INTERNET OF THINGS: SECURE APPLICATION IN INDUSTRIAL WIRELESS SENSOR NETWORKS

by

Roy Mason Fisher

Submitted in partial fulfillment of the requirements for the degree

Master of Engineering (Computer Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology
UNIVERSITY OF PRETORIA

April 2015

SUMMARY

INTERNET OF THINGS: SECURE APPLICATION IN INDUSTRIAL WIRELESS SENSOR NETWORKS

by

Roy Mason Fisher

Supervisor(s):	Dr. G.P. Hancke
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Master of Engineering (Computer Engineering)
Keywords:	Internet of Things, Security, Transport Layer Security, Open Source Technologies, Hardware Platforms.

The Internet of Things has become a hugely popular field of research. This is due to the effect that the application of this range of technologies could have on an individual's daily life. These technologies could be applied in a range of applications from the 'smart home' to a more connected and 'smarter' industrial application. The Internet of Things is a paradigm in which a range of technologies are used to provide for ubiquitous communication between nearly all objects in the world. Currently the approach has been to make use of a collection of proprietary technologies and hardware in order to provide this ubiquitous connection between devices. This focus on proprietary (closed source) hardware and technologies has come about as a result of the belief that open source hardware and software is inferior, especially for industrial use. This effect is compounded by the fact that proprietary hardware is designed to operate optimally with other hardware made by the same company. Often this technology is not inter-operable with hardware from other competing companies. Another strongly held belief is that the current range of Internet of Things devices are low resource devices with a limited range of capabilities. This belief has led to the development of a range of specialized protocols specifically within this domain to provide advanced capabilities (such as secure communication) in a form that these low resource devices can make use of.

Through the course of this research it is shown that not only are the available new devices

powerful enough to make use of the standard protocols that are available but that an open source approach to the design and development of the application ensures that the devices are inter-operable. This comes as a result of the fact that the development process for open source technologies is far more inclusive and built to the community standards rather than to a specific company's specifications. Through the course of this research it is shown that open source technologies allow for a more capable and inter-operable device to be created. These open source technologies also open up the possibility of creating customized devices from a commercial-off-the-shelf devices (COTS) where devices are modified to work in the required application.

The COTS approach together with open-source modular design approach also allows for the upgrading of individual technologies within the system. An example, consider that the ZigBee protocol is currently the preferred communication technology. If in the future a better performing technology becomes available a simple upgrade of the ZigBee component within the system allows for rapid upgrading of the entire system to the new standard. The transparent design, development and maintenance approaches to these new technologies also allow for a better Plug and Play approach to modular system development.

OPSOMMING

INTERNET VAN DINGE: BEVEILIGING VAN TOEPASSINGS BINNE INDUSTRIËLE DRAADLOSE SENSOR NETWERKE

deur

Roy Mason Fisher

Studieleier(s):	Dr. G.P. Hancke
Departement:	Elektriese, Elektroniese en Rekenaar-Ingenieurswese
Universiteit:	Universiteit van Pretoria
Graad:	Magister in Ingenieurswese (Rekenaaringenieurswese)
Sleutelwoorde:	Internet van Dinge, Sekuriteit, Vervoer-laag Sekuriteit, Oopbron-tegnologie, Hardeware Platforms

Die "Internet van Dinge" het ’n uiters gewilde gebied van navorsing geword. Dit is omdat die gebruik van hierdie tegnologie ’n groot impak kan hê op individue se daaglikse lewens. Hierdie tegnologie kan toegewend word in ’n verskeidenheid toepassings, en kan in enige toepassings vanaf ’n "slim huis" tot meer gekoppelde "intelligente" industriële stelsels voorkom. Die "Internet van Dinge" is ’n paradigma waarin ’n reeks tegnologie gebruik kan word vir alomteenwoordige kommunikasie tussen amper alle voorwerpe in die wêreld. Tans word daar meestal gebruik gemaak van verskeie enkel-verskaffer-tegnologie en -hardeware om die alomteenwoordige konneksie tussen voorwerpe te voorsien. Hierdie fokus op enkel-verskaffer (geslote-bron) -tegnologie is meegebring deur die gevoel dat ope-bron-hardeware en -sagteware minderwaardig is, veral in industriële toepassings. Dit word erger gemaak deur die feit dat enkel-verskaffer-hardeware ontwerp is om net optimaal te funksioneer saam met ander hardeware vanaf dieselfde maatskappy. Dikwels is hierdie tegnologie nie aanpasbaar by hardeware van kompeterende maatskappye nie. Nog ’n sterk oortuiging is dat huidige "Internet van Dinge"- toestelle min hulpbronne en dus lae vermoëns het. Dit het gelei tot die ontwikkeling van ’n reeks gespesialiseerde protokolle, spesifiek binne hierdie domein, om gevorderde kapasiteit (bv. beveiligde kommunikasie) moontlik te maak vir toestelle met lae

vermoëens.

Deur die loop van hierdie navorsing word gewys dat die beskikbare nuwe toestelle nie net kragtig genoeg is om van die standaard protokolle gebruik te maak nie, maar ook dat ’n ope-bron aanslag tot die ontwerp en ontwikkeling van toepassings meebring dat toestelle ook inter-aanpasbaar is. Dit gebeur as gevolg van die feit dat die proses van ontwikkeling vir ope-bron-tegnologie baie meer inklusief is, en gebou word volgens gemeenskaplike standaarde, eerder as die spesifikasies van ’n enkele maatskappy. Deur die loop van hierdie navorsing word aangetoon dat ope-bron-tegnologie toelaat vir ’n meer geskikte en inter-aanpasbare toestel. Hierdie ope-bron-tegnologie maak dit ook moontlik dat doelgerigte toestelle geskep kan word vanuit ‘kommersiële-vanaf-die-rak’ produkte (‘COTS’), waarin toestelle omskep word om in die benodigde toepassings te werk.

Die ‘kommersiële-vanaf-die-rak’ benadering saam met oopbron-modulere ontwerp maak ook voorsiening vir die opgradering van enkele tegnologië binne die stelsel. As voorbeeld, tans geniet the Zigbee-protocol voorkeur as kommunikasietegnologie. As daar in die toekoms ’n meer geskikte tegnologie beskikbaar word, sal ’n eenvoudige opgradering van die Zigbee-komponente binne die stelsel voorsiening maak vir ’n vinnige opgradering van die hele stelsel tot die nuwe standaard. Die deursigtige ontwerp, ontwikkeling en instandhoudingsbenadering tot hierdie nuwe tegnologie maak ook voorsiening vir ‘n beter "prop en speel" benadering tot modulêre stelsel-ontwikkeling.

LIST OF ABBREVIATIONS

3G	Third Generation Network
4G	Fourth Generation Network
6BR	IPv6 Border Router
6LoWPAN	IPv6 over Low Power Personal Area Network
AH	Authentication Header
ARM	ARM® Technologies
BMS	Building Management System
BR	Border Router
CoAP	Constrained Application Protocol
COTS	Commercial off the Shelf
CSIR	Council for Scientific and Industrial Research
DTLS	Datagram Transport Layer Security
ESP	Encapsulated Security Payload
GHz	Giga-Hertz
GSM	Global System for Mobile Communications
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronic Engineers
IKE	Internet Key Exchange
IOT	Internet of Things
IP	Internet Protocol
IPSec	Internet Protocol Security
IPv6	Internet Protocol Version 6
IWSN	Industrial Wireless Sensor Networks
KB	Kilo Byte
KBps	Kilo Bytes per second
kbps	kilo bits per second
M2M	Machine to Machine
MAC	Medium Access layer
MHz	Mega-Hertz
NFC	Near Field Communication

PSK	Pre-shared key
REST	Representational State transfer
RFID	Radio Frequency Identification
TLS	Transport Layer Security
VPN	Virtual Private Network
WSN	Wireless Sensor Networks

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

1.1.1 Context of the problem

The current implementations of the Internet of Things rely very heavily on proprietary devices. These devices leave very little choice up to the individual implementing the application. These devices have a hidden architecture which enforces the inability of the implementor to make adjustments. The implementation of applications based on this closed source or proprietary device often means an implementor must make trade-offs between different available options. Proprietary devices are often also considered to be much more expensive than their counterparts and are designed to work with a company's devices and not to a standard. Making use of proprietary devices is however not the only option available. Additionally due to recent worldwide events individuals and industries are becoming increasingly focused on the issue of digital (or online) security. The low resource nature of the devices deployed within the Internet of Things means that a secure approach to data movement is generally difficult to achieve. Due to the nature of the Internet of Things, and the areas of possible application, security of these devices should be a primary research area when we consider any application of the associated technologies.

Open source technologies has seen a major interest boom over recent years. Open source software and hardware have become something that many people see as a method for the generation of reliable and capable devices. Instead of focusing on creating devices that are able to communicate with their own products, open source developers create their devices

to operate across a range of devices by ensuring they either meet or create standards for similar product development. These standards are then published and other individuals can expand, improve or freely utilize any sections or parts as they see fit (or according to a GNU licence published with the technology). Security technologies created through an open source approach are often considered to be more secure. This is due to the fact that anybody can read and improve these technologies, starting from an already partially implemented system. This also ensures that no mistaken or intentional back doors exist within the technology [1].

Open source technology has the potential to expand the Internet of Things to include cheaper, more powerful devices that are designed by the community for the community. These devices are free to expand, improve or utilise; however the developer of the Internet of Things application sees fit. This will allow for simple and cost effective Internet of Things applications that are able to be inexpensively upgraded and changed through the years. An open source approach has the potential to save industry and home developers both time and money. An industry developer that has similar requirements to a system already deployed could make minor changes and simply utilise the created system as is; saving a large amount of development and research time and money.

The device and community that is believed to be the ideal open source approach for the Internet of Things is the 'credit card sized' computer community. The most well known devices are the Raspberry Pi and the Beaglebone. These two devices are believed to be the perfect options for the creation of an open source implementation of the Internet of Things. In order to ensure the security of these devices an open source security toolkit that has seen wide adoption for the 'normal' Internet will be used. This is known as the OpenSSL toolkit and large companies such as Google, Facebook and many banks make use of this toolkit in order to provide security for their current Internet applications.

1.1.2 Research gap

The application of open source hardware and software to the Internet of Things, although it has been mentioned, has not been studied in detail. The creation of the Internet of Things is possibly going to create a massive paradigm shift and the technology that is going to be applied to this area needs to be thoroughly researched and understood. The capabilities of

these open source approaches to hardware and software needs to be examined and determined if they satisfactorily meet the stringent requirements demanded by the future Internet of Things and applications making use of this technology.

1.2 RESEARCH OBJECTIVE AND QUESTIONS

The research aims to provide clarity as to the capabilities of open source hardware and software when applied within the Internet of Things. The research will also aim to ensure that the technologies discussed will be capable of being deployed within an Internet of Things application. Through the research the following questions will be answered:

- Is it possible, within an Industrial Internet of Things, to replace proprietary hardware with open source alternatives?
- What open source alternatives exist that could be applied within the embedded section of an Internet of Things application?
- What are the known security risks, and methods used to prevent them, within the Internet of Things?
- Are existing security protocols available and capable of being deployed within the Internet of Things? As a replacement or alternative to the application specific protocols being developed for application within the Internet of Things?

Through this research each of these questions will be examined and explored in detail through experimentation and the creation of a blueprint for a possible Internet of Things application.

1.3 HYPOTHESIS AND APPROACH

Internet of Things applications will benefit greatly from the implementation of a non transport control protocol based security mechanism. This security mechanism will take the form of a protocol applied to data communication between a gateway device and a larger cloud based platform. The Raspberry Pi is capable of being deployed as this gateway device and is capable of achieving a satisfactory throughput of data with an advanced security protocol

such as Datagram Transport Layer Security. This protocol will ensure the safety of the communicated data and will allow for a complete, capable and robust implementation of an Internet of Things applications.

1.4 RESEARCH GOALS

The final goal of the research will be to provide a working example of a Raspberry Pi gateway powered Internet of Things implementation. The use of powerful open source technologies will also be validated and proven to be possible. The final research will provide for a blueprint of a possible Internet of Things architecture. The aim is to provide an inexpensive, powerful, secure and robust Internet of Things architecture that will allow for a developer to implement the technologies according to his requirements.

1.5 RESEARCH CONTRIBUTION

The main research contribution will be the validation of the possibility to deploy large scale open source projects that can outperform the more traditional and expensive proprietory implementations. A new option of a gateway device will be provided and the most appropriate security protocol will be investigated. With collaboration from the CSIR a commercial off the shelf Internet of Things node will also be provided and this approach to creating an embedded application will be explored. An investigation into the unique security requirements for the Internet of Things will be completed. This unique and revolutionary application domain will require the application of a different set of technologies in order to secure it.

1.6 RELATED PUBLICATIONS

The research found within this document originates from a number of publications done by the author of this dissertation [2], [3] and [4].

Two of the above mentioned related publications are conference proceedings. The third publication is a journal article published in 2015. The initial conference paper, which is now published within the book series *Cryptology and Information Security Series*, was done to determine the security capabilities of commonly used Internet of Things platforms, *Roy Fisher and Gerhard Hancke, "SSL usage in commercial Internet of Things platforms," in Volume*

11: Radio Frequency Identification System Security, ser. Cryptology and Information Security Series. IOS Press, 2013, vol. 11, pp. 69-82. Each individual platform went through a test confirming their capabilities with regards to expected SSL best practice as well as current SSL security concerns. Testing against current security concerns allowed for the estimation of how rapidly the developers of the platforms respond to security threats/issues that may affect their users. This allows us to see how much concern the developers show to current security trends that may affect their users. Section 3.5 and 4.2 are based on the work found in [3].

The second conference paper dealt specifically with the different capabilities of Datagram Transport Layer Security and Transport Layer Security for application within a streaming Internet of Things application. The research found interesting facts that under certain conditions, Datagram Transport Layer Security can outperform Transport Layer Security, *R. Fisher and G. Hancke, "DTLS for lightweight secure data streaming in the internet of things," in 3PGCIC 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 2014.* Section 3.7 and section 4.3.4 are based on the work found in [2].

The journal article is a comparison between open source solutions and the more traditional built for purpose options. During the course of this research a commercial off the shelf solution was also considered. The advantages of open source versus the other options were provided and a guide to implementing a COTS based industrial Internet of Things was provided. A large amount of related research has been provided within this paper specifically with regards to the individual capabilities of each device that may be included within a Internet of Things application, *R. Fisher, L. Ledwaba, G. Hancke, and C. Kruger, "Open hardware: A role to play in wireless sensor networks?" Sensors, vol. 15, no. 3, pp. 6818–6844, 2015. [Online]. Available: http://www.mdpi.com/1424-8220/15/3/6818.* Sections 3.6 - 3.7 and 4.3 - 4.4 are based on this work done in [4].

CHAPTER 2

LITERATURE STUDY

2.1 CHAPTER OBJECTIVES

The following chapter will discuss the current status of the research within the field of Internet of Things. The chapter aims to introduce the reader to the Internet of Things by explaining some of the key terms and requirements and ultimately allow for the reader to have a greater degree of understanding within the area in which this research has occured. The final sections within this chapter will begin to introduce the current technologies available within the Internet of Things paradigm, which will be expanded upon over the course of the next chapter.

2.2 OPEN SOURCE

A recent development within the hardware and software world is the idea of open source. Open source is the publishing of all related material to either hardware or software development online and making it freely available. For hardware devices, these documents will include all the circuit diagrams and development documents created during the development process. Open source software is a similar approach where all the code and documents are published online. An open source approach allows the individuals making use of the technology to use or change anything published online as they see fit. Sometimes this is to a specific licence requirement.

2.3 INTERNET OF THINGS

The Internet of Things is rapidly becoming a world wide phenomenon. The term was first used during the 1990's during an IBM presentation [5]. The Internet of Things is the paradigm used to describe the moment when more devices are communicating with each other than people across the Internet [6]. In order to achieve this paradigm everyday objects will require Internet access in order to send and receive data; thereby creating an Internet of Things. This paradigm will create a pervasive embedded network within the world in which people live and work [7]. This will allow for the physical world to be interacted with through the Internet using embedded sensors and actuators. The actuators will allow for actions to be carried out on the world and the sensors would allow for the world to be sensed and understood. The combination and interaction of the sensors and actuators will create the embedded intelligence that the developers can utilise.

Using the Internet of Things could potentially allow for a great range of applications. This ranges from building automation all the way to automatically managing the work force within an area or for a company [8]. Creating an application for building automation is becoming one of the favoured approaches to combating climate change. Wirelessly controlled lights and carefully controlled building automation will allow for a personalised and well controlled lighting and environmental control within a building [9]. Building management systems often rely on an assortment of different embedded systems operating together within the building [10]. Building management systems (BMS) have begun to be very widespread and allow for a greater degree of information regarding the individuals currently within the building and the systems with which the individuals are interacting. This is due to the focus of more and more people into managing environmental issues in a more controlled and managed manner.

Allowing for all devices in the world to connect both with humans and each other will create a completely new paradigm in which the Internet will exist. This new Internet paradigm will involve a world of interconnected networks allowing for machine to machine communication (M2M). The sensors and actuators deployed will enable this paradigm to interact and enhance the technological applications available for use within the current connected world. The total number of connected devices could exceed the number of humans on the planet, rough estimates place the total number of devices at around twenty-six billion [11] [12]. This could greatly affect not only the world in which humans live but also the cyber world in which

humans communicate and increasingly spend a large amount of their time.

A number of factors have led to the possibility of the creation of the Internet of Things. The major driver is the range of networking options that are available and the reduction in cost. The options that are currently available to developers are wide ranging. Devices involved have evolved to easily and cheaply connect to mobile and fixed data solutions having created an environment in which embedded Internet devices are a possibility[13]. Two of the major factors that are leading to the creation of an Internet of Things are:

- A range of devices of all sizes, capabilities and form factors
- Ultra Scalability

Additionally cloud based technologies provide for an additional option and personalization capability that was not previously available for applications within an Internet of Things application. Each of these technologies have been in separate development for many years and have matured to a point where they can be combined and used within an Internet of Things application. Mobile technologies such as 3G/4G will create a more mobile and capable Internet of Things. Using these mature technologies with/alongside a well managed networking implementation will allow for a robust and secure application of the Internet of Things.

The Internet of Things can be broadly broken down into five distinct layers. Each of these layers performs a task or set of tasks and they combine to provide an operational environment for Internet of Things applications [14]. The five layers are: perception layer, access layer, internet layer, service management layer and application layer. The perception layer's main function is to perform environment monitoring using sensor nodes and other sensing equipments. The access layer has the job of providing the perception layer with access to the higher up layers. The next higher up layer is the internet layer which is responsible for allowing the lower layers access to the larger Internet of Things network. This is where the traditional Internet happens within the Internet of Things. Real-time management and control over the vast amounts of data that will be created is provided by the service management layer. This is typically provided by the use of big data technologies. The final layer is the application layer. This layer integrates the function of underlying systems and allows for the creation of external applications that developers can make use of.

An important consideration when considering Internet of Things applications is that the technology is only the enabler for the application, the outcome of the Internet of Things of providing ambient intelligence is the main focus and the technology should ideally become part of the 'background' [15].

2.3.1 Internet of Things versus Internet as we know it

Many people believe that the Internet of Things is the logical evolution of the Internet as we currently see it today and in the passed. In the early days of the Internet single mainframe devices were used to communicate across the Internet. As time has passed and the technology has increased more and more devices are gaining the ability to communicate across the Internet. Initially it was personal computers, it has now extended to include; e-readers, tablets, cellphones, watches, and some vehicles. The Internet of Things will take objects not designed to communicate across the Internet (in the same was as a laptop or cellphone is) and enable them to communicate as freely as if they had been designed for only that purpose. This will mean that there will be a massive number of additional devices communicating across the Internet. Current solutions that are used on the Internet are not always capable of being applied to the Internet of Things alternative. As an example an Internet of Things enabled coffee machine will be able to warn the owner when his coffee pod supply runs low. Giving the coffee pods advanced communication capabilities is not sensible, and not practical due to the lack of a power supply, but allowing them to communicate with the owner through the more advanced coffee machine is more sensible; this is possible through the use of the new technologies that have been created. In an industrial environment a similar scenario can be envisioned involving monitoring employees within a factory making use of only their employee nametags. In many cases enabling an Internet of Things capability means working with the current resources that are available to the devices, in many cases this could be working with very limited resources, such as described previously.

