MASTER'S THESIS

A Distributed Security Scheme to Secure Data Communication between Class-0 IoT Devices and the Internet

James King
2015

Master (120 credits)
Master of Science in Information Security

Luleå University of Technology
Department of Computer Science, Electrical and Space engineering

LULEÅ UNIVERSITY OF TECHNOLOGY

Table of contents

Abstract

Constrained devices are devices with limited resources such as CPU, memory (ROM and RAM), and battery life. These devices often function as sensors collecting information, machine to machine (M2M) or smart devices controlling electrical appliances and services [1]. When these devices are connect to a network they become known as “things” and become part of the “Internet of Things” (IoT), a network of objects such as embedded computers, controllable and intelligent automated devices (smart devices), and sensors with the ability to connect and exchange data with other devices and services [2] [3]. IoT devices are being connected to the internet to allow for the collection and exchange of data with web servers and cloud data centers. Efforts are being made to standardize IoT devices and how they communicate with the web. Data communication in the web is primarily conducted using HTTP which was not designed for constrained environments and carries a lot of overhead. Other protocols for more tailored for IoT such as CoAP has been developed by Constrained RESTful environments (CORE) as part of the Internet Engineering Task Force (IETF) and is a specialized application layer, web transfer protocol designed to be used with devices such as resource constrained IoT [4].

In this thesis security is defined as the protection of data from unauthorized interference or monitoring by ensuring confidentiality, integrity, and authenticity of data. Protocols such as CoAP and HTTP do not secure data transmissions by default. Appropriate security measures must be applied such as cryptography to ensure the security of data. In 2014 the Open Web Applications Security Project (OWASP) ranked the top ten security issues facing IoT devices and placed the lack of encryption in data transmissions as number four on the list [5]. To solve this problem security protocols are needed. Transport Layer Security protocols are added to HTTP and CoAP in an effort to secure communication. HTTP is secured using TLS and CoAP using DTLS [6]. While much work has been done by IETF to minimize the resource requirement, DTLS is still a heavy weight protocol and many IoT devices fall short of the minimum resources needed to support it. The use of cryptographic functions in DTLS adds much complexity and demand for resources on an IoT device [4]. As a result devices with system resources below the minimum of 10KB Random Access Memory (RAM) and 100KB Read Only Memory (ROM) are considered to be too constrained to effectively support the transport layer

security mechanisms needed to provide secure communications over the internet. These devices are known as class-0 devices [1].

This thesis focuses on securing data exchanged between class-0 devices and the internet. A design-science research methodology (DSR) was chosen for this thesis as it helps us to gain knowledge and understanding of the problem domain. DSR is a problem solving paradigm involving the building and implementation of an artifact to solve identified problems. In this case the artifact takes the form of a security solution for class-0 devices. The solution builds on existing research by combining elements of different research solutions to create a more secure solution. This solution helps to solve gaps in security left by existing solutions through the use of symmetric cryptography in data objects and IoT security gateways which act as intermediaries between devices and the internet. The goal of this research is to provide a security solution for devices which do not have the resources necessary to effectively implement the recommended TLS based protocols for secure communication over the internet. The solution provides confidentiality to data traveling between device and gateway while also providing confidentiality, integrity and authenticity to data traveling across the internet. The solution works by delegating demanding security processes to an IoT gateway which securely processes communications to and from the internet using HTTPS (SSL/TLS). Security of data being passed between device and gateway is provided with AES symmetric encryption at the data link and data object layers. The performance of the solution is measured by timing the security process of the IoT device while also measuring the resource requirements of applying the solution to the device. By applying the solution to an IoT test environment we were able to evaluate its performance and ability to satisfy the design requirements set out in the design stage. The solution ensured security confidentiality of data from IoT device to gateway and confidentiality, authenticity, and integrity of data from gateway to server. Using this solution it was possible to perform data object encryption with minimal impact on device resources (additional 0.5Kb RAM and 0.47Kb ROM) and with minimal delay to existing device functionality (additional 0.46 seconds). Heavier security mechanisms were applied at the gateway including TLS/SSL based data communication to the final destination. Deep packet inspection was used to verify data was secured at both stages of transfer.

Glossary

AES	Advanced Encryption Standard
CoAP	Constrained Application Protocol
CPU	Central processing unit
DSR	Design science research
DTLS	Datagram Transport Layer Security
ECC	Elliptic curve cryptography
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
JSON	JavaScript Object Notation
KB	Kilobyte
LAN	Local area network
MHz	Megahertz
MTU	Maximum transmission unit
OWASP	Open Web Application Security Project
PKI	Public key infrastructure
POC	Proof of Concept
PSK	Pre-shared key
RAM	Random-access memory
ROM	Read-only memory
RPi	Raspberry Pi
SD	Secure Digital
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
WPA	Wi-Fi Protected Access

Table of Figures

1 Introduction

In this section we will discuss IoT devices including common protocols and standards used in these devices. The section starts with a description of the devices and an introduction to a classification system for IoT devices based on device resources. Following on from this we will discuss protocols and standards for IoT and how they relate to those of resource rich devices. Finally some background information will be provided into security protocols available to IoT devices and the limitations of these protocols.

1.1 Background

1.1.1 Internet of Things devices and protocols

Constrain devices are devices with limited resources such as CPU, memory (ROM and RAM), and battery life. Devices often function as sensors collecting information, machine to machine (M2M) communication or smart devices controlling electrical appliances or services [1]. When these devices are connected to a network they become known as “things” and become part of what is known as the “internet of things”. IoT is a network of objects such as embedded computers, controllable and intelligent automated devices (smart devices), and sensors, with the ability to connect and exchange data with other devices and services [2] [3]. IoT has many applications such as home automation, manufacturing, environmental monitoring, medical and health care systems, and transportation. By being connected to the internet IoT devices can potentially provide information and services in the physical world to anyone, anytime, anywhere. With IoT users can access their device information which has been uploaded and stored on a web server or cloud storage, and interact with and control their devices through web, mobile, and cloud interfaces and applications.

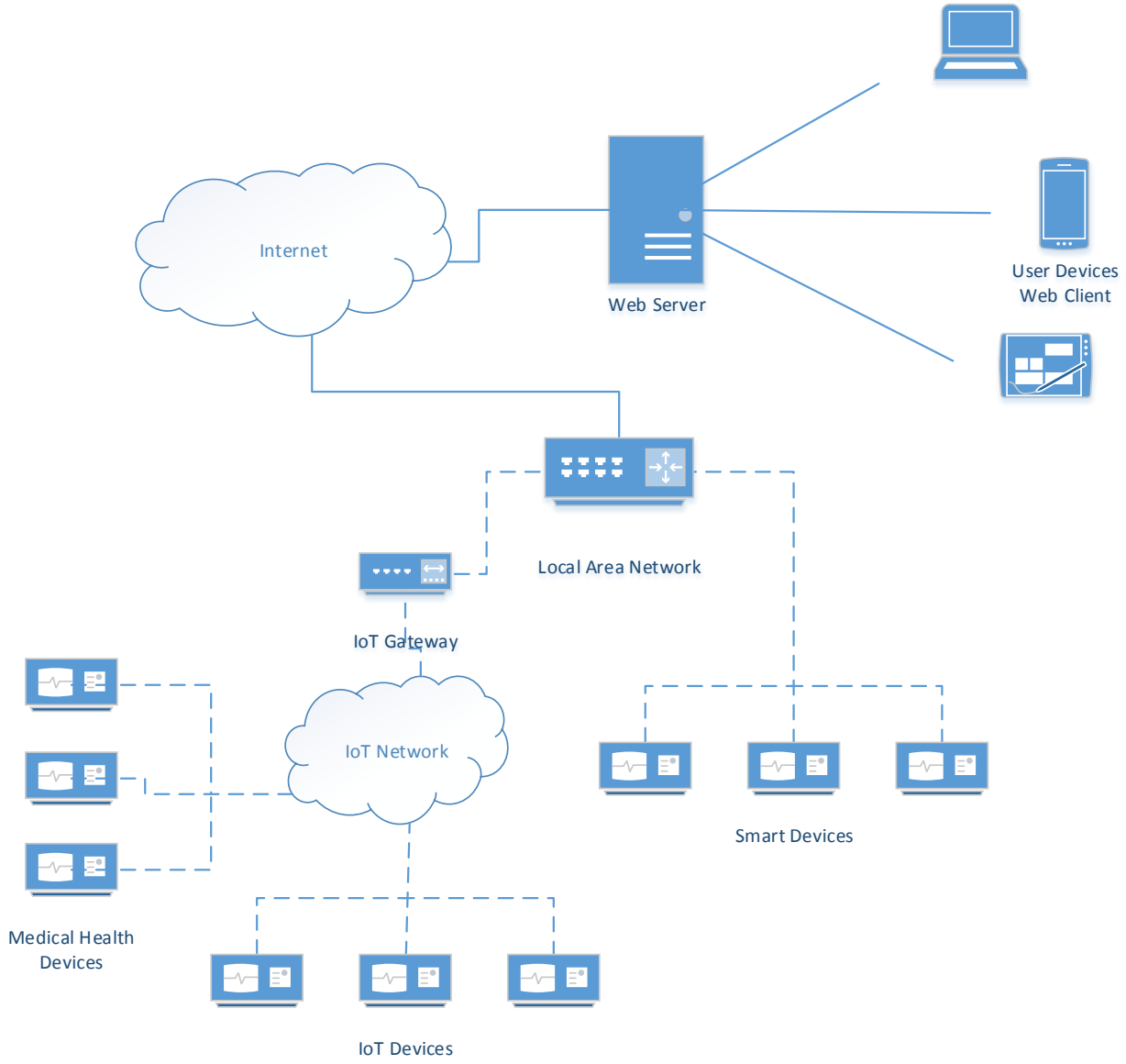

Figure 1 - IoT deployment and application scenarios

Figure 1 depicts a typical IoT network in a home. The figure shows different IoT devices connected directly to an Internet router or by using an IoT gateway which acts as a bridge between the constrained IoT network and the internet. In this figure IoT devices connect to the internet to upload data and to receive requests from a server. Users can access uploaded data from their devices through a web interface and control them using web applications installed on the server.

Devices can communicate using web standards and protocols similar to that of a web stack such as HTTP, MQTT, and CoAP to allow them to interact with, browse, and share data over the internet increasing their usefulness and accessibility [7]. This data may be sensitive in nature such as sensor readings for equipment or medical devices and so precautions must be taken before information is uploaded for analysis or storage. By connecting to and transmitting data over a public network such as the internet, devices and data transmissions become exposed to attack and must be secured. Challenges arise due to the resource constrained nature of some devices and their ability to support the security mechanisms needed for secure communication. A lack of well implemented security may lead to a leakage of personal information collected by these devices [8]. In recent years there has been an increase in the availability of personal sensor devices such as medical and healthcare devices which have the ability to collect medical data about the user and typically have a wireless interface for communication of that data. These IoT devices can be small, wearable, wireless, battery powered sensors which are physically attached to the user's body for the observation of the user's health including blood pressure, heart, and insulin levels [9]. Using a gateway such as a smart phone or wireless receiver these device can communicate this data through the gateway and across the internet to a health professional in a remote locations for analysis and observation [8].

While these devices face many of the security challenges of other IoT devices, such as confidentiality of data, they also face additional challenges due to their highly constrained nature such as supporting resource heavy security mechanisms [9]. Security attacks on personal medical devices are not new. During a cyber-security conference in 2011 a research architect demonstrated the vulnerabilities of a wireless insulin pump by taking unauthorized control of the device and sending commands that would result in lethal dose of insulin for a user. The demonstration showed the lack of encryption and authentication mechanisms within the device making it relativity easy for someone with knowledge of the device to control it [10]. A similar demonstration was given in 2012 but this time using a pacemaker, a device design to control the heartbeat of the user. The demonstration showed how an attacker could potentially delivery a lethal shock directly to the heart from the pacemaker by delivering commands to its wireless receiver [11].

While the focus of this thesis is security of devices which lack adequate resources to support secure data communication many devices with ample resource still suffer security issues. In a study of security and privacy risks from household appliances Notra et. al (2014) examined three popular IoT devices, specifically the Phillips Hue light-bulb, the Belkin WeMo power switch, and the Nest smoke-alarm. The study demonstrates a lack of encryption, appropriate authentication, integrity checks, and privacy implications of these devices. [12] Other examples of data leakage and privacy violations have be seen in the news such as the case with the Samsung SmartTV [13] which allows users to control the TV through voice commands. The TV was also recording those commands and transmitting them to a third-party unbeknownst to the user. To avoid confusion when discussing resource constrained IoT devices Bormann et al. (2014) [1] devised a classifications scheme to differentiate between IoT devices based on their system resources.

