

PERFORMANCE EVALUATION OF LIGHTWEIGHT CRYPTOGRAPHIC
ALGORITHMS FOR INTERNET OF THINGS SECURITY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

SELAHATTİN POLAT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

IN
THE DEPARTMENT OF CYBER SECURITY

JULY 2019

Approval of the thesis:

**PERFORMANCE EVALUATION OF LIGHTWEIGHT CRYPTOGRAPHIC
ALGORITHMS FOR INTERNET OF THINGS SECURITY**

Submitted by SELAHATTIN POLAT in partial fulfillment of the requirements for the degree of
Master of Science in Cyber Security Department, Middle East Technical University by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, **Graduate School of Informatics**

Assoc.Prof. Dr. Aysu Betin Can
Head of Department, **Cyber Security**

Prof. Dr. Nazife Baykal
Supervisor, **Information Systems, METU**

Assist. Prof. Dr. Tolga Yalçın
Co-Supervisor, **SICCS, Northern Arizona University, USA**

Examining Committee Members:

Assoc.Prof. Dr. Cengiz ACARTÜRK
Cognitive Science Dept., METU

Prof. Dr. Nazife Baykal
Information Systems, METU

Asst.Prof. Aybar Can Acar
Medical Informatics Dept., METU

Assoc.Prof. Dr. Sedat Akleylek
Computer Engineering Dept., OMU

Asst.Prof. Dr. Murat Perit Çakır
Cognitive Science Dept., METU

Date: 26.07.2019

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Selahattin POLAT

Signature :

ABSTRACT

PERFORMANCE EVALUATION OF LIGHTWEIGHT CRYPTOGRAPHIC ALGORITHMS FOR INTERNET OF THINGS SECURITY

Polat, Selahattin

MSc., Department of Cybersecurity

Supervisors: Prof. Dr. Nazife Baykal, Assist. Prof. Dr. Tolga Yalcin

July 2019, 42 pages

Widespread deployment of mobile and embedded devices in everyday use has brought up not only new concepts and application areas such as Internet-of-Things (IoT) but also several security and privacy problems. In theory, it is possible to mitigate most of these problems by implementing well-known and standardized security algorithms and techniques on IoT devices. However, in practice, it is rather difficult, if not impossible, to implement standard security algorithms on these devices due to their limited resources and processing power. Instead, algorithms specifically tailored for such devices, which are also known as lightweight algorithms, are favored. In this thesis, we will investigate the suitability and adaptability of the lightweight cryptographic algorithms on IoT devices, and compare their implementations with those of standard algorithms. We will realize our implementations on the Arduino Uno, which is widely used in several embedded applications and preferred as a target development platform for its low price-performance ratio. We will mainly focus on block ciphers and hash functions, which are the fundamental components of many cryptographic protocols. Among these protocols, Internet Protocol Security (IPSec) suite and DTLS are perhaps from the most well-known and commonly used ones. With our study, we plan to provide results that may be guidelines for existing and future lightweight implementations of IPSec, DTLS and other security protocols on IoT devices.

Keywords: IoT security, lightweight cryptography, block ciphers, hash functions, Arduino Uno.

ÖZ

NESNELERİN İNTERNETİ GÜVENLİĞİ İÇİN HAFİF KRİPTO ALGORİTMALARININ PERFORMANS DEĞERLENDİRMESİ

Polat, Selahattin

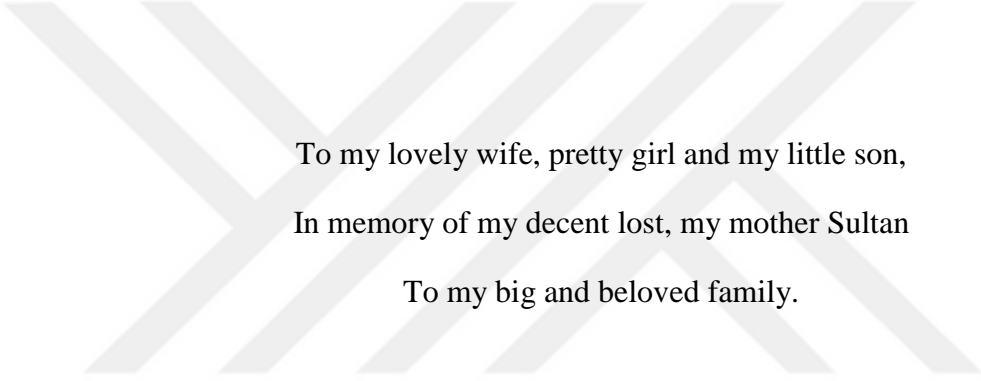
Yüksek Lisans, Siber Güvenlik Bölümü

Tez Yöneticisi: Prof. Dr. Nazife Baykal, Yrd. Doç. Dr. Tolga Yalcin

Temmuz 2019, 42 sayfa

Mobil ve gömülü sistemlerin günlük hayattaki yaygın kullanımı, Nesnelerin İnterneti gibi yeni kavram ve uygulama alanlarının yanında birçok yeni güvenlik ve gizlilik sorunlarını da beraberinde getirmiştir. Geniş boyuttaki internet alanında, uygun güvenlik çözümlerinin kullanımıyla söz konusu problemlerin birçoğunun üstesinden gelinebilir. Bununla birlikte, gömülü sistemlerin sınırlı kaynakları ve işlemci gücü, standart güvenlik algoritmalarının bu cihazlara gerçekleşmesini imkânsız olmamakla birlikte uygulanabilir olmaktan çıkarmaktadır. Bu tezde, hafif kript algoritmalarının uygunluğu ve uyumluluğunu inceleyip, Nesnelerin İnterneti cihazlarındaki performanslarını standart algoritmalarla karşılaştıracğız. Algoritma uygulamamızı, birçok gömülü sistemde yaygın olarak kullanılan ve düşük fiyat ve performans oranı nedeniyle hedef platform olarak tercih edilen Arduino Uno üzerinde gerçekleştireceğiz. Araştırmamızda kript protokolünün temel bileşenini oluşturan blok şifreler üzerine odaklanacağız. İnternet Protokolü Güvenliğı (IPsec) suiti ve Datagram Taşıma Katmanı Güvenliğı (DTLS) protokolleri bunlar arasında en çok bilinen ve ortak olarak kullanılanlardır. Çalışmamızla, Nesnelerin İnterneti cihazlarındaki mevcut ve gelecek hafif IPsec ve DTLS ve diğ er güvenlik protokollerinin gerçeklemeleri için rehber olacak sonuçlar sağlamayı hedefliyoruz.