2.4 INTERNET OF THINGS APPLICATION AREAS

The possible applications for the Internet of Things are very wide ranging and extremely varied. In the following sections a detailed description of some of these possible applications will be attempted. Due to the nature of the Internet of Things a list of the possible applic-

ations will never be exhausted. The uses, implementations and capabilities of the Internet of Things is constantly changing and updating as the technology matures and begins to get more widespread use and implementation.

In reality just about any application could be conceived for the Internet of Things and the technology could be relatively easily deployed within many fields and systems. This list found below is not exhaustive and covers only the major application areas.

2.4.1 Retail

A large number of approaches are available for retail applications that wish to make use of the Internet of Things technologies. Using the Internet of Things technologies and applying technologies to retail applications can be done in a number of methods. Real-time monitoring and control of supply chains and in store monitoring can be completed using technologies such as RFID and NFC to effectively track and monitor the supply chain [16]. Using these technologies, and the Internet, it is also possible to rapidly receive additional information for the client using NFC and the Internet [17].

The application of these could greatly increase the efficiency of retail applications. 'Smart' online shopping and delivery is one application that could have the greatest impact on individual people.

2.4.2 Smart Cities

One of the main areas of interest for the Internet of Things is within automation. Automation will be applied to many different areas: ranging from buildings all the way up to cities [8]. Municipalities are interested in automating the process of reading water and electricity meters, controlling the traffic in large cities and allowing for the digitization of entire areas through automation techniques and artificial intelligence. Smart cities are a current major research focus, due to the benefits that could be gained in terms of planning and city management.

2.4.3 Industrial Wireless Sensor Networks (IWSN)

The main focus of the application of the Internet of Things to an industrial environment is in improving the automation efficiency of the industrial system [16]. A generic implementation will involve the application of some form of identification into the product and environment around the factory, this is generally done making use of RFID tags. Using readers and other advanced technologies the system keeps track of the products and systems as they interact throughout the operation of the factory. Both sides of the industrial application will see a number of advantages through the application and use of this technology.

Many wireless sensor network industrial applications have already been deployed and tested. The automation of the Industrial landscape has been occuring for a number of years and the capabilities of local in-house sensor networks has been studied at length through the application and development of wireless sensor networks. Many of the worlds largest industrial companies have some form of wireless sensor network already deployed to manage their factories. By enabling these in-house wireless sensor networks to have Internet capabilities will allow for the creation of an industrial Internet of Things. Using one of the architectures to be discussed a powerful conversion to an Internet of Things is not an extremely complicated proposal to achieve [18] [19].

Using the Internet of Things in an industrial application will improve upon the data that is currently being created and collected [20]. It will allow for easier processing of the data, as the data will all be centralised and using powerful cloud platform techniques useful statistics and information can be acquired from this collected data. Unlike a wireless sensor network implementation, which historically is implemented with weaker devices, the Internet of Things uses a powerful central platform and a weaker set of embedded devices to enable some advanced features. This creates the opportunity to do more with the data that is received from the sensors and actuators allowing for a greater sense of intelligence in and around the users. This intelligence can be hugely beneficial to an industrial application where small savings can have a large impact over the lifetime of the project [21].

2.4.4 Healthcare

Healthcare is one of the major areas in which the Internet of Things technologies could have a great impact. This area of application is where the most noticeable impact could be seen [15]. The monitoring and managing of ill patients could assist in the healing of a great number of people. Remote doctors and nurses will improve the healthcare system greatly.

With the increasing age of the population within many countries the application of the Internet of Things to healthcare within a home or hospital could have a huge impact on many peoples lives [22]. The number of people aged over eighty years of age is expected to double within the next century, this is especially true within European populations. The use of technology within a home to monitor and encourage good health is a popular idea. Many wearable devices are beginning to arrive on the market with the ability to monitor heart rate and other key areas. Large corporations like Apple and Google are actively developing applications to aid the user in monitoring their health [23]. The best area of application will be within the 'smart home' of the not so distant future. Monitoring individuals while they sleep and move around their homes during normal day to day activity will provide medical professionals with a huge range of useful data. Using big data processing techniques to assist with diagnosis will allow for the early detection of many known diseases and may increase the chances of finding a cure or preventative measure [24]. This will also allow for medical professionals to better understand what effect certain activities have on the patient during a normal day.

These techniques and approaches are commonly known as e-healthcare. E-healthcare currently exists in three major approaches. Electronic personal record (EPR) is the act of keeping electronic records of patients, no active monitoring or sensing. The information is available to registered healthcare professionals and is assumed to be regularly updated. Current examples include the Discovery healthcare client application known as HealthID [25]. The next approach involves pro-active monitoring and alerting of current health problems. This approach allows doctors to receive a current issue alert if a patient experiences a heart attack or a similar medical emergency. The final approach is known as remote diagnosis. This allows for the remote monitoring, identification and in some cases treatment of patients from a remote location.

To enable better monitoring of people throughout the entire day e-health applications are becoming more mobile. This means they are aiming to be worn or carried with an individual during the course of the average day.

When creating implementations for this specific area of application for the Internet of Things a wide range of concerns need to be considered. In e-health an Internet of Things application needs to take a number of issues into consideration [15]; namely:

- Focus on the patient needs to be the primary concern
- Any electronic emmitance needs to be patient friendly. Radio frequency or any moving parts must be carefully controlled
- A number of approaches are available to provide care to a patient and all options should be carefully considered
- An e-health application is considered mission critical and reliability is a primary concern

2.5 INTERNET OF THINGS ARCHITECTURE APPROACHES

2.5.1 Border router based approach

A typical Internet of Things border router based approach is shown in image 2.1. A border router is a device that has been used to bridge the gap between the embedded devices and the larger Internet [26]. These devices have allowed for the connection between the more well known/mature wireless sensor networks and the larger Internet as a whole. To enable this type of communication the gateway devices must support multiple communication protocols. The devices also need to be able to translate between these protocols allowing for seamless communication across multiple network technologies. Often the communication technologies available for the border routing device is a combination of GSM (or other mobile technology) and WIFI or similar Intenet technology. These devices are one of the current methods of creating a Internet of Things application.

An actual deployment of an Internet of Things application may contain many sub gateways allowing for segregation between the lower level and sub networks that are formed in the

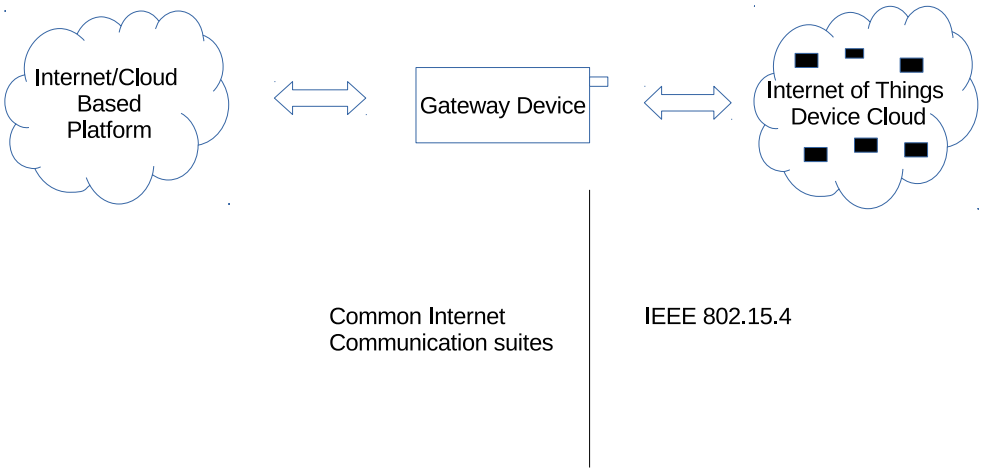

Figure 2.1: Typical Border Router Based Implementation

system. This allows for both larger and more powerful networks to be created [27] [28]. The network is then broken down using two levels of gateway devices. The top level device, being the only powerful device of this network, commonly known as the main gateway. This gateway device is primarily responsible for communicating data between the Internet and the in house sensor devices either directly or through other gateways. The inner layer of gateways are known as terminal gateways. These terminal gateways are the ones that will communicate directly with the embedded sensor devices.

Figure 2.1 shows a standard IoT implementation. The device cloud embedded within the environment performs the role of a typical wireless sensor network. This wireless sensor network has been significantly upgraded to enable advanced Internet capabilities. The upgraded wireless sensor network combined with the advanced cloud based platforms communicate through the use of powerful gateway devices that allow for standardized communication utilizing standard Internet technologies and sensor network technologies [29].

The combination of the wireless sensor network and the powerful platforms creates two separate networking zones. These individual networking zones, due to their nature, must utilize devices with a range of differing capabilities.

Internet of Things device cloud

The device cloud typically have two major functions: enable the sensing of the environment in which they are deployed and in some instances they allow for the interaction within this same environment (by making use of an actuator). This local network is a combination of devices that must have the following capabilities: must be self organising, allow for the determination of routes and failed nodes, allow for data protection and allow a range of these devices to be able to communicate with one another. Depending on the chosen protocol the devices need to be able to communicate with one another.

Gateway

Devices within the Internet of Things architecture make use of gateway devices in order to communicate. These gateways are typically responsible for translating the information between the embedded device cloud and the wider Internet devices [30]. A large number of options exist for the differing requirements of the gateway devices in the Internet of Things. These gateway devices typically spend most of their time performing simple routing of data across the Internet. They are also responsible for applying any advanced features to the data. The most important one being the application of advanced security features onto the data.

Cloud based platforms

The implementation of the cloud based platforms can be done in two separate ways. The servers constituting the back end devices can either be deployed in house or the developers decide to make use of a cloud based platform in order to more effectively manage these server devices [3]. Making use of the cloud allows the deployment to be more reliable and redundant. Cloud platforms world wide are gaining more acceptance with businesses and are becoming a more standard method to deploy hardware resources. A large body of research exists into the advantages and disadvantages of each method of deployment [31]. A range of platform options exist namely: Axeda, Bugswarm, Carriots, Thingspeak and XIvely. This aim of this piece of the architecture is to analyse and present data in a method that both the end user and other devices in the architecture can utilise.

2.5.2 Direct Internet connection based approach

This method of connecting the Internet of Things is slowly gaining ground but is currently limited by current device capabilities. Advanced protocols such as 6LoWPAN and CoAP have allowed for the embedded devices in the Internet of Things to gain the ability to communicate directly with the Internet at large [8]. This approach makes use of a multi protocol gateway similar in operation to current gateways that operate widely across the Internet.

When using this approach the gateway from the above implementation is simplified greatly as it is no longer required to translate between the common Internet protocols and the more common protocols found on a traditional wireless sensor network rather performing as a simple router. This gives the ability to have a less expensive gateway device with a generic implementation across multiple application types. The disadvantage of this approach is to possibly increase the cost of the embedded sensor devices. This is due to the need for these devices to support more advanced protocols which will require additional capabilities.

The increase in the connectivity options of these devices can be somewhat attributed to the new addressing standard that has been released for the Internet. This standard allows for a much larger number of addresses, 128 bit address instead of the standard 32 bit address, some estimates place the number of addresses available per square meter of the earth to at least 2000 whereas other sources have much larger estimates at 667×10^21 [32]. Whichever source is to be trusted the number of IP addresses available has dramatically grown, possibly allowing for everything to be connected to the Internet.

2.6 COMMUNICATION AND COMPUTATION TECHNOLOGIES

A number of technologies have lead to the explosion in interest for the Internet of Things. Some of these technologies have recently been created and some have been around for many years and are only being adapted and re-applied within this new area of application. Technologies such as WIFI and Bluetooth are finding new applications within this new application domain. This advancement in both low power and a significant decrease in size for the communicating parts has lead to a possibility of embedding sensors within everyday objects to allow for ambient intelligence [33]. This requirement will mean that many separately designed technologies will need to learn to work together and co-operate to ensure an efficient

and capable implementation of the Internet of Things application.

2.6.1 Communication Technologies

2.6.1.1 IEEE 802.15.4

The IEEE 802.15.4 protocol stack has been designed specifically for low power and low capability devices [34]. The IEEE 802.15.4 protocol specifies only the lower layers of the protocol namely: the physical layer and the MAC (medium access layer) layer [35]. The protocol stack allows for the upper layers to be specified by other protocols and thus allows for a great deal of flexibility and interconnectivity when implemented within an Internet of Things environment. The protocol allows for 250, 40 and 20 kbps communication rates; multiple addressing modes; automatic network establishment; power management and 16 channels in 2.4 GHz, 10 channels in 915 MHz and 1 in the 868 MHz band respectively.

2.6.1.2 Zigbee

In recent years Zigbee has seen possibly the most interest. Its ease of application, cost and capabilities have endeared it to both the hobby developer world and the industrial built for purpose world [36]. Zigbee is based on the IEEE 802.15.4 protocol. Zigbee as a protocol expands the 802.15.4 lower level protocol and adds the network and application layer to the network for complete communication between different applications running across the network. Many commercial implementors of both WSN and IoT projects are planning to use the Zigbee protocol stack within their applications [9]. It is well known for both it's reliability and capability. The protocol stack allows for many modes of operation and has native support for many networking and routing capabilities including the current wide ranging device support.

Without adding additional capabilities to the Zigbee and 802.15.4 protocol stacks it is currently not possible to connect it directly to the Internet. This is due to the fact that the Zigbee protocol is not interoperable with Internet Protocol (IP) based communication. Additionally the Zigbee protocol only supports communication speeds ranging from 20 kbps to 250 kbps, which is much slower than modern WiFi speeds (802.11ac could reach 1.3Gbps). The additional IP based addressing capabilities need to be added through additional protocol

support on board either the border routing device or directly on the sensor node. Research into this has been conducted and a working prototype does currently exist [37]. Although promising, these approaches are not considered to be the preferred method of implementing the system due to the limited testing that has been performed.

2.6.1.3 WirelessHART

Similar in operation to the Zigbee Standard defined above, WirelessHART is a protocol suite developed to operate above the IEEE 802.15.4 protocol [38]. This protocol has been developed with a number of industrial partners and has recently been accepted and drafted as an IEEE standard.

2.6.1.4 RFID

Radio Frequency Identification is one of the powerful technologies that will require the application of a gateway device to provide for the translation between the embedded network and the wider Internet devices.

Radio Frequency Identification is a mature technology that has been around for many years. It is seeing an increase in use in the Internet of Things and is becoming one of the preferred methods in which to tag and monitor the movement of equipment or supplies through the application [39]. RFID tags can be seen as the single technology that could enable the wide adoption of the Internet of Things within people's daily and work lives. The diverse capabilities that this technology allows mean that it can be employed in almost any application, and provide a meaningful use [40]. Embedding RFID sensors within objects and placing RFID readers in the correct location could allow for the Internet of Things to become a realised system of application. RFID has applications in the commercial, industrial, private and even public domain. It is a cheap, powerful and very versatile technology that can be used to provide a constant flow of information to the implementors of any Internet of Things applications [14] [41].

RFID is a technology that provides communication capabilities within a number of frequency bands. The term RFID often covers a large cross-section of available technologies and is used to represent a group that focuses on using radio frequencies to communicate and interact.

The identification part of the name is the capability of the devices to be tagged and then passively monitored as they interact with the world in which they are deployed. RFID's ease of use has led it to being considered as a backbone technology for the Internet of Things. This means that the technology has become a major part of any Internet of Things applications. As an example in a commercial application the individual items could be tagged, allowing for a complete record of intransit locations and environmental conditions. The containing box could be tagged allowing for a box level information. Finally a group of boxes on a pallet could be tagged allowing for easy monitoring [39].

2.6.1.5 6LoWPAN

6LoWPAN (IPv6 over Low Power Personal Area Networks) is the protocol that will allow for the connection of the individual sensor nodes to the Internet. Up until recently (last two years) the online world has been operating on IPv4. In fact results from Google show that people accessing their servers over IPv6 are as low as 7% of their total customer base [42]. Recently IPv6 has become the new standard for addressing in an Internet environment. IPv6 is a relatively new technology that will allow for a huge number of extra addresses to be made available. IPv6 expands the 32 bits available for Internet addresses and instead uses 128 bits. Dramatically increasing the number of addresses available by an order of approximately $8 \times 10^2 8$ [43]. IPv6 has been designed to be used with traditional Internet devices whereas the Internet of Things is designed to be implemented by low power and resource defficient devices.

6LoWPAN is ideally supposed to be the protocol that addresses this complexity and optimized IPv6 for operation within the Internet of Things. The main focus of 6LoWPAN is to compress the header of the packets and thereby allow for a more compact packet which is easier to handle for the low resource devices and still fully support the IPv6 operation. The protocol also specifies specifics when dealing with routing problems that arise due to the embedded nature of the Internet of Thigns. The routing will typically occur across technologies and through different localised geographical locations. Using these technologies will allow for the inclusion of all the sensors into a single network thereby creating uniquely identifiable embedded objects and a truely universal Internet of Things [44]. This reshifting of the design will allow for a more homogenous access across applications and services [45].

2.6.1.6 Mature technologies

The two main technologies that make up the mature section of the Internet of Things are WIFI and Bluetooth. Both of these technologies are widely implemented and both have been used extensively previously in many application areas. These two technologies are also being used within the Internet of Things. WIFI is not as popular as 802.15.4 based protocols because of the lack of low power support, with the protocol designed to provide the optimal communication for as long as possible, thereby maxing out the power consumption. Bluetooth is also getting more application because of the possibility of using the technology in conjunction with an individuals cellphone. Allowing for direct simple communication with a large number of people within an area as most people have a bluetooth enabled phone.

2.6.2 Computation Technologies

One other section of the technologies that require a brief discussion is the software that enables the overall applications. Over the years a number of operating systems have begun to see a greater use within the Internet of Things. Some operating systems have even been created specifically for the Internet of Things. The three most well known are: Contiki OS, TinyOS and one of the Linux flavours of operating systems.

2.6.2.1 Contiki OS

Contiki is designed to be an event driven operating system. It has basic multithreading support and is deigned to keep the base system lightweight and compact [46]. Contiki is implemented using the C programming language and is capable of running on devices with extremely low resources. The entire operating system requires less than 2KB of RAM and can run on a extremely low speed processor typically in the order of a few megahertz, but it is also capable of running more powerful standard systems. The operating system even has a power saving mode built in to allow for a greater amount of power to be conserved when operational requirements are low for the embedded devices. The operating system is completely open source and has gained a large following of developers and designers and a highly active community.

2.6.2.2 TinyOS

TinyOS is an advanced operating system that operates similarly to Contiki OS. TinyOS is an open source event driven operating system currently being widely used within embedded designs. The operating system is well maintained and the community is very active in terms of development and upgrades. It is important to note that unlike the other operating systems discussed, TinyOS is not programmed using the C or C++ language. The operating system uses its own operating system langugage which is nesC. NesC is known as network embedded systems C which is optimised for event-driven operations. NesC is an extension to the C programming language.

2.6.2.3 Linux

Linux is a more heavyweight operating system. It has been deployed with limited success within the Internet of Things. The operating system has been the model used for the development of the other Internet of Things operating systems. It is also the operating system used by many of the servers, gateways and more high power devices within the Internet of Things. There are many Linux variants available and some have been written in order to the be utilized within low resource systems, this could allow them to be used within the Internet of Things.

2.6.2.4 Riot

Riot OS is a relatively new operating system. It is a community designed and developed open source operating system. It has native C and C++ support thereby allowing for easy and rapid development of powerful applications. The operating systen is designed to have a low requirement for resources and requires less than 1.5kB of RAM and less than 5kB of ROM [47]. Riot is not an event driven operating system like Contiki or TinyOS. This means that it is possibly more suited for application on the Internet of Things due to its ability to operate in a more standard manner when performing networking tasks.

2.7 CONCERNS

A number of concerns have been raised regarding the possible implementations of the Internet of Things within both the social and legal aspects of the application. A number of open research areas still exist and a huge effort is being done to try and address these issues.