Name	**RAM**	**ROM**
Class 0, (C0)	<<10Kb	<<100Kb
Class 1, (C1)	~ 10 Kb	~ 100 Kb
Class 2, (C2)	~ 50 Kb	~ 250 Kb

Table 1 - Classes of Constrained Devices (KB = 1024 bytes)

This table allows us to classify constrained IoT devices based on their available resources. Class-0 devices are considered to be too resource constrained to support secure communication over the internet. Class-1 devices have enough resources to support constrained communication protocols such as CoAP and in some cases transport layer security protocols such as DTLS. The ability to support these protocols effectively may vary depending on their configuration. Class-2 devices are considered to have enough resources available to support heavier web protocol stack such as HTTP over TLS [1]. These resource savings are important when dealing highly constrained devices as the amount of available resources will be a big deciding factor in what security mechanisms the device can support. Each layer of the IoT stack uses protocols and standards that are chosen for their suitability and efficiency in constrained environments. Devices communicate with the web using different protocols such as HTTP and CoAP and in order to standardize communication with the web communication REST (REpresentational State

Transfer) methods are used. REST uses standard HTTP type methods (GET, POST, PUT, DELETE) and different media types such as JSON, XML, and ATOM to retrieve web pages and send data across the internet [14]. Utilizing these web standards and methods allows for a lighter way to manipulate data in a web browser or application making REST for communication in constrained environments such as IoT and smart devices. RESTful systems typically communicate with HTTP [14] but in the case of IoT, CoAP is used to minimize resource overhead [4]. CoAP reduces packet headers from 1000's of bytes to 100's of bytes and when combined with other resource constrained protocols such as 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) a network layer protocol, the size of the information being sent over the network can potentially be reduced even further to 10's of bytes. [15]

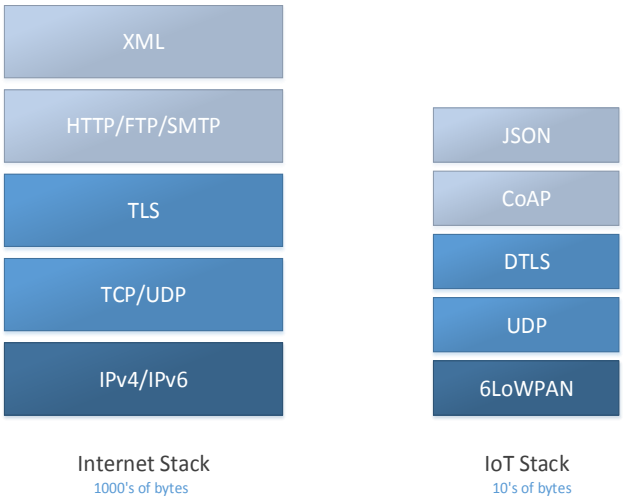

Figure 2 - Comparison between web stack and IoT stack

The object format JSON (JavaScript Object Notation) is also used in the IoT stack instead of other formats such as XML. JSON is a lightweight data-interchange and storage protocol for structuring data in a way which is easy for humans to read and write, and for machines to generate [17]. The design goals of JSON were for it to be minimal, portable, textual, and a subset of the programming language JavaScript. JSON's lightweight nature makes it more suitable for constrained devices than other heavier protocols such as XML where a larger resource overhead is required [18] [19]. Work has been conducted on JSON to add security to the format leading to JSON becoming a popular method of data exchange. The JOSE (JavaScript Object Signing and

Encryption) project adds the ability of encrypt and sign data being transferred, as well as cryptography key exchange and secure data encoding. This is known as object layer security and is considered an additional layer of security on top of existing security protocols. [19]

1.1.2 Security in IoT devices

In this thesis security is defined as the protection of data from unauthorized interference or monitoring by ensuring confidentiality, integrity, and authenticity of data. Confidentiality of data is defined as the protection of data from disclosure to unauthorised persons, parties or systems. Integrity is defined as the preventions of falsification or modification of data by unauthorised persons. Authenticity refers to the verification of the identity of a device or system. [19] [20]

Hypertext Transfer Protocol (HTTP) is an application layer protocol for distributed, collaborative, hypermedia information systems and has been used for data communication on the World Wide Web since 1990 [21]. HTTP as a standalone protocol and is considered insecure as it sends data in plaintext and does not apply security mechanisms to protect data. With the increase of sensitive data being transmitted over the internet a more secure method was required. This lead to the development of HTTP over Secure Socket Layer (SSL) and then its successor Transport Layer Security (TLS). This combination became known as HTTPS, a protocol for secure communication designed to prevent eavesdropping, tampering, or message forgery using cryptography [23]. HTTP and HTTPS run over the transport layer protocol Transmission Control Protocol (TCP) which provides reliability, error protection and flow control for data transmission. These data checking features require additional systems resources to ensure the communication is reliable [24].

HTTP and HTTPS were not design for IoT devices with resource limitation and so a more efficient protocol was developed specifically for constrained resource devices. The Constraint Application Protocol (CoAP) is a specialised application layer protocol design for resource constrained devices such as IoT [4]. Like HTTP, CoAP operates using REST methods and was design so that the two protocols could easily interface with each other [4]. Unlike HTTP which typically runs over TCP, CoAP by default runs over UDP, a less complex protocol which

required less header information than TCP making it more suitable for constrained devices. UDP by default does not check data transmission for errors and so it is considered an unreliable protocol as a result [4]. Due to the constrained nature of IoT, devices have limited resources and so are limited to which protocols and mechanisms they can support. While these devices may have limited functionality they may still have the ability to collect and transmit personal sensitive information across the internet to web or cloud services. This presents challenges as data being exchanged over the internet is potentially exposed to attack and must be secured [3]. In an effort to standardise constrained device communication organisations such as the IETF have developed efficient web standards for constrained devices such as CoAP [4]. Security standards have also been created with the goal of securing IoT data exchange across networks by adapting existing TLS security protocols for constrained devices. The resulting DTLS protocol provides a method for securing data communication in some IoT devices. DTLS protects data confidentiality, integrity, and authenticity of CoAP communications in a similar way that TLS protects HTTP communication on the web i.e. HTTPS [6] [4]. While suitable for some IoT devices, DTLS is still a heavy weight protocol and so devices must have sufficient resource to run it while still being able to perform the devices intended function e.g. temperature sensor collecting data.

1.1.3 Cryptography in IoT

Transport Layer Security (TLS) works by using cryptography to ensure a secure a reliable connection for data communication. [23] Data is encrypted by the sender using the cryptographic public key of the recipient. The data is then sent across the internet to the recipient. Only the recipient's private key will be able to decrypt the data and this key is kept private and secure by the recipient. TLS is also used to establish a secure session by using asymmetric cryptography to secure exchange symmetric keys which are then used for the bulk of the data exchange. Asymmetric cryptography methods require more resources to operate than symmetric cryptography as security handshake and key exchange must take place. As Class-0 devices have limited resources asymmetric cryptography and (D)TLS protocols are too resource demanding for these devices [24].

While asymmetric cryptography may be too resource intensive to secure communication in Class-0 devices, symmetric encryption offers an alternative solution with minimal resource demand [25]. Symmetric cryptography involves encrypting data with a single encryption key which is shared between multiple devices. Any devices which possess the key decrypt data from other devices which possess the key. As the key is shared there is a higher risk that it may fall into the wrong hands and so it must be kept safe. If more than two or more IoT devices share a key those devices become part of a group and are authenticated as part of that group rather than as individual devices. Advanced Encryption Standard (AES) [26] is one such symmetric standard which operates at fast speeds and requires fewer resources than (D)TLS making it very suitable for Class-0 constrained devices [24]. AES inputs data as 16 byte (128-bit) blocks which are then encrypted using a cryptographic key of 128-bit, 192-bit, and 256-bit in size [26]. The larger the key size the greater the security and resource requirement on the device to encrypt and decrypt. Symmetric encryption can be applied at different layers of the communication stack such as the data link layer (e.g. Wireless transmissions) and to specific objects of data with in a message such as sensor readings.

1.2 Problem Statement

1.2.1 Problem Description

Constrained IoT devices have limited resources and so are limited to the protocols and standards they can support. Efforts have been made by groups such as the IETF to develop protocols and standards more suited for constrained environments, such as (D)TLS and CoAP, by increasing efficiency and minimising the resource requirements [6] [4]. Despite these efforts there is still a range of devices which fall short of the minimum resources needed to support such technologies on top of existing application code. These devices are known as “Class-0” devices as they fall short of the minimum threshold (100Kb ROM, 10Kb RAM) to support secure communication using recommended TLS based solutions [1]. (D)TLS is an adaptation of the TLS protocol and has a heavy resource footprint in addition to existing application code in the device [24]. It is considered too resource heavy for Class-0 devices and may perform poorly on some Class-1 devices, with connection times as slow as 24 seconds for a secure transmission [25]. This thesis

examines ways of securing devices too constrained to effectively support the security standards recommended for secure communication. While Class-0 devices lack the resources necessary to effectively support secure communication of data they may still have the capability to transmit data insecurely across the internet or with-in the LAN insecurely [1]. As a result the unsecure communication from these devices would be vulnerable to attack. This falls in line with the OWASP Internet of Things top 10 (2014) security issues which places a lack of transport layer security for IoT devices as number 4 on its list. OWASP recommends avoiding sending any sensitive information unencrypted across networks or the internet by applying SSL/TLS where ever possible or find an alternative method to secure the data. If not addressed, this issue could results in data loss or compromise of account credentials depending on the nature of the data being communicated [5]. An alternative method for securing these highly resource constrained devices is needed.

1.2.2 Research Question

Q: How can security be provided for data being communicated between highly constrained IoT devices and the internet?

1.2.3 Purpose/Research Objectives

The purpose of this research is to provide security of data communications between IoT devices and the internet by examining existing research and building on this research to create a solution that overcomes the challenges inherent to highly constrained devices. A design-science research methodology (DSR) will be used in this thesis to gain an in-depth understanding of the problem area. Through an iterative process of analysis, design, implementation, and evaluation an IT artifact will emerge as a solution to the identified problems. In this case the artifact will take the form of an IT solution for highly constrained IoT data communication security. The objective of this thesis is to solve the identified problems using the designed artifact. Through evaluation and communication the performance of the artifact will be communicated to the reader and the conclusion will be added to the existing knowledge base of this research area.

1.3 Motivations for Research

It is expected that IoT markets will surge over the next five years [26] bringing millions of new IoT devices online and communicating with the web. This study is motivated by the need to secure sensitive data being transmitted across the internet from highly constrained IoT devices. While much work has been conducted into securing data transmissions from constrained devices there is a minimum resource requirement needed to support these security mechanisms. Class-0 devices have been deemed too resource constrained to support secure transmission of data potentially exposing sensitive information to attack. A method to secure data being transmitted from these devices is needed to ensure data privacy and security.

1.4 Delimitations

The scope of this research will be focused around securing the contents of data being transmitted between the device and the internet. The internet destination will take the form of a HTTP server with an enabled web service to receive and process sensor data. This research will not focus on the collection methods, storage or use of data once it has reached its destination but instead the security of the data contents as it is being communicated between source and destination. The focus of this research is to provide additional security to sensitive information which may be communicated from devices too highly constrained to support TLS protocols.

2 Literature Review

The challenge to securing data communication in highly constrained IoT devices is the limited resources available to support security technologies. The amount of available resources in a device also depends greatly on the devices intended purpose, functionality and the remaining resources once this functionality has been implemented. In order for a device to be categorised as Class-0 its resource must be below the resource threshold as outlined by Bormann et al. [1] with less than 100kb ROM and/or less than 10Kb RAM. An example of a Class-0 device is the Arduino Uno [27], an 8-bit microcontroller with 16MHz CPU, 32Kb RAM, 2Kb ROM. The RAM is the processing memory of the application operating on the device while the ROM is the code size of the application. In the case of the Uno the device has sufficient RAM to perform sizable tasks but lacks the ROM (code space) needed for security mechanisms.

The goal of this literature review is to determine if a gap in research exists in securing data communication from highly constrained devices and the internet. The objective of this literature review is to find the most recent and relevant solutions to solving the problems of securing data communication from highly constrained devices to the internet. Different solutions will be examined based on the resource requirements and ability to fully or partially solve the problem. The advantages and disadvantages of each will be discussed at the end of this chapter. Based on this review, comparisons of each solution will be made and gaps in research will be identified if they exist.

2.1 Transport Layer Security

Security of the transport layer for some Class-1 and Class-2 devices can be provided by (D)TLS over CoAP or HTTP. Like HTTP, CoAP as a standalone protocol does not contain the security features necessary to secure data communication. To fix this a variation of TLS was developed to run under CoAP and over UDP called Datagram Transport Layer Security (DTLS) [6]. DTLS contains many features of TLS such as data encryption and authentication, with added features to deal with UDP's unreliability [4]. CoAP over DTLS has been termed CoAPs. The minimal

code size and memory consumption of using DTLS was presented by Kumar et al. (2013) [24] for the IETF in a DTLS implementation guide.

DTLS Implementation requirements		
	ROM (Kb)	RAM (Kb)
State Machine	8.15	1.9
Cryptography	3.3	1.5
DTLS Record Layer	3.7	0.5
Total	15.15	3.9

Table 2: DTLS Memory Requirements in KB

Table 2 presents the minimum memory consumption for a prototype DTLS implementation on a constrained device including State Machine, Cryptography, and DTLS Record Layer. This is simply a guideline as figures will vary with the configuration of the protocol. The more flexible the configuration the more memory required. With this in mind the resources available to the device may have an influence over what type of security implementation is chosen for DTLS. The memory requirements outlined in the table suggest that an effective implementation of DTLS would incur a heavy memory footprint for a constrained IoT device in both ROM and RAM would not be practical in Class-0 devices. At just 2kb the ROM size on Arduino Uno is relatively small compared to Class-1 devices and is far too insufficient to support security technologies such as (D)TLS. Class-0 devices are considered too resource constrained to support TLS protocols such as DTLS and so cannot support secure communication across the internet using this method. While these devices cannot support secure communication they can still communicate with servers on the internet and could potentially transmit sensitive information unprotected. As a result an alternative method for secure communication from Class-0 devices is required.