Anahtar Sözcükler : Nesnelerin İnterneti güvenliğı, hafif kriptoloji, blok şifreler, özet fonksiyonlar, Arduino Uno



To my lovely wife, pretty girl and my little son,
In memory of my decent lost, my mother Sultan
To my big and beloved family.

ACKNOWLEDGMENTS

First of all, I would like to express to gratitude to my supervisors Prof. Dr. Nazife Baykal, Assist. Prof. Dr. Tolga Yalcin for their supports throughout my METU education process.

Besides my supervisors, I would like to thank to the all good METU family. I learned lots of things from METU.

I also would like to thank to the Turkish Army and devote my study to the founder of Turkish Republic Mustafa Kemal ATATURK who remarks the science as the true mentor in life, his comrades and colleagues.

Besides all these thanks, I owe a special thanks and gratitude to the Dr. Tolga Yalcin. This thesis is the result of his relentless support to me.

TABLE OF CONTENTS

ABSTRACT	iv
ÖZ	v
DEDICATION	vi
ACKNOWLEDGMENTS.....	vii
TABLE OF CONTENTS	viii
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF ABBREVIATIONS.....	xiii
CHAPTER	1
1. INTRODUCTION	1
1.1. Lightweight Security	2
1.2. Motivation	3
1.3. Thesis Organization.....	3
2. SECURITY FOR THE INTERNET OF THINGS	5
2.1 IoT Definitions	6
2.2 IoT Protocol Stack for Constrained Devices	7
2.3 The Constrained Application Protocol (CoAP).....	9
2.4 Message Queuing Telemetry Transport (MQTT)	9
2.5 Datagram Transport Layer Security (DTLS)	9
2.6 IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL).....	10
2.7 IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN)	10
2.8 Internet Protocol Security (IPsec)	10
2.9 IEEE 802.15.4 Standard for Low-Rate Wireless Networks	11
2.10. Security Solution Implementations for IoT	11
2.10.1. IEEE 802.15.4 Security Implementations.....	11
2.10.2. IPsec Implementations	12
2.10.3. DTLS Implementations.....	12
2.10.4. Other Implementations.....	12
3. LIGHTWEIGHT SECURITY SCHEMES	13
3.1. Lightweight Block Ciphers	13

3.1.1. Advanced Encryption Standard (AES).....	14
3.1.2. SIMON and SPECK.....	15
3.1.3. RoadRunner	15
3.1.4. Present	16
3.1.5. Rectangle	16
3.1.6. Pride.....	16
3.1.7. SparX.....	16
3.1.8. RC5.....	17
3.1.9. LED	17
3.1.10.LBlock.....	17
3.1.11.Fantomas	17
3.1.12. Skinny	17
3.2. Hash Functions.....	18
3.2.1. MD-5	18
3.2.2. US Secure Hash Algorithm 1 (SHA-1)	18
3.2.3. SHA-256.....	18
3.2.4. SHA-3 (KECCAK).....	19
3.2.5. Keyed-Hashing for Message Authentication (HMAC).....	19
4. EVALUATION METHODOLOGY.....	21
4.1. Choice of Implementation Platform.....	21
4.1.1. Arduino Platform.....	21
4.1.2. Arduino Uno Board	22
4.2 Bench-marking criteria for Block Ciphers	22
4.3. Selection of Cryptographic Algorithms	23
4.4. Code Implementation	24
5. EVALUATION RESULTS	25
5.1. Evaluation of Lightweight Block Ciphers.....	25
5.1.1. Evaluation of Code Sizes	25
5.1.2. Evaluation of SRAM Usage	27
5.1.3. Evaluation of Execution Time.....	28
5.1.4. Evaluation of Throughput	30
5.1.5. Overall Evaluation.....	31
5.2. Evaluation of Hash Functions and HMAC.....	32
6. CONCLUSION AND FUTURE WORK	35
6.1. Limitations.....	36

6.2. Future Works36
REFERENCES.....37



LIST OF TABLES

Table 1. De-facto IoT Protocol Stack	8
Table 2. Look-up Table for the Main Characteristics of the Lightweight Block Ciphers	14
Table 3. Code Size of Lightweight Block Ciphers	26
Table 4. SRAM Usage	27
Table 5. Execution Time of Chosen Block Ciphers.....	29
Table 6. Throughput of Chosen Block Ciphers	30
Table 7. Overall Comparison of Block Ciphers.....	32
Table 8. Hash Functions Characteristics.....	32
Table 9. HMAC and Hash Functions Results	33

LIST OF FIGURES

Figure 1. Number of IoT connected devices.....	6
Figure 2. IoT Three-tiered Architecture.....	7
Figure 3. Features and scope of an IoT system.....	8
Figure 4. Publish/subscribe process utilized by MQTT.....	9
Figure 5. Block Cipher Code Sizes in Bytes on Arduino Uno	26
Figure 6. SRAM Usage of Lightweight Block Ciphers	28
Figure 7. Execution Time (Per Bit) of Lightweight Block Ciphers	29
Figure 8. Throughput (bytes/ms) of Lightweight Block Ciphers.....	31



LIST OF ABBREVIATIONS

3G	Third Generation of Wireless Mobile Telecommunications Technology
6LBR	6LoWPAN Border Router
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
AES	Advanced Encryption Standard
AH	Authentication Header
AMQP	Advanced Message Queuing Protocol
ARM	Advanced RISC Machines
ARX	Modular Addition, Rotation, Xor
ASM	Assembly Language
CIA	Confidentiality, Integrity, Availability
COAP	Constrained Application Protocol
CPU	Central Processing Unit
DDS	Data Distribution Service
DTLS	Datagram Transport Layer Security
EEPROM	Electrically Erasable Programmable Read-Only Memory
ESP	Encapsulation Security Payload
ETSI	European Telecommunications Standards Institute
FELICS	Fair Evaluation of Lightweight Cryptographic Systems
HMAC	Hash Based Message Authentication Code
HTTP	Software Development Kit
HW	Hardware
ICSP	In Circuit Serial Programming
ID	Identity
IDC	International Data Corporation
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IoT	Internet of Things
IP	Internet Protocol
IPsec	Internet Protocol Security
IPv6	Internet Protocol version 6
ISO	International Organization for Standardization
ITU	International Telecommunication Union
I/O	Input / Output
KB	Kilobyte
LED	Lightweight Encryption Device