2.7.1 Privacy

Inserting embedded sensors within the world in which people live and work will create a huge amount of data about the interaction of people within the environment around them [48]. Privacy has become a deep concern for many people globally, including the recent revelations by Edward Snowden relating to the United States government spying and data collection [49]. These revelations have made the general public very aware of the privacy implications and using the public Internet can make them vulnerable. This large amount of data raises two main areas in which the collection of data can affect the privacy of individuals [50].

Some of the Internet of Things solutions have proposed allowing the user to decide on what information is collected about himself.

2.7.1.1 Consented collection

Consented collection is where the individual about which the data is being collected is aware and has consented to the collection of this information. This consent could be physically received in terms of a license or terms of use agreement [48]. The consent could also be implied. This implication could be received when the user deploys the application within their own environment in which they live. By deploying the application the owner is implying consent for data collection [6].

2.7.1.2 Non-consented collection

The major concern is the collection of data of individuals without their consent. A person entering into an area with a deployed Internet of Things application who is unaware of its existence would be an example of non-consented collection. This could potentially have a very negative effect on the widespread application of the Internet of Things [48]. More and

more people are becoming aware of the data that they generate online and the manner in which this information is used.

2.7.2 Security

The huge number of devices that will be connected through the Internet of Things will require for the implementation of a capable security system [48]. Due to the implementation reality of the Internet of Things the devices will be deployed and left alone for long periods of time, which could greatly increase the risk related to the Internet of Things applications [16].

Many attacks exist that could affect the Internet of Things some of which are described below:

- Eavesdropping - wireless communication allows for an attacker to attempt to collect some of the data that is being communicated
- Man-in-the-middle - attacking device intercepts and retransmits the data after inspecting what is sent
- Denial of Service - an attacker prevents use of the resource by continually flooding the network disallowing access to the device
- Traffic analysis - analysing the data traffic movements to determine weak points within the network

The sheer number of security breaches currently being experienced across the spectrum of Internet applications is a cause for concern for a paradigm that relies so much on the current design of the Internet to provide core functionality [51]. A Internet of Things application that is not security conscious could have dramatic and dangerous effects on the users of this technology. The damage done could range simply from personal information being stolen, or to a patient not receiving, or receiving the wrong, medical treatment in a connected hospital example. The design of the security of these systems needs to take a security centric approach to these systems.

The application of security to the Internet of Things could have a massive effect on the

application deployment and requirements that are implied [52] [53] [54]. One key concern for the Internet of Things is the underlying security protocol that will be chosen. The security protocol must allow for the protection of the data but must not make the entire application unuseable by individuals. Typically the application of security to a system will negatively impact on the features and the ease of use of a system [55]. This is one of the key areas that need to be explored within the realm of the Internet of Things and the fact that the entire system is designed to provide functionality for users must be remembered.

CHAPTER 3

METHODS

3.1 CHAPTER OVERVIEW

This chapter will cover the possible approaches available for the implementation of a secure Internet of Things. It will introduce a wide range of options that are available for the approaches of securing all the applications. The final implementation plan will be discussed towards the end of the chapter. This will detail a complete overview of the design of the system that will be implemented.

3.2 SECURING COMMUNICATION

To facilitate the adoption of the Internet of Things by individuals, companies and other entities, one of the core design features must ensure that all the information sent across it must be secure. From the research two distinct areas of consideration arise within implementations of the Internet of Things. The embedded network and the more public connection to the Internet will both have specific requirements that must be met to ensure that the communication is secure [56].

Currently a range of options exist for securing the communication within Internet of Things applications, with each of the network sections assuming their own approaches to ensuring secure data transmission. As an example the embedded network approach is mainly to make use of the security features provided through the IEEE 802.15.4 protocol suite.

Using the current technologies the focus of the research is based on the gateway devices that

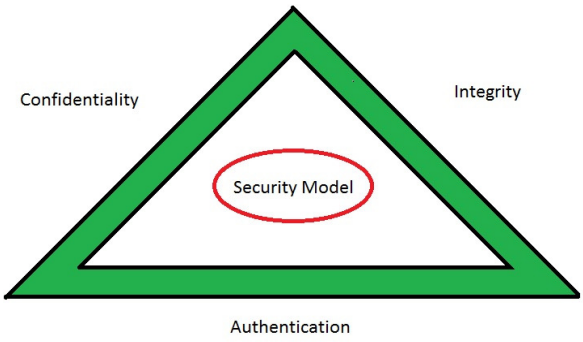

Figure 3.1: CIA Triad

enable the communication between the wider Internet and the embedded sensor networks that are deployed. The IEEE 802.15.4 protocol has built in security measures that ensure the data sent is secure [34]. This leaves us with a smaller (although geographically much larger) more focused area of the application design in which to focus our research. We ensure that the embedded devices available are able to communicate across a range of the current protocols and design a gateway device that can provide secure communication to the platforms. In order to complete the security review the individual platforms are examined in order to assess their security capabilities.

It is widely popularised that the connection of WSN to the wider Internet will be hampered by the capabilities of the embedded devices that are used. These devices are generally expensive and have low power capabilities. It is believed that the use of a powerful, inexpensive device based on the ARM architecture will be able and capable of being applied within an Internet of Things application [56].

3.2.1 Secure Communication Requirements

Typically a application is considered to be secure if it implements three primitive security requirements, namely: confidentiality, integrity and authentication. These three requirements are known as the CIA triad and allow for a communications implementer to be certain that a level of security has been applied to his communication [57]. For a secure solution all three of the triad branches need to be included and operate in unison.

3.2.1.1 Confidentiality

Confidentiality is the ability of the communication to remain hidden from people possibly trying to view it. A number of methods are available that will allow for the application of confidentiality to communication. The main method of providing confidentiality is with the use of encryption technologies which make use of cryptographic protocols and functions. User IDs and passwords are the current most used procedure for confidentiality, however other options do exist. With the increase in the use of technology and the greater application of security to applications many other forms of encryption types have been used. These types could use "biometric verification" (such as device fingerprinting) and security tokens or smart cards [58].

3.2.1.2 Integrity

Integrity is the ability of the communication to maintain consistency and accuracy. This in turn ensures the trustworthiness of the data over its entire lifestyle. Some of the most commonly used integrity mechanisms are secure hash sums, functions or by using message authentication codes. These functions create a unique signature of the contents of the data [59].

3.2.1.3 Authentication

Authentication is the ability of the communication to ensure that the communication occurs between the correct individuals [5] [60]. The methods most commonly used to provide authentication typically involves the use of public and private certificates. Public and private key encryption could also be used to provide for the authentication of the communicating parties. It is important to note that a certificate will often contain a key; but it will also provide other information about the key. For example; the issuer, what is the aim of use, validity dates as well as providing other meta-data specifically involving its contained key. A key on the other hand is the piece of information used to perform the cryptographic or security functions.

Transport Mode	*Tunnel Mode*
Only the data payload is encrypted and authenticated. The destination/client device and IP header information can be seen by all devices through which the packet passes.	The entire packet is encrypted and authenticated. Mini-tunnels are created between devices and each device can only see the next device within the communication stream. All IP header information is protected and a new header is added with the next hop address only.

Table 3.1: Modes of IPSec Operation

3.2.2 Internet Protocol Security (IPSec)

Internet Protocol Security is one of the end-to-end security mechanisms currently available for implementation within the Internet of Things. IPSec has seen wide application within virtual private networks (VPNs). IPSec is a suite of protocols for the implementation of secure end-to-end transmission. The Authentication Header (AH), Encapsulated Security Payload (ESP) and Internet Key Exchange (IKE) are the three protocols that allow for secure communication [54]. Similar to the Transport Layer Security Handshake protocol defined in the next section; IKE manages the establishment of the secure keys and secure transfer of protocol specific information (version information, cryptographic protocol support etc) between the client and server. This process leads to the creation of a Security Association between the two communicating devices, this includes keys and other important information (cryptographic protocols supported etc). This Security Association information is then used by the ESP and AH protocols to provide for secure communication between the two devices. The AH protocol provides for authentication between communicating devices and the ESP provides for confidentiality.

IPSec has two primary methods of operation: Tunnel Mode and Transport Mode.

3.2.3 Transport Layer Security (TLS)

Transport Layer Security also known as Secure Socket Layer security is a protocol developed to provide confidentiality, integrity and authentication to the communication that is

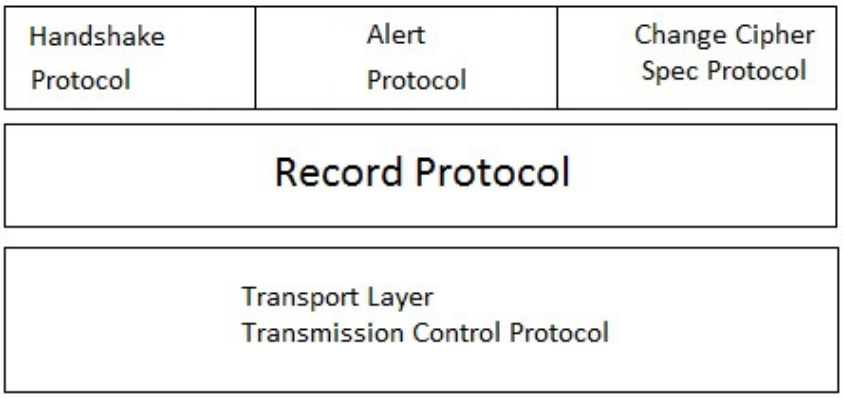

Figure 3.2: TLS/SSL Protocols

used.

3.2.3.1 TLS operation

The operation of TLS/SSL is best described by looking at the protocols that it consists of, which can be seen in the image below. The figure 3.2 is a piece of the open systems interconnect model that describes the communication of data.

As can be seen from figure 3.2 the over-arching TLS protocol is actually made out of four sub protocols. These are:

- Handshake protocol
- Record Protocol
- Alert protocol
- Change Cipher Spec Protocol

It is important to note the TLS/SSL protocol runs on top of the reliable Transmission Control Protocol (TCP). This ensures reliable end to end communication of data. All four of these protocols combine together to create the powerful and widely used TLS/SSL protocol. This protocol has become the industry standard for the secure communication of data across the Internet. Many websites make use of this technology to protect and secure the data that is

Protocol	Description
Handshake	The major protocol used to establish the intial requirements for all the communication that will follow. The handshake stage involves a large amount of setup work. A detailed picture of the process can be seen in figure 3.3. The client initiates the communication. During the hello phase of the communication certificates are swapped and the requirements for the creation of the sessions symmetric keys are exhanged. During this phase the authentication of the server, and optionally the client, is carried out. The pre-master secret is securely communicated through the use of the servers public key.
Alert	This protocol is used mainly for the implementation of event notification. Any session errors that occur between the client and the server during the course of communication are communicated making use of this protocol.
Change Cipher Spec	Any changes that are made to any of the cryptographic requirements are communicated by making use of this protocol.
Record	As can be seen from figure 3.2 the other three protocols are implemented to run above the *record* protocol. This is the protocol that ensures confidentiality and integrity to all the communications transmitted between the client and server. It accepts communication from the above protocols and other sources and applies encryption and integrity checking mechanisms (secure hashing, message authentication codes) to provide for a secure and reliable communication stream.

Table 3.2: Table discussing the four TLS/SSL sub-protocols

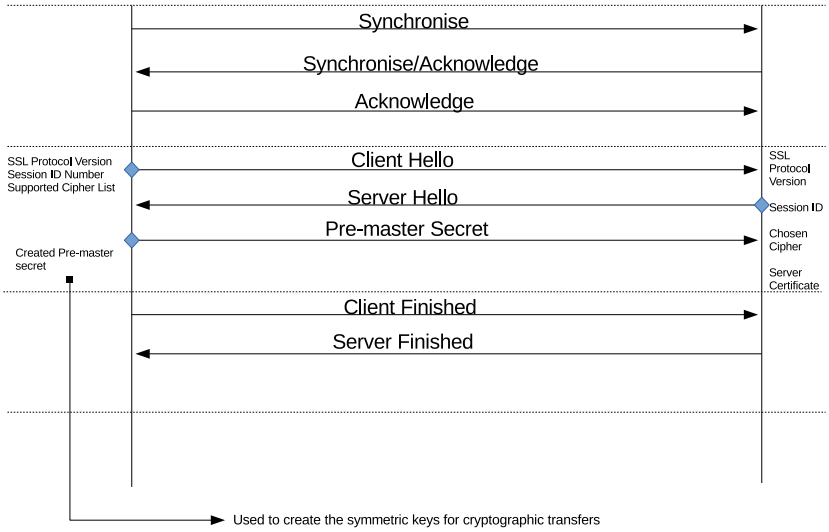

Figure 3.3: TLS/SSL - Handshake Protocol [57]

transmitted and received. Although it has seen wide use within the current Internet, it is not widely seen as a protocol suited to being used within the Internet of Things. The devices and volume of data force the use of a less resource intensive means of securing the data.

3.2.4 Datagram Transport Layer Security (DTLS)

Over the years it was noticed that TLS/SSL was not well suited for certain applications. In applications that require a large amount of data to be sent and received rapidly it was highly evident. A good example of this would be online gaming. This is where the impact of TLS was most pronounced. A solution to this problem was created by a group of Stanford researchers and the result was known as Datagram Transport Layer Security (DTLS) [61]. As the name suggests the protocol is the implementation of TLS/SSL but instead of basing it on the reliable TCP, the protocol suite instead operates on the unreliable UDP (user datagram protocol) [2]. This prevents retransmission of lost data which in turn allows for the improvement of the throughput speed of the data. It is worth noting that the unreliable nature of UDP may negatively affect the transmission, particularly if the percentage of data lost is unacceptably high. DTLS provides for the three security primitives as seen in the security triad figure 3.1 in the same way as TLS/SSL.

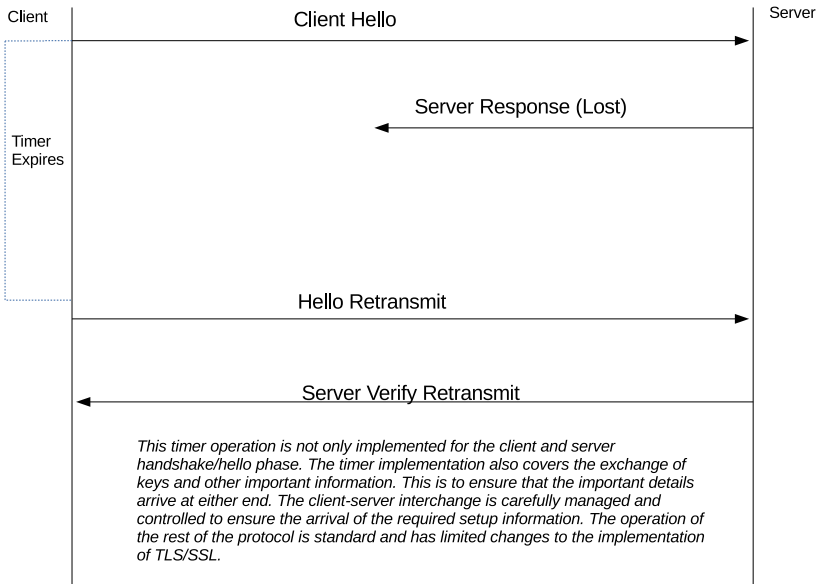

Figure 3.4: DTLS Timer Implementation

The practical operation of DTLS is very similar to TLS as the protocol is designed to mimick it perfectly. This means that the DTLS protocol implements the four sub-protocols of TLS known as:

- Handshake protocol
- Record protocol
- Change Cipher Spec protocol
- Alert protocol

Figure 3.4 shows the method by which certain important information (cryptographic keys, certificates) are reliably transmitted across the unreliable UDP stream. It must be noted that although some data retransmission occurs, ideally it will only occur during the setup phase and not the data transmission phase. This will ensure fast throughput and reliability for the data to be transmitted. One other important feature is the inclusion of a sequence number in the packets transmitted during the handshake phase. This ensures that the packets that arrive out of order, or that are forcibly retransmitted out of order (in an attack scenario), can be re-ordered or ignored depending on requirements.

It is believed that this protocol, although not created for the Internet of Things, is a possible solution to the problem of transmitting data securely without negatively affecting the operation of the system.

Currently it is believed that traditional security architectures (IPSec, DTLS and TLS) are not suited for application within the Internet of Things without some form of adjustment [54] [53].

3.3 OTHER CONCERNS

Two major approaches are available for the implementation of the Internet of Things. The first approach is built for purpose and the second approach is to make use of open source hardware or software all the while utilizing a commercial off the shelf approach (COTS). All these approaches are available for either the gateway devices or for the embedded devices within the application.

3.3.1 Built for purpose

A built for purpose approach makes use of manufacturer created devices that have been designed to work together to create a Internet of Things application. These devices make use of proprietary hardware or software and thereby limit the amount of customization that the customer can apply. These systems also sometimes make use of proprietary protocols that prevent the application from randomly allowing other devices to connect.

3.3.2 Commercial off the shelf

The next approach is to make use of a commercial off the shelf (COTS) device. COTS devices make use of open source software and hardware designs to create a solid base for the user to develop their code. This approach creates solid base that follows the open source guidelines allowing for any device following these guidelines to be used in conjunction with these approaches.

3.3.2.1 Raspberry Pi

The Raspberry Pi is a credit card sized computer that has been designed for teaching applications [62]. The powerful capabilities of the Raspbian operating system allows for the possible application of the Raspberry Pi to the Internet of Things. This device is believed by me to be a possible device to replace the gateway devices used within the Internet of Things. The advanced capabilities of the operating system will allow for the implementation of the advanced requirements of the Internet of Things. The key reason for this approach is the price of the Raspberry Pi, costing only $35 compared to similar gateway devices costing in the region of $400.

3.4 OVERALL SYSTEM DESIGN

The design that I chose was to create a new gateway device making use of the advanced capabilities of the Raspberry Pi [63]. The gateway will be based on the Raspberry Pi and will make use of the DTLS protocol in order to provide secure communication between the gateway device and the server collecting the data. The embedded network is created using embedded devices that support the 802.15.4 protocol. The gateway must be able to translate the communication between the embedded network and the wider Internet of Things.

In the following sections the tests used to confirm the operation of the created system are described.

3.5 PLATFORMS

The capabilities of the Internet of Things platforms are only limited by the resources of the individual companies due to their deployment in a cloud fashion. In theory each platform is limitless due to this design choice. In reality however the platform is limited to the total resources of the company. For the implementation of our system the important features that need to be tested are the overall security implementations of each platform.

A range of both commercial and open-source platform options are available for the implementation of the centralized data storage and processing facilities [64]. A number of the large corporations have deployed in house wireless sensor networks that make use of these plat-

forms [65]. Making use of a pre-deployed cloud platform allows the developers to be certain that the application is elastic enough to manage a huge amount of data.

3.5.1 Platform security

As has already been discussed security is a major concern for applications aiming to make use of Internet of Things technologies, this is especially relevant within industrial implementations[66]. In order to ensure that the security of the entire application is adequate each individual section will need to be tested. The security implementations of both the embedded networks and the devices communicating across the Internet must be adequate. In order to test the security of the platforms a standard test is used that confirms a number of basic security requirements have been met. The current industry standard for secure communication is to use HTTPS (SSL)[67] , and it is therefore not surprising that it is used extensively in IoT platforms. The security test used was created by Qualys SSL labs and tests the SSL capabilities of each of the servers[1]. In order to adequately test a cloud based implementation the address that is used to receive data from the devices was used as the point of contact. This ensures that the correct SSL/TLS implementation is tested. By using the Qualy SSL Labs tools a ranking can be assigned to each of the servers. The tool tests the platform strength in four important areas: key exchange, certificate validity, protocol support and cipher suite strength. The test produces a complete view of the current implementation of the security procedures for each of the platforms tested. The full discussion on how the tool calculates the results is available in the SSL guide [68]. The main aim of performing these tests was to determine the current state of platform security that is available for use within the Internet of Things applications.

3.5.1.1 Certificate

The certificate is rated according to either being a valid or invalid certificate. An invalid certificate results in a zero score being awarded and an automatic fail for the server. Any of the reasons below will force a certificate to be considered invalid:

- Domain name mismatch

[1]Found at https://www.ssllabs.com/ssltest/index.html

- Certificate not yet valid
- Certificate Expired
- Use of a self signed certificate
- Use of an untrusted certificate authority (CA)
- Use of a revoked certificate

Any of these errors result in a website achieving a zero result for the test and being assigned a F final mark. In all of the communication it is important to ensure that any private key is kept safe and secure and all public keys are incorporated into a certificate and signed by a relevant certificate authority.