One solution to this problem would be select more resource rich devices as devices become more powerful and cheaper over time. This strategy would ensure the problem would be overcome for

future devices but as Class-0 devices contain fewer resources they will be cheaper to produce and may remain popular with manufactures for budget IoT devices products to reduce costs [1]. Single purpose legacy devices may also remain in service of long periods of time and as private networks become web-facing these devices may become unintentionally exposed to attack.

An alternative solution suggested by Bormann et al. (2014) [1] would be to implement a security gateway for Class-0 devices. A security gateway acts as an intermediary for the IoT device by applying TLS security to data from the device before it is forwarded to the internet. The gateway would require systems resources large enough to support heavier security and communication protocols required for secure data communication over the internet. This may take the form of a router, microcomputer or smart phone. Doukas et al (2012) [28] released a paper on a security architecture using a gateway to provide security in IoT medical health devices. This paper attempts to solve security and privacy issues surrounding the aggregation of data by health IoT devices with gateways using public key infrastructure (PKI), data encryption, and digital certificates. The architecture offers a device-gateway-destination solution where heavier security processing is handled by the gateway instead of the device. The gateway offers security to data being transported between gateway and internet destination using asymmetric encryption to preserve data confidentiality, integrity, authentication, and authorization. The architecture does not focus on data communication from device to gateway. The gateway received data, processes it and re-encrypted using asymmetric encryption before being forwarded to the server. As a result data passing through the gateway would be exposed during processing and may be vulnerable if an attacker gains control over the gateway. In addition the destination will be able to authenticate data from the gateway but not from the devices.

Due to the complex nature of cryptography it is considered to be a resource heavy mechanism but efforts have been made to test the performance of various cryptography protocols and algorithms on highly constrained devices. Sethi (2012) [29] conducted research on the implementing of an architecture based on CoAP for IoT to add security by providing authentication and integrity of data being passed over the network. An implementation of this research was then applied to different constrained IoT devices to evaluate the performance and practicality of such an implementation. It was determined from the experiment that not only is

asymmetric public-key cryptography on constrained devices possible but some configurations can be efficient on some devices. The configuration of the cryptographic mechanisms depends greatly on the purpose and the resources available (e.g. time critical, limited memory). Some of the configurations did not run effectively on Class-0 devices taking long times to compute or requiring too much memory and so care must be taken in the configuration of the asymmetric cryptography. The results of the experiment showed Ecliptic Curve Cryptograph (ECC) algorithms provided strong encryption in acceptable processing times while requiring minimal RAM from the device. This may offer security solution for the exchange of symmetric cryptography keys between devices, gateway and server. Using asymmetric encryption for exchanging the bulk of data may not be feasible in a Class-0 devices but using it for secure symmetric key exchange followed by the use of symmetric keys for the bulk of data exchange would be more feasible [30].

2.2 Data Object Security

Data encryption can be applied at different layers of the IoT stack. A solution for less resource constrained devices was presented by Vucinic, M. et al. (2014) [31] that proposed a security architecture based on CoAP and uses DTLS for cryptography key exchange. The architecture is known as Object security architecture for the Internet of Things (OSCAR) and it builds the existing DTLS end-to-end security solution by also providing additional features DTLS does not such as caching, mapping to HTTP, and asynchronous message exchange. While this solution is too resource intensive for Class-0 devices it does suggest using object security as part of the architecture by providing encryption for data objects using (JOSE, Cryptography Messaging Syntax (CMS)). [31].

By encrypting data at the object level data passing between device and server could be handled and processed at the gateway without being exposed to the gateway. The data packet would contain an encrypted payload such as a sensor reading which could only read at the final destination. Object layer security exists at the application layer inside the payload of a transmission packet. Objects in this context refer to a container of information which has been

formatted to be human readable. Different data formats exist for the web including JSON, XML, and YAML. It's worth noting that object layer security applies cryptography to a data object but the header information such as source and destination address remains exposed. This can be used as an additional layer of protection but does not serve a comprehensive security mechanism on its own.

2.3 Data link layer security

Symmetric cryptography between device and gateway can be performed in the Data Link layer. IoT devices can use different wireless standards to communicate at the physical and data link layers such as Bluetooth low energy, ZigBee, 6LoWPAN (IEEE 802.15.4) and conventional Wi-Fi (IEEE 802.11). Each method has optional security mechanisms to secure communication between device and wireless Access Point (AP). IEEE 802.15.4 and 802.11n provides security of the wireless transmission with Wi-Fi Protected Access version 2 (WPA2) which utilizes security encryption techniques such as AES to provide data encryption and authorization [32]. This protects confidentiality, integrity and authentication of data from device to AP. While the addition of security has little impact on conventional Wi-Fi communication which has a transmission unit (MTU), IEEE 802.15.4 protocols such as 6LoWPAN are designed to use minimal resources in an effort to save resources and battery power and so adding security has a more significant impact. When stacking additional layers of security such as DTLS on top the remaining space in a transmission unit for device data such as sensor data becomes small. The 802.15.4 standard has a maximum frame overhead of 25 bytes leaving 102 bytes of the frame to spare. When link layer security (WPA2 AES 128-bit) is applied an additional overhead is imposed, e.g. 25 bytes for AES-CCM-128, leaving 81 bytes of space remaining [33].

Applying cryptography at the Data Link layer also has a cost to resources as it required additional data to be transmitted in a data leaving less room for payload information. In a study by Granjal et al. (2012) [34] evaluated the impact of security mechanisms on low-powered devices and application while identifying the limitations of devices to support proposed security mechanisms. They evaluated various cryptography algorithms for performance and resource demand. They noted that at a physical and data link layer 6LoWPAN communications has a low

throughput of 127 bytes and therefor payload space is scarce. This presents problems when applications require larger payloads or security mechanisms are applied as more space is required than what is available resulting in an increase in the number of frames and resources required to transmit data. They recommend finding compromise between transmission security and device functionality as the larger the data payload and resource required by the device software the less available resources for security features. As such resource constraints and device functionality play a factor in the selection of physical and data link layer protocols. While the size of data transmissions being sent from a device are not counted as part of the IoT device classification it is still an important factor to consider as many devices run on battery power and the more data being transmitted the more power being used for that transmission. Considerations need to be taken with choosing and configuring a data transmission standard depending on the devices purpose and available resources.

2.6 Gap in research

In the literature review we have examined the problems in securing data communication from Class-0 devices and what solutions are available to solve these problems. Finding a solution to securing data communications for Class-0 devices is about finding a balance of what security measures are supportable and if there is enough resource remaining for the devices functionality e.g. small sensor device. In this literature review we present existing solutions which tackle different issues surrounding IoT security. Much work has been conducted into securing IoT devices but the threshold of minimum resources needed to support these measures leaves many devices lacking sufficient resources to support secure communication.

Research Solutions	Features	Support
Doukas. et al. (2012) [28] *Enabling Data Protection through PKI encryption in IoT m-Health Devices*	• Gateway to Internet data security w/ SSL. • Suitable for all Class devices. • Does not secure device to gateway.	C0, C1, C2 Protects communication of all device classes across internet.
Sethi (2012) [29] *Security in Smart Object Networks*	• Examination of cryptographic protocols performance in IoT devices. • Testing asymmetric cryptography algorithms. • Add security to CoAP protocol.	C0, C1, C2
Rescorla et al. (2012) [6] *Datagram transport Layer Security V1.2*	• Transport Layer security protocol adapted for constrained devices. • Lower flexibility decreases overhead • Offers confidentiality, integrity, authenticity. • Too heavy resource requirements for C0 and some C1.	C1*, C2 *Depending on configuration and flexibility of DTLS. Elements of solution relevant to C0.
Vucinic, M. et al. (2014) [31] *OSCAR: Object Security Architecture for the Internet of Things*	• (D)TLS based solution. • Additional Object layer security of data payload using symmetric encryption. • Improves on existing DTLS of DTLS protocol with asynchronous traffic, caching, and group communication. • Too heavy resource requirements for C0 and some C1.	C1*, C2 *Depending on configuration and flexibility of DTLS. Elements of solution relevant to C0

Table 3 - Literature Review Comparison

Table 3 compares the different solutions examined in the literature review against its applicability to solving the problem. Each solution solves a different aspect of problems in securing IoT devices. For this thesis the focus is on Class-0 devices and how these solutions help to solve the problems in securing communication from these devices.

Doukas et al (2012) [28] attempts to secure data communication in medical devices with limited resources by deploying an IoT security gateway as an intermediary between device and destination. This solution solve the issues with communication of data over the internet it leaves gaps in security from the device to the gateway. Data may be intercepted before it reaches the gateway or after if the gateway has been compromised. While this solution focuses on communication between gateway and internet other solutions attempt secure data from the device. A solution by Vucinic, M. et al. (2014) [31] designed for Class-1 devices uses an additional layer of security by securing data objects inside a transmission payload e.g. sensor readings. Object layer security with symmetric encryption is possible on Class-0 devices and it would be much less resource demanding that transport layer security [29]. While object layer security on its own does not offer effective security it may be possible to combine it with a security gateway implementation to create a more comprehensive layered security solution. Additional security can be provided between device and gateway at the data link layer be securing wireless transmissions with AES symmetric encryption. As symmetric keys are shared between two or more parties (in this case device and server) it may be necessary to update and manage active keys. A method for secure key exchange can be provided using asymmetric cryptography (RSA 1024-bit) as suggested by Sethi (2012) [29] [30] but would be too resource heavy for bulk data transfer [30].

Existing solutions address different elements of the challenges secure communication of data in Class-0 devices but no single solution provides a full solution to the problem. This thesis proposes to build on existing research by combining different elements of these solutions to create a more secure method of data communication specifically for Class-0 devices.

3 Research Methodology

This thesis will offers a security solution to challenges faced by IoT device with insufficient system resources to support secure communication. The proposed solution will take the form of a method involving symmetric and asymmetric encryption and a security gateway which acts as an intermediary between devices and the internet. Security processes which are too resource

intensive for Class-0 devices are delegated to the gateway where data is processed into a secure form before being transmitted across the internet. This solution will provide confidentiality, integrity, and authenticity of data being transported across the internet and confidentiality of data as it passes between device and gateway. The solution will be designed to meet the requirements outlined in the research objectives. It will then be implemented as a proof-of-concept on a Class-0 device and tested for performance, ability to satisfy requirements and solve identified problems. The purpose of this research is to provide organizations and developers working in the area of IoT development with an alternative solution to some of the security challenges faced by highly constrained IoT devices which will help them to ensure their systems are as secure as possible. The output of this research will act as a guide for the development of similar systems. It is hoped that this thesis will be useful and informative to those who read it and may spark further research in this area. This thesis will also contribute to the existing pool of knowledge in IoT security.

A design-science research methodology (DSR) will be used in this thesis to gain an in-depth understanding of the problem area with the final goal of finding a solution to these problems. DSR will be applied to the problem area as it is proactive problem solving paradigm. In design science, knowledge and understanding of a problem domain and its solution are achieved in the building and implementation of a designed artifact [35]. The artifact in this case will take the form of a security method involving Class-0 IoT devices and a security gateway. Once the system design is finalized the gateway will be implemented as a proof-of-concept. The design process can be broken up into several steps which are outlined below. These steps are based on the design-science research process model (DSRP) [36]. There are different ways to describe the design process but I have chosen the method outlined by Ken Peffers as it puts an emphasis on knowledge creation and flow, and highlights the outputs of each stage.
Figure 3 - Design Science Research Methodology depicts the DSRP at each stage:

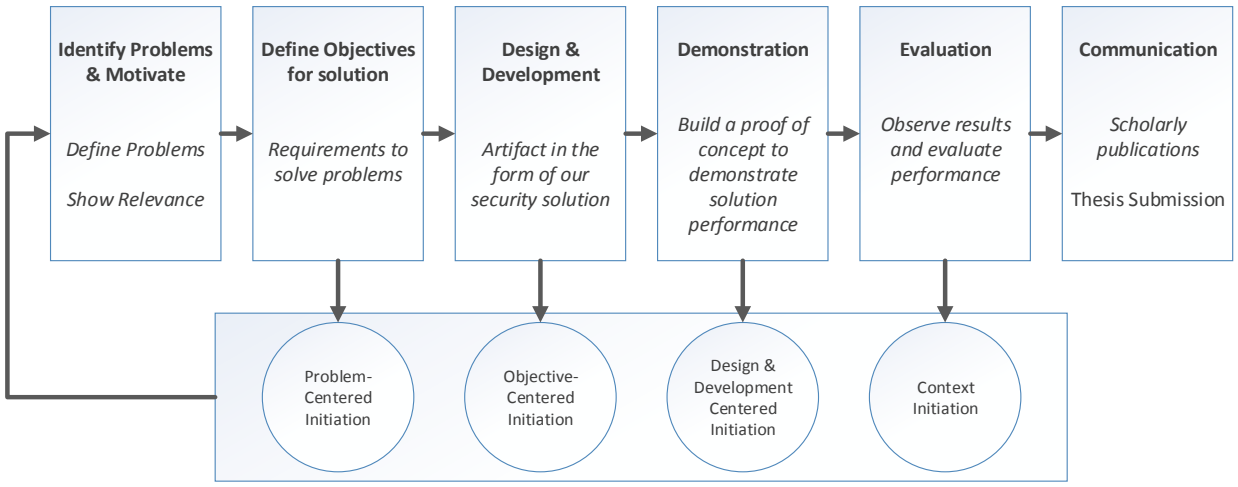

Figure 3 - Design Science Research Methodology [36]

3.4.1 Step 1 - Identify Problem & Motivate

The first step in the design-science process will be to define the problems in detail. The problems identified are centred on the security of data communications between device and internet based destination. Many other security concerns may exist such as the security of data at rest but these are not discussed here as they are outside the scope of this research.

