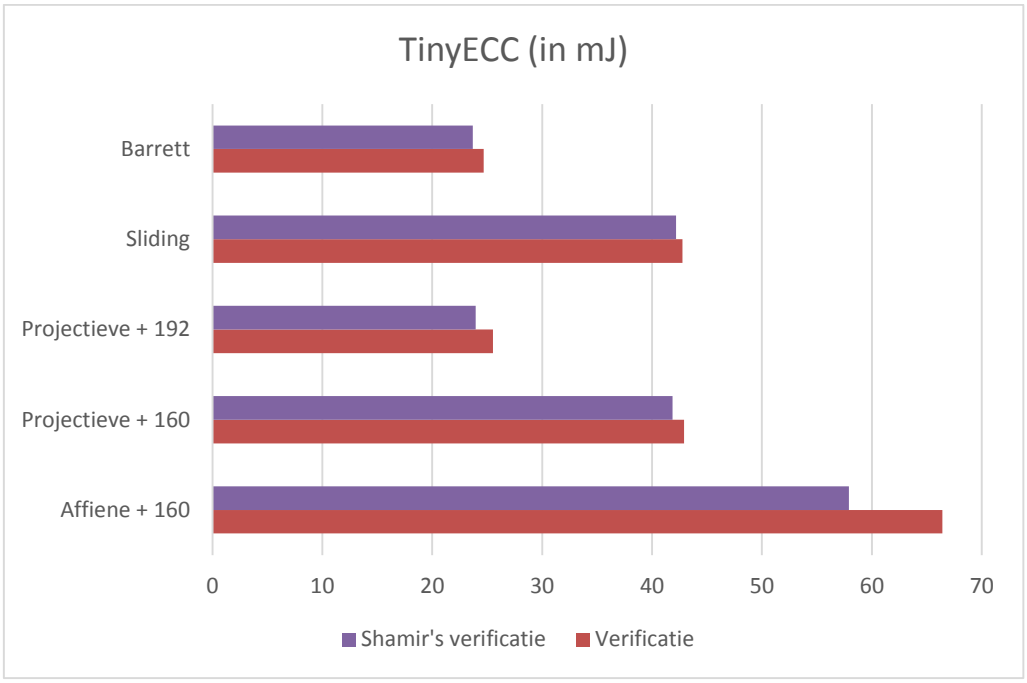


Figuur 17: RAM-geheugen van Relic

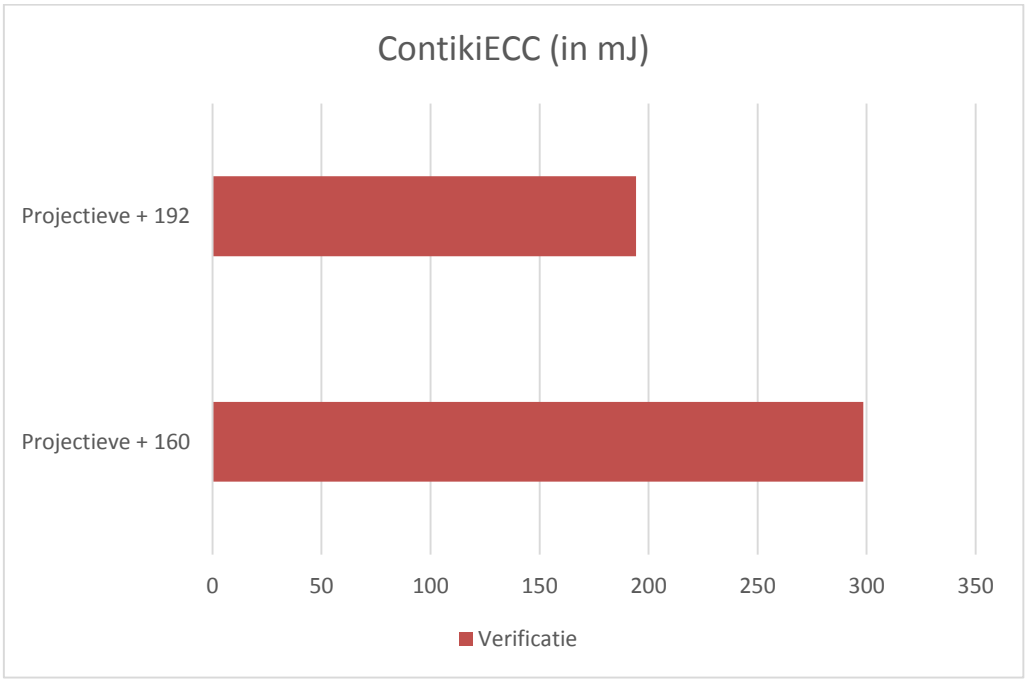
Zoals bovenvermeld beschikken de *sensor nodes*, die vandaag de dag op de markt zijn, over weinig RAM-geheugen. In deze thesis gebruikte, Zolertia Z1, *hardware sensor node* beschikt over ongeveer 8kB RAM-geheugen. Door deze gebrek aan RAM-geheugen was het *uploaden* van sommige optimalisaties die ContikiECC biedt niet mogelijk. Sliding window algoritme en Shamir's verificatie algoritme optimalisaties van ContikiECC zijn hierdoor niet toegepast. Evenzo kan men hier concluderen dat TinyECC naar minste RAM-geheugen vraagt bij het gebruik ervan. Verder ziet men ook dat wanneer men meer reductie wilt in de uitvoeringstijd, men meer RAM-geheugen moet weggeven.