3.5.1.2 Protocol support

In calculating the score of the protocol support section; each protocol is assigned a score out of 100 based on known weaknesses and strengths. The scores for the worst supported and the best supported are added together and an average is calculated. Copied below in Table 3.3 detailing the scoring guide:

As can be seen from table 3.3 the SSL protocols are generally considered to be the weakest and the TLS protocols are considered to be the strongest. These rankings are generally aligned with the number of security flaws that the protocol is plagued by. The percentages are based on the current best practices for a secure SSL implementation on the Internet. These best practices are based on current trends and weaknesses shown in security research on SSL communication.

3.5.1.3 Key exchange

The key exchange scoring guide is shown in table 3.3. Servers which fall prey to known exploits are given a zero ranking otherwise the length of the servers private key determines the strength of the cryptography. The key exchange strengths are scored against the length of the key. In SSL communication during the handshake phase the longer the key the harder

Table 3.3: Protocol Support Rating, Key Exchange Rating Guide and Cipher Strength Rating by SSL Labs [68]

Protocol	Score
SSL 2.0	20%
SSL 3.0	80%
TLS 1.0	90%
TLS 1.1	95%
TLS 1.2	100%

Key Exchange Aspect	Score
Weak Key (OpenSSL flaw)	0%
Anonymous key exchange	0%
Key Length < 512 bits	20%
Key Length < 1024 bits	40%
Exportable Key Exchange	40%
Key Length < 2048 bits	80%
Key Length < 4096 bits	90%
Key Length > 4096 bits	100%

Cipher Strength	Score
0 bits (no encryption)	0%
< 182 bits	20%
< 256 bits	80%
>= 256 bits	100%

for an outside party to decode the initial phases of communication between a client and a server. Initially this key is used in SSL as the encryption key used to communicate the session encryption key between client and server.

3.5.1.4 Cipher suite strength

The cipher strength is calculated in the same way as the protocol support score. The strongest supported score is added to the weakest supported score and divided by 2. The scores for the cipher suite are based on the cipher length and can be seen in table 3.3. These percentage scores are again based on best practices in industry implementations.

3.5.2 Additional security concerns

Below some of the prominent security concerns related to SSL can be seen. These have been published about but are not covered in the scores given above.

3.5.2.1 RC4

The results obtained from the security scans that have been completed also highlight valid concerns with RC4 based encryption schemes. RC4 has for many years been considered to be less secure than other related encryption, this is mainly due to its use in WPA/TKIP encryption for WIFI connections. The initial concerns involving the RC4 algorithm were concerned with the manner in which RC4 was employed within the overall encryption structure. New more recent attacks instead target the RC4 algorithm and affect a much larger percentage of data communication [69]. The recent attacks are capable of decrypting any RC4 based encrypted data. The attack can only be prevented by completely disallowing the use of RC4 based encryption schemes. There are a number of countermeasures that can be employed to assist in mitigating the damage the attack is capable of inflicting.

3.5.2.2 Current security concerns

As a means to measure the response of the platform developers to current security concerns, three current (recently discovered) security problems are tested against the SSL implementation currently implemented for each of the respective platforms. If the platform developers are security conscious the three security threats should have already been patched. This gives an indication for a developer as to how rapidly the platform maintainers will respond to known security issues.

The CRIME attack exploits the TLS compression to inject predictable data and gain access to the information by using that data [70]. Disabling support for compression is a method to ensure that the CRIME attack is successfully prevented. The BEAST attack exploits a weakness in client side SSL setup. The attack is capable of decrypting encoded information stored within secure cookies and allowing a third party to view the information [71].

The Lucky Thirteen attack is an exploit released in 2013 by the Information Security Group at the Royal Holloway University. The attack is a new method of the Oracle padding attack which was previously believed to have been fixed. New versions of open source security toolkits, such as OpenSSL, have been updated and now successfully mitigate this attack.

Newer security concerns such as the Heartbleed bug for OpenSSL have also been discovered and patched by the developers of the toolkits used to provide the SSL/TLS protocols. It is assumed that the developers of large platforms have set their devices to receive automatic updates and are therefore immune to security risks associated with the base toolkit [72].

3.6 GATEWAY EXPERIMENTS

The next important component within an Internet of Things application is the gateway device. The main responsibility of this device is ensuring that communication between the embedded devices and the wider Internet (in our example the platform) is possible and secure. This leaves two important characteristics that need to be tested for the gateway device, namely the performance capabilities and the current security architectures that are available. Up until now the focus of research has been on creating separate security protocols for use within the Internet of Things (due to the low resource nature of the devices involved), it is the authors belief that such limitations are no longer applicable with the new generation of devices [73]. These new devices have higher processing capabilities due to the new advances within microcontroller design and architectures. The tests have been designed to compare the capabilites of these new devices against the more traditional devices designed specifically for use within the Internet of Things.

When considering the performance capabilities of gateway devices in an industrial implementation the most important characteristic is communication performance. To prevent data bottlenecks and slow transmission when measuring performance three key areas must be considered: reliability, throughput and response time [28].

Reliability measures the confidence in a transmitted packet being received by the cloud platform. For a gateway in an industrial application high reliability is required. High reliability in this sense means an ability to provide accurate and real-time information [74]. This enables the application implementer to be sure that transmitted data is received by the cloud platform. High reliability will ensure that data transfer rates are not negatively affected by retransmission of data across a low resource network. Typically networks based on the 802.15.4 protocol stack do not implement transmission control protocol [2]. This means that reliable transmission is not guaranteed by the communication protocol. A high reliability will allow for a greater degree of confidence that transmitted data has reached the destination.

Reliability must also take into account the order in which the packets are received. In a time sensitive implementation, that has limited resources, any additional processing must be avoided [75].

Throughput is a measure of the amount of data that the system can process per second. The gateway implementation will need to take data from one communication technology and retransmit on the other end. This will typically involve receiving data from a 6LoWPAN network and re-transmitting it on a normal IPv6 or IPv4 network. The higher the throughput of the device the better for the overall implementation of the application. Another measure of the throughput is the latency of the device. Latency can also be called response time. This is a measure of the time it takes for another device to respond. A low latency is preffered as this means that the gateway is operating more efficiently.

Due to the nature of the information that will be transferred across the Internet by an Internet of Things application the additional process of adding security to the communication needs to be benchmarked [76]. The three major requirements for the Internet of Things security is confidentiality, integrity and authentication [59][77]. These three requirements are known as the primitive security objectives. The processing requirements of these primitive objectives are extensive and will be a good benchmarking tool for the devices but these devices will also be required to implement these objectives in communication in order to secure the communication [78]. Confidentiality is the ability to hide the data that is transmitted. Integrity is the ability to ensure that the data that is transmitted is the same as the data that is received. Authentication is used to ensure that the individuals communicating are the people that are expected to be part of the communication. Using all three of these objectives allows us to ensure data security when communicating.

To complete the benchmarking of the possible devices two approaches could be used: built in toolkits for benchmarking; or the creation of scripts that perform the required benchmarking.

3.6.1 Open source implementation

The Raspberry Pi has often been labeled as a ‘hobby’ device [63]; this leaves questions as to whether or not this device is capable of operating as a gateway or in any time/data

sensitive application. Due to the increasing community interest in the Raspberry Pi the device has undergone a large amount of testing and benchmarking already. Another device to consider for implementation is the Beaglebone. This device has been benchmarked but is not considered as an easy alternative to the Raspberry Pi. The focus of the paper is to supply a current state of the art for implementation of a Internet of Things application. The Beaglebone, although being a very capable device, has not seen the interest and development from third party individuals that the Raspberry Pi has enjoyed. The Beaglebone has a less well supported operating system known as Angstrom Linux. Although the operating system on both devices can be changed, the paper looks at the default design of the devices and the Raspberry Pi's Raspbian operating system has more support and software packages due to its Debian heritage.

During the initial benchmarking it was shown that the Beaglebone and the Raspberry Pi perform on similar levels. However due to the fact that it is slightly easier to expand the capabilities of the Raspberry Pi, it was decided that the Raspberry Pi would be used in the testing.

3.6.2 Proprietary implementation

The proprietary implementation of the gateway devices involves the use of the Digi Connectport devices acting as the gateway for the embedded Internet of Things sensors to the wider internet. The two devices that are tested are the Connectport X2 and the Connectport X4. Both of these devices have LAN and Zigbee networking capabilities. Unlike the Raspberry Pi these devices have a very limited set of customization options available. They both run the Digi implemented operating software that has a specific focus for operating requirements [79] [80].

3.6.3 Experimental details

A set of experiments were conducted that showed the capabilities of the Raspberry Pi with regards to the requirements for an open source gateway device namely, reliability, throughput and security. The aim of these experiments was to determine whether or not the Raspberry Pi is a device capable of acting as a gateway for the Internet of Things in an industrial application. Tests were done to show each of the requirements and measure the Raspberry

Pi's capabilities with regards to these requirements. Each of the tests will be discussed in the sections below. Due to the interest in the Raspberry Pi many benchmarking tests have already been completed. These benchmarking tests (specifically the OpenSSL tests) use the built-in benchmarking capabilities provided within the OpenSSL development kit. Although being a good performance comparison between the different devices available, it does not provide a real world comparison of the devices. The benchmarking tools allow for uniform data to be worked upon in a structured and controlled loop only utilising high speed sections of onboard hardware (RAM). This is the ideal operating environment for benchmarking but does not provide a real world view of operational capabilities. A real world implementation involves more operational calls to external input output devices and interrupts from other devices. This is why when completing the implementation these external features where simulated within the code allowing for a more realistic view of the real world operational capabilities of the device.

It is important to note that in order for secure communication to be implemented a certification authority was created in-house. This was done in order to ensure that both the client and the server devices had access to useable and secure certificates in order to facilitate the required communication. This certification authority, although not being verified externally, will be more than sufficient if a developer implements an in-house application server.

3.6.3.1 Reliability test

The first set of tests were done to confirm the reliability of the Raspberry Pi. Reliability testing was done using a widely known network benchmarking tool. This is used to ensure that the packets sent and received are reliably delivered. It should be noted that our definition of reliability is somewhat limited. We have decided to only focus on the successful delivery of packets between the receiver and transmitter. This was to provide us with a solid starting point for comparison against the two security protocols tested later in the section. Although being a limited approach it provides us with a good basis for testing the security protocols. The reliability could be affected by nodes moving in or out of the network range as well as other possible faults that could arise. The test consisted of a laptop connecting to the Raspberry Pi via the laptops WiFi and a USB WiFi receiver on the Raspberry Pi. The Raspberry Pi hosted a WiFi access point and was also connected to a desktop computer via

the onboard ethernet capabilities. Each of these separate networks was setup in a subnet. The Raspberry Pi had IP forwarding enabled switching the packets between the ethernet and WiFi networks. Making use of the Iperf benchmarking suite the performance was measured [81]. When measuring reliability of wireless communication, distance of the transmitter from the receiver can play an important role.The tests were run at three distance intervals, with the laptop moving away from the Raspberry Pi.

The Connectport devices are designed for reliability as they are created to be deployed in a industrial environment. The Connectport Devices have a similar setup structured for the reliability testing. Due to the wired nature of their communication distance testing was not required.

3.6.3.2 Throughput tests

The throughput testing was completed using two separate tests. The raw data bandwidth was measured using the Iperf benchmarking suite for data of different sizes. Additionally the Iperf benchmarking suite has the capability to simulate the communication of up to one hundred devices. A test was run that simulated the communication of a large number of devices across the WiFi channel to the Raspberry Pi and onto the server connected via ethernet. This simulation was run three times and the average results was calculated. The latency of the communication was also measured by pinging the laptop and desktop computer through the Raspberry Pi.

Secure throughput tests - Raspberry Pi command line

A Python script was created that could call the required OpenSSL command line tool. The Python script built the required command and then called the command using the built in set of OpenSSL command line functions on the Raspberry Pi. The command was executed on a file that was greater than 500MB and the time to complete the required security objective was measured. The authentication of the system is not included in the results below as the times were consistently less than one second. This leads to unreliable results, so it is assumed that a key is used that is encrypted with the message as authentication. These tests were run a number of times and the average throughput calculated. The tests were completed using a block size of 8196 bytes. These tests assumed that the entire file has the required security

objective applied and is then transmitted as an entire package by some underlying protocol for example a TCP based unencrypted communication. The underlying protocol will then break the file into packets as required. Once received the entire file could be decrypted using a preshared key. For the integrity tests the secure communication of the signature of the file was assumed to exist. The raw processing and throughput speeds are more important in terms of benchmarking the Raspberry Pi. A simple solution is to transmit this information as part of the file in the first 16 bytes of the message. By doing this no major additional processing is required, it will simply be the attachment of x bytes onto the file. The aim of these tests is to determine the processing and communication time required to protect the communicated data for each of the primitive security requirements.

Secure throughput tests - Raspberry Pi Python wrapper

This implementation broke up the file into equal sized packets (1024 bytes). Each packet then had a combination of integrity, confidentiality or authentication applied to it and was then transmitted. These tests were done to better simulate the size of data that would be received from the in-house network as this is generally small packets from a large number of devices. The aim was to show the added overhead when dealing with the small size data packets that are typically going to be received and transmitted by the Raspberry Pi. The M2Crypto Python wrapper was used to interface with the OpenSSL C functions [82]. An additional advantage of using a trusted and established wrapper for secure communication allowed us to confirm our results.

The next set of tests were completed using the best performing cipher suite and message digest to provide confidentiality, integrity and authentication. A system was then implemented that hashed, signed and finally encrypted each 1024 byte size message and transmitted it along the network. The message digest used was MD5 as this has proven to be a fast hashing algorithm. The signing algorithm was RSA based and encryption was implemented using the AES 256 CBC algorithm.

Secure throughput tests - proprietary implementation

Similar testing was completed for the the Digi devices. The testing allowed for secure sockets layer based communication of data (built from the data collected from the embedded network)

between the Digi device and a suitable stand alone computer.

3.6.4 Datagram Transport Layer secure communication results

Once thorough testing of the gateway device making use of Transport Layer Security was completed testing of the Datagram Transport Layer Security protocol was completed. In order to provide a concise comparison of the results DTLS was compared to TLS through a series of tests. Using a similar approach to the tests already completed, the Raspberry Pi implemented the DTLS or TLS protocol to provide security to the communication, the protocols were compared against transmitted packet size and the throughput and number of packets lost/received and packets sent was compared. The test simulated the Raspberry Pi operating as a border routing device. As an additional comparison the setup was also compared against the IPSec protocol in order to provide a complete overview of the capabilities of the device.

The secure streaming test was constructed using a TLS or DTLS client and corresponding server; written in the C programming language using the OpenSSL security toolkit [83]. The server was hosted on a high power desktop machine. Two sets of tests were run to determine the throughput capability of the protocols and the Raspberry Pi's on a local network. The initial test altered the packet size of the data portion of the TLS and DTLS Record protocol. The test was conducted for ten seconds in which the total number of packets sent and received was recorded.

3.7 NODES

Many different approaches have arisen to creating an Internet of Things application. Some of the current range of embedded enabling devices for the Internet of Things device cloud are introduced below [84]. The most important capabilities of the nodes of the networks are the support communication protocols, power requirements and general input and output capabilities. Other important features such as the supported operating systems for the devices are secondary concerns, but do still remain important to consider when a developer decides to implement an Internet of Things application.

It was noticed that the divide between open source and closed source seems to be that the

newer devices follow an open source approach whereas the older versions follow a closed source approach. The Waspmote, Lotus, Sprouts, Firefly and TelosB are open source. The Micaz and Imote2 are closed source devices. A few of the devices mentioned as open source are not completely open source due to external problems and the inability to use the advantages of being an open source device. A good example of this is the TelosB device, the ability to advance or add attachments to the device is limited because of the design. So although being an open source device, due to external complications it does not gain the advantages of other open source devices and is therefore very similar to a closed source device.

3.7.1 Available nodes

Each of the sections below will provide some background to some of the currently available devices that can be used in an Internet of Things application.

3.7.1.1 Waspmote Pro

The Waspmote Pro is a new open source, wireless sensing node from cooking hacks by Libelium, improving on their initial Waspmote node [85]. The Waspmote Pro can be used in various industries such as gas and events monitoring, smart metering, agriculture and radiation detection owing to a number of sensor boards available for use with the node. This node falls under our built for purpose devices as it has been designed for a specific task and contains specific sensors and equipment.

3.7.1.2 Lotus

Lotus is an advance WSN node built around the ARM Cortex M3, 32-bit processor [86]. It is fitted with a 802.15.4 compliant on-board radio associated with the Zigbee protocol. The Lotus is well suited for applications requiring Acoustic, Video, Vibration and Other High Speed Sensor Data, Condition Based Maintenance, Industrial Monitoring and Analysis and Seismic and Vibration Monitoring. Another node that falls under our built for purpose category.

3.7.1.3 Sprouts

The Sprouts node is a developing node from Queens University [87]. Developed as part of industry related research, the node has attracted attention in the oil and gas mining, steel production and power grid monitoring sectors as an event monitoring node. This is one of the nodes that falls under our open source options with many expansion capabilities.

3.7.1.4 Firefly 2.2

Developed at Carnegie Mellon University, the Firefly 2.2 is a real time sensor networking node designed for use in industrial control and automation, smart home monitoring, inventory and personnel tracking and hazardous environment monitoring as well as general embedded system education [88]. This is one of the nodes that falls under our open source options with many expansion capabilities.

3.7.1.5 Older options

The newer options that have been discussed above are not the only options that are available. The other options to be discussed below are: TelosB, Micaz and Imote2. There are a number of other options that could be discussed and a complete list of node options are available [84].

Some other examples are WeC, Rene, Dot Mica, Spec, Cricket, EyesIFX, Tmote Sky, Shimmer, Stargate, Sun SPOT, IRIS, NetBridge, BTNode, V-Link and a few others [84]. Many of these examples are closed source. These devices were all created before 2009 and can be traced back to the designs used within the TelosB, Micaz or Imote2. We have therefore decided to make use of these three devices to use as a representative of the older devices that are available. The oldest versions are well represented by the TelosB and Micaz (two of the first iterations of the Internet of Things) and through the years the upgrades can all be represented through the Imote2. These decisions were made after carefully studying the components used to create the different iterations of all these available devices.

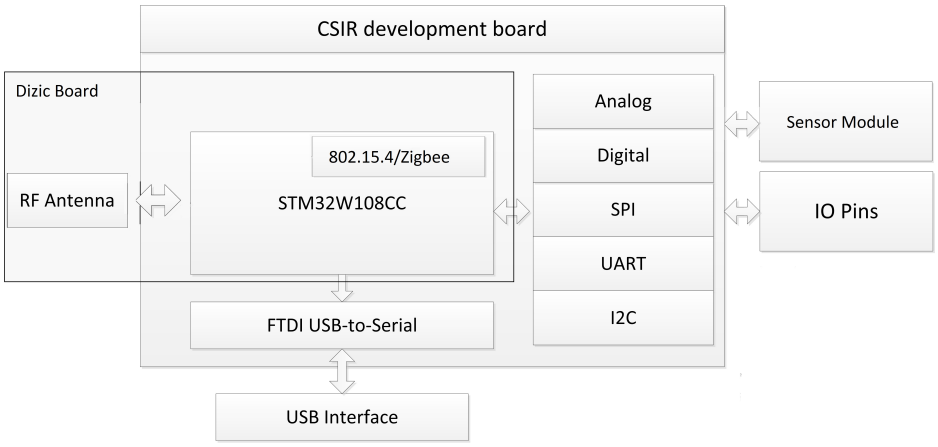

Figure 3.5: The CSIR Internet of Things Node Functional Diagram

3.7.1.6 Rapidly prototyping new devices

Alternative options to the ones listed do exist. One of the major options that we have is to create our own device by making use of COTS components [89]. This allows us to personalise our devices to perfectly suit our requirements. The advantage gained from being able to create a node that specifically addresses your requirements must be done with the other devices in mind. The huge range of technologies used to create the Internet of Things often means that, within a single application, a large range of technologies are used. For example the 802.15.4 protocol is not the only protocol being implemented within this section of the architecture. Industrial production lines using the more mature WiFi protocol also exist [90]. WiFi technology is based on the 802.11 protocol and has been in use in both residential and industrial applications for many years. The 802.11 protocol, although capable of being implemented within the device cloud section of the Internet of Things, was originally designed for implementation within devices with high resource allowance (power, throughput and large memory space) [91]. The standard has evolved and the current implementation, 802.11ac, focuses on high data throughput and multiple access [92], both are less important for the Internet of Things device cloud where a large number of devices communicate irregularly. Using a device such as the Raspberry Pi and a plugin XBEE or WirelessHart device will enable a powerful range of capabilities for an embedded sensor. These are a few of the considerations that must be taken into account when developing a node.