No.	Problem Description
P1	Class 0 devices cannot support secure data communication
P1.1	Class 0 devices may still communicate insecurely with the web
P2	Gateway offers a partial solution but creates a security gap
P2.1	Gateway presents a valuable target for attack
P2.2	Security gap between device and gateway

Table 4 - Problem Summary

P1 - Class 0 devices cannot support secure data communication

Class-0 devices cannot support transport layer security protocols needed to secure communications, such as DTLS. DTLS is considered to be a heavy weight protocol and so not all IoT devices have the resources necessary to support it. Devices which meet the minimum

resource requirements (~100Kb ROM, ~10 Kb RAM) to support the protocols required for secure communication are known as “Class 1” devices. Devices with resources below this threshold (<<100KB ROM, <<10KB RAM) are known as “Class-0” devices and are considered too resource constrained to support secure data communication [1] [24].

P1.1 – Class 0 devices may still communicate insecurely with the web

While Class-0 devices lack the resources necessary to effectively support secure communication of data they may still have the capability to transmit data insecurely across the internet. As a result the unsecure communication from these devices would be vulnerable to interception and attack. This falls in line with the OWASP Internet of Things top 10 (2014) security issues which places a lack of transport layer security for IoT devices as number 4 on its list. OWASP recommends avoiding sending any sensitive information unencrypted across networks or the internet by applying SSL/TLS where ever possible or find an alternative method to secure the data. If not addressed, this issue could results in data loss or compromise of account information depending on the nature of the data being communicated [5]. While projects exist for securing communication between IoT devices and the internet, these are based around the (D)TLS protocol so are inherently unsuitable for Class-0 devices [37]. An alternative method of securing communication between highly constrained class 0 IoT devices and the internet is needed.

P2 – Securing communication between the device and the gateway

One solution would be the implementation of a security gateway between IoT device and the internet as suggested by Bormann et al. (2014) [1]. Heavier security mechanism such as (D)TLS which are too resource demanding for Class-0 devices could be moved to a gateway device residing on the local area network (LAN) between the device and the internet. The gateway would have ample resources to perform the security functions needed to securely transmit and receive data on behalf of the device. While the gateway may offer protection for data from external attacks originating from the internet it would not protect data from internal attacks from within the LAN as communication between the device and the gateway would be unsecured. This would also fail to provide security between the device and the destination and so additional security mechanisms between device and gateway would be required to close the security gap.

P2.1 - A gateway presents a valuable target for attack

Using a gateway as an intermediary raises additional problems. Data may be processed into a more secure form in the gateway before it is transmitted over the internet but the contents of that data would be insecure until it is processed and sent. This makes data vulnerable if the gateway were compromised by an attacker. This presents an additional challenge whereby the contents of the data transmission should only be readable by the device and intended destination. A means of protecting the data contents (e.g. sensor data) inside a data packet is needed in the event the gateway is compromised.

P2.2 - Security gap between device and gateway

Data being sent from the device to the gateway may be exposed to attack if an attacker is listening in on wireless transmissions. Data link layer security may provide security of the transmission but if an unauthorised entity is able to connect to the wireless network they could potentially be able to intercept the data transmission.

3.4.2 Step 2 - Define Objectives

The objectives of the solution will be based on the identified problem from step 1. The objectives are the requirements which need to be met in order to solve the identified problems in Step 1.

No.	Artifact Requirement	Related Problem
R1	Provide security of data between class 0 device and destination	P1, P1.1, P2, P2.1
R2	Secure data transported between gateway and destination	P1.1
R3	Secure data between device and gateway	P2.1
R4	Securing data passing through the gateway	P2.1
R5	Perform efficiently, timely, without data loss	-

Table 5 - Defined Objectives

R1: Provide security of data between class 0 device and destination

In order to provide security of data between class 0 devices and webserver data confidentiality must be provided between source and destination. As a solution to securing Class-0 devices which cannot support the necessary protocol to provide secure communications Bormann et al.

(2014) [1] has put forward the idea of using a gateway to secure traffic between the device and destination. Using a gateway will separate the process of transmitting data into two parts; data transmission between device and gateway, and data transmission between gateway and internet destination. Each part of the journey will require different security mechanisms which are chosen based on what can be supported.

R2: Secure data transported between gateway and destination

Strong security protection must be placed on data traveling over the internet. Using gateways will help to protect data from external attacks by providing additional security to data before it is transmitted over the internet. As recommended by OWASP in order to protect sensitive data being transported strong and correctly implemented TLS and encryption standards should be applied [5].

R3: Secure data between device and gateway

The level of security in data traveling between the device and gateway is limited to what the device can support based on its resources. We know that DTLS is too resource demanding to work on class 0 devices so an alternative is needed in order to close the security gap and prevent attacks on data from the within the LAN. Security must be provided to data traveling between the device and the gateway.

R4: Securing data passing through the gateway

If data is secured between the device and the gateway it may still be vulnerable once it reaches the gateway for processing. If an attacker gains control of the gateway they may be able to access data as it is being processed. This makes the gateway a valuable target and so data must be protected while being processed in the gateway. To protect data contents they should not be exposed to the gateway. A mechanism is needed to protect the contents of data packets while being processed in the gateway.

R5: Preform efficiently, timely, without data loss

While security is the primary goal, performance will also be measured on timing, delay and loss of data being transmissions. The device and gateway must also be able to effectively support the

security mechanisms to secure data while still being able to transmit the data effectively. The resource requirements will be measured against the limited resources available.

3.4.3 Step 3 - Design

In this stage requirements will be analyzed and development decisions will be made and written into the artifact design. This will be the blueprint on which the development process for implementing. Once the design is complete and mapped out a proof-of-concept (POC) will then be implemented from the finished design and prepared for demonstration and evaluation. This is an iterative process and with each stage a review of the requirements and objectives will be made to ensure the system development is on track. *"A design artefact is complete and effective when it satisfies the requirements and constraints of the problem it was meant to solve."* [35].

3.4.4 Step 4 - Implementation and Evaluation

Tests will be conducted on performance and effectiveness of the POC and data will be collected. The performance of the POC will be measured against how well it meets the objectives outlined in step 2 of the DSR process. Data will be collected on the resource requirements and processing times of the POC in the IoT device. Packet analysis will also be conducted to ensure data is secure at each stage of transfer. Conclusions will be drawn based on how effective the system performs against these criteria, and the pros and cons of the mechanism, standards and technologies used. It is hoped that lessons learned through this work will help others developing similar systems. The POC will then be evaluated to see how well it solves the problems identified in Step 1 and satisfies the design requirements outlined Step 3.

3.4.5 Step 5 - Communication

The results from the research will be presented in the thesis, then discussed and reflected upon. A presentation will also be conducted with supervisors and fellow students. It is hoped that these results and reflections will acts as helpful recommendations to those who read this thesis. It is

expected that some area requiring further research will be identified in this stage leaving the door open for other researchers to continue on from this work.

4 Design of Security Solution

This chapter will focus on the design of the security solution. The design is divided into three components: Device, IoT security Gateway, and Web Server. Each component is discussed in detail along with a description of how data is communicated from one stage to the next. The design is based on the design requirements outlined in Step 2:

- Provide security of data between Class-0 device and destination
- Secure data transported between gateway and destination
- Secure data between device and gateway
- Securing data passing through the gateway
- Perform efficiently, timely, without data loss

It is expected that the final solution design will fulfill these requirements and will be used as the foundation for the implementation of the proof of concepts in chapter 5.

4.1 Securing Device to Gateway

In order for a device to be categorised as Class-0 its resource must be below the resource threshold as outlined by Bormann et al. [1] with less than 100kb ROM and/or less than 10Kb RAM. A device like the Arduino Uno (16MHz CPU, 32Kb RAM, 2Kb ROM) may be tasked to controls appliances, actuators, services, or collect data from sensors. Security from between device and gateway can be provided using hardware based symmetric encryption of the Data Link Layer as part of the wireless protocol (e.g. IEEE 802.15.4, IEEE 802.11n). Wireless security modes offer hardware AES symmetric encryption at the Data Link Layer. Figure 7 - depicts the encryption and decryption of data as it passes wirelessly from device to gateway. The method of transfer depends on the available hardware. Wireless transmission can be provided using a IEEE 802.15.4 module such as a ZigBee or 6LoWPAN interface but alternative are available.

When connecting to a network devices are secured with a pre-shared key (PSK) which is installed on each authorised device and is required for communication with the gateway. Any unauthorised devices listening in on traffic will not be able to decrypt data without the PSK. While this protects data from entities without the PSK it leaves data exposed if an attacker manages to compromise the wireless security or capture the PSK from another device.

4.2 Securing Data in the gateway

Security protocols which are too heavy to run directly from the Class-0 device are delegated to the gateway. The gateway acts as an intermediary with ample resources to support these security measures and secure data before sending it over the internet. Data sent from the IoT device will be sent to the gateway using protocols such as CoAP and HTTP and sent across the internet using HTTPS (HTTP over TLS) to the web server. The payload of the packets will be formatted as JSON objects and encrypted using AES 128-bit or 256-bit symmetric encryption. This data object will exist inside the transmission payload while the packet header information such as source and destination address remains unencrypted.

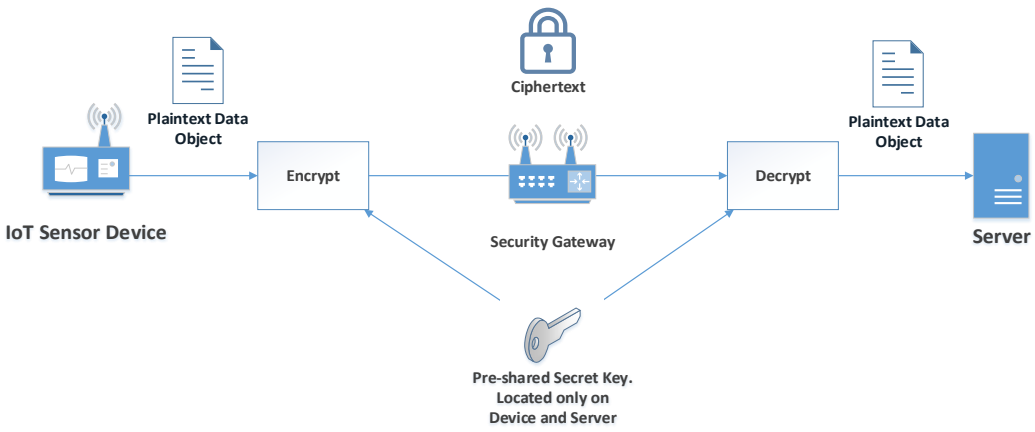

Figure 4 - Data Object Encryption

The JSON object will not be readable by the gateway or any other intermediary entity other than the intended destination. Similarly if the server sends a command back to the device the data

object is encrypted using the pre-shared symmetric key, and is forwarded to the device for decryption.

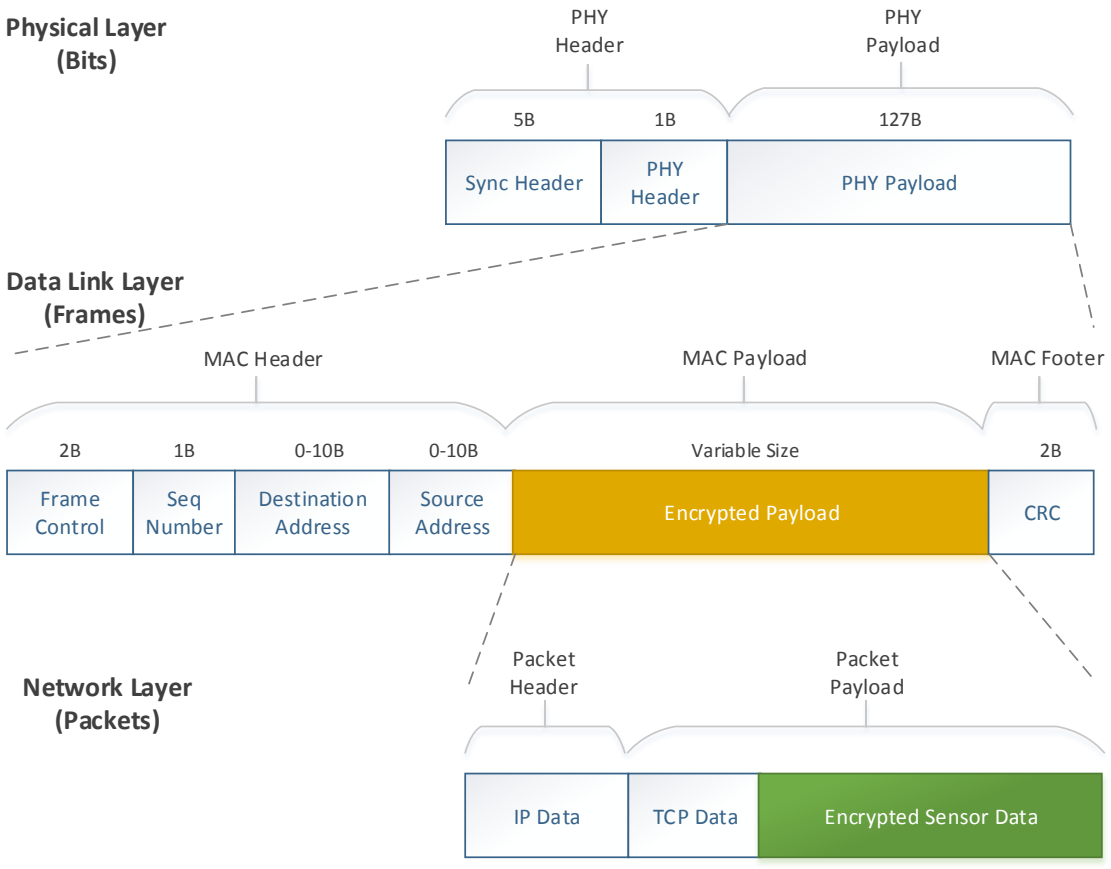

Figure 5 - Device to Gateway Security

Figure 5 depicts the two layers of security applied to data transmitted from the device to the gateway. Security is applied at the Data Link layer in the form of hardware based AES encryption secured with a PSK. Only authorised devices should be in possession of the PSK. The second layer of security is applied to the contents of the data object only. Addressing and source information remain are not encrypted in this layer. The data object is encrypted with a symmetric key which has only been shared with the server so no intermediaries will be able to decrypt the data.