LLN	Low-Power and Lossy Network
LTS	Long Trail Design Strategy
LWC	Lightweight Cryptography
MAC	Media Access Control
MCU	MicroController Unit
MD4	Message Digest Algorithm 4
MD5	Message Digest Algorithm 5
MHZ	Megahertz
MQTT	Message Queuing Telemetry Transport
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OSI	Open Systems Interconnection
PDA	Personal Digital Assistant
PHY	Physical
PKI	Public Key Infrastructure
PWM	Pulse Width Modulation
RAM	Random-Access Memory
RFC	Request For Comment
RFID	Radio-Frequency Identification
ROM	Read-Only Memory
RPL	IPv6 Routing Protocol for Low-Power and Lossy Networks
SHA	US Secure Hash Algorithm
SPN	Substitution-Permutation Network
SRAM	Static Random-Access Memory
SSL	Secure Sockets Layer
SUN	Smart Utility Networks
SW	Software
S-BOX	Substitution Boxes
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
USB	Universal Serial Bus
W3C	World Wide Web Consortium
WPAN	Wireless Personal Area Networks
WTS	Wide Trail Design Strategy
XMPP	Extensible Messaging and Presence Protocol

CHAPTER 1

INTRODUCTION

When Tim Berners-Lee introduced World Wide Web at CERN in Switzerland in the 1980s (A short history of the Web, 2019), he probably had no idea that his invention would eventually lead to a world where even kinder garden kids would be playing with Internet tablets and their parents would be using smart phones with much higher processing power than the supercomputers of his time. It is without a doubt that information technology has developed with an unprecedented speed and changed our lives in a way that even the wildest science fiction movies could not have predicted. Star Trek communicators lack the ultra-high definition (UHD) display capability, which is almost standard to any middle-to-high range smart-phone we use today (Star Trek (film), 2019). We can do almost anything from the comfort of our homes, from buying groceries to the most complex banking operations, thanks to the advances in information technology within the last two decades. It is like a fairy tale of technology.

However, as with all fairy tails, there is a dark side to this one as well: This much advancement has also brought unconceivable problems with it, security being the primary problem. There is almost no single day without news of a new Internet fraud scheme. We all are now somehow familiar with terms like hacker/hacking, virus, spyware, phishing, spam, etc., which either had a completely different meaning before Internet, or didn't exist at all (Cambridge Dictionary, 2019).

Fortunately for the end users, and unfortunately for the people who are trying to maintain security, the Internet as we know it is not the end. It is just the beginning. The next step in this connectivity dream (or nightmare) is connecting “things”! At first, we were more than happy with high-speed Internet on our computers. Then came the mobile Internet on our phones, tablets and smart phones, which made us even happier. But it hasn't stopped there. Now, scientists and engineers are connecting everything to the Internet, whereas this everything includes home appliances (to build “smart homes” of the future), medical

devices (for real-time patient-monitoring), transportation vehicles (not just for monitoring but also for smart/autonomous driving), buildings, manufacturing lines, agricultural facilities, etc. The list goes on forever... Soon we will be able to monitor a bottle of orange juice through its evolution from a single seed all the way to its bottling and arriving at the neighborhood (or even online) market. Smart devices like Amazon's Alexa already performs science-fiction-like tasks on a daily basis: It wakes up its owner, turns on his/her favorite music, and while he/she is taking the morning shower, turns on the toaster (Getting Started with the Alexa Skills Kit, 2019). This new evolution is known as the Internet-of-Things (IoT).

This all sounds so fancy and unbelievable. But it is already reality, in a way a scary reality. No one wants his/her Alexa to be hacked, sending his/her banking information, or even simply morning home videos to third parties, nor wants the toaster to start a home fire.

With the "classical" Internet, it is somewhat easier to handle. Cryptographic algorithms and security procedures have been developed in parallel with the Internet technologies. The processing capacity of modern computers are more than enough to handle even the most complex cryptographic algorithms. Furthermore, developers of Internet had foreseen future security problems and have developed Internet Protocol Security (IPSec) suite in parallel (K.Seo, 2005).

Unfortunately, it is hard to say the same for IoT environments, which mostly consist of resource-constrained devices. No manufacturer wants to put a processor into the toaster, if the processor's price is more expensive than, or even close to, that of the toaster. As a natural consequence of their environments and usage scenarios, IoT devices have limited capabilities and resources in terms of processing power, memory and battery. Most of these devices do not even have any security support, which is unacceptable! Cryptographic algorithms and security schemes which can cope with the limited capabilities of these devices are necessary. This brings us to the notion of "lightweight security".

1.1. Lightweight Security

As its name implies, lightweight security deals with the same security concerns as its conventional counterpart, however in a reduced, i.e. "lightweight", capacity. It too covers a wide range of topics including both cryptographic algorithms and protocols. Despite being a relatively new topic, it has been attracting researchers, thanks to the rapid deployment of IoT. The main question in field is to achieve "sufficient" level of security given the limited capability of the target platform, which in fact is a challenging target. Conventionally, security within an algorithm is ensured by the complexity of the underlying security algorithms and hence the corresponding operations. These operations do not only require heavy processing, but are also memory and storage consuming. Clearly, it is a practical impossibility to provide any of these on an IoT platform. New

algorithms and approaches have to be introduced. This alone is also not sufficient. Implementation of these algorithms have to be optimized for certain platforms.

1.2. Motivation

Optimized implementation of lightweight security schemes is the main focus of our study. However, we limit our focus only to cryptography part of lightweight security, i.e. lightweight cryptography. It too is a very wide subject, and covers both hardware and software implementations. While there has been tremendous amount of research study and even commercial work done on hardware implementations (Nemati, et al., 2015). Software part is somehow neglected, at least in relative terms. Most of the software development in lightweight cryptography is on asymmetric algorithms (Hosseinzadeh J. H., 2016). Symmetric algorithms are not as thoroughly investigated. Therefore, we will only focus on software implementations of lightweight symmetric cryptography, which covers both lightweight block ciphers and hash functions.

Software implementation of a lightweight cryptographic algorithm requires selection of a certain platform or a family of platforms. In our case, we made a bold decision to go with a very cheap device and see the limits of resource efficiency and performance we can reach. We selected Arduino Uno as our development platform and implemented all our algorithms on this platform.

Algorithm selection was also another challenge for us. While it is a relatively new subject that covers only the last decade, lightweight cryptography has become overly popular and countless algorithms has been developed by cryptographers all around the globe. While some algorithms specifically targeted certain products, like mCrypton from Samsung (Lim C.H., 2006), some of them were developed for solely scientific curiosity in an effort to challenge existing algorithms in terms of a certain optimization target. In fact, NIST has recently launched a lightweight cryptography competition in an effort to end this chaos and standardize a lightweight algorithm or a family of algorithms for IoT applications (Lightweight Cryptography, 2019).