Creation of your own node requires only a basic understanding of electronics, communication

interfaces and embedded programming. To illustrate the feasibility of rapidly prototyping a self-built solution, thus we created a simple node to test alongside the nodes discussed above. The node that was prototyped is based around a DiZiC module [2], with a simple 'break-out' board and additional communication and power circuitry. The node has full communication capabilities based on the IEEE 802.15.4/Zigbee protocol stacks. The developed nodes functional diagram is given in figure 3.5. This shows the simple design of the implemented node and the range of communication options available to the node. The main chip was chosen to ensure a wide range of compatibility with as many of the available Internet of Things operating systems as possible. The other components were chosen to ensure that the device remained as cost effective as possible but could still perform the required operations. An advantage of making use of such a design is to ensure that the device meets your individual requirements and is able to meet the goals of your application but remain easy to both replace and upgrade. This ensures that the entire application remains robust, cheap and flexible, all of which ensure a future proof design. This node that we have created is capable of performing all the functions that the other devices can achieve, when compared against a number of the other available options, and at a much lower cost even when considering we were ordering small quantities.

[2] http://dizic.com/

CHAPTER 4

RESULTS

4.1 CHAPTER OVERVIEW

This section will detail all the results from the tests that were completed. These tests have been described in the previous section and the overall implementation of the Internet of Things application will be tested near the end of the chapter. The focus of the tests are to ensure the following:

- Security - at the minimum meeting the three primitive security objectives of **C**onfidentiality, **I**ntegrity and **A**uthentication.
- Capability - ensuring the devices are capable of providing the required service
  - Throughput - able to handle the data
  - Reliability - available and able to send and receive the data

A brief discussion of the results can be found in the chapter that follows. A more comprehensive overview discussion of the results can be found in the next chapter. A discussion on the tests can be found in the previous chapter.

4.2 PLATFORMS

The results for the SSL capabilities tests can be found in this section. These tests were designed to test the SSL implementations currently deployed within platforms on the Internet

of Things. The certificates used within this test were provided by the platform servers. The client devices were not authenticated by the server throughout the tests.

4.2.1 Platform test results

4.2.1.1 Arkessa

Arkessa is a commercial Internet of Things platform. They provide services to many large companies and therefore the security of their platform should be well maintained. The results of the Arkessa SSL test can be found in Table 4.1.

Table 4.1: Arkessa, Axeda and Bugswarm SSL Results

Arkessa		Axeda		Bugswarm	
Overall Result	*C*	*Overall Result*	*B*	*Overall Result*	*F*
Certificate	100	Certificate	100	Certificate	0
Protocol Support	85	Protocol Support	85	Protocol Support	85
Key Exchange	40	Key Exchange	80	Key Exchange	90
Cipher Suite	60	Cipher Suite	90	Cipher Suite	90

The overall rating for Arkessa was capped to a C value because of the low marks scored in both key exchange and cipher strength. The low marks for the cipher suite are scored because of the support for very weak cipher algorithms. The platform offers support for a range of cipher algorithms, however, within this list is support for four algorithms with relatively short key lengths (56 and down) these are considered to be weak cipher types. The platform preferred cipher algorithms are 128 bit and above. This platform successfully mitigates both the BEAST and CRIME attacks. The platform supports RC4 based encryption algorithms. Steps need to be taken to ensure that recent confirmed attacks for RC4 based encryption do not adversely affect the secure communication.

4.2.1.2 Axeda

Axeda is another commercial Internet of Things platform. The company provides an Internet of Things service to many large companies. On their website they claim to currently support

over a million connected devices [93]. Unfortunately due to the fact that the platform does not have the option for a free account; the only platform that could be tested was the developers platform, however this does provide for a good example of the level of SSL security that the developers of the platform could deploy.

As can be seen from Table 4.1 the overall result for Axeda is relatively good. Although having a relatively high set of scores the grade is capped to a B because the platform does not mitigate two possible attacks. The two attacks are the CRIME attack and the BEAST attack and have been described above. The platform supports RC4 based encryption algorithms. Steps need to be taken to ensure that recent confirmed attacks for RC4 based encryption do not adversely affect the secure communication. The preferred method of prevention is to disable support for RC4 based communication.

4.2.1.3 Bugswarm

Bugswarm offers options for both paid and free users. The free users have a number of restrictions, for example limiting the number of devices that can be connected.

Table 4.1 shows that the platform for Bugswarm has completely failed the test. The platform fails the test because the certificate that it supplies does not match with the web address of the platform. The platform reported that the certificate was for "buglabs". This certificate had expired and thus needs to be rectified by the company for their clients. The certificate chain of trust is not complete and therefore not a trusted certificate. This untrusted certificate makes the communication vulnerable to man-in-the-middle attacks (MITM) [68].

The rest of the SSL requirements are met and this resulted in the cipher suites, protocol support and key exchange all scoring good marks. This platform also does not mitigate the BEAST attack, however it successfully mitigates the CRIME attack by not supporting compression based communication. The platform supports RC4 based encryption algorithms. Steps need to be taken to ensure that recent confirmed attacks for RC4 based encryption do not adversely affect the secure communication.

4.2.1.4 Carriots

Carriots is similar to Bugswarm in that they offer an initial free startup option allowing ten devices to be connected. Once the ten devices have been used a pay-as-you use principle applies.

Table 4.2: Carriots, Xively and Evrythng SSL Results

Carriots		Xively		Evrythng	
Overall Result	*B*	*Overall Result*	*A*	*Overall Result*	*B*
Certificate	100	Certificate	100	Certificate	100
Protocol Support	90	Protocol Support	90	Protocol Support	85
Key Exchange	80	Key Exchange	80	Key Exchange	90
Cipher Suite	90	Cipher Suite	90	Cipher Suite	60

Although the platform achieved high scores in all the sections, the final grade is capped to a B because of the platforms inability to mitigate the BEAST attack. This platform successfully mitigates the CRIME attack as compression is not supported. The platform is vulnerable to RC4 based exploits as it supports RC4 based encryption algorithms. Steps need to be taken to ensure that recent confirmed attacks for RC4 based encryption do not adversely affect the secure communication.

4.2.1.5 COSM/Pachube/Xively

COSM[1] (previously known as Pachube) is one of the original open source Internet of Things Platforms. COSM is one of the most well known and widely used Internet of Things platforms by open source developers.

As can be seen from Table 4.2 the security implementation of the platform scores very high and the platform achieves a grade A rating. Although the platform scores the same scores as the Carriots platform, by prioritizing TLS 1,2 and RC4 encryption for other protocol versions the platform effectively mitigates the BEAST attack. By disabling support for compression the platform also successfully mitigates the CRIME attack. The platform supports RC4

[1]During the writing of this report COSM changed its name again to Xively

based encryption algorithms. As explained above encryption based on the RC4 algorithm is widely considered to be insecure.

4.2.1.6 Evrythng

Evrythng is another commercially available option with current industry customers. Just like Carriots, Everythng offers free developer accounts. This allows possible consumers to test their product before they decide to implement it.

A number of security issues are prevalent in this platform. The main issue is the support for a weak cryptographic algorithm. The algorithm in question is the use of DES, which has a key length of 56 bits. Another issue that the platform presents is allowing support for client-initiated renegotiation. Although not compromising the security of the information being sent, this opens the platform up to a Denial-of-Service (DoS) attack. This platform does not mitigate the BEAST attack but does successfully mitigate the CRIME attack by disabling support for TLS data compression. The platform supports RC4 based encryption methods and is therefore also vulnerable to the recently discovered RC4 exploits.

4.2.1.7 Exosite

Exosite offers similar solutions to Carriots. Having a pay as you use principle as well as a small developer account which is free for small implementations.

Table 4.3: Exosite, Grovestreams and iDigiTab SSL Results

Exosite		Grovestreams		iDigiTab	
Overall Result	*F*	*Overall Result*	*B*	*Overall Result*	*A*
Certificate	100	Certificate	100	Certificate	100
Protocol Support	85	Protocol Support	85	Protocol Support	85
Key Exchange	80	Key Exchange	80	Key Exchange	90
Cipher Suite	60	Cipher Suite	90	Cipher Suite	90

The platform receives a grade of F because it allows for insecure client renegotiation. This allows for man-in-the-middle attacks. Other security issues with the platform are the support for two weak cipher suites both of which have a strength of 56 bits. The platform also

supports many other cipher suites with a higher strength. This platform does not mitigate the BEAST attack. The platform supports RC4 based encryption methods and is therefore also vulnerable to the recently discovered RC4 exploits. The platform successfully mitigates the CRIME attack by disabling support for compression.

4.2.1.8 Grovestreams

Grovestreams is a platform that is still under development. The current release is only scheduled as a beta version. They are allowing completely free accounts to any individual that registers on their site. The accounts are free for as long as the platform is a beta version.

The platform scored well in all sections and supports two protocol types. The platform supports both SSL 3.0 and TLS 1.0; neither of these standards are known to be one hundred percent secure. The platform favours the stronger cipher suites (anything over 128 bit strength). The platform does not mitigate the BEAST attack but does successfully mitigate the CRIME attack by disabling support for compression. The platform is also vulnerable to RC4 based exploits as support for RC4 based encryption is still supported.

4.2.1.9 iDigi/Device Cloud by Etherios

Etherios offer a free account allowing the connection of five devices. This is similar to other commercial Internet of Things platforms.

This platform scores well (as seen in table 4.3) in all of the categories and is therefore awarded with a high grade symbol. The platform does not support a large number of cipher suites, however, all of the supported suites have high strength enhancing the overall security. The platform successfully mitigates both the CRIME and BEAST attacks, however it is vulnerable to RC4 exploits as RC4 based encryption is still supported.

4.2.1.10 SensorCloud

This platform like many others offers both free and paid accounts. The free accounts have a few restrictions placed on them; including the amount of data that is allowed to be transmitted

through the platform as well as the number of connected devices allowed to interact through the platform.

Table 4.4: Sensorcloud, Thingspeak and Yaler SSL Results

Sensorcloud		Thingspeak		Yaler	
Overall Result	*B*	*Overall Result*	*B*	*Overall Result*	*C*
Certificate	100	Certificate	100	Certificate	100
Protocol Support	85	Protocol Support	85	Protocol Support	85
Key Exchange	90	Key Exchange	80	Key Exchange	40
Cipher Suite	90	Cipher Suite	90	Cipher Suite	60

The platform, although scoring very high marks which should have placed it within an A grade, has instead been graded as a B because of a number of key security flaws. The platform supports client initiated renegotiation which would place it in danger of a Denial of Service (DoS) attack. The platform also does not mitigate the BEAST attack but does mitigate the CRIME attack. The platform supports RC4 based encryption methods and is therefore also vulnerable to the recently discovered RC4 exploits.

4.2.1.11 Thingspeak

Thingspeak is an open-source platform that is free to use. The platform is designed to allow any user to connect as many devices as they want.

The platform supports a good collection of both cipher suites and protocol versions. The platform however allows for compression and is therefore vulnerable to the CRIME attack. The platform does not mitigate the BEAST attack. The platform does not support RC4 based encryption methods and is therefore not vulnerable to the recently discovered RC4 exploits.

4.2.1.12 Yaler

Yaler is slightly different to other Internet of Things platforms. Not only does it allow a user to send data from an embedded device to the platform, but also allows the platform to act as a relay, allowing external users to access the embedded device through the platform; even

if the device is behind a firewall or similar security feature.

The platform has a number of security issues. The main security concern is the strength of the public key used during the handshake phase of SSL communication. The second concern is the large number of weak security ciphers that are supported. Even ciphers that have weaknesses within their algorithms are supported. The platform supports client initiated renegotiation opening up the possibility of a DoS attack on the platform. The platform does not mitigate the BEAST attack but does successfully mitigate the CRIME attack. The platform supports RC4 based encryption methods and is therefore also vulnerable to the recently discovered RC4 exploits.

4.2.2 Device management

Not only is it important to protect the information that the platform is receiving, it is also important to ensure that the data that is being received by the platform originates from a legitimate device/source [94]. Many of the platforms use a single API master key that gets sent from the device during any HTTP/S based request. Typically this master key is a long string of characters. Although allowing for a degree of security this does not specify device level access and allows any device or individual that obtains the key to send updates to the platform. Another more recent approach, adopted by many of the platforms, is to first register each device with the platform and grant each device a specific key. The devices are also associated with a single stream and do not have the capability to update another information stream. As an example a RFID tag reader will only be able to update information concerning the tags that it has read and not information about the lighting conditions within the factory. This type of cross information stream updating was possible with the older single key approach. Although the new method requires a bit more user administration, the security and control benefits gained far outweigh the extra administration requirements. Rogue devices can be singled out and the information received from them can be terminated remotely.

4.3 GATEWAY

This section will detail the results from the tests described in the previous chapter concerning the devices operating in the gateway role for the operation of the Internet of Things. In all of

Distance	Result
<1 meter	0% packets dropped
5 meters	0%packets dropped and 1 arrived out of order
>10 meters	0% packets dropped

Table 4.5: Table summarizing results collected from numerous Reliability tests with a Raspberry Pi

the below tests a certification authority was created that issued both the client devices and the server devices with a public and private key pair. All private keys are stored locally and the public keys were incorporated into a certificate to be used in the communication. These certificates and keys are securely distributed to the devices either during setup, via a USB, or some other secure method.

4.3.1 Reliability tests

The results for the test can be viewed in table 4.5. As can be seen from these results the Raspberry Pi is capable of achieving reliable data transmission at distances up to 10 meters with very few packets either dropped or arriving out of order. Each of the test

The proprietary X2 and X4 devices have reliability and response capabilities expected of a networking device in an implementation. The devices achieve a 0% packet drop rate and a very low response time averaging $< 10ms$.

4.3.2 Comparison of throughput results against individual security objective

The results for the raw (unsecure) data throughput tests can be seen in table 4.6. This data shows that the larger the information being sent through the Raspberry Pi the more throughput is being achieved, the increase is due to the larger data sizes being able to stream at maximum speed for longer periods due to all the communication requiring setup time. The data also shows that the Raspberry Pi does not have a large effect on the data flow as the communication is limited by the WIFI standard used namely 802.11g which achieves a maximum of 22 Mbps throughput [95].

Data Size	Result
15 MB (megabytes)	11.5 Mbps (Megabits per second)
18.2 MB	15.2 Mbps
60 MB	21.5 Mbps
200 MB	22.2 Mbps

Table 4.6: Table summarizing results from Throughput test with a Raspberry Pi

For the Iperf communication of multiple devices simulation, the Raspberry Pi was capable of achieving an average throughput for 100 communicating devices of 15.8 Mbps.

The ping testing was repeated 10 times; each test involved continuously performing pings for 20 seconds and using the calculated values. The average time to receive a response was 80.011 ms. The minimum response time was 12.669 ms and the maximum response time was 205.143 ms. This large swing in response time could potentially negatively affect the performance of communication through the Raspberry Pi. The large swing could be due to the single core processor that is included with the Raspberry Pi. The response to one of the pings is delayed because the processor is performing some other set of tasks (other processor interrupts) before it can return to the task of responding to the request.

These results show that very little overhead is introduced through the use of SSL on the Raspberry Pi for the Internet of Things.

4.3.2.1 Secure throughput results - Raspberry Pi command line

The results for the implementation applying only confidentiality are shown in table 4.7. The data throughput using the recommended encryption cipher AES with a 256 bit key achieves 3.20268 MB/s. The two fastest performing algorithms are Blowfish and AES both achieving greater than 3 MB/s throughput. Tests were also completed with regards to the cipher chaining method. The best performing chaining method was ECB (electronic code book) this was expected because of the simplicity of the algorithm. The next best performing chaining method was CBC. AES 256 using CBC achieved a throughput of 3.279 MB/s whereas the ECB achieved a throughput of 3.421976 MB/s.

Confidentiality Algorithm	Throughput (MB/s)
AES 128 bit key	3.34744
AES 192 bit key	3.318837
AES 256 bit key	3.20268
Blowfish	3.317878
DES	2.590775
Triple DES	1.458663
RC4	3.58629

Table 4.7: Table summarizing results from Python and Command line implementation of Communication with Confidentiality

One important consideration is RC4, it would appear that by adding I/O operations the throughput of this algorithm is greatly reduced. The operation of RC4 is being delayed due to the I/O operations that need to be performed. Whereas without the file I/O the RC4 stream cipher could rapidly process the data due to its stream nature complementing the continuous loop cycle of the benchmark test. It is now being hampered by the low speed of the I/O operations but the cipher is still able to perform at the fastest rate.

The results in table 4.8 show the results achieved when adding integrity to the communication with the OpenSSL command line tool. As can be seen the results for just integrity far outperform the throughput of adding confidentiality. The best performing hashing function was MD4 achieving a throughput of 28.82474 MB/s. These results are achieved by implementing the hash on the complete file and then transmitting the hash value along with the file.

Figure 4.1 shows the results for the implementation of both integrity and confidentiality on the data sent from the Raspberry Pi. These results show that a data throughput rate of 2.962641 MB/s is achievable using the AES 128 encryption with MD5. Similar speeds can be achieved using AES 128 with MD4 and RC4 with SHA1. The more secure implementation of AES 256 with MD4 achieves 2.852727 MB/s. These values show that by adding I/O operations the speed of the system is slowed drastically. It is important to note that in these tests, authentication of the communication is handled by the knowledge of the pre-shared key.

Integrity Algorithm	Throughput (MB/s)
MD4	28.82474
MD5	24.85157
ripemd160	15.347
SHA	19.09429
SHA1	21.68956
SHA 224	15.2319
SHA 512	8.411452
Whirlpool	1.566124

Table 4.8: Table summarizing results from Python and Command line implementation of Communication with Integrity

Confidentiality Algorithm	Throughput (MB/s)
AES 256 CBC	2.221412771
Blowfish CBC	2.211554
DES CBC	1.84385782
Triple DES CBC	1.138862268
RC4	2.696817074

Table 4.9: Table summarizing results from Python OpenSSL wrapper implementation of Communication with Confidentiality

4.3.2.2 Secure throughput results - Raspberry Pi python wrapper

The first set of results were for the implementation of the described system passing messages of size 1024 bytes and encrypting, then sending the data, this can be found in table 4.9. As can be seen the throughput achieved is very similar to the throughput achieved for the previous implementation using the command line. Only CBC mode was tested because from the previous experiments it became clear that this was the best performing mode. It is clear from these results that AES and Blowfish again achieve the fastest throughput. This is expected and confirms the results found regarding the fastest cipher [96].