4.3 Securing Data from Gateway to Server

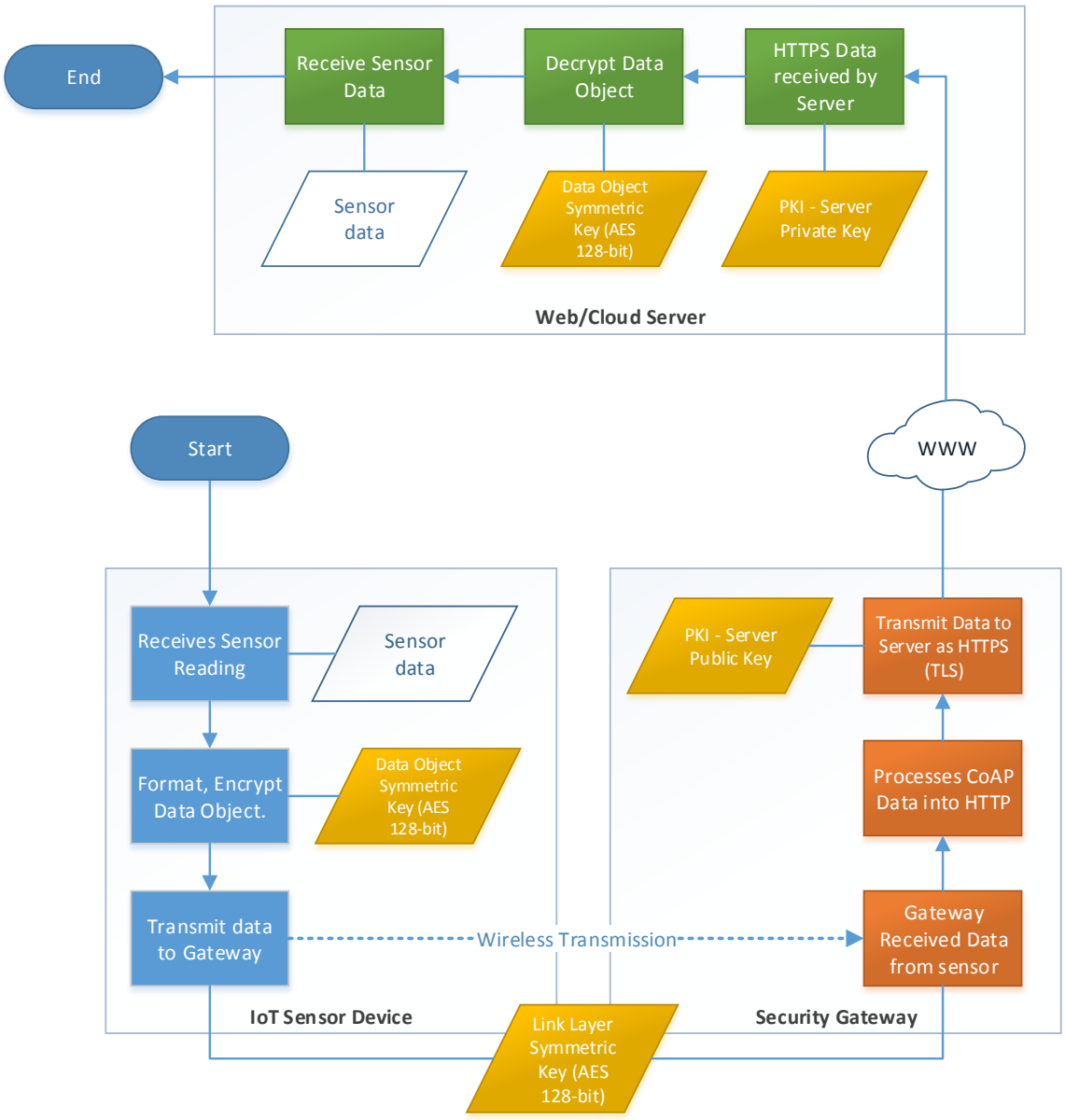

Figure 6 - Device to Server, Solution Workflow

Gateways are computational devices with enough resources to run operating systems and protocols necessary to securely transfer traffic across the internet. A gateway may take the form of a microcomputer with a Linux based operating system. An example of a microcomputer would be the RPi model B has a 700 MHz single core CPU, 512MB SDRAM, Ethernet port, and

an SD-card reader as for on-board storage. The gateway has sufficient resources to apply heavier security and communication protocols which cannot be supported by the IoT device. An addition wireless transmitter can be attached to the RPi to connect wirelessly with IoT devices.

Figure 6 depicts the process of transmitting data from device to server. Data objects (sensor readings) are formatted as JSON and encrypted using the AES 128-bit before being transmitted to the gateway wirelessly. Wireless transmissions are secured using WPA2 PSK and AES 128-bit PSK. Only authorised devices and the gateway possess the PSK so traffic is protected from attackers eavesdropping on wireless transmissions.

Once data is received by the gateway it is processed into HTTPS and prepared for transmission to the server. The gateway is configured with Secure Socket Layer (SSL) tools which are used to create a secure HTTPS connection between gateway and server. From this we can forward secure communications to the server over the internet using asymmetric cryptographic. Messages being transmitted to the server are encrypted with the server's public key which is installed in the gateway. Only the server can decrypt messages using its corresponding private key. The private key is located on the server and is not shared with any other devices. This maintains confidentiality of information as it passes over the internet. Once the HTTPS packets are received by the server they are decrypted using the private key. The encrypted data object can then be decrypted using the symmetric secret key from the originating device, in this case our class-0 IoT device. If the key is only present of a single IoT device and the server it can be used to authenticate data received from either party. If the key is shared with multiple devices the devices are authenticated as part of a group. Data integrity can be provided at the object layer by providing a cryptographic hash of the data and encrypting it with the data before it is transmitted. The destination will be able to perform a check on data received from the IoT device and verify it matches the included hash. While this may be possible through the use of hashing algorithms such as SHA and MD5 it was not implemented in this solution and is an area for further work.

4.4 Processing Data Server Side

On the server side a RESTful web service is setup to receive information from the gateway. A secure HTTPS connection must be established between server and gateway. Data is received data through a secure socket (HTTPS) which is established using the server's privacy certificate. Once data is received the data contents are then extracted. The data is still encrypted using the device/server symmetric encryption key. Using the key and an AES algorithm the data is decrypted, processed and stored in a database. In order for a HTTPS connection to be established the server must first have a security certificate installed. Certificates must also be signed by a trusted certificate authority in order to be trusted. Before the connection between the server and gateway security certificates received must be verified by a trust third party to ensure the connection is not compromised.

5 Implementation and Evaluation

A proof-of-concept (POC) has been developed and demonstrated in detail and its performance will be evaluated. This implementation is based on the solution outlined in the design section. In this section we will talk about each part of the implementation including the IoT device, gateway, and the web server.

Figure 7 gives an overview of the workflow process of each stage, starting from the IoT device through the gateway to the server. Different layers of security are applied to data at the device and the gateway before finally being forwarded across the internet to the server. Object layer security is applied to data at the device before being passed to the gateway over a secure wireless connection. Transport Layer Security is applied at the gateway before data is final passed to the server for processing and analysis.

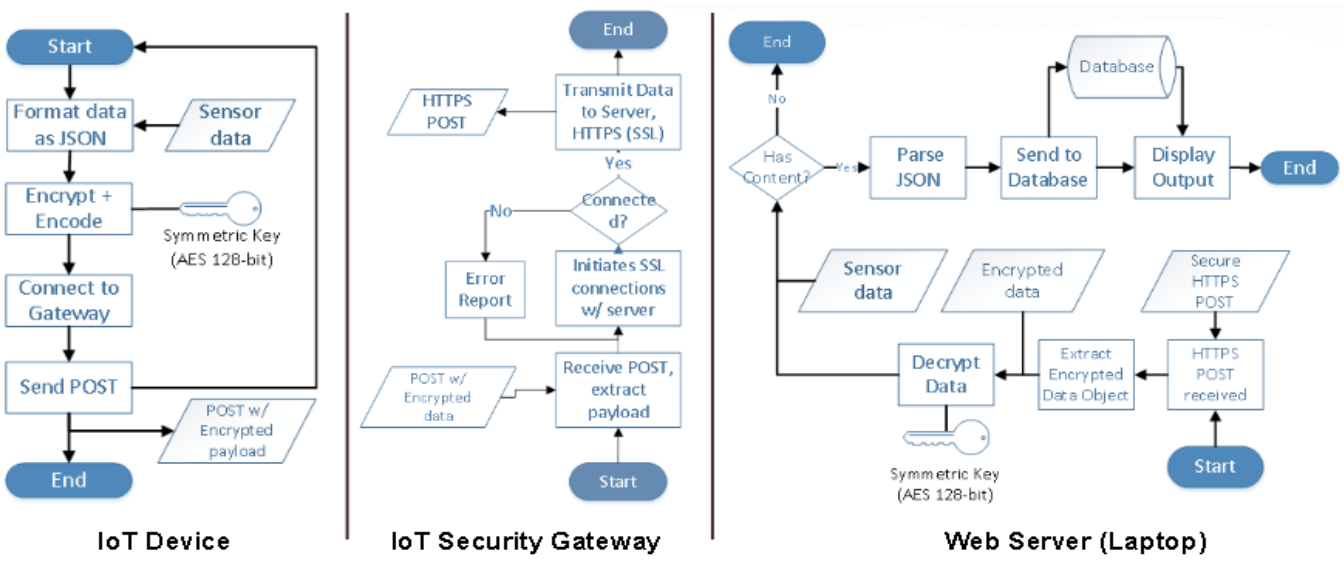

Figure 7 - Solution workflow overview

5.1 Class-0 IoT Device

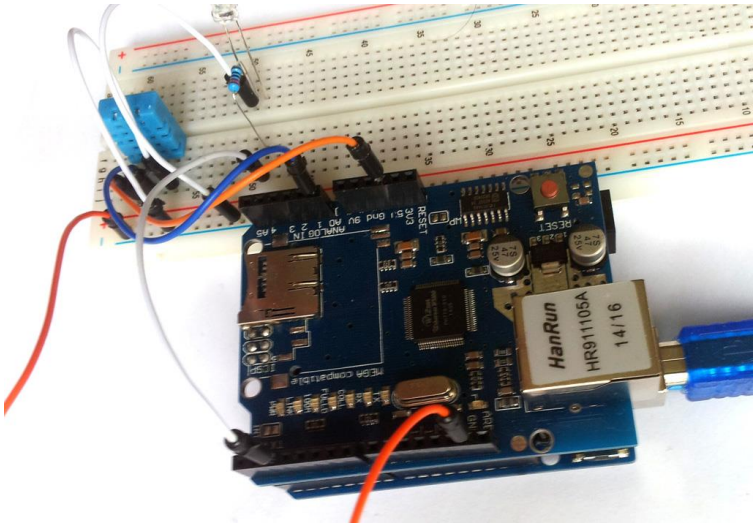

Figure 8 - Arduino Uno + Ethernet Shield + DHT11 Sensor

For the IoT device underlying hardware an Arduino Uno microcontroller was used with an additional Ethernet shield added for connectivity. A wireless shield could be used as an alternative but for the POC the Arduino is connected directly to a wireless router via Ethernet cable. Connected to the Arduino is a DHT11 temperature and humidity sensor. The Arduino connects to and reads data from the sensor and then parses the data into JSON format.

Figure 9 depicts the core functionality of the implementation. The device automatically begins the sensor reading process when the device is connected to a power source and continues to repeat the process until the power is disconnected.