6.2.4 Energie

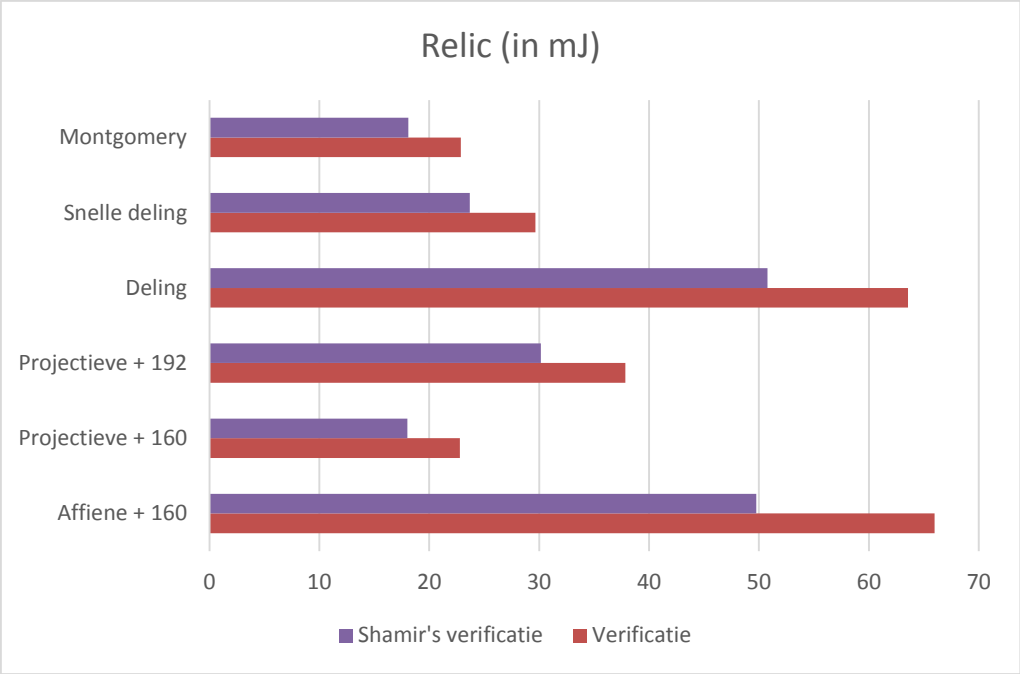
Energieverbruik is een belangrijk punt van zorg op dit moment voor de *sensor nodes* in draadloze sensornetwerken. Dit komt omdat de *sensor nodes* batterij-gevoed zijn en hierdoor weinig beschikken van *power*. Het grotendeel van de gebruikte toepassingen in draadloze sensornetwerken verbruiken energie in de µJ. Maar dit verandert tot mJ van energieverbruik bij gebruik van publieke cryptografie in draadloze sensornetwerken. In de drie komende figuren, namelijk Figuur 17, 18 en 19, wordt er voor ieder bibliotheek apart bekeken naar energieverbruik per gebruik van gekozen optimalisatie.



Figuur 18: Energieverbruik van TinyECC



Figuur 19: Energieverbruik van ContikiECC



Figuur 20: Energieverbruik van Relic

Uit deze figuren blijkt dat Relic het minste energieverbruik heeft, te vergelijken met de twee andere bibliotheken. Bij gebruik van de optimalisaties in Relic vraagt het nog minder energie. Hieruit blijkt dat Montgomery ook minder energieverbruik heeft dan alle andere optimalisaties. In verificatie fase neemt Shamir's verificatie algoritme de minimale energie. Dit allemaal is doordat het minder rekentijd nodig heeft voor het berekenen van de bewerkingen.