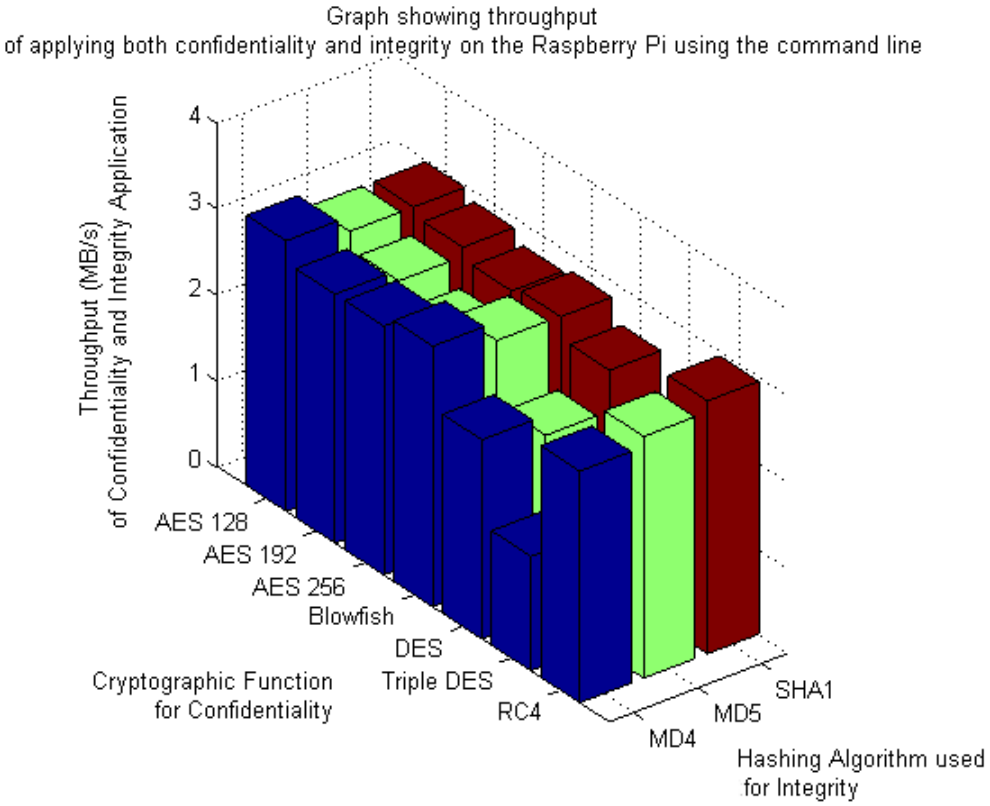

Figure 4.1: Throughput results when applying both Confidentiality and Integrity for Command Line application

The results of applying all of the primitive security objectives using the results from the previous tests are that the throughput capability of applying all of the primitive security objectives was 1.5778 MB/s.

Tests were not implemented to confirm the capabilities of signing and hashing each protocol as there is no point in applying each primitive security objective separately in a comparison test. One more test was completed. This test measured the throughput of the Raspberry Pi when implementing confidentiality and integrity on the device. This test was done to mimic the operation using a static key for authentication. This test achieved a throughput of 1.764865 MB/s. This shows that the process of signing the information does not account for a very large percentage of the processing power of the processor. This confirms the results that where achieved in the first benchmarking results.

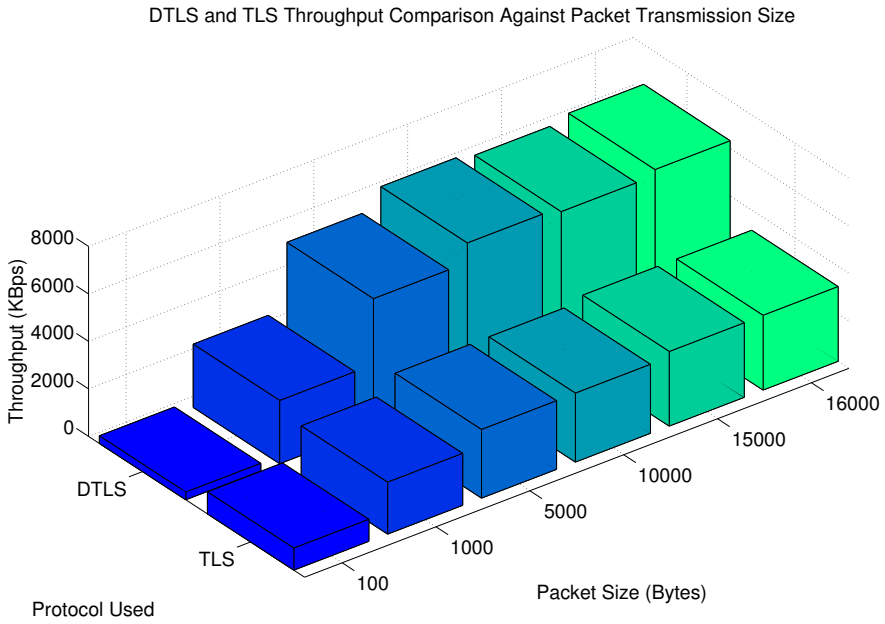

Figure 4.2: DTLS versus TLS throughput for different packet sizes

4.3.2.3 Secure throughput results - proprietary implementation

The results show that the X4 device performs significantly worse than the X2 device achieving only 113 KB/s of data throughput compared to the X2 achieving 1.4 MB/s. Both these devices perform worse than the Raspberry Pi. This is surprising since the two Digi devices are specifically designed for the application.

4.3.3 Throughput of security protocol and packet size comparison

These results indicate that the Raspberry Pi is capable of achieving a maximum throughput rate of 6 324.48 KBps for DTLS communication and 3 146.24 KBps for TLS communication. These results help to confirm that the previous tests were valid. Table 4.10 shows the percent of packets that are lost during the course of the communication. This shows that for smaller packets the communication is less reliable and for larger packets the converse is true. This is to be expected due to the congestion that occurs on the communication medium. This congestion is also the reason that the TCP based TLS performs better when the packet sizes are smaller. The smaller packet size means that there is a larger number of individual packets. TCP has built-in features to manage flow control (congestion control) this allows for

the better handling of the smaller packet numbers [97]. It is important to note that full DTLS and TLS were implemented for these tests. Not only is the information being transmitted secure but the server and client are also being authenticated.

The 16 000 byte packet size was chosen due to the maximum packet size for DTLS and TLS being $2^{14} = 16384$ bytes [98]. It can be seen that both protocols have a higher throughput capability when dealing with a larger packet size. This is also due to the bottleneck forming onto the connection medium. A large number of very small packets arriving very quickly puts the lower level hardware under pressure whereas for the larger packets moving the data through the lower levels of the Open Systems Interconnect model seems to perform an automatic flow control.

Once the packet sizes reach into the kilobytes the advantage gained through control mechanisms by TLS is lost and the nature of the UDP based DTLS leads it to be far superior in terms of throughput. From a packet size of over 5 000 bytes the throughput is around 100% better for UDP based DTLS than the TCP based TLS. This is attributed to the reliable nature of the TLS protocol and the requirement for a response to be received for each transmitted packet.

The Raspberry Pi has a 100 Mbps Ethernet port onboard. This device is theoretically capable of achieving a data throughput of 12 500 KBps. However in practice the overhead of all the networking protocols often results in a lower actual throughput. For the used switch the speed rating can not be found. The impact from DTLS and the overhead of underlying protocols forces the device to only operate at around 60% efficiency. As a comparison high definition video streaming requires around 2 400 kbps [99].

The next set of tests that were completed where done to ensure that the small testing period of 10 seconds did not negatively affect the results achieved. The best performing packet size was run over three different time periods, namely: 30 seconds, 60 seconds and 120 seconds. The results of these tests can be found in figure 4.3. Running over these time durations allows us to ensure that the data stream and communication medium are stable. As can be seen these results confirm the results achieved by the previous tests. It is also seen that the DTLS communication seems to become more efficient with time as the longer the connection is kept available the higher the throughput is achieved.

Packet Size (Bytes)	Percent Lost
100	5%
1000	6%
5000	1.5%
10000	1%
15000	<0.5%
16000	<0.5%

Table 4.10: Table showing DTLS packet sizes against percentage of dropped packets

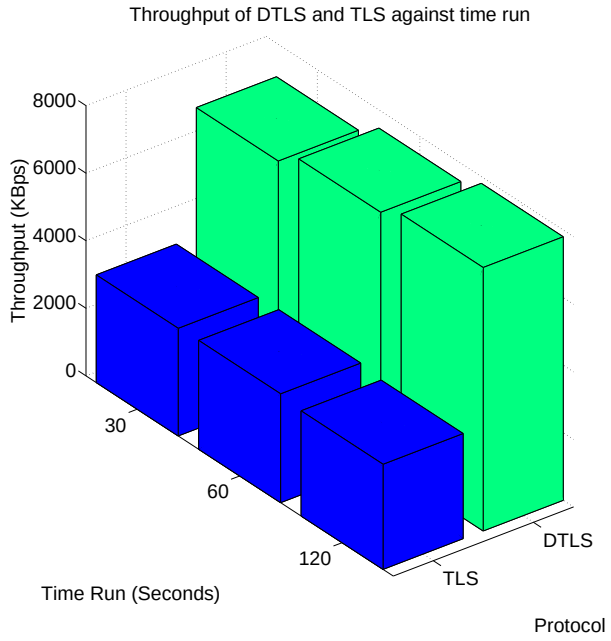

Figure 4.3: DTLS versus TLS throughput for different run lengths

Protocol	Throughput (KBps)
TCP	2450
UDP	131.26

Table 4.11: Table showing the IPSec throughput results

After successfully testing the capabilities of both DTLS and TLS the final test completed was to examine the capabilities of IPSec as a security protocol. The setup involved a Raspberry Pi connected in point-to-point mode directly to a laptop and the Iperf tools where used to perform the tests on the connected network. In order to maintain parity between the tests the exact same network setup as the other tests was used [100]. These final tests resulted in the throughput capability for the IPSec protocol as shown below in table 4.11. The tests were performed in two ways. The first way was to use the Raspberry Pi as the platform device and then to use the Raspberry Pi as the client device.

It is expected that the protocol will perform worse than either DTLS or TLS. This is due to the previous research that has been completed that shows that this protocol performs significantly worse [101]. The results show that the capability of the Raspberry Pi to communicate using IPSec are greatly hampered when implemented in the Internet of Things. The greatest surprise is the difference between the TCP and UDP protocols. This can only be attributed to the TCP protocol's built in flow control. The use of IPSec is possible and the performance is similar to that achieved for TLS based communication.

4.3.4 Power requirements

The power requirements of the implementation will be of the utmost importance. Being able to ensure that the power requirements are met and the power is well controlled will allow for a more robust application. In order to understand what effect each of the protocols have on the application power requirements a study into this was completed.

Figure 4.4 shows the impact that each of DTLS or TLS protocol have on the power consumption of the Raspberry Pi. These results show some surprising results. One of the important things to note is the differing power charts. DTLS has a lower power consumption as the packets increase whereas TLS power consumption increases as the packet sizes increase.

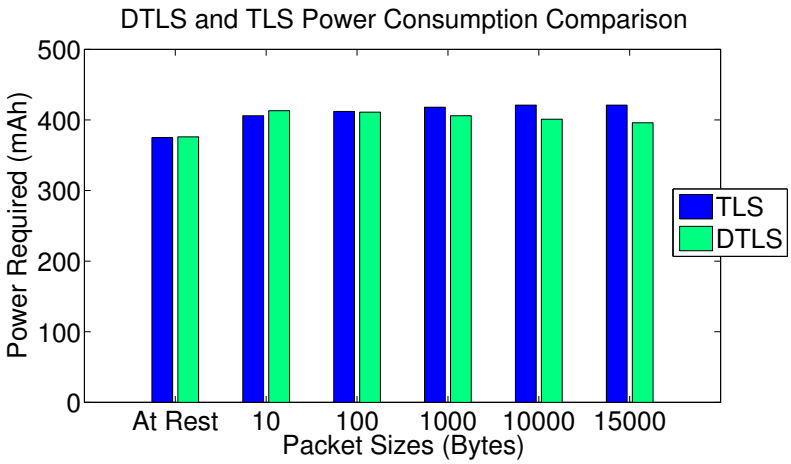

Figure 4.4: DTLS and TLS power required

Protocol	Power Consumed (mAh)
TCP	415
UDP	391

Table 4.12: IPSec Power Consumed

Table 4.12 shows the power consumption of the Raspberry Pi when applying IPSec across either of the two protocols. Due to the performance tool used, the comparison for IPSec only occurs between the two protocols. Unlike the other comparisons completed the measurements do not occur across ranging packet sizes. It is possible to see that, similar to the other tests completed, the UDP protocol outperforms the TCP protocol for implementations. The two protocols perform similarly but the UDP protocol again shows its capability to be used in place of the TCP protocol and its capability to be applied within the Internet of Things.

Noticeably it can also be seen that, the differences between the security options available, in terms of the power used, do not have that much of an impact when applying different security tools. The major concern is the amount of throughput that is capable of being achieved and the power analysis shows us that the capabilities of the Raspberry Pi as a gateway device are possible. It is important to note that this comparison has not had the opportunity of comparing other major gateway devices against the Raspberry Pi.

4.4 NODES

This section will cover the results of the tests for the nodes described in the previous chapter.

4.4.1 Device comparison

The primary comparison of the devices will look at their basic functionality and capability [87]. The initial testing will look at the power consumption of each of the devices.

The list of devices above can be broken down into two separate categories; traditional wireless sensor network nodes and non-traditional (devices that were not originally created to perform the role of a wireless sensor network node). The Raspberry Pi and the Beaglebone are the non-traditional WSN nodes as they have not been implemented with the limitations of the WSN in mind. These devices have not been developed to use the minimal amount of power etc. This means that these devices perform very well in certain instances of the benchmarking.

4.4.2 Power comparison

The two graphs seen in figure 4.5 and figure 4.6 show the power consumption of all of the tested devices. These graphs show the expected current consumption (in mAh) against the duty cycle (% of time spent sending and receiving information) for these individual Internet of Things nodes. They provide estimations based on the expected power consumption with values for the sleep and non sleep being used to provide a guide. For the time that the nodes are not operational the device spends its time in sleep mode, attempting to conserve as much energy as possible. The most economical of all of the devices is the Sprouts device. This device successfully achieves less than 180 μAh in full active mode (continuously awake). Some of the other economical devices are the Waspmote, which achieves a maximum of 15 mAh in full active mode. The older devices that have been compared here all perform very similarly. These are the TelosB and the Micaz, they both achieve around 25 mAh for full awake operation. The Firefly node achieves a similar current requirement of 24.8 mAh for continuous operation. These devices are followed closely by the Imote2 and Lotus platform each achieving 44 mAh and 66 mAh respectively for continuous operation. Using these graphs and by knowing the power limitations of the application being designed for a suitable node

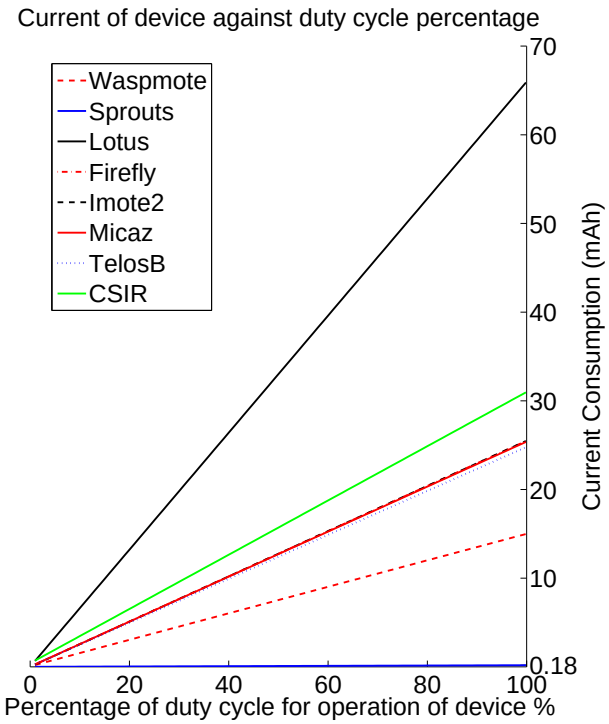

Figure 4.5: Wireless Sensor Nodes current consumption against duty cycle

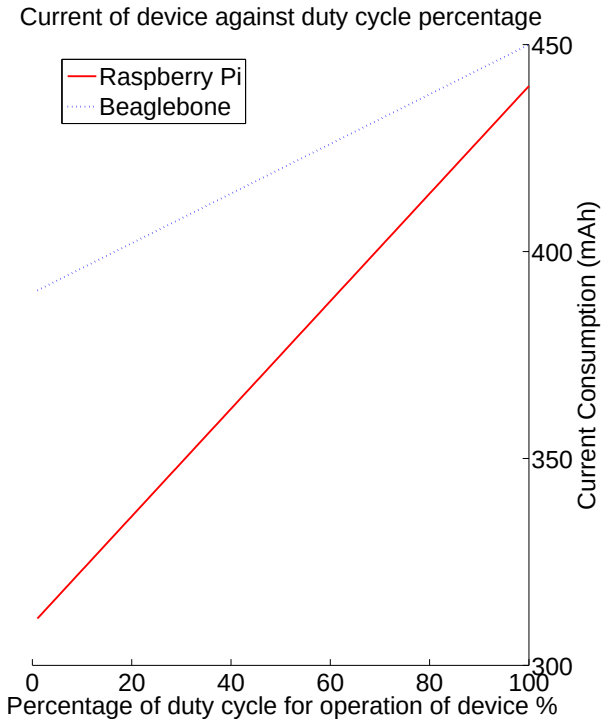

Figure 4.6: Non-traditional nodes current consumption against duty cycle

can be chosen beforehand. This allows for adequate planning and can help prepare a suitable budget when deciding on which node to use for the Internet of Things application.

As can be seen from the graphs shown above the CSIR's rapidly prototyped COTS node has only slightly higher power requirements, mostly due to the more powerful processor that is included on the board, than a cluster of current sensor nodes options. The node achieves 31 mAh when operating at 100% and 400 nAh when in sleep mode. The sleep mode current is high for the device but the operational current is similar to what is required for the other nodes.

The power consumption of the non-standard nodes (by non-standard means any device that is not originally created for use within the Internet of Things/WSNs) is significantly higher. The sleep and non-sleep of these nodes is defined differently to the sleep and non-sleep of the traditional nodes. A traditional WSN node sleep typically involves a state in which most power consuming hardware is off and waiting for a wake command either from the onboard processor or another device in the network. The sleep for the Raspberry Pi and the Beaglebone is defined as a state of rest, where little to no processing is occuring [2]. The traditional sleep (for WSN without the expansion board) can be achieved but it is very difficult to wake the device and takes a significant length of time longer than the more traditional nodes. As can be seen from figure 4.6 the Raspberry Pi consumes 44 mAh for continuous operation and the Beaglebone consumes 450 mAh for continuous operation.

4.4.3 Capabilities comparison

In the current world of computing more cost typically gives you greater capability. A comparison was completed between the cost of the nodes and the processing capability that is received for the cost. Processing capability is not the only performance based metric that can be used. This metric was chosen as it is generally the one most linked to the cost of the node and potentially the one with the most impact on actual performance.

The cost of the Firefly device is currently not known as it is still under development. The Sprouts device is open source and although the cost of the completed device is not available, the bill of materials cost was used as a capable individual could in theory create one of these

[2] A expansion board for the Raspberry Pi can be added which will enable a WSN node like sleep mode

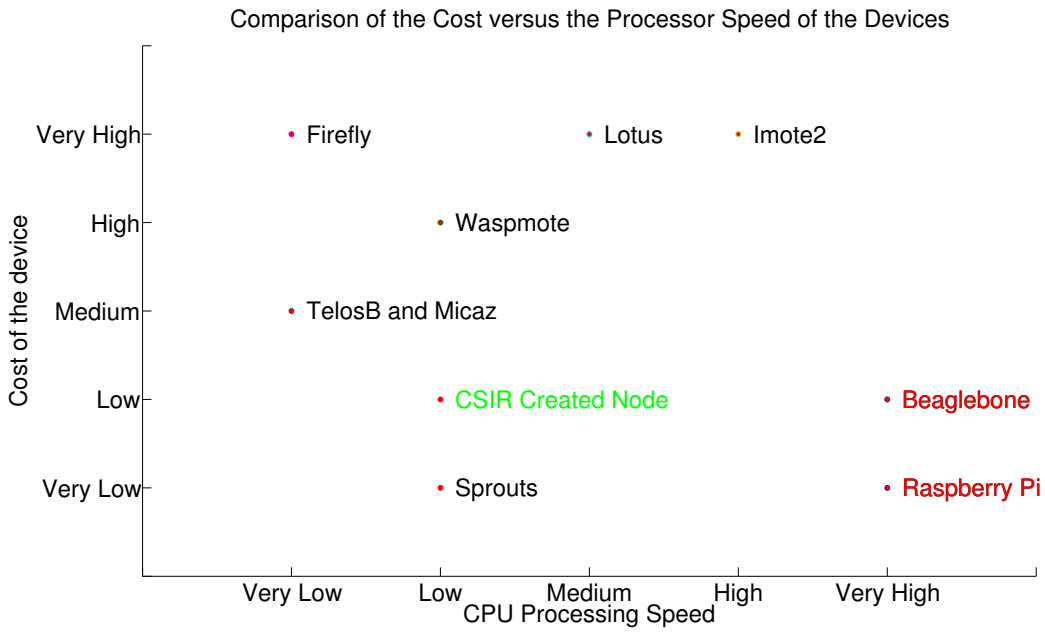

Figure 4.7: Comparison of processing speed against cost for WSN nodes (COTS and open platforms indicated in green and red respectively).

devices themselves. Surprisingly in this comparison (figure 4.7) the most powerful devices are also the cheapest. The Sprouts node is the cheapest complete WSN node with the CSIR's own rapid prototype device coming in as being one of the cheapest available as well. Although both Sprouts and the CSIR made device are low on processing and power, depending on the application this should not be a drawback from implementation within a wireless sensor network. The low power requirement will allow for long term deployments within a Internet of Things applications such as those within embedded environment monitoring applications and with a careful design the low processing power will not negatively affect the performance of the application.