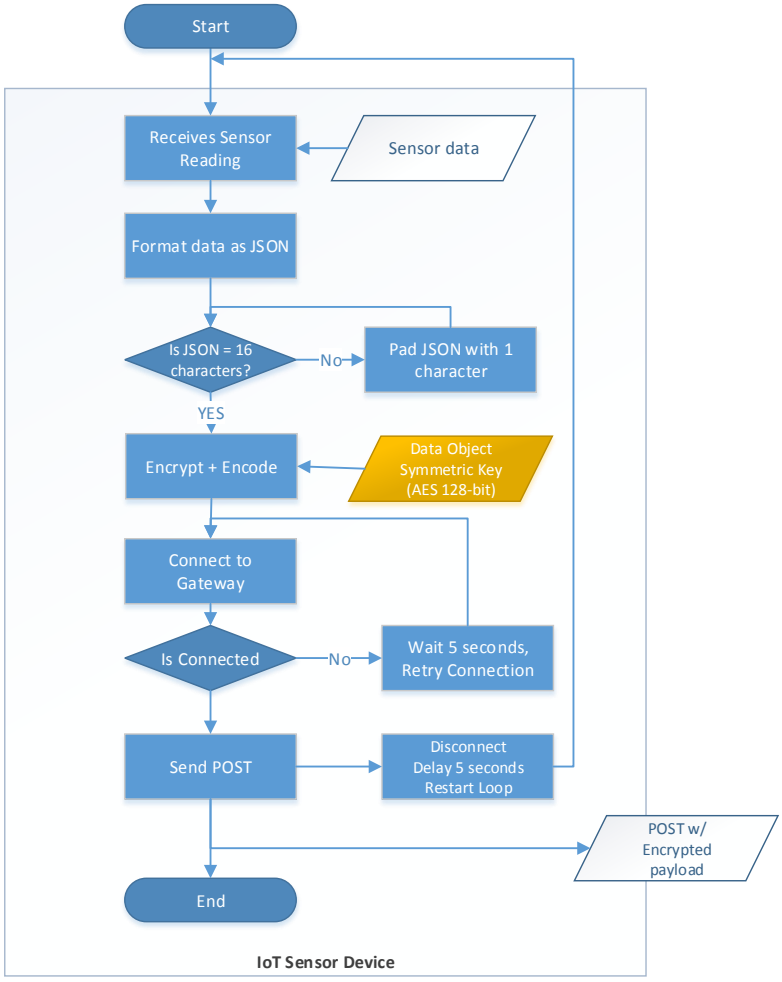

Figure 9 - IoT Sensor Block Diagram

Figure 10 depicts the workflow of the code segment from the Arduino for processing the data. The temperature data is parsed as JSON and padded to 16-bytes, as this is the required block size for AES. The data is then encrypted using an AES 128-bit encryption library [38]. The encrypted output may contain special characters which are not web friendly or human readable so it is encoded to the base64 character set to make it easier to transmit to the server.

```
String data = "{\"temp\":\"";
            data += t;
            data += "\"}";
            while (data.length() < 16) {
                        data += "*";
 }
// Parsed JSON data. Pad data with “*” if less than 16 characters
// “t” = Temperature variable from sensor

uint8_t key[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p'};
// Declare AES 128-bit key
data.toCharArray(dataEnc, sizeof(dataEnc));
// Convent JSON to character array
aes128_enc_single(key, dataEnc);
// Encrypt with AES 128-bit single block cipher
base64_encode(dataEncoded, dataEnc, 16);
// Encode with base64
```

Figure 10 - Arduino code for the JSON data parsing, encryption and encoding

A web client has been setup on the Arduino which establishes a connection to the IoT gateway. Once a connection is established the Arduino uses a POST method to send data to the gateway via HTTP. The encrypted data is added to the contents of the HTTP POST before being sent. CoAP could be used here as a more efficient alternative but would require additional coding and configuration to implement on the Arduino. Once the POST is sent, the Arduino receives a response back from the gateway confirming the POST was received, waits for period of time, and then restarts the processes from the beginning.

```
if (client.connect(server, 80)) {
            Serial.println("Connected");
            client.println("POST /index.php HTTP/1.1");
            client.println("Host: 192.168.2.102"); // RPi address here
            client.println("Content-Type: text/plain");
            client.print("Content-Length: ");
            client.println(sizeof(dataEncoded));
            client.println();
            client.println(dataEncoded);
            Serial.println("Data Sent");
}
else {   Serial.println("Connection failed");        }

// print response from server in serial reader
if (client.available()) {
          char c = client.read();
          Serial.print(c);
}
```

Figure 11 - Arduino POST code

Figure 11 shows the formulation of the POST in the Arduino code sketch. The header information is coded with the destination IP address and web service “index.php”. The encrypted sensor data is added to the contents of the post through the variable “dataEncoded”. If an error is received while attempting to connect to the gateway the response is read when the POST reaches the gateway. If no errors are received the connection is established and the packet is sent.

0000	50 4f 53 54 20 2f 77 65 62 73 65 72 76 65 72 2f	POST /webserver/webclient.php HTTP/1.1
0010	77 65 62 63 6c 69 65 6e 74 2e 70 68 70 20 48 54	Host: 192.168.2.102
0020	54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 31 39	Content-Type: text/plain
0030	32 2e 31 36 38 2e 32 2e 31 30 32 0d 0a 43 6f 6e	Content-Length: 24
0040	74 65 6e 74 2d 54 79 70 65 3a 20 74 65 78 74 2f	
0050	70 6c 61 69 6e 0d 0a 43 6f 6e 74 65 6e 74 2d 4c	ZGioFzoApFk9CfV9XFQhxQ==
0060	65 6e 67 74 68 3a 20 32 34 0d 0a 0d 0a 5a 47 69	
0070	6f 46 7a 6f 41 70 46 6b 39 43 66 56 39 58 46 51	
0080	68 78 51 3d 3d	

Figure 12 - Captured TCP Packet from Sensor

Figure 12 shows a captured TCP packet after it is transmitted from the sensor and reassembled by the gateway. The packet includes the header information such as destination and source address. It also includes the encrypted sensor data in the POST contents i.e. “ZGioFzoApFk9CfV9XFQhxQ==”.

5.2 IoT Security Gateway

Figure 13 - Raspberry Pi Setup

The IoT gateway was built on a Raspberry Pi (RPi) model B [39]. The RPi is a microcomputer with ample resources to perform the heavier security processes too resource intensive for the IoT device. The RPi contains a 700-MHz CPU, 512 MB, and a SD card reader which acts as its storage memory. In this case an 8GB SD card was used for storage. The RPi can be configured with a range of Linux based operating systems. For this solution a version of Debian Linux [40] optimised for the RPi called Raspbian [41] has been used. The RPi connects to the wireless router through a wireless USB adapter. The RPi has been configured with a PSK to access the wireless network which is secured with WPA2 AES 256-bit symmetric encryption.

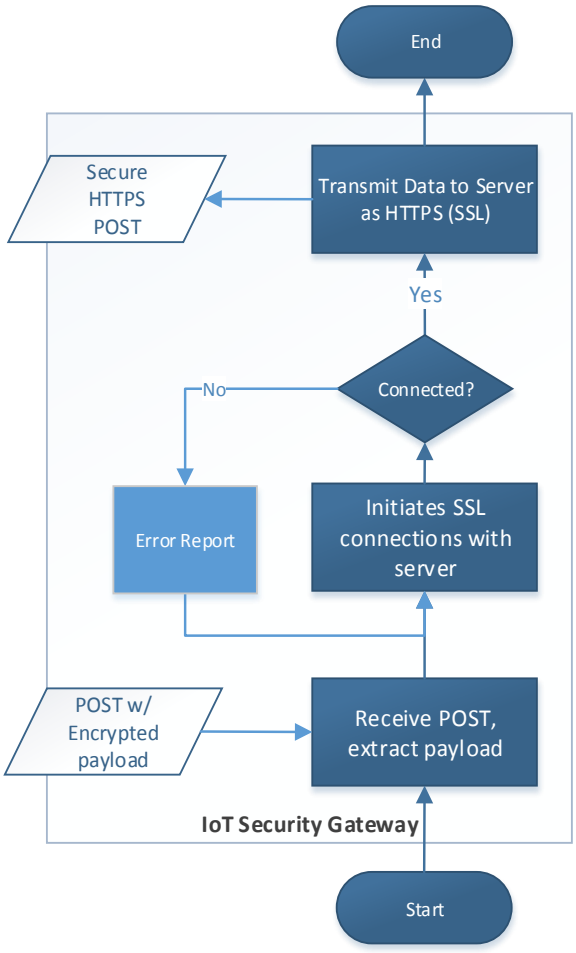

Figure 14 - IoT Gateway Block diagram of web service

```
function forwardJson(){
	$data = file_get_contents('php://input');
	$ch = curl_init('https://192.168.2.100/webserver/webservice.php');
	curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
	curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE); // Disable CA verification
	curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE); // Disable CA verification
	curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
	curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
	curl_setopt($ch, CURLOPT_HTTPHEADER, array(
		'Content-Type: application/json',
		'Content-Length: ' . strlen($data))
	);
}
```

Figure 15 - Gateway code function for processing received sensor data

A web application running on an Apache web server has been setup on the RPi to receive and process data from the IoT device. Figure 15 depicts the core function in the gateway code. When a POST is received from the Arduino the encrypted payload (sensor data) is extracted. The gateway does not contain the symmetric key to decrypt data from the Arduino and simply forwards it to the server over a secure connection. This case the server address is "192.168.2.100". The RPi connects to the server using a SSL connection and posts the data to the server in a HTTPS POST. For the POC the security certificate was not signed by a certificate authority and so when the IoT gateway attempts to connect to the server the verification of the certificate with a trusted third party was disabled in the code (VERIFYPEER and VERIFYHOST). In a real world environment this would be unsafe and by disabling the verification the gateway would not be able to ensure that the connection has not been tampered with.

5.3 Web Server

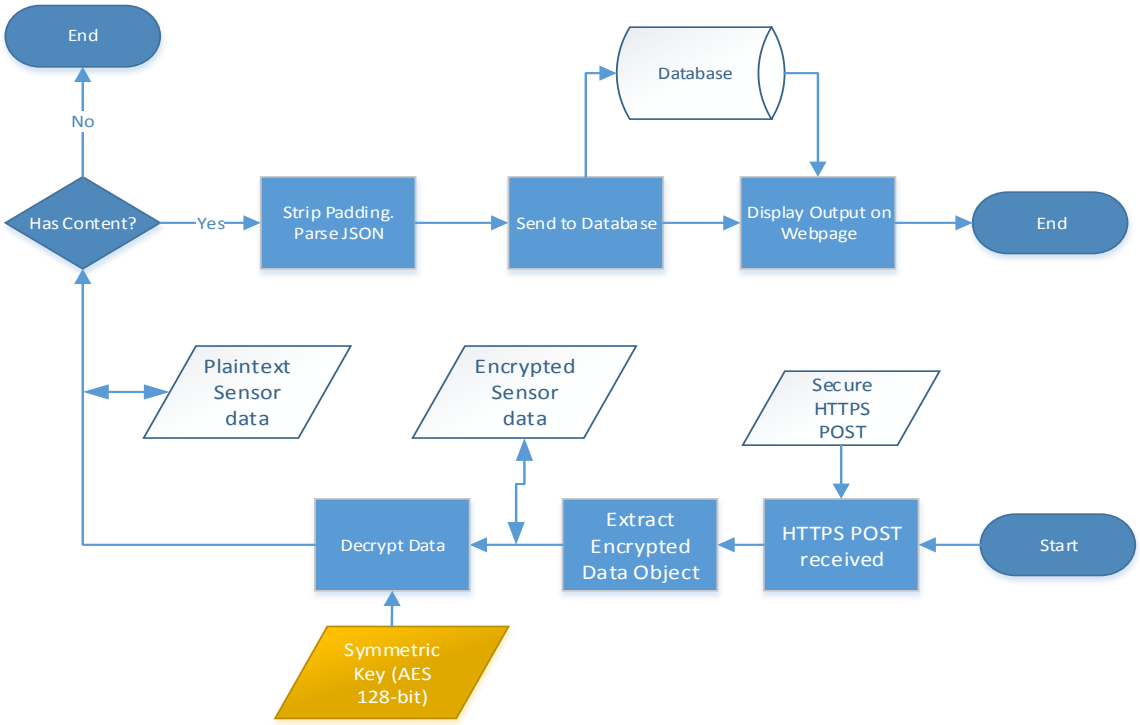

Figure 16 - Web Server Block Diagram

For the POC a server was setup on a laptop within the local area network. This server represents the online server to which data would be transmitted. An apache web server was installed and configured on the laptop. A security certificate was created and the server was setup to receive HTTPS connections using SSL/TLS. When the connection is established by the gateway a POST is sent to the server containing the encrypted sensor data. Figure 16 depicts the web service on the server. The service extracts the data, decodes it from base64 to its original encrypted form, and then runs the encrypted data through a decryption algorithm. When the data is decrypted it is read as JSON and the values are feed into a database. A webpage was also created to demonstrate the working solution. The webpage allows the user to see the latest sensor results which have been stored in the database.

```
function decrypt_data($data, $iv, $key) {
   $cypher = mcrypt_module_open('rijndael-128', '', 'ecb', '');
   if(is_null($iv)) {
      $iv = mcrypt_create_iv(mcrypt_enc_get_iv_size($cypher), MCRYPT_RAND);
   }
   // initialize encryption handler
   if (mcrypt_generic_init($cypher, $key, $iv) != -1) {
         // decrypt
         $decrypted = mdecrypt_generic($cypher, $data);
         mcrypt_generic_deinit($cypher);
         mcrypt_module_close($cypher);
         return $decrypted;
   }
   return false;
}
if($ctext != NULL){         $res = decrypt_data(base64_decode($ctext), null, $key);            }
else{     echo "No data received \n";       }
```

Figure 17 - Web Server code

In Figure 17 we can see how the web service handles the decryption process. The cipher text and the symmetric key are inputted into the function. The cipher text is decoded from base64 into its original encrypted form. It is then processed using a "rijndael-128" cipher which is another reference for AES 128-bit. The final stage of the process is to parse the decrypted output and upload it to a database along with the date and the original encrypted message for reference.