6.2.5 Samenvatting

Eerste en belangrijkste informatie die vereist is om de publieke sleutel te implementeren is om te bepalen hoeveel niveau van beveiliging er nodig is. Indien men een *sensor node* in de militaire of een andere beveiligingsomgeving toepast, moet men een hoger niveau van veiligheid beschouwen. In dit geval spreekt men van een veiligheidsniveau van tenminste 192bits elliptische kromme cryptografie. In andere geval, wanneer de *sensor node* in een vreedzame zone wordt geplaatst en alleen vereist is voor het bieden van het verzamelen van informatie, is een 160bits gebaseerde elliptische kromme cryptografie genoeg om berichtintegriteit te waarborgen. Bijgevolg, hoe hoger men het niveau van de veiligheid wilt uitvoeren, hoe hoger de ROM, RAM-geheugen en energiebehoefte wordt.

Na onderzoek en gebruik van deze bibliotheken heeft men besloten om ook de optimalisaties die, geschikt zijn in de gekozen bibliotheken, te gebruiken en *uploaden* op de *sensor nodes*. Zoals uit de resultaten is gebleken zorgen deze optimalisaties voor efficiëntie bij gebruik. Maar niet alle optimalisaties, die aanwezig zijn in de bibliotheken, waren toepasbaar. Zo had men bij ContikiECC geen mogelijkheid om Shamir's verificatie en Sliding window algoritme te gebruiken. In Tabel 12, hieronder, vindt men een overzicht van alle optimalisaties per bibliotheek die toepasbaar zijn op Zolertia Z1.

Tabel 11: Op Zolertia Z1 toepasbare optimalisaties per bibliotheek

Optimalisaties	**Relic**	**ContikiECC**	**TinyECC**
Simpele vermenigvuldiging	Toepasbaar	Toepasbaar	Toepasbaar
Sliding window vermenigvuldiging	Niet beschikbaar	Niet toepasbaar	Toepasbaar
Montgomery reductie	Toepasbaar	Niet beschikbaar	Niet beschikbaar
Barrett reductie	Niet beschikbaar	Niet beschikbaar	Toepasbaar
Affiene coördinatensysteem	Toepasbaar	Niet beschikbaar	Toepasbaar
Projectieve coördinatensysteem	Toepasbaar	Toepasbaar	Toepasbaar
Simpele verificatie	Toepasbaar	Toepasbaar	Toepasbaar
Verificatie van Shamir	Toepasbaar	Niet toepasbaar	Toepasbaar
SECG160 curve	Toepasbaar	Toepasbaar	Toepasbaar
SECG192 curve	Toepasbaar	Toepasbaar	Toepasbaar

Onder Contiki OS kan er best gebruikt worden van Montgomery reductie van Relic bibliotheek. Dit omdat deze beste prestaties levert zowel in de uitvoeringstijd, ROM, RAM-geheugen en als in energieverbruik. Bij nood aan publieke sleutelcryptografie onder TinyOS kan men Barrett reductie optimalisatie gebruiken. Deze geeft evenzeer beste resultaten indien men het vergelijkt met andere optimalisaties van TinyECC, onder TinyOS.

Hoofdstuk 7

Conclusie en uitbreidingsmogelijkheden

In deze thesis wordt er een haalbaarheidsstudie voor het implementeren van publieke sleutelcryptografie in draadloze sensornetwerken gepresenteerd. Er is ook een vergelijkende studie gemaakt tussen mogelijke publieke sleutelcryptografie algoritmen in draadloze sensornetwerken. Publieke sleutelcryptografie is steeds duur en moeilijk toepasbaar op een *sensor node* die aanwezig is in een draadloos sensornetwerk. Er is echter in deze thesis gebleken dat Contiki OS en TinyOS ook een hoog niveau van veiligheid met kleine sleutelgroottes kunnen bereiken, door het toepassen van de gebruikte bibliotheken.