The next two cheapest are the Raspberry Pi and the Beaglebone. Both of these devices will require an XBee module (or similar) for wireless communication. These modules cost about $30 - $40. This will not change the ranking of these two devices on the scale. This graph shows that some of the least powerful devices are in fact the most expensive. The Lotus node and the Imote2 node both cost $300, a massive sum to be paying for a WSN node. The built for purpose devices end up being expensive possibly due to the ease of plug and play capability. The additional cost for the Raspberry Pi and Beaglebone is getting the devices to

communicate successfully with the network through the GPIO serial ports. Due to the open source nature many tutorials and online examples exist to assist with this problem.

4.4.4 Features comparison

This comparison looks specifically at the hardware and software capabilities of the individual devices[3].

[3]Costs shown in the table are rough guides as large discounts can be achieved when ordering large quantities

Node	OS	CPU Architecture (bits)	CPU Speed (MHz)	Network Std	No. of Sensors	Language	I/O Pins	Release Date	Cost ($)
Waspmote	None	8	14.75	Various (Depends on Implemen- tation)	2	C/C++	21	2013	210
Sprouts	Linux	32/8	32	Zigbee, BLE	2	C	0	2011	20
Lotus	Mote Runner / TinyOS	32	10-100	Zigbee	0	nesC	0	2011	300
Firefly	NanoRk	8	7.3	Zigbee	0	C	20	2010	Unknown
Imote2	TinyOS	32	13-416	Zigbee	0	nesC	12	2006	300
Micaz	TinyOS	8	16	Zigbee	0	nesC	4	2004	100
TelosB	TinyOS or Contiki	16	16	Zigbee	3	nesC	16	2005	120
CSIR Node	Contiki	32	24	802.15.4/Zigbee	0	C/C++	17	2012	80
Raspberry Pi	Raspbian, Android, Many more	32	700	None but expandable	0	Any for Linux OS	0 GPIO and UART ,I^2C SPI	2012	35
Beagle-bone	Angstrom, Raspbian, Many more	32	720	None but expandable	0	Any for Linux OS	65 GPIO and UART ,I^2C SPI	2012	89

Table 4.13: Table showing features of the WSN nodes

As can be seen the features of the devices differ greatly. One important feature is the operating system of the node chosen. The operating system has impact into the stability, capability and security of the device. A well developed and continuously updated operating system can allow for new powerful features to be deployed to the nodes without making any hardware changes. The Raspberry Pi and Beaglebone both have many options available and many of these options are well supported in terms of continuous updates. TinyOS and Contiki are both open source with active communities driving development. NanoRK is a operating system created by Carnegie Melon University for use on their WSN node. Mote Runner is a set of tools to aid the running of the node from IBM [102]. Table 4.13 shows that the features of the devices can differ greatly. However it also shows that the current communication choice for Internet of Things embedded nodes is the Zigbee protocol. This is due to the large number of devices that support this communication standard as can be seen from table 4.13. A number of the devices also include built-in onboard sensors that can be used for sensing applications giving a reduction of the cost for certain applications when these onboard sensors are used. Some of the devices also include a range of GPIO (general purpose input output) ports available on the devices. These devices allow for advanced features to be added to the application. As can be seen a range of features are available to the nodes.

It can be seen from the table that the created CSIR node, although being a rapid implementation, has similar capabilities to the other nodes tested. This shows that the use of the circuits available and a simple working knowledge of general electronic skills can allow one to create powerful and cheap devices. Our node has full 32 bit capability, is coded in a standard language (C++) and has a range of expansion capabilities through the generic I/O pins. The node is also significantly cheaper than many of the built for purpose devices.

The table 4.13 shows many of the capabilities of each of the individual devices. Some of the additional expansion capabilities are included in footnotes at the bottom of the page and these contain links to developer webpages showing a large list of expansion capabilities available for the Raspberry Pi and the Beaglebone.

CHAPTER 5

DISCUSSION

5.1 CHAPTER OVERVIEW

This chapter provides an overview of the results obtained in the previous chapter. The aim is to provide a concise and over-arching view of the results achieved through the set of tests in the previous chapter.

5.2 PLATFORMS

As a summary of the section detailing the results, section 4.2.1, table 5.1 has been included. These results vary greatly and highlight the number and range in security of options that are available for IoT development.

These tests provided an initial study of the security of the communication between the embedded devices and the cloud platforms typically used in Internet of Things applications. Currently, none of the implementations are completely secure but a good number of the implementations and platforms do provide for a acceptable level of security. In a number of application environments this limited security capability will be sufficient. When communicating across any public form of a network there will always be a security risk. It is not feasible to limit all forms of attack that could affect data communication across any public infrastructure. Although posing a plausible security risk the BEAST attack described in the results section may be considered to be less harmful to the overall Internet of Things application. In order to successfully carry out this attack an attacker must first ensure the victim accesses a link to a BEAST program. If the embedded device is autonomous it is

Table 5.1: Summary of evaluates scores and resistance to specific weaknesses

Platform	*SSL Result*	RC4	BEAST	CRIME
Arkessa	C	N	Y	Y
Axeda	B	N	N	N
Bugswarm	F	N	N	Y
Carriots	B	N	N	Y
COSM/Pachube/Xively	A	N	Y	Y
Evrythng	B	N	N	Y
Exosite	F	N	N	Y
Grovestreams	B	N	N	Y
iDigi/Device Cloud	A	N	Y	Y
Sensorcloud	B	N	N	Y
Thingspeak	B	N	N	N
Yaler	C	N	N	Y

unlikely that such an attack is possible unless the attacker gains control of the device. If this is the case the greater security risk is the loss of control of the device. This leads to the fact that each and every single Internet of Things application needs to be considered on its own merits when deciding on which platform needs to be used. It is advised that the decision of which platform used is guided by the requirement's of each individual and the security achieved. One concern is that the implementation of good security principles done in a bad way can in itself compromise the security of the application. This reinforces the need to look at the security of the application as a core requirement in conjunction with all the other requirements of the system.

5.3 GATEWAY

From the results achieved when completing the tests for the gateway devices within the overall Internet of Things applications, we can see a number of important points that we can consider.

5.3.1 Secure gateway devices

The tests show that both TLS/SSL and DTLS can be applied within an Internet of Things application, with DTLS achieving 6 324.48 KBps and TLS/SSL achieving 3 146.24 KBps. It is also shown that the cryptographic cipher chosen is not the most relevant factor when considering the overall impact on throughput for the devices. The biggest factor that directly affects the throughput of the different devices when enforcing secure communication across Internet of Things applications is the packet size used within the secure algorithm.

The testing proves that although the application of security processes and procedures to the Internet of Things applications does have an impact on the performance of the devices involved; the benefit that is gained from having a secure application and thereby protecting the data that is involved in the communication far outweighs the negative effects of the secure algorithms on the performance of the application as a whole.

5.3.2 Raspberry Pi as an open source gateway

Comparing the results from the Raspberry Pi and the Beaglebone tests against the results achieved by the proprietory gateways provides for a reasonable assumption that not only are the Beaglebone and Raspberry Pi capable of achieving the required throughput but can in some cases easily outperform the much more expensive proprietary devices. It must be stated though that in some cases the Raspberry Pi may perform worse due to the single core nature of the architecture that the device is built upon. This can be avoided, and in many cases discounted, due to the fact that many embedded systems will want to limit the amount of communication that the embedded sensors/nodes perform in order to preserve the limited battery life of these devices. This will limit the chances that the single core Raspberry Pi gets swamped by the embedded system and is unable to handle the sending and receiving of data on the more traditional side of the network.

As an open source alternative not only is the cost of the device cheaper but due to a greater need to follow established protocols and systems during the creation of the devices a much more interoperable application exists. This grants a huge deal of advantages when dealing with the Raspberry Pi and similar devices. The ability to replace a wireless communication module on the device and suddenly be able to communicate with a range of new devices

simply and easily is something that will allow the embedded application to last longer and be more robust than other proprietary or closed source implementations. It is possible for these closed source devices to meet similar standards as the open source alternatives but this is rarely done. Instead the manufacturers prefer to provide you with a range of devices that work well together and communicate effectively but are not interoperable with devices from other manufacturers or with other open source systems.

5.3.3 Secure communication

In order to achieve secure communication a number of protocols have been examined. The results have been received and it can be seen that secure communication from the cheaper open source devices often outperforms the capabilities of the more specialized proprietary devices.

We see from the results that a number of factors affect the communication throughput of the secure data. Some of the factors that we have shown to affect the throughput of the data can be seen below:

- Level of security
  - CIA support
- Size of packets
- Protocol chosen
- Underlying algorithms
- Underlying implementation of the chosen protocol

All of the above have an effect on the throughput capabilities of the devices. The single factor that has the highest level of impact when compared directly is the level of security chosen. The use of all three of the primitive security objectives results in having the highest impact on the communication. Of all of these primitive security objectives, applying confidentiality is the one that has the single greatest impact on data communication. Authentication and Integrity application have a minimal effect on the data communication. This is due to the complicated

functions and procedures that are utilised to provide confidentiality to communication when applying cryptographic functions to the data to be communicated. These functions make use of a large amount of the limited resources available for operations on the device.

The other major factor that affects the data communication abilities of the device is the protocol that is chosen as the results show that the DTLS protocol is more capable of achieving a higher throughput when dealing with larger packet sizes. When dealing with smaller packet sizes a TLS protocol is favoured instead. These two protocols are both capable of achieving high data rates and should be applied according to the requirements of the individual applications. It is important to note that the Raspberry Pi is capable of being applied as a Gateway device with the results that are achieved.

The common theme that runs through all of the results is that no single solution/roadmap exists for the Internet of Things. Each application must be considered according to the requirements and a decision must be reached as to what is mission critical and what is not. An application that has the requirement to transfer very small packets of data (< 1000Bytes)continuosly will benefit from the application of TLS based security measures due to the high throughput speeds achieved. Applications that require constant transfer of small to large packets of data will benefit greatly from the raw throughput speeds that can be achieved from the application of DTLS based security. As an example TLS applied to individual embedded nodes that transmit directly to the centralised servers and don't require the gateway device to be involved will benefit from the simplicity and throughput to be achieved. A gateway device that collects the information from the embedded system and then continuosly updates the central server will benefit from a DTLS based system.

The developer is also able to benefit from the implementation of the Raspberry Pi within the system. This device is much cheaper and tests show it performing much better than the devices built for purpose for application within the Raspberry Pi. As a general purpose gateway device the Raspberry Pi is able to replace these more expensive devices with limited changes required. Additionally, due to its open source nature, the Raspberry Pi allows for easy upgrades as new communication technologies become available.

5.4 NODES

There are a variety of nodes used within Internet of Things applications. This leads to a huge number of discrepancies in the capabilities of individual nodes. A concise list of the capabilities and advantages of using individual nodes needed to be provided. This is what was aimed to be achieved through the duration of the research, a complete list of the current state of the capabilities of the Internet of Things node devices.

During the duration of the research it was noticed that the cost of using a large number of the nodes was high. Individual purchase cost for these devices sometimes ranged into the hundreds of dollars or thousands of rands, but the cost of a Raspberry Pi or Beaglebone is much lower around $30. It was therefore decided to make a comparison and decide whether or not it was possible to apply the Raspberry Pi or Beaglebone device as an Internet of Things node. From the results we can see that these devices could be deployed with a few disadvantages or considerations that need to be made before hand. Additionally in these comparisons a device that was created as an alternative node by the CSIR was compared and included as an option of creating a node from off the shelf components in a COTS approach.

It must be noted that all of these options are fairly easy to achieve and with modern, powerful and cheap components being made by the large technology companies; the availability and interest in these devices has surged greatly. This COTS approach may become the prefered method of adoption for a large number of organisations and institutions.

5.4.1 Devices

There are a number of considerations that need to be taken into account when implementing any Internet of Things applications. The range of tests that were completed across the range of devices show that there are a large number of devices available for applications. This large number of devices allows there to be a large range of options available when implementing Internet of Things applications. This means that the devices typically have one area of speciality and are therefore designed to carry out a specific task instead of being more general purpose. For example it can be seen that the Raspberry Pi has a huge number of features and is considered a powerful device. This large number of features and massive

amount of customization options has created a large online community which has created projects that utilise the Raspberry Pi in a range of possible applications and roles. These customization options come at a cost though; within our testing it was seen that the large amount of features and high capabilities of the device come with the cost of having a large power requirement. It is noticeable from the results that there is a direct relationship between the features of the device and the power requirements. The most noticeable impact between the devices is the CPU speed and its impact on the power requirements of the devices. One of the unexpected relationships appears between the cost of the individual devices and the number of features. The most feature rich devices have the lowest costs (Raspberry Pi and Beaglebone) whereas the most expensive devices have the lowest power requirements. This is due to the specialized hardware and to a smaller extent the software that is developed to be used with these devices. The Raspberry Pi make excellent use of the fact that they are open source devices and allow the community to submit updates and changes to the code and hardware that will improve the system in any way. These updates mean that money can be saved by the developer and changes can be implemented according to the requirements of the end user.

This range of devices highlight why the COTS approach may be the best available option. With a limited amount of knowledge or by making use of online forums a huge number of implementations exist that can be deployed by a developer of the technology and this can be relatively cheap and robust with easy options in order to upgrade components thereby saving cost across the long years these applications will be deployed.

5.5 CONCLUDING REMARKS

Through all of this, it is shown that the range of application requirements that we can find within the implementation of the Internet of Things no one option is available that can meet them all. The research has shown that the open source approach to the Internet of Things does have a number of advantages over the other traditional approaches. The cost is generally lower; there are more general purpose and robust devices that can be deployed. An application that makes use of these advantages effectively and allows for the design to meet the requirements will easily be able to deploy an open source, secure and capable Internet of Things application meeting either industry or individual requirements.

5.5.1 Open source concerns

Over the last year a number of key weaknesses have arisen with the use of the open source approach to both hardware and software design.Most notably is the recent errors found within the OpenSSL library known as the Heartbleed bug [72]. The developers of the OpenSSL library made the mistake of allowing for a buffer overflow attack that allowed confidential memory data to be compromiseable. Although rapidly patched the number of devices that were affected were huge because of the widespread use of the library. The number of people that are directly responsible for maintaining this piece of code responsible for this error is small, normally about two people. These small teams create additional concerns. If the documentation for the design of open source hardware or software is not well maintained and if something happens to these developers the library or hardware may be unuseable by other people.

CHAPTER 6

CONCLUSIONS

6.1 CHAPTER OVERVIEW

This chapter will have the aim of detailing exactly what conclusions have been extracted from the set of results that have been completed.

6.2 FOCUS OF RESEARCH

The focus of the research was with the technologies that are currently used within the Internet of Things. Focus of the research was to determine the capabilities of the current technologies that are available for the Internet of Things. The technologies that where investigated range from specific application areas within the more well known Wireless Sensor Networks through to the much newer and demanding Internet of Things. The focus was on ensuring that the application of open source ideals, in terms of both hardware and software, to the Internet of Things has the capability of providing a meaningful contribution to future applications. When speaking of capability the focus was on ensuring a reliable, stable and secure implementation of the application. The research ensured that all three subject areas where investigated, namely; the embedded devices providing the sensing and actuators; the gateway section that provides for the communication between the embedded devices and the wider Internet and finally the powerful cloud based platforms that enable the processing, displaying and storing of the large amounts of data available through the Internet of Things.

The research was conducted through the implementation of a complete Internet of Things application. Testing was completed and the initial hypotheses were validated through exper-

imentation across a range of devices. The arguments are stacked up by physical testing and experimentation completed on the actual devices. The tests also cover a number of theoretical values, specifically for the capabilities comparison, an in depth study was completed as to the availability of these devices. The arguments made within the document are backed up by in depth research and conducted research.

6.3 CONCLUSIONS

The Internet of Things has become a major area of interest for the world and a large amount of interest in research and industrial application is being experienced. The experimentation that has been completed shows that a capable, reliable and cheap Internet of Things is possible. An open source implementation can in certain situations provide for a more powerful and capable application within the wide Internet of Things. The open source approach is a dedicated approach that has allowed for an expansion of inexpensive devices that the customer/applicator can utilise in order to create power specialised applications. The key point that must be considered is that individual applications need to make choices according to the requirements specific for their application. The approach chosen by the CSIR shows that another option is available. The device created from this COTS approach shows that the options available for the Internet of Things applications are wide and can be hugely specialized to requirements. A limited knowledge of electronics can allow for an application developer to create a device specialised to his needs. This is the approach that is recommended from the research that has been presented.

The tests also show that the Raspberry Pi is capable of being deployed in a number of roles within an Internet of Things applications. The results show that the Raspberry Pi is capable of being deployed within the embedded section or as a gateway device. The power requirements is the one major concern for the embedded section within the Internet of Things, although these concerns are valid simple measures can be taken to limit the effect that this has on the Internet of Things application.

As a gateway device the Raspberry Pi or Beaglebone can easily be deployed as a gateway device in the Internet of Things. The capabilities of the Raspberry Pi and Beaglebone are shown to outperform both of the industry devices that they were compared against. They are able to easily control the communication between the embedded network and the larger

Internet as a whole. The small impact that the application of security to the communication between the embedded network and the platforms on the Internet must be noted. The security of the application on the Internet of Things can allow for a much faster and more powerful implementation of the Internet of Things.

The two approaches chosen for securing the Internet of Things mainly DTLS and TLS are both capable of securing the Internet of Things. DTLS is the most capable when applied within the Internet of Things especially when applied to large packet sizes. The impact is noticeable once applied to the Internet of Things but this impact is easy for the Internet of Things to handle and able to be applied. An approach that is recommended is to not have a single gateway device. Splitting the embedded network amongst a number of gateway devices and applying the required security to the communication is capable once applied.

All of the above leads to the fact that an open source, cheap, reliable and secure Internet of Things is possible. Making use of a secure DTLS protocol and an open source inexpensive device such as the Raspberry Pi is capable of being deployed within the Internet of Things as either a gateway device or less likely as embedded device. The preffered approach for the embedded section would be to create a device from a commercial off the shelf approach.

6.3.1 Research objectives discussion

6.3.1.1 Open source hardware/software within an industrial Internet of Things

There are a huge number of open source alternatives that are available for use within the industrial Internet of Things. Through the course of the research it has been noted that the capabilities of the new open source devices have greatly increased in recent years. The communities that are associated with these open source devices are very well supported and have a huge following, especially for the Raspberry Pi. Through all of the completed tests it can be seen that these devices are capable of being deployed within the industrial Internet of Things. The results of these tests can be seen in the results section. Not only is it possible to apply open source hardware within the industrial Internet of Things but open source software is already seeing a huge following within this area.