With our study, we aim to provide guidelines to researchers and designers, who may want or need to explore capabilities and requirements of software implementations of symmetric cryptography algorithms on IoT platforms. Therefore, we tried to select a set of algorithms, which have attracted highest attention from the cryptography community and have been intensively implemented on different platforms. We hope that our implementations on this selection of algorithms will also provide guidelines for the NIST competition.

1.3. Thesis Organization

In Chapter 2, we give a brief summary of security protocols that are used for Internet of Things. This is crucial for a better understanding of the in-depth literature review of

lightweight security schemes presented In Chapter 3 together with lightweight security considerations. A detailed definition of IoT and security solutions for IoT are also introduced in this chapter. An overview of the algorithms implemented in this thesis is given as well.

It is followed by Chapter 4, where we explain our evaluation methodology, together with information regarding the implementation platform, namely Arduino Uno board. Research and implementation methodology for the target algorithms and the benchmarking criteria as well as implementation details (libraries used, parameters set, etc.) are also discussed in this chapter.

In Chapter 5, implementation results are presented and resource evaluations are made with respect to these results.

Finally, in Chapter 6, we summarize our conclusions, challenges and limitations we have faced, and possible directions for future research

CHAPTER 2

SECURITY FOR THE INTERNET OF THINGS

Internet of Things (IoT) will be the future of Internet, which will connect not only traditional computers, but also countless number of devices (things) we use in our daily lives ranging from tablets, smart phones to the smallest appliances at home. Most of the IPv6 network traffic will be initiated by IoT, maybe not in terms of average packet size, but number of packets for sure (Raza, 2013). IoT will be the means to combine several smart/dummy things with the help of Internet to constitute a smart synergy between humans and objects which surrounds them. Furthermore, cloud services will constitute to an indispensable part of IoT by incorporating data collected from “things” used everyday users and enterprises into the limitless collective of information and knowledge (Ray, 2016). Data mining algorithms will mine this data and make right decisions for the systems and services (Chovatiya, 2017).

It is practically impossible to have an exact numbers of devices that connect to the Internet every day. Even predictions vary considerably. Figure 1 depicts figures by statista.com, according to which, 75.44 billion IoT devices are predicted to be installed between 2015 and 2025 (Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions), 2019). International Data Corporation ([IDC](#)) forecasts IoT spending reaching up to \$1.2 trillion in 2022. (IDC Forecasts Worldwide Technology Spending on the Internet of Things to Reach \$1.2 Trillion in 2022, 2019)

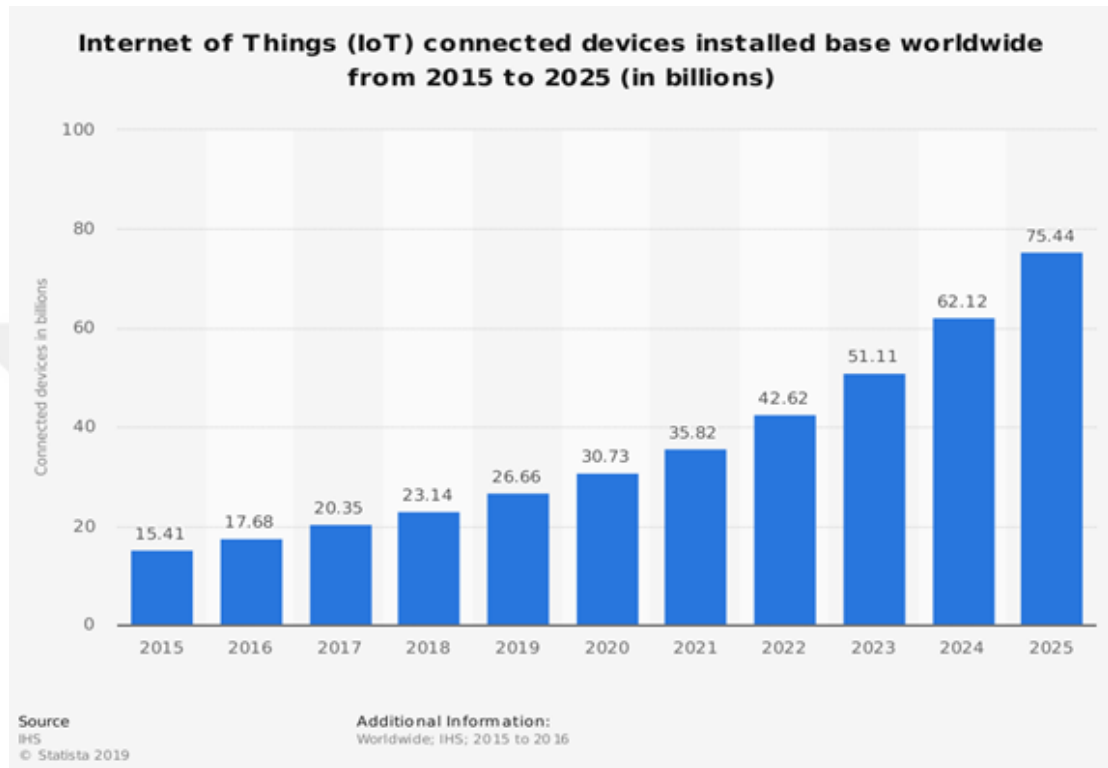


Figure 1. Number of IoT connected devices

2.1 IoT Definitions

The term “Internet of Things (IoT)” was first used by Kevin Ashton, a technologist from England and also the co-founder of the Auto-ID Center at the Massachusetts Institute of Technology, in 1999 (Kramp T., 2013). It has seen been the widely accepted terms for ubiquitous computing devices and even formal definitions have been made by standardization organizations.

According to IETF, “Internet of Things is the network of physical objects or "things" embedded with electronics, software, sensors, and connectivity to enable objects to exchange data with the manufacturer, operator and/or other connected devices.”

Gartner defines IoT as “the network of physical objects that contain embedded technology communicate and sense or interact with their internal states or the external environment” (The Internet of Things, 2019).

United Nations' information and communication agency ITU defines the IoT as “ubiquitous network” which is consist of networks and networked devices “available anywhere, anytime, by anything and anyone” (History of IEEE, 2019).

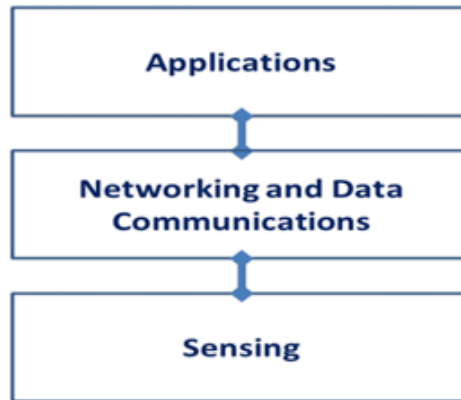


Figure 2. IoT Three-tiered Architecture.