Alsook is er een detailbeschrijving van alle parameters die nodig zijn om elliptische kromme cryptografie te evalueren en te gebruiken in Contiki OS en TinyOS besproken. Hierin worden drie cryptografische bibliotheken geanalyseerd, namelijk TinyECC, ContikiECC en Relic. Deze worden gebruikt met aandacht voor *timing* prestaties, ROM en RAM-geheugen gebruik binnen Contiki OS en TinyOS. TinyECC is de keuze voor gebruik binnen TinyOS, terwijl de Relic bibliotheek de voorkeur heeft binnen Contiki OS. Dit omdat de Relic bibliotheek betere prestaties geeft dan ContikiECC.

Zonder gebruik van de optimalisaties is elliptische kromme cryptografie te duur op draadloze sensornetwerken. Evaluaties van de prestaties en de *resource* efficiëntie van het ECDSA-algoritme op de Zolertia Z1 *hardware* zijn ook gepresenteerd. Uit experimenten is er gebleken dat het mogelijk is om elliptische kromme cryptografie met zijn optimalisaties te gebruiken op draadloze sensornetwerken.

In deze thesis heeft men zich steeds gefocust op één publieke sleutelcryptografie algoritme. Hiervoor is er elliptische curve cryptografie gekozen, dat meer efficiënt is voor gebruik in *lightweight* gebaseerde draadloze sensornetwerken. Binnen elliptische kromme cryptografie heeft men enkel ECDSA-algoritmes gebruikt. De andere algoritmes binnen elliptische kromme cryptografie, zoals ECDH, ECIES en ECMQV, kunnen gebruikt en vergeleken worden om de volledige elliptische kromme cryptografie onder Contiki OS en TinyOS te analyseren.

Bibliografie

[1] "ES&S." [Online]. Beschikbaar: http://www.acro.be/NL/ess.php?id=102. [Geopend: 15-sep-2015].

[2] "Contiki Operating System." [Online]. Beschikbaar: http://www.contiki-os.org. [Geopend: 15-sep-2015].

[3] "MANTIS OS." [Online]. Beschikbaar: http://mantisos.sourceforge.net. [Geopend: 15-sep-2015].

[4] "TinyOS." [Online]. Beschikbaar: http://www.tinyos.net. [Geopend: 15-sep-2015].

[5] A. Dunkels, "Full TCP/IP for 8-bit architectures," *Proc. 1st Int. Conf. Mob. Syst. Appl. Serv. - MobiSys '03*, pp. 85–98, 2003.

[6] L. Bruno, M. Franceschinis, C. Pastrone, R. Tomasi, and M. Spirito, "6LoWDTN: IPv6-enabled Delay-Tolerant WSNs for Contiki," *2011 Int. Conf. Distrib. Comput. Sens. Syst. Work.*, pp. 1–6, Jun. 2011.

[7] L. Casado and P. Tsigas, "ContikiSec : A Secure Network Layer for Wireless Sensor Networks under the Contiki Operating System," pp. 1–16.

[8] A. R. Sedghi and M. R. Kaghazgaran, "Data Security via Public-Key Cryptography in Wireless Sensor Network," vol. 2, no. 3, pp. 1–11, 2013.

[9] R. Berger, "Introduction to Wireless Sensor Networks," *2009 NI Tech. Symp.*, 2009.

[10] C. Townsend and S. Arms, "Wireless Sensor Networks:," no. Ic, pp. 575–589.

[11] J. Walters and Z. Liang, "Wireless sensor network security: A survey," *Secur. Distrib. …*, pp. 1–50, 2007.

[12] J. Deng, R. Han, and S. Mishra, "Enhancing Base Station Security in Wireless Sensor Networks," *Science (80-.).*, no. April, pp. 1–17, 2003.

[13] S. Zamani and M. Jafari, "Security in WSN." .

[14] S. Mohammadi, "A Comparison of Link Layer Attacks on Wireless Sensor Networks," *J. Inf. Secur.*, vol. 02, no. 02, pp. 69–84, 2011.

[15] M. Çakiroğlu and A. T. Özcerit, "Jamming detection mechanisms for wireless sensor networks," 2008.

[16] M. Conti, R. Di Pietro, L. Mancini, and a Mei, "Emergent properties: detection of the node-capture attack in mobile wireless sensor networks," *… ACM Conf. Wirel. …*, pp. 214–219, 2008.

[17] M. Messai, "Classification of Attacks in Wireless Sensor Networks," *Icta*, no. April 2014, pp. 23–24, 2014.

[18] W. Znaidi, M. Minier, and J. Babau, "An Ontology for Attacks in Wireless Sensor Networks," pp. 1–17, 2008.

[19] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," *Proc. 2nd ACM Int. Conf. Embed. Networked Sens. Syst.*, pp. 162–175, 2004.