6.3.1.2 Open source alternatives to embedded devices within an industrial Internet of Things

Following on from the research into the open source device capabilities, it was interesting to see exactly how many open source devices are available for implementation within the Internet of Things. As a comparison it was interesting to note that many devices have been designed in recent years for application within the Internet of Things. These devices are generally built for a specific purpose by a company. From the research it was noted that often open source devices were better for the Internet of Things. This was due to the more open design process. Allowing all developers to see all the design stages and make adjustments to suit their individual needs allows for a more appropriate device to be utilised within their application. This was specifically confirmed when the devices were tested and the capabilities directly compared between each other, chapter 4 results section.

6.3.1.3 Known security concerns and existing security protocols

As has already been shown within the previous sections, chapter 3 and chapter 4, there are a huge number of inherrent risks when considering the security concerns within the Internet of Things. A number of solutions have already been shown to be adequate for application within a traditional Internet or a Internet of Things application. A large amount of research has been performed in creating a new generation of security protocols to protect the modern Internet of Things. From the completed research it is evident that the new generation of devices are powerful enough to interact within these Internet of Things applications. Making use of these exisiting and tested security protocols will allow for a more simple and interoperable Internet of Things.

6.3.2 Future work

Due to the huge amount of interest being generated for the Internet of Things, research into the field is highly valued and incredibly necessary. The research conducted within this document shows that current assumptions regarding devices within the Internet of Things have become outdated. New generations of devices are powerful enough to implement the full suite of technologies already developed to protect the Internet of Things. One specific area of

research would be into the possible application of public key infrastructure into an Internet of Things application. With the powerful devices available and the new generation of devices, such as the Intel Edison and Galileo development boards, becoming available application of these technologies previously considered to be too intensive for the Internet of Things may actually become the perfect technologies for this specific application domain.

REFERENCES

[1] A. Boulanger, "Open-source versus proprietary software: Is one more reliable and secure than the other?" *IBM Systems Journal*, vol. 44, no. 2, pp. 239–248, 2005.

[2] R. Fisher and G. Hancke, "DTLS for lightweight secure data streaming in the internet of things," in *3PGCIC 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2014.

[3] R. Fisher and G. Hancke, "SSL usage in commercial internet of things platforms," in *Volume 11: Radio Frequency Identification System Security*, ser. Cryptology and Information Security Series. IOS Press, 2013, vol. 11, pp. 69–82.

[4] R. Fisher, L. Ledwaba, G. Hancke, and C. Kruger, "Open hardware: A role to play in wireless sensor networks?" *Sensors*, vol. 15, no. 3, pp. 6818–6844, 2015. [Online]. Available: http://www.mdpi.com/1424-8220/15/3/6818

[5] A. Ukil, J. Sen, and S. Koilakonda, "Embedded security for internet of things," in *Emerging Trends and Applications in Computer Science, 2011 2nd National Conference on.* IEEE, Mar. 2011, pp. 1–6.

[6] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[7] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, Sep. 2012.

[8] M. Jung, J. Weidinger, W. Kastner, and A. Olivieri, "Building automation and smart

cities: An integration approach based on a service-oriented architecture," in *2013 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Mar. 2013, pp. 1361–1367.

[9] D. Egan, "The emergence of ZigBee in building automation and industrial controls," *Computing and Control Engineering*, vol. 16, no. 2, pp. 14–19, 2005.

[10] G. Bovet, J. Hennebert, and others, "Introducing the web-of-things in building automation: A gateway for KNX installations," in *10th international Conference on Informatics in Control, Automation and Robotics (ICINCO 2013)*, 2013.

[11] M. O'Neill, "The internet of things: do more devices mean more risks?" *Computer Fraud & Security*, vol. 2014, no. 1, pp. 16–17, Jan. 2014.

[12] J. Rivera and R. Van Der Meulen, "Gartner says the internet of things installed base will grow to 26 billion units by 2020," 2014. [Online]. Available: http://www.gartner.com/newsroom/id/2636073

[13] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson, "M2m: From mobile to embedded internet," *Communications Magazine, IEEE*, vol. 49, no. 4, pp. 36–43, 2011.

[14] Y. Liu and G. Zhou, "Key technologies and applications of internet of things," in *2012 Fifth International Conference on Intelligent Computation Technology and Automation (ICICTA)*, Jan. 2012, pp. 197–200.

[15] F. Nasri, N. Moussa, and A. Mtibaa, "Internet of things: Intelligent system for healthcare based on WSN and android," in *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, Jan. 2014, pp. 1–6.

[16] A. Iera, C. Floerkemeer, J. Mitsugi, and G. Morabito, "The internet of things," *IEEE Wireless Communications*, vol. December, pp. 8–9, 2010.

[17] M. Friedewald and O. Raabe, "Ubiquitous computing: An overview of technology impacts," *Telematics and Informatics*, vol. 28, no. 2, pp. 55–65, May 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736585310000547

[18] D. Zuehlke, "SmartFactory - towards a factory-of-things," *Annual Reviews in Control*, vol. 34, no. 1, pp. 129–138, Apr. 2010.

[19] V. Gungor and G. Hancke, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 4258–4265, Oct. 2009.

[20] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[21] C. Kruger and G. Hancke, "Implementing the internet of things vision in industrial wireless sensor networks," in *IEEE International Conference on Industrial Informatics (INDIN)*, 2014.

[22] M. Chan, E. Campo, D. Estève, and J.-Y. Fourniols, "Smart homes - current features and future perspectives." *Maturitas*, vol. 64, no. 2, pp. 90–7, Oct. 2009. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19729255

[23] B. Popper, "How apple and google plan to reinvent health care." [Online]. Available: http://www.theverge.com/2014/7/22/5923849/how-apple-and-google-plan-to-reinvent-healthcare

[24] A. Jara, M. Zamora, and A. Skarmeta, "Knowledge acquisition and management architecture for mobile and personal health environments based on the internet of things," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Jun. 2012, pp. 1811–1818.

[25] Discovery Health, "Health ID - your medical information in your doctor's hands," 2014. [Online]. Available: https://www.discovery.co.za/portal/individual/medical-aid-news-jul13-medical-history

[26] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "IOT gateway: Bridging wireless sensor networks into internet of things," in *2010 IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC)*, Dec. 2010, pp. 347–352.

[27] X. He, Y. Ma, and M. Mizutani, "Data transmission mechanism for multiple gateway

system," in *Information Science and Service Science and Data Mining (ISSDM), 2012 6th International Conference on New Trends in*, Oct. 2012, pp. 98–101.

[28] T. Sauter and M. Lobashov, "How to access factory floor information using internet technologies and gateways," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 699–712, Nov. 2011.

[29] M. Castro, A. J. Jara, and A. F. Skarmeta, "An analysis of m2m platforms: Challenges and opportunities for the internet of things," in *Innovative Mobile and Internet Services in Ubiquitous Computing, 2012 Sixth International Conference on*. IEEE, jul 2012, pp. 757–762.

[30] L. Deru, S. Dawans, M. Ocaña, B. Quoitin, and O. Bonaventure, "Redundant border routers for mission-critical 6lowpan networks," in *Real-World Wireless Sensor Networks*, ser. Lecture Notes in Electrical Engineering, K. Langendoen, W. Hu, F. Ferrari, M. Zimmerling, and L. Mottola, Eds. Springer International Publishing, Jan. 2014, no. 281, pp. 195–203.

[31] R. Piyare and S. R. Lee, "Towards internet of things (IOTS):integration of wireless sensor network to cloud services for data collection and sharing," *International journal of Computer Networks & Communications*, vol. 5, no. 5, pp. 59–72, Sep. 2013, arXiv: 1310.2095. [Online]. Available: http://arxiv.org/abs/1310.2095

[32] Editor IEEE Spectrum, "Oops! how many IP addresses?" Mar. 2007. [Online]. Available: http://spectrum.ieee.org/tech-talk/semiconductors/devices/oops_how_many_ip_addresses

[33] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi, "From today's INTRAnet of things to a future INTERnet of things: A wireless- and mobility-related view," *IEEE Wireless Communications*, vol. 17, no. 6, pp. 44–51, 2010.

[34] S. Sajjad and M. Yousaf, "Security analysis of IEEE 802.15.4 MAC in the context of internet of things (IoT)," in *2014 Conference on Information Assurance and Cyber Security (CIACS)*, Jun. 2014, pp. 9–14.

[35] Institute of Electrical and Electronics Engineers (IEEE), "IEEE 802.15 WPAN task

group," Nov. 2014. [Online]. Available: http://www.ieee802.org/15/pub/TG4.html

[36] L. Lo Bello and E. Toscano, "Coexistence issues of multiple co-located IEEE 802.15.4/ZigBee networks running on adjacent radio channels in industrial environments," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 2, pp. 157–167, May 2009.

[37] M. Jung, J. Weidinger, W. Kastner, and A. Olivieri, "Heterogeneous device interaction using an IPv6 enabled service-oriented architecture for building automation systems," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC '13. New York, NY, USA: ACM, 2013, pp. 1939–1941. [Online]. Available: http://doi.acm.org/10.1145/2480362.2480722

[38] T. Lennvall, S. Svensson, and F. Hekland, "A comparison of WirelessHART and ZigBee for industrial applications," in *IEEE International Workshop on Factory Communication Systems, 2008. WFCS 2008*, May 2008, pp. 85–88.

[39] G. Gaukler, "Item-level RFID in a retail supply chain with stock-out-based substitution," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 2, pp. 362–370, May 2011.

[40] S. Cataudella, R. D. Ambra, M. Rampacci, V. Sbragaglia, F. Antonucci, J. d. Río Fernandez, C. Costa, J. Aguzzi, P. Menesatti, A. Manuel Lázaro *et al.*, "Versatile application of rfid technology to commercial and laboratory research contexts: fresh fish supply-chain and behavioural tests," *SARTI*, 2011.

[41] O. Ondemir, M. Ilgin, and S. Gupta, "Optimal end-of-life management in closed-loop supply chains using RFID and sensors," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 3, pp. 719–728, Aug. 2012.

[42] Google.com, "Ipv6 – google," 2015. [Online]. Available: https://www.google.com/intl/en/ipv6/statistics.html

[43] J. Hui and D. Culler, "IPv6 in low-power wireless networks," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1865–1878, Nov. 2010.

[44] L. Deru, M. Van de Borne, M. Ocana, and S. Dawans, *6lbr A deployment-ready 6LoWPAN Border Router solution based on Contiki*, 2013. [Online]. Available: http://cetic.github.io/6lbr/

[45] A. J. Jara, S. Varakliotis, A. F. Skarmeta, and P. Kirstein, "Extending the internet of things to the future internet through IPv6 support," *Mobile Information Systems*, vol. 10, no. 1, pp. 3–17, Jan. 2014. [Online]. Available: http://dx.doi.org/10.3233/MIS-130169

[46] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks, 2004*, Nov. 2004, pp. 455–462.

[47] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. Schmidt, "RIOT OS: Towards an OS for the internet of things," Apr. 2013. [Online]. Available: https://hal.inria.fr/hal-00945122/

[48] K. O'Hara, "The fridge's brain sure ain't the icebox," *IEEE Internet Computing*, vol. 18, no. 6, pp. 81–84, Nov. 2014.

[49] S. Dredge, "Edward snowden revelations have had limited effect on privacy," 2014. [Online]. Available: http://www.theguardian.com/technology/2014/nov/25/edward-snowden-privacy-open-thread

[50] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer Networks*, vol. 76, pp. 146–164, Jan. 2015.

[51] M. Kenney, "Cyber-terrorism in a post-stuxnet world," *Orbis*, vol. 59, no. 1, pp. 111 – 128, 2015.

[52] H. Suo, J. Wan, C. Zou, and J. Liu, "Security in the internet of things: A review," in *2012 International Conference on Computer Science and Electronics Engineering*. Ieee, Mar. 2012, pp. 648–651.

[53] O. Garcia-Morchon, S. L. Keoh, S. Kumar, P. Moreno-Sanchez, F. Vidal-Meca, and

J. H. Ziegeldorf, "Securing the IP-based internet of things with HIP and DTLS," in *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '13. New York, NY, USA: ACM, 2013, pp. 119–124. [Online]. Available: http://doi.acm.org/10.1145/2462096.2462117

[54] A. De Rubertis, L. Mainetti, V. Mighali, L. Patrono, I. Sergi, M. Stefanizzi, and S. Pascali, "Performance evaluation of end-to-end security protocols in an internet of things," in *Software, Telecommunications and Computer Networks (SoftCOM), 2013 21st International Conference on*, 2013, pp. 1–6.

[55] M. Walker, *CEH Certified Ethical Hacker All-in-One Exam Guide*, 1st ed. McGraw-Hill Osborne Media, 2011.

[56] J. Granjal, E. Monteiro, and J. S. Silva, "Security in the integration of low-power wireless sensor networks with the internet: A survey," *Ad Hoc Networks*, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870514001619

[57] W. Stallings, *Network Security Essentials*, 5th ed. Prentice Hall, 2013.

[58] L. Peterson and B. Davie, *Computer Networks A Systems Approach*, 5th ed., R. Adams and N. McFadden, Eds. Burlington: Morgan Kaufman, 2012.

[59] M. Cheminod, L. Durante, A. Valenzano, and S. Member, "Review of security issues in industrial networks," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 1, pp. 277–293, 2013.

[60] A. Ukil, S. Bandyopadhyay, J. Joseph, V. Banahatti, and S. Lodha, "Negotiation-based privacy preservation scheme in internet of things platform," in *Proceedings of the First International Conference on Security of Internet of Things*. ACM, 2012, pp. 75–84.

[61] N. Modadugu and E. Rescorla, "The design and implementation of datagram TLS," in *Network and Distributed System Security Symposium (NDSS)*, 2004.

[62] Allied Electrical, *Raspberry Pi Model B*, 2013. [Online]. Available: http://www.alliedelec.com/lp/120626raso/

[63] C. Edwards, "Not so humble raspberry pi gets big ideas," *Engineering and Technology*, vol. 8, no. 3, pp. 30–33, 2013.

[64] Postscapes, "Tracking the Internet of Things," p. 1, 2012. [Online]. Available: http://postscapes.com/internet-of-things-platforms

[65] M. Castro, A. J. Jara, and A. F. Skarmeta, "An Analysis of M2M Platforms: Challenges and Opportunities for the Internet of Things," *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 757–762, Jul. 2012.

[66] R. H. Weber, "Internet of Things – New security and privacy challenges," *Computer Law & Security Review*, vol. 26, no. 1, pp. 23–30, Jan. 2010.

[67] SSL.com, "Q10241 - FAQ: What is SSL?" p. 1, 2005. [Online]. Available: http://info.ssl.com/article.aspx?id=10241

[68] SSL Labs, "SSL Server Rating Guide," 2013.

[69] K. Paterson, "On the Security of RC4 in TLS and WPA," p. 1, 2013. [Online]. Available: http://www.isg.rhul.ac.uk/tls/

[70] Ivanr, "CRIME: Information Leakage Attack against SSL/TLS," p. 1, 2013. [Online]. Available: https://community.qualys.com/blogs/securitylabs/2012/09/14/crime-information-leakage-attack-against-ssltls

[71] I. Ristic, "Mitigating the BEAST attack on TLS," p. 1, 2013. [Online]. Available: https://community.qualys.com/blogs/securitylabs/2011/10/17/mitigating-the-beast-attack-on-tls

[72] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, and V. Paxson, "The matter of heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ser. IMC '14. New York, NY, USA: ACM, 2014, pp. 475–488. [Online]. Available: http://doi.acm.org/10.1145/2663716.2663755

[73] B. Villaverde, D. Pesch, R. De Paz Alberola, S. Fedor, and M. Boubekeur, "Constrained application protocol for low power embedded networks: A survey," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, July 2012, pp. 702–707.

[74] V. Ç. Güngör and G. P. Hancke, *Industrial wireless sensor networks: Applications, protocols, and standards*. CRC Press, 2013.

[75] Q. Xianli, F. Mingchao, and S. Bin, "Coal mine gas wireless monitoring system based on WSNs," in *2011 Second International Conference on Digital Manufacturing and Automation (ICDMA)*, Aug. 2011, pp. 309–312.

[76] G. Hancke, K. Markantonakis, and K. Mayes, "Security challenges for user-oriented RFID applications within the internet of things," *Journal of Internet Technology*, vol. 11, no. 3, pp. 307–313, 2010.

[77] A. Abu-Mahfouz and G. Hancke, "Distance bounding: A practical security solution for real-time location systems," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 16–27, Feb. 2013.

[78] A. Melonyan, A. Gampe, and M. Pontual, "Facilitating remote laboratory deployments using a relay gateway server architecture," *Industrial Electronics, IEEE Transactions on*, vol. 61, no. 1, pp. 477–485, 2014.

[79] Digi, "ConnectPort x2 datasheet," 2013. [Online]. Available: http://www.digi.com/pdf/

[80] Digi, "ConnectPort x4 datasheet," 2013. [Online]. Available: http://www.digi.com/

[81] Anonymous, *Special Computing*, 2013. [Online]. Available: https://specialcomp.com/beagleboard/bone.htm

[82] Python Community, *M2Crypto: A Python crypto and SSL toolkit*, 2013. [Online]. Available: https://pypi.python.org/pypi/M2Crypto

[83] Anonymous, "OpenSSL About," p. 1, 2013. [Online]. Available:

http://www.openssl.org/about/

[84] I. F. Akyildiz and M. C. Vuran, *Wireless Sensor Networks*. John Wiley & Sons, Jun. 2010.

[85] Libelium, "Waspmote technical datasheet," 2013. [Online]. Available: www.libelium.com/products/waspmote

[86] Memsic, "Lotus datasheet," 2011. [Online]. Available: http://www.memsic.com/userfiles/files/Datasheets/WSN/

[87] A. Kouche, "Towards a wireless sensor network platform for the internet of things: Sprouts WSN platform," in *2012 IEEE International Conference on Communications (ICC)*, Jun. 2012, pp. 632–636.

[88] C. University, "Firefly 2.2 datasheet," 2010. [Online]. Available: http://www.nanork.org/attachments/download/77

[89] A. Kruger, A. Abu-Mahfouz, and G. Hancke, "A rapid prototype sensor network gateway using commercial off-the-shelf components," in *IEEE International Conference on Industrial Technology (ICIT)*, March 2015.

[90] T. Refaat and R. Daoud, "WiFi implementation of wireless networked control systems," in *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, 2010, pp. 145–148.

[91] U. Varshney, "The status and future of 802.11-based WLANs," *Computer*, vol. 36, no. 6, pp. 102–105, Jun. 2003.

[92] E. H. Ong, J. Kneckt, O. Alanen, Z. Chang, T. Huovinen, and T. Nihtila, "IEEE 802.11ac: Enhancements for very high throughput WLANs," in *2011 IEEE 22nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2011, pp. 849–853.

[93] Anonymous, "Axeda," p. 1, 2013. [Online]. Available: http://www.axeda.com/

[94] G. A. N. Gang and L. U. Zeyong, "Internet of Things Security Analysis," in *Internet*

Technology and Application (iTAP), 2011 International, 2011, pp. 1–4.

[95] Institute of Electrical and Electronics Engineers (IEEE), "IEEE standard for information technology — telecommunications and information local and metropolitan area networks — part 11 : Wireless LAN medium access control (MAC)," 2007.

[96] T. Bingmann, *Speedtest and Comparsion of Open-Source Cryptography Libraries and Compiler Flags*, 2008. [Online]. Available: http://panthema.net/2008/0714-cryptography-speedtest-comparison/

[97] J. Postel, "Transmission Control Protocol," 1981.

[98] W. Stallings, "SSL: Foundation for Web Security," *The Internet Protocol Journal*, vol. 1, no. 1, p. 1, 1998.

[99] Anonymous, "Bit Rates for Live Streaming," p. 1, 2014.

[100] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf: The tcp/udp bandwidth measurement tool," *htt p://dast. nlanr. net/Projects*, 2005.

[101] A. De Rubertis, L. Mainetti, V. Mighali, L. Patrono, I. Sergi, M. Stefanizzi, and S. Pascali, "Performance evaluation of end-to-end security protocols in an internet of things," in *Software, Telecommunications and Computer Networks (SoftCOM), 2013 21st International Conference on*. IEEE, 2013, pp. 1–6.

[102] Unknown, "IBM research - zurich | computer science | mote runner," 2014. [Online]. Available: http://www.zurich.ibm.com/moterunner/faq.html