Institute of Electrical and Electronics Engineers (IEEE) devised IoT as “Internet of Things envisioning a self-configuring, adaptive, complex network that interconnects ‘things’ to the Internet through the use of standard communication protocols and stated a three-tier architecture of IoT as shown in Figure 2 (The Internet of Things, 2019). According to IEEE, the interconnected things have physical or virtual representation in the digital world, sensing/actuation capability, a programmability feature and are uniquely identifiable. The representation contains information including the thing’s identity, status, location or any other business, social or privately relevant information. The things offer services, with or without human intervention, through the exploitation of unique identification, data capture and communication, and actuation capability. The service is exploited through the use of intelligent interfaces and is made available anywhere, anytime, and for anything taking security into consideration.” Figure 3 is a depiction of this definition. (Towards a definition of the Internet of Things (IoT), 2019).

2.2 IoT Protocol Stack for Constrained Devices

Many standardization groups and organizations made efforts to standardize the IoT protocols such as World Wide Web Consortium (W3C), Internet Engineering Task Force (IETF), Institute of Electrical and Electronics Engineers (IEEE) and the European Telecommunications Standards Institute (ETSI) (A. Al-Fuqaha, 2015). In this section, we will summarize these protocols, and give detailed information for the ones chosen within the scope of this thesis.

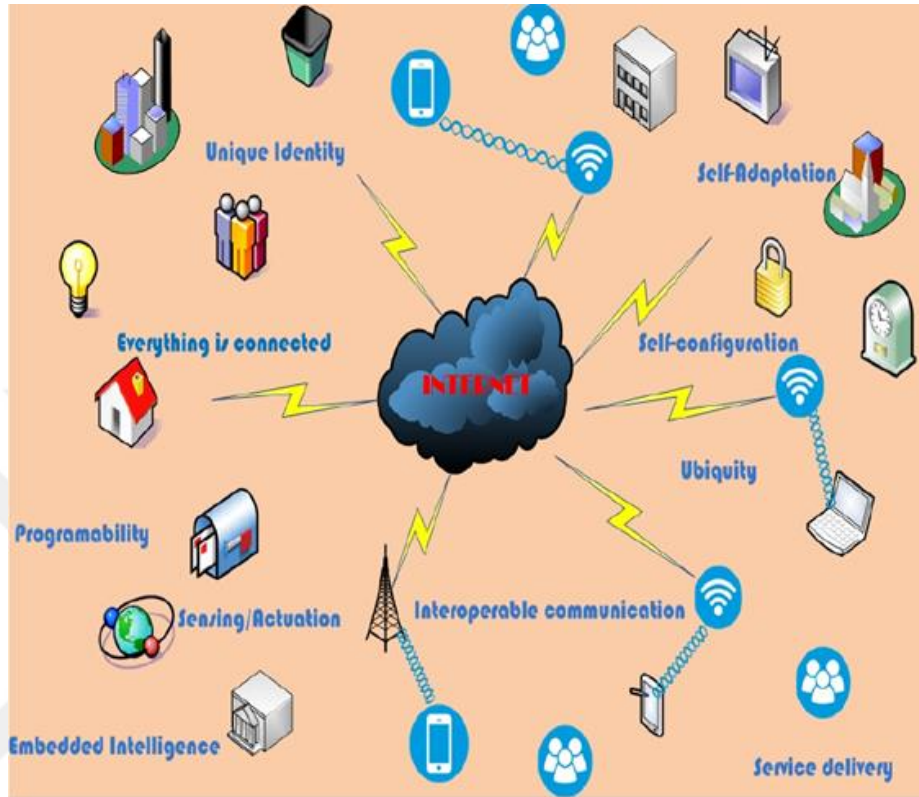


Figure 3. Features and scope of an IoT system.

Table 1 gives a high level picture of the de-facto IoT protocol stack according to Internet Protocol (TCP/IP) Suite.

Internet Protocol (TCP/IP) Layers		IoT Protocols		IoT Security Protocols
Application		COAP, MQTT, AMQP, XMPP, HTTP		User-defined
Transport		UDP		DTLS
Network	Routing	RPL		RPL security
	Encapsulation	6LoWPAN	IPv6	IPsec
Link		IEEE 802.15.4		802.15.4 security

Table 1. De-facto IoT Protocol Stack

2.3 The Constrained Application Protocol (CoAP)

CoAP is a web protocol designed by Internet Engineering Task Force (IETF) for constrained devices and networks. By constrained devices, usually 8 bit MCUs with tiny RAM and ROM are referred to. CoAP make transfer of data for such devices. It is not just a compressed version of HTTP. It does more than that by fulfilling special requirements such as scarce power, memory, small processing resources and machine-to-machine communication of IoT devices. CoAP uses UDP as transport layer protocol. It has URI and Content-type support. The protocol ensures asynchronous message exchanges. With the use of Datagram Transport Layer Security (DTLS), CoAP can be secured at the transport layer (Shelby, 2014).

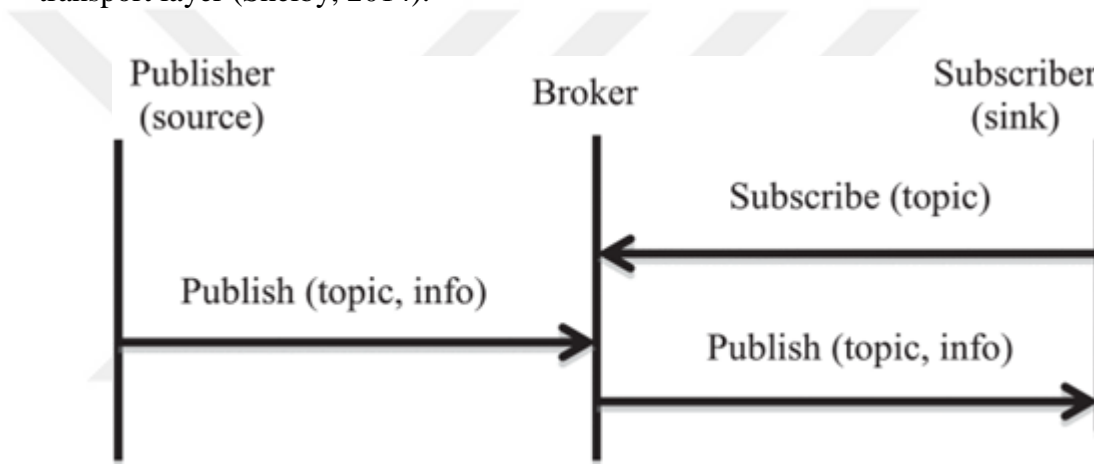


Figure 4. Publish/subscribe process utilized by MQTT.