[20] "SICS Swedish ICT." [Online]. Beschikbaar: https://www.sics.se/. [Geopend: 15-sep-2015].

[21] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," *Proc. ACM SIGPLAN 2003 Conf. Program. Lang. Des. Implement. - PLDI '03*, p. 1, 2003.

[22] "Z1 Datasheet," pp. 1–20.

[23] J. Lopez, "Unleashing public-key cryptography in wireless sensor networks," 2006.

[24] NIST, "Recommendation for Key Management - Part 1: General," *NIST Spec. Publ. 800-57*, vol. Revision 3, no. July, pp. 1–147, 2012.

[25] K. Piotrowski, P. Langendoerfer, and S. Peter, "How Public Key Cryptography Influences Wireless Sensor Node Lifetime," *Proc. Fourth ACM Work. Secur. Ad Hoc Sens. Networks*, p. 169, 2006.

[26] W. Diffie, P. van Oorshot, and M. Wiener, "Authentication and Authenticated Key Exchange," *Des. Codes Cryptogr.*, vol. 2, pp. 107–125, 1992.

[27] M. England, “Including The Generalisation To Higher Genus The Burn 2007,” 2007.
[28] Certicom, “ECC TUTORIAL.” [Online]. Beschikbaar: https://www.certicom.com/10-introduction. [Geopend: 10-okt-2015].
[29] “Cryptography.” [Online]. Beschikbaar: http://crypto.stackexchange.com/questions/3907/how-does-one-calculate-the-scalar-multiplication-on-elliptic-curves. [Geopend: 12-dec-2015].
[30] W. Chou, “Elliptic curve cryptography and its applications to mobile devices,” *Univ. Maryland, Coll. Park. USA*, pp. 1–23, 2003.
[31] “Relic.” [Online]. Beschikbaar: https://code.google.com/p/relic-toolkit/. [Geopend: 10-okt-2015].
[32] “ContikiECC.” [Online]. Beschikbaar: https://github.com/punyal/Contiki_3-IPsec/tree/master/core/lib/contikiecc. [Geopend: 11-okt-2015].
[33] A. Liu and P. Ning, “TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks,” *Proc. - 2008 Int. Conf. Inf. Process. Sens. Networks, IPSN 2008*, pp. 245–256, 2008.
[34] C. Research, “Standards for Efficient Cryptography, SEC1: Elliptic Curve Cryptography,” vol. 1, no. Sec 1, 2000.
[35] S. Toledo, “Exercise in Embedded Computing : Contiki Basics,” pp. 1–2, 2009.
[36] “msp430-size.” [Online]. Beschikbaar: http://www.linuxcertif.com/man/1/msp430-size/. [Geopend: 21-okt-2015].
[37] “msp430-ram-usage.” [Online]. Beschikbaar: https://github.com/cetic/python-msp430-tools/blob/master/scripts/msp430-ram-usage. [Geopend: 10-okt-2015].
[38] X. Jiang, P. Dutta, D. Culler, and I. Stoica, “Micro Power Meter for Energy Monitoring of Wireless Sensor Networks at Scale,” *Proc. 6th Int. Conf. Inf. Process. Sens. networks - IPSN ’07*, p. 186, 2007.
[39] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, “Software-based on-line energy estimation for sensor nodes,” *Proc. 4th Work. Embed. networked sensors - EmNets ’07*, p. 28, 2007.
[40] M. Varchola and T. Güneysu, “MicroECC: A Lightweight Reconfigurable Elliptic Curve Crypto-processor,” 2011.
[41] M. Varchola, “A Very Lightweight Reconfigurable Elliptic Curve Crypto-Processor,” p. 615.
[42] D. Hankerson, a J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. 2003.
[43] M. Knežević, F. Vercauteren, and I. Verbauwhede, “Faster interleaved modular multiplication based on Barrett and Montgomery reduction methods,” *IEEE Trans. Comput.*, vol. 59, no. 12, pp. 1715–1721, 2010.
[44] D. E. Knuth, “The art of computer programming,” *Sorting Search.*, 1981.
[45] Z. Cao, R. Wei, and X. Lin, “A Fast Modular Reduction Method,” *Eprint.Iacr.Org*, pp. 1–12, 2014.

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Impact van publieke sleutel cryptografie op draadloze sensornetwerken

Richting: **master in de industriële wetenschappen: elektronica-ICT**
Jaar: **2016**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Gultekin, Umit

Datum: **15/01/2016**