5.4 System Evaluation

The system is evaluated based on its performance and ability in meeting the outlined requirements from 3.4.2 Step 2 - Define Objectives to solving the problems identified in 3.4.1 Step 1 - Identify Problem & Motivate. Below is a table matching each requirement to its corresponding problem. In addition the solution should perform in a timely manner and not be subject to an unacceptable amount of packet loss or failure. It should also be supportable with respect to the resources of Class-0 devices.

No.	Problem Description	Objectives
P1	Class-0 devices cannot support secure data communication	• R1 - Provide security of data between class 0 device and destination. • R5 - Perform timely, minimal data loss
P1.1	Class-0 devices may still communicate insecurely with the web	• R1 - Provide security of data between class 0 device and destination. • R2 - Secure data transported between gateway and destination.
P2	Gateway offers a partial solution but creates a security gap	• R1 - Provide security of data between class 0 device and destination.
P2.1	Gateway presents a valuable target for attack	• R1 - Provide security of data between class 0 device and destination. • R4 - Secure data passing through gateway
P2.2	Security gap between device and gateway	• R1 - Provide security of data between class 0 device and destination. • R3 - Secure data between device & gateway

Table 6 - Evaluation Criteria

The requirement R1 is a general requirement which applied to the whole solution. If requires that the system provide confidentiality of data from the device to destination and integrity and authenticity of data as it passes from device to gateway and from the gateway to the final destination. R1, R2, R2.1, R2.2 relate to the communication of data between the device and the gateway. These requirements are met using encryption of data at the object and data link layer. The solution uses symmetric encryption to encrypt data on the IoT device and transmit it within the contents of transmission. The transmission between the device and gateway are secured with WPA2 AES 256-bit although this is subject to change depending on the transmission technology chosen.

Using cryptography required additional resources from the device. In order to meet R5 the solution must have minimal impact on the existing system resources. In the Arduino the base

application before the encryption functionality was added requires 16Kb RAM (50% of 32Kb total) and 0,5 Kb ROM (28% of Total). Adding cryptographic functions (AES 128-bit CBC) requires an additional 0.5Kb RAM and 0.47Kb ROM. Even with the Arduino minimal ROM size this additional resource requirement would be acceptable for most Class-0 devices (<< 100KB ROM). The processing time to encrypt a sensor reading in the Arduino takes less than 1 second (0.46 sec). This satisfies the requirement R5 - Perform timely, minimal data loss. The requirement may change depending on the application of the device and the nature of its constraints. It is assumed that a resource impact of 0.5Kb RAM and 0.47Kb ROM with an additional processing time of 0.46 seconds is an acceptable burden on most Class-0 systems. Additional efforts could be made to reduce these numbers further using more efficient algorithms and protocols and configuration.

```
// AES 128-bit Single Block Encryption
Plain Text: {"temp":"22"}***
Encrypted Data: dh¨:
Encoded base64: ZGioFzoApFk9CfV9XFQhxQ==

Time: 46msec

// AES 256-bit Single Block Encryption
Plain Text: {"temp":"22"}***
Encrypted Data: k3LøíŠxÖ^oè~vMÛ
Encoded base64: azMcTPjNinjWXm/ofnZN2w==

Time:  57msec
```

Figure 18 - Arduino Encryption Processing Times

In Figure 18 we can compare processing times for two different encryption key sizes. When the key size was increased to 256-bit there was a 25% increase in processing time to 0.57 seconds and 1% increase in ROM usage to 0.63Kb. Again for most applications the increases would be an acceptable and would provide stronger encryption as a result. With AES data is inputted into the algorithm in 16 byte blocks. If the input is larger than 16 bytes it is divided into subsequent blocks. If a subsequent block falls short of the 16 bytes padding is applied to increase the size of the data to 16 bytes. During the test a small single line JSON string was created with temperature

reading from the sensor. The result was 12 bytes in size. 3 bytes of padding were added to the string before it was encrypted and then encoded using base64. The final result increased the data size to 24 bytes, a 100% increase on the initial 12 bytes of data. This increase may require additional resources when transmitting data depending on the transmission method being used. IEEE 802.15.4 based standards such as 6LoWPAN have a limited frame size. When header information and data link layer security is applied such as WPA2 the remained transmission space can be as little as 81 bytes [33].

```
POST /index.php HTTP/1.1
Host: 192.168.2.102
Content-Type: text/plain
Content-Length: 24
ZGioFzoApFk9CfV9XFQhxQ==
```

Figure 19 - HTTP POST sent from Arduino

In Figure 19 we can see the POST that is sent from the Arduino to the gateway. In total the POST size is 119 bytes and so it would have exceeded the maximum size for a single frame transmission in the 6LoWPAN configuration described above. Careful considerations of data sizes must be taken to prevent frames being divided up into multiple segments and requiring additional resources such as battery power to transmit. For this test Ethernet and an external power supply were used so battery power was not a consideration and frame sizes were not as restricted (MTU 1500 bytes). As an alternative to HTTP, constrained communication protocols such as CoAP can greatly reduce the header size of a packet where frame sizes are limited.

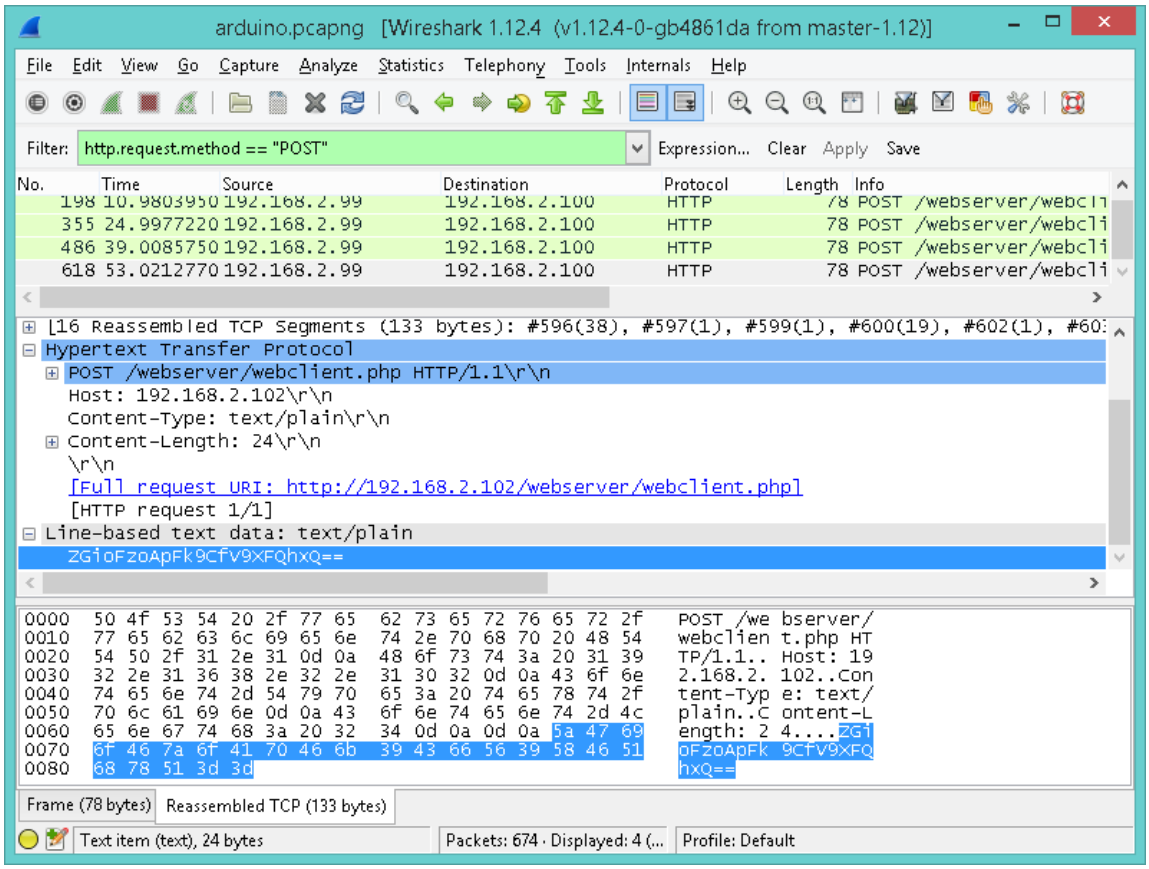

Figure 20 - IoT POST packet capture

Figure 20 shows a captured packet as it is transmitted from the IoT device. From this we can see the encrypted data of the POST highlighted in the bottom of the screenshot. We can also see the header information from the packet which has not been encrypted. An attacker would first need to gain access to the network either through direct access to the gateway or with a PSK for the network to be able to capture the data. With the additional encryption applied to data objects, even if the attacker had access to the network or the gateway the attacker would not be able to read the data without the cipher key that was used to encrypt it. This satisfies the requirements:

- R3 - Secure data between device & gateway
- R4 - Secure data passing through gateway
- R1 - Provide security of data between class 0 device and destination. (Partially satisfies for this leg of the journey)

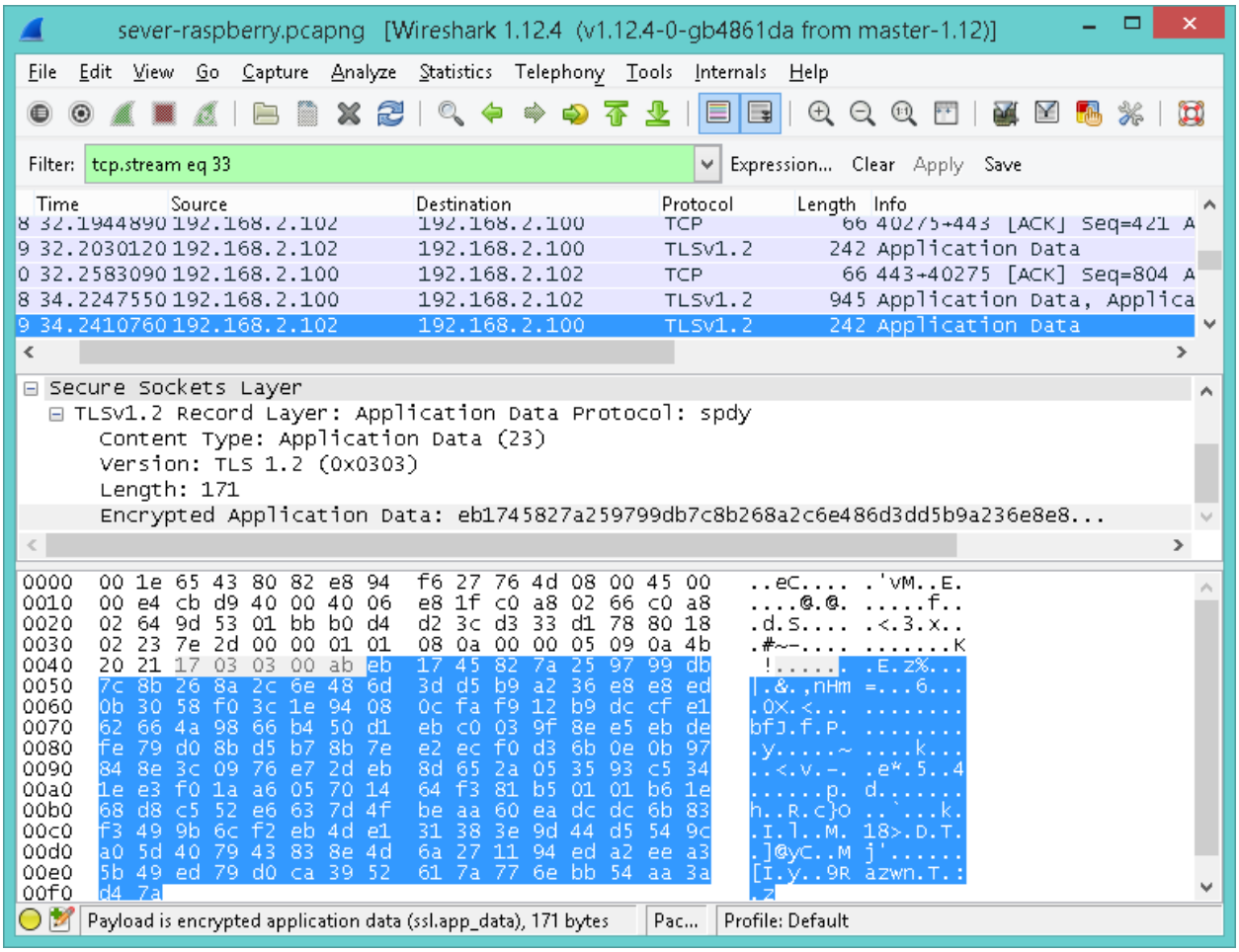

Figure 21 - Web Server Encrypted TLS Packet Capture

When the data reaches the gateway it is processed into HTTPS using RSA 2048-bit and a session key, and securely forwarded to the server. Figure 21 we can see the encrypted application data being received by the server. Using the network protocol analyser called Wireshark we can analyse the traffic exchange between server and gateway. The received data is encrypted using a session key so it is not readable to any unauthorised users eavesdropping on traffic. This satisfies the requirements:

- R1 - Provide security of data between class 0 device and destination. (Partially satisfies for this leg of the journey)
- R2 - Secure data transported between gateway and destination.