2.4 Message Queuing Telemetry Transport (MQTT)

MQTT is a data centric message transfer protocol with a client-server topology for the machine-to-machine communication. It uses publish and subscribe pattern to determine the unicast, multicast and broadcast message distribution. Data is sent from device to a broker and published to the client by the broker. Figure 4 shows the publish/subscribe process of MQTT protocol. It is a lightweight message transfer protocol thanks to small overhead and low exchanged data. MQTT was originally designed by IBM, but now it is an open protocol (Andrew Banks, 2019). There are also application protocols such as XMPP, AMQP, HTTP, AMQP and DDS, for which a comparison is given in Table 2 which will help to give a general understanding of these protocols (A. Al-Fuqaha, 2015).

2.5 Datagram Transport Layer Security (DTLS)

DTLS protocol is used to secure the transport layer. It ensures security and privacy of applications. DTLS is a variety of Transport Layer Security (TLS) protocol. While it guarantees similar security, it also protects the datagram semantics. DTLS protocol builds

on the idea of “TLS over datagram transport”. TLS uses TCP, while DTLS instead uses UDP, which in fact is the main designing reason of DTLS in the presence of TLS. DTLS protocol uses cipher suites and cryptographic algorithms to enhance privacy (N.Modadugu, 2012).

2.6 IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL)

RPL is an IPv6 routing protocol for Low-Power and Lossy Networks (LLNs) which is introduced by IETF in March 2012. RPL is designed to work in LLNs which utilizes constrained devices as both nodes and routers. The protocol makes routing possible in environments with high loss and low data rates. RPL is constructed for one-to-one, one-to-many or many-to-one traffic types. There are security fields in RPL protocol which ensures integrity and replay protection. These additional fields allow security features such as confidentiality and delay protection (T. Winter, 2012) (Negi, 2017).

2.7 IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN)

6LoWPAN is an adaptation protocol between IPv6 and Low-Power and Lossy Networks (LLNs) which is designed by IETF. It enables transmission of IPv6 packets to the LLNs, specifically IEEE 802.15.4. The starting point of the protocol was the opinion that IP should be implemented on all networks and devices, so that all information systems can be connected with one protocol. The protocol fragments the big IPv6 packets, compresses header – sometimes even 4 bytes, in order to make them suitable for IEEE 802.15.4.

6LoWPAN is designed according to “pay as you use” principle. The packet size is adjusted with respect to amount of transmitted data. This idea will help the constrained devices that sends just small packets to serve longer time periods (Mulligan, 2007) (G. Montenegro, 2007).

2.8 Internet Protocol Security (IPsec)

IPsec is a group of protocols and services that ensures the security of IP layer. These services and protocols fulfill several security requirements. Security provided to the communication at the IP layer also ensures security at the upper layers. This is one of the main advantages of the IPsec solution.

IPsec provides privacy with encryption, ensures the integrity of the message, prevents replay attacks, facilitates exchange of Public Key Infrastructure (PKI) keys and negotiation of security parameters. The protocol supports two security modes—transport and tunnel, for different network scenarios. In order to provide these services, IPsec uses three main protocols:

- IPsec Authentication Header (AH) - provides integrity and authentication,
- IPsec Encapsulation Security Payload (ESP) – provides encryption for privacy,
- Internet Key Exchange (IKE) – protocol to exchange keys and decide on security associations.

These three protocols use several other sub protocols to provide mentioned services. (K.Seo, 2005) (S.Kent, IP Authentication Header, 2005) (S.Kent, IP Encapsulating Security Payload, 2005) (S.Kent, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC7296, 2014).

2.9 IEEE 802.15.4 Standard for Low-Rate Wireless Networks

802.15.4 is a wireless standard developed by The Institute of Electrical and Electronics Engineers (IEEE) for the transmission of data through low data rate and low power consumption wireless personal area networks (WPANs). The standard describes the physical (PHY) and media access control (MAC) layer of the Open Systems Interconnection (OSI) model for WPANs. While PHY specifies modulation, frequency, power and other remaining specifications of the link, MAC describes the data scheme.

802.15.4 standard defines some subcategories for different application areas including industrial applications, active (battery powered) radio-frequency identification (RFID) uses, smart utility networks (SUNs) and various other country-specific applications. It is the de-facto wireless standard for WPANs. (IEEE Standard for Low-Rate Wireless Networks, 2016)

2.10. Security Solution Implementations for IoT

We introduced the protocol stack for IoT at the above. Now we will have a look to the security solutions and their implementations. IoT needs multi-layer security solutions to provide confidentiality, integrity and authentication. To provide these security requirements, there are standardized solutions; IEEE 802.15.4 Security, IPsec, DTLS.

2.10.1. IEEE 802.15.4 Security Implementations

It has a security sublayer which provides node-to-node security at the link layer (IEEE Standard for Low-Rate Wireless Networks, 2016). This standard uses Advanced Encryption Standard [AES] cipher suites for security services. Hui Lin implemented IEEE 802.15.4 security services in his study and showed the practicability of the standard (Lin, 2014). But implementing the just IEEE 802.15.4 security services just provides node-to-node security and for each node the same procedure is repeated. IEEE 802.15.4

Security doesn't provide an end-to-end security for the system and not replace network and transport layer security solutions.

2.10.2. IPsec Implementations

IPsec ensures end-to-end security for IoT. However, in its original form, it is a "heavy" protocol for the IoT. As a result, there has been considerable research in order to come up with optimized implementations which will reduce IPsec's heavy burden on constrained devices.

In a study where Raza et al implemented a lightweight version of IPsec, he presents a specification for AH and ESP headers and concluded that lightweight IPsec is feasible for WLANs. He further asserts that using lightweight IPsec is a better alternative than IEEE 802.15.4 security services (S. Raza S. D., 2011).

Glissa and Meddeb proposed a new security protocol implemented at adaptation layer-6LoWPAN and ensures end-to-end security like IPsec. They named it 6LowPsec. The protocol constitutes end-to-end security channel between nodes and the 6LoWPAN Border Router (6LBR) by using IEEE 802.15.4 security services (Glissa, 2018).

2.10.3. DTLS Implementations

Raza and others also implemented lightweight DTLS for the CoAPs. They compressed DTLS header and showed the possibility of using this for the CoAPs (S. Raza H. S., 2013). In another study, Caposelle reported performance improvements in DTLS implementations on constrained networks. (A. Caposelle, 2015)

2.10.4. Other Implementations

In addition to standard DTLS and IPsec solutions, there are also a few niche ones. Ukil proposes a different solution from them, where they designed a lightweight security scheme providing object security based authentication and key management. Implementation results show that their scheme is much more feasible than DTLS and IPsec (Bose, 2014).

In (De Rubertis, 2013), the authors compared the performance of IPsec and DTLS with different benchmarks. Their results show that DTLS has higher resource use compared to IPsec. They concluded IPsec to be more suitable for constrained networks and devices than DTLS.

In another study by Raza and others, IPsec and IEEE 802.15.4 security services performance are compared. They found that IPsec performs better than IEEE 802.15.4 security services (Shahid Raza, 2012).

CHAPTER 3

LIGHTWEIGHT SECURITY SCHEMES

In the recent years, in parallel with the increasing popularity of IoT, researchers have steadily increased their efforts to come up with novel schemes which will reduce the heavy resource burden of security on constrained networks and devices. In fact, this had led to the birth of what-is-known-as “lightweight security”. While initial efforts were on the lightweight implementation of standard schemes, the last decade has seen countless number of new security algorithms (ciphers, hash functions, etc.) specifically tailored for lightweight security. They have proved to have several advantages over standard schemes in terms of resources usage, cost, implementation ease and lifetime.

3.1. Lightweight Block Ciphers

Conventional cryptography is developed for the computers and devices that have no resource computation problem. However, IoT devices have completely different requirements. They should be cost and power/energy effective, which in turn means resource-constrained. Such constraints are, by nature, contradictory with the specifications of conventional cryptographic algorithms, whose main target is to provide high security at all costs. The main focus of lightweight cryptography is to solve this problem (William J. Buchanan, 2017).

Lightweight cryptography is, in simple terms, the art of designing algorithms that ensures ample security using limited resources. Some lightweight algorithms are optimized for software implementations while some others for hardware implementations. As stated earlier, countless lightweight algorithms have already been introduced, with new ones being introduced every year. In our study, we chose the most popular ones for evaluation on Arduino Uno platform. We have also implemented standard algorithms like AES, MD-5 and SHA for a fair comparison between standard and lightweight algorithm.

Table 2 depicts main characteristics of our choice of lightweight block ciphers. In the rest of this chapter, we will briefly introduce these algorithms.

Algorithm	Year	Block size (bit)	Key Size (bit)	Rounds	Structure Type	Target Platform
AES	1998	128	128/192/256	10/12/14	SPN	SW, HW
Simon / Speck	2013	32	64	32	Feistel	SW, HW
		48	72/96	36		
		64	96/128	42/44		
		96	96/144	52/54		
		128	128/192/256	68/69/72		
Roadrunner	2015	64	80/128	10/12	Feistel	SW
Present	2007	64	80	31	SPN	HW
Rectangle	2015	64	80 / 128	24/28/32	SPN	SW,HW
Pride	2014	64	128	20	SPN	SW
SparX	2016	64/128	128/256	24/32/40	Feistel	SW
RC5	1994	64	128	20	Feistel	SW,HW
LED	2011	64	80	48	SPN	HW,SW
Lblock	2011	64	80	32	Feistel	HW,SW
Fantomas	2014	128	128	12	SPN	SW
Skinny	2016	64	64/128/192	32/36/40	SPN	HW,SW
		128	128/256/384	40/48/56		

Table 2. Look-up Table for the Main Characteristics of the Lightweight Block Ciphers

3.1.1. Advanced Encryption Standard (AES)

AES is the block cipher algorithm standard by National Institute of Standards and Technology (NIST). It is based on the Rijndael algorithm. AES encrypts/decrypts 128-bit wide blocks of data using either 128, 192 or 256-bit long keys. It uses Substitution-Permutation Network (SPN) construction. AES is optimized for both software and hardware implementations (NIST, 2001).

AES encryption and decryption relies on a round and inverse round function, respectively. Four different steps are performed within the round function. First step is called SubBytes, where each byte within the data block is substituted with another byte using a lookup table,

also known as a substitution box, S-box in short. For the second and third steps, 16 bytes of data are organized in a 4-by-4 matrix. In the second step, ShiftRows(), each row in this matrix is rotated by a different amount, whereas in the third step, MixColumns(), each column of the state is mixed with other columns for maximum diffusion. These two steps constitute the permutation layer, whereas the first step is the substitution layer. In the last step, AddRoundKey(), a round key is added to the state. All these functions are executed in each round except the last one, where MixColumns() is skipped.

Number of rounds to be executed for each block of data depends on the key size. It is 10, 12 and 14 for 128, 192 and 256 bits of key, respectively. The round keys are generated from the original key through a process known as key expansion, which can be either performed in parallel with encryption or done in advance. Exact inverses of these functions are executed for decryption (NIST, 2001).

3.1.2. SIMON and SPECK

Simon and Speck are lightweight block cipher algorithms designed by National Security Agency (NSA). The algorithms are the result of a project aims to present the world a flexible, secure and analyzable lightweight crypto (LWC) algorithms. They use Feistel Network structure and have good performance both on software and hardware.

Simon is specifically optimized for hardware implementations, whereas Speck is optimized for software platforms. They have several block and key length options. It is possible to choose from 32, 48, 64, 96, 128 bits of block length and 64, 72, 96, 128, 144, 192, 256 bits of key length depending on the requirements and needs of the target application. Similar to AES, round numbers depend on the chosen block and key length (32, 36, 42, 44, 52, 68, 69, 72). The Simon round uses bitwise XOR, bitwise AND and left circular shift operations. The Speck round uses bitwise XOR, addition modulo 2^n and left circular shift operations. Addition is replaced with subtraction in decryption. Simon and Speck family algorithms have been specifically designed to have very simple round operations (R. Beaulieu, 2015).

3.1.3. RoadRunneR

RoadRunneR is a software efficient LW block cipher developed by Baysal et al. The design goals of the developers are low decryption overhead, efficiency in 8-bit CPUs and provable security. The algorithm has 64 bits of block length and 80 or 128 bits of key length. The number of rounds is either 10 or 12 for the key length of 80 or 128 bits, respectively. It too uses a Feistel Network structure as well as SPN type functions. RoadRunneR uses bitslice S-box, bitwise XOR, shift and rotate operations. (Şahin, 2015)

3.1.4. Present

Present is a hardware-oriented lightweight block cipher developed by Bogdanov et al in 2007. The algorithm has 64 bits of block length and 80 or 128 bits of key length. It has SPN type structure with 31 rounds. The round function includes bitwise XOR, S-box layer and bitwise permutation. International Organization for Standardization (ISO) specified this algorithm as the standard lightweight block cipher in 2012 (A., 2007) (O'Neill, 2012).