6 Conclusions

This thesis has focused on the security of data being communicated between highly constrained IoT devices and the internet. A Class-0 device does not have sufficient resources to support the transport layer security mechanism needed to securely transport data directly to across the internet. This thesis attempts to build on existing research by addressing gaps in existing solutions. Using data object encryption and a security gateway between the device and the internet a more layered security solution has been developed. The security gateway provides a means for sending data from the device across the internet using HTTPS (TLS). An extra layer of protection is applied to data by encrypting its using AES before it leaves the IoT device. Using symmetric encryption confidentiality of data can be secured between device and the intended server destination. Sensitive data being passed to the gateway cannot be read by the gateway and is forwarded securely using TLS to the final destination in its encrypted state. Using data link layer security (WPA2 AES 265-bit) the connection between the device and gateway is further secured against any unauthorised entities try to intercept a transmission. These additional layers of security have an acceptable level of impact on device resources. While outside of the scope it is important to consider the security of the physical device and the storage of encryption keys. If a device is compromised the shared key could be captured and unauthorised entities could intercept traffic.

While integrity of data is provided from the device to the gateway through Data Link Layer security and from the gateway to the server with HTTPS, integrity is not assures directly between device and server. This leaves room for further work by ensuring integrity of data from when it leaves the device or server. While integrity is ensured with SSL/TLS between gateway and server an integrity check is not performed on the device or the server for the data object. Using hashing algorithms such as SHA or MD5 it may be possible to implement an integrity check on a Class-0 device. While an attacker cannot read the data without the encryption key it may be possible to compromise the integrity of the data e.g. replay attack were old data is retransmitted as new. Using a hashing algorithm data can be protected against modification or replication.

Authenticity of data and secure key exchange can be provided with asymmetric encryption. With symmetric encryption if a key is compromised all devices containing that key should be updated with a new key. Key management was not implemented in this solution but through the use of asymmetric encryption it may be possible to introduce it to a Class-0 device. Research has been conducted on asymmetric cryptography on low powered devices and it was determined that it is possible although not recommended for bulk data exchange [30]. Asymmetric encryption is more resource demanding than symmetric encryption so is best used for the secure exchange of keys for faster symmetric encryption algorithms. By using asymmetric cryptographic algorithm such as RSA the authenticity of data can be secured as only the sender will possess the private key for encrypting its data and only the servers corresponding public key can decrypt the message. While this functionality maybe possible it would require additional memory space for key storage. A single public key can be as large as 1.2KB so for a device with limited storage space like the Arduino Uno an external SD card would be required to store keys. Using a RESTful API it is possible to implement a web service on the device to received commands from the server or gateway. Using asymmetric encryption a Class-0 device could be configured to receive a key update if the old key is compromised, out-of-date, or to establish a secure session for a period of time. This is an area for further research.

In order for this solution to be practical in a real work scenario a broader range of devices and functions would need to be deployed and tested with the gateway in a many-to-one scenario. There should also be functionality for devices within the same LAN to communicate with each other through the gateway in a one-to-one scenario. A device-gateway-device scenario could be securely implemented using asymmetric cryptographic keys installed on each device for the exchange of symmetric session key while also using the IoT gateway to perform the heavier security functions. Building on this further multiple gateways could also be connected using SSL to securely connect remote sites to create a secure many-to-many device scenario.

4 Bibliography

[1] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained Node Networks," Internet Engineering Task Force (IETF), Informational 2070-1721 , 2014.

[2] J. Höller et al., *From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence*, 1st ed.: Elsevier, 2014.

[3] R. Khan, Univ. of Genova (UNIGE), Genova, Italy DITEN Dept., S.U. Khan, R. Zaheer, and S. Khan, "Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges," in *Frontiers of Information Technology (FIT), 2012 10th International Conference*, Islamabad, 2010, pp. 257 - 260.

[4] Z. Shelby, ARM, K. Hartke, C. Bormann, and Universitaet Bremen TZI, "The Constrained Application Protocol (CoAP)," Internet Engineering Task Force (IETF), Standards Track 2070-1721, June 2014. [Online]. https://tools.ietf.org/html/rfc7252

[5] D Miessler, C Smith, and J Haddix, "OWASP Internet of Things Top Ten Project," Open Web Application Security Project, Informational 2014. [Online]. https://www.owasp.org/index.php/OWASP_Internet_of_Things_Top_Ten_Project#tab=OWASP_Internet_of_Things_Top_10_for_2014

[6] E. Rescorla, Inc. RTFM, N. Modadugu, and Inc. Google, "Datagram Transport Layer Security Version 1.2," Internet Engineering Task Force (IETF) , PROPOSED STANDARD 2070-1721, 2012.

[7] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices," , 2011, pp. 97–129.

[8] M. Abomhara, Univ. of Agder, Grimstad, Norway Dept. of Inf. & Commun. Technol., and G.M. Koien, "Security and privacy in the Internet of Things: Current status and open issues," in *Privacy and Security in Mobile Systems (PRISMS)*, Aalborg, 2014, pp. 1 - 8.

[9] A. Dohr et al., "The Internet of Things for Ambient Assisted Living," in *Information Technology: New Generations (ITNG)*, Las Vegas, NV , 2010, pp. 804 - 809.

[10] A. Mohan and Golden Valley, MN, USA Honeywell Res. Labs., "Cyber Security for

Personal Medical Devices Internet of Things," in *Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference*, Marina Del Rey, CA , May 2014, pp. 372 - 374.

[11] Stilgherrian. (2011, October) CSO Online - Lethal medical device hack taken to next level. [Online]. http://www.cso.com.au/article/404909/lethal_medical_device_hack_taken_next_level/

[12] Darlene Storm. (2012, October) Pacemaker hacker says worm could possibly 'commit mass murder'. [Online]. http://www.computerworld.com/article/2473402/cybercrime-hacking/pacemaker-hacker-says-worm-could-possibly--commit-mass-murder-.html

[13] S. Notra, UNSW, Sydney, NSW, Australia Sch. of Electr. Eng. & Telecommun., M. Siddiqi, H.H. Gharakheili, and V. Sivaraman, "An experimental study of security and privacy risks with emerging household appliances," in *Communications and Network Security (CNS), 2014 IEEE Conference*, San Francisco, CA, 2014, pp. 79 - 84.

[14] Shane Harris. (2015, February) Your Samsung SmartTV Is Spying on You, Basically. [Online]. http://www.thedailybeast.com/articles/2015/02/05/your-samsung-smarttv-is-spying-on-you-basically.html

[15] Roy Thomas Fielding, *Architectural Styles and the Design of Network-based Software Architectures (Ph.D.)*. Irvine, USA: University of California, 2000.

[16] Raza Shahid, Trabalza Daniele, and Thiemo Voig, "6LoWPAN Compressed DTLS for CoAP," in *8th IEEE International Conference on Distributed Computing in Sensor Systems*, 2012, pp. 287-289.

[17] JSON. (2015, May) Introduction to JSON. [Online]. http://json.org/

[18] Moataz Soliman, Tobi Abiodun, Tarek Hamouda, Jiehan Zhou, and Chung-Horng Lung, "Smart Home: Integrating Internet of Things with Web Services and Cloud Computing," in *IEEE International Conference on Cloud Computing Technology and Science*, 2013, pp. 317-320.

[19] N Nurseitov, M Paulson, R Reynolds, and C Izurieta, "Comparison of JSON and XML Data Interchange Formats: A Case Study," in *CAINE*, 2009, pp. 157-162.

[20] M. Vucinic, Grenoble Alps Univ., Grenoble, France Grenoble Inf. Lab., B. Tourancheau, F.

Rousseau, and A. Duda, "OSCAR: Object security architecture for the Internet of Things," in *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium*, Sydney, NSW , June 2014, pp. 1 - 10.

[21] K Islam, Weiming Shen, and Xianbin Wang, "Security and privacy considerations for Wireless Sensor Networks in smart home environments," *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on* , p. 627, May 2012. [Online]. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6221884

[22] W. Stallings, *Cryptography and Network Security: Principles and Practice 3rd*. NJ, USA : Pearson Education, 2002.

[23] Roy T. Fielding et al. (1999, June) Hypertext Transfer Protocol -- HTTP/1.1. [Online]. https://tools.ietf.org/html/rfc2616

[24] E. Rescorla, T. Dierksv, and Inc RTFM, "The Transport Layer Security (TLS) Protocol Version 1.2," Internet Engineering Task Force (IETF), Standards Track RCF5246, March 2008. [Online]. http://tools.ietf.org/html/draft-ietf-tls-tls13-05

[25] Defense Advanced Research Projects Agency, "Tranmission Control Protocol," Information Sciences Institute, University of Southern California, California , Protocol Specification RFC: 793, September 1981. [Online]. https://tools.ietf.org/html/rfc793

[26] S. Kumar, S. Keoh, and H. Tschofenig, "A Hitchhiker's Guide to the (Datagram) Transport Layer Security Protocol for Smart Objects and Constrained Node Networks," IETF - LWIG Working Group, Internet-Draft 2014.

[27] H. Cam, Arizona State Univ., AZ, USA Dept. of Comput. Sci. & Eng., S. Ozdemir, D. Muthuavinashiappan, and P. Nair, "Energy efficient security protocol for wireless sensor networks," in *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, 2003, pp. 2981 - 2984 Vol.5.

[28] "Announcing the Advanced Encryption Standard (AES)," National Institute of Standards and Technology (NIST) , Federal Information Processing Standards Publication 197 FIPS PUB 197, 2001.

[29] S. Keoh, S. Kumar, O. Garcia-Morchon, Philips Research, and University of Glasgow, "Securing the IP-based Internet of Things with DTLS," LWIG Working Group,

Informational 2013. [Online]. https://tools.ietf.org/html/draft-keoh-lwig-dtls-iot-02

[30] G. Press. (2014, August) Forbes - Internet of Things By The Numbers: Market Estimates And Forecasts. [Online]. http://www.forbes.com/sites/gilpress/2014/08/22/internet-of-things-by-the-numbers-market-estimates-and-forecasts/

[31] Arduino. (2015, March) Arduino Uno. [Online]. http://www.arduino.cc/en/Main/arduinoBoardUno

[32] C. Doukas, Univ. of Aegean, Athens, Greece Dept. of Inf. & Commun. Syst. Eng., I. Maglogiannis, V. Koufi, and F. Malamateniou, "Enabling data protection through PKI encryption in IoT m-Health devices," in *Bioinformatics & Bioengineering (BIBE), 2012 IEEE 12th International Conference*, Larnaca, Nov 2012, pp. 25 - 29.

[33] M. Sethi, "Security in Smart Object Networks," Aalto University, 2012.

[34] Aalto / Ericsson Research, J. Arkko, A. Keranen, H. Rissanen, Ericsson M. Sethi, "Practical Considerations and Implementation Experiences in Securing," Internet Engineering Task Force (IETF), Informational 2012.

[35] S S Kolahi, P Li, M Argawe, and M Safdari, "WPA2 Security-Bandwith Trade-off in 802.11n Peer-Peer WLAN for IPv4 and IPv6 Using Windows XP and Windows 7 Operating Systems," *Computers and Communications (ISCC), 2012 IEEE Symposium*, pp. 575-579, July 2012.

[36] J. Granjal, Coimbra, Portugal Univ. of Coimbra, E. Monteiro, and J. Sa Silva, "On the feasibility of secure application-layer communications on the Web of Things," in *Local Computer Networks (LCN), 2012 IEEE 37th Conference*, vol. October, 2012, pp. 228 - 231.

[37] Alan R Hevner, Jinsoo Park, Sudha Ram, and Salvatore T March, "Design science in information systems research," *MIS quarterly*, vol. 28, no. 1, p. 75, 2004.

[38] Tuure Tuunanen, Marcus A Rothenberger, Samir Chatterjee Ken Peffers, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45-77, 2007.

[39] M. Kovatsch, M. Lanter, and Z. Shelby, "Californium: Scalable cloud services for the Internet of Things with CoAP," in *IEEE - Internet of Things (IOT), 2014 International Conference* , Cambridge, MA, 2014, pp. 1-6.

[40] DavyLandman. (2015, May) DavyLandman - GitHub. [Online]. https://github.com/DavyLandman/AESLib

[41] Raspberry Pi Foundation. (2015, March) Raspberry Pi - Model B. [Online]. http://www.raspberrypi.org/products/model-b/

[42] Debian. (2015, May) Debian. [Online]. http://www.debian.org/

[43] Raspbian. (2015, May) Raspbian. [Online]. https://www.raspbian.org/

[44] Ed. Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann. (2012, November) Internet Engineering Task Force (IETF). [Online]. http://datatracker.ietf.org/doc/rfc6775/