3.1.5. Rectangle

Rectangle is both hardware and software friendly lightweight block cipher designed by Zhang et al. The designers' idea was to come up with an algorithm that is lightweight and fast by using bit-slice techniques. It has an SPN structure. Rectangle algorithm's block length is 64 bits and key lengths are 80 and 128 bits with 24, 28 and 32 rounds. The algorithm introduces new kind of S-box and a new model of permutation layer. Rectangle's contribution to the LWC is the adaptation of bit-slice techniques, ensuring fast and cost-effective software and hardware implementations (Zhang, 2015).

3.1.6. Pride

Pride is a software-oriented lightweight block cipher developed by Albrecht et al in 2014. The authors have concentrated on the design of linear layer to come up with a software-efficient algorithm. Pride has an SPN structure and uses 64 bits of data block and 128 bits of key with 20 rounds. The algorithm has the most efficient software implementation in 8-bit micro-controllers compared to all other lightweight block ciphers (Albrecht M.R., 2014), except NSA's Speck.

3.1.7. SparX

SparX is an ARX (Addition, Rotation, XOR) based block cipher. It is designed with newly presented "long trail design strategy" (LTS) instead of "wide trail design strategy" (WTS) in 2016. WTS designs use efficient and small substitution boxes (s-boxes) and costly linear layers while new LTS design uses big (ARX-based) s-boxes and light linear layers.

32-bit S-boxes are used in Sparx. The algorithm uses 64 bits of data and 128 bits of key with 3 rounds each with 8 steps; 128 bits of data and 128 bits of key with 4 rounds each with 8 steps; or 128 bits of data and 256 bits of key with 4 rounds each with 10 steps (Dinu D. &., 2016) (Sehrawat, 2018).

3.1.8. RC5

RC5 uses ARX two-branched Feistel-Network structure. Rives presented this algorithm back in 1994. It can be implemented on both software and hardware but it was especially optimized for software implementations. The algorithm is word-oriented meaning that all operations take place on words. Data-dependent rotations is the main attribute of this algorithm. The block length can be 32, 64 or 128 bits. It can have different key lengths. And round numbers can be chosen according to tradeoff between security and speed (R.L., 1995).

3.1.9. LED

Also known as Lightweight Encryption Device, LED has an SPN structure. The authors aimed an algorithm that has solid hardware and feasible software implementation. It uses cryptographic operations similar to AES, such as MixColumnSerial, ShiftRows and SubCells. The SubCell function's S-box is the same as PRESENT's. The algorithm has simple key expansion model. The block length is 64 bits and key lengths are 64, 80 and 128 bits (Guo, 2011).

3.1.10.LBlock

LBlock is a type of Feistel Network structure. It is stated that the algorithm can be implemented feasibly not only on hardware but also on software platforms, especially 8-bit MCUs. The algorithm has 64-bit block and 80-bit key length with 32 rounds. The round function consists of substitution layer, permutation layers. These layers include 4-bit S-box, 32-bit permutations and shift operations (Wu W., 2011).

3.1.11.Fantomas

Fantomas is an LWC block cipher algorithm designed to provide security especially against side-channel attacks. It is an algorithm from LS-designs family. LS-designs are a combination of L-boxes (look-up table) and S-boxes. Their target implementation areas are both hardware and software (especially 8-bit MCUs). The algorithm has 128 bits of block and 128 bits of key length with 12 rounds (Vincent Grosso, 2014).

3.1.12. Skinny

Skinny is a block cipher family designed to challenge with the SIMON algorithm. It uses another input with the key to bolster the security against the known attacks. The algorithm has an SPN structure. It includes a light diffusion layer and key schedule. The designers asserts that Skinny has strong resistance against differential, linear and side-channel attacks compared to the NSA designed Simon block cipher family. The authors of the

algorithm is stated that the algorithm has one of the best performance for ASIC implementations. It's block lengths are 64 and 128 bits. And key lengths change between 64 and 384 bits. (Beierle, et al., 2016)

3.2. Hash Functions

Security algorithms must also provide authentication functionality (to ensure confidentiality and integrity) in addition to encryption. This is achieved by use of cryptographic hash functions and hash based message authentication codes (HMAC). As in block ciphers, there have been recent studies to design lightweight versions of these protocols. However, they are mostly still in infant mode. In our study, we will be focusing only on the standard hash functions and their use in HMAC.

3.2.1. MD-5

MD-5 is a hash algorithm standardized in RFC-1321 (Request for Comment) in 1992. Its design principles are similar to those of its predecessor MD-4. MD-5 takes message or data and processes it in 512 bits wide blocks. If data length is smaller than 512 bits, it is padded using a predefined scheme. Afterwards, each 512 bits wide block is processed using 4 rounds of operations. Upon completion of processing of all blocks, a 128 bits wide message digest output is generated. The algorithm assures that if the data changes, so will the message digest with a high probability (Rivest, 1992). But, because of the collision attacks conducted on the algorithm, it is not anymore safe to use MD-5 (Wang X., 2005).

3.2.2. US Secure Hash Algorithm 1 (SHA-1)

SHA-1 is a hash algorithm standardized in RFC-3174 in 2001. SHA-1's design principles are similar to MD-4 and MD-5. SHA-1 also processes data in 512 bits wide blocks and uses a padding scheme for smaller blocks. SHA-1 algorithm uses 80 rounds to process each data block. In the end, a 160 bits wide message digest is output. It too assures change of message digest with the smallest change in data with high probability (D. Eastlake, 2001). According to a collision attack announced by Google Security Blog, SHA-1 is not safe anymore (Marc Stevens (CWI Amsterdam), 2019).

3.2.3. SHA-256

SHA-256 is a hash algorithm standardized as the US Secure Hash Algorithm Suite (SHA and HMAC-SHA) in RFC-4634 in 2006, and later adopted by NIST. It processes 512 bits wide data blocks and produces 256 bits wide message digest. The algorithm assures that if the data changes, so will the message digest with a high probability (Eastlake 3rd, 2006).

3.2.4. SHA-3 (KECCAK)

NIST started a new hash algorithm process in 2005. Later they announced a competition for the SHA-3 Cryptographic Hash Algorithm in 2007. This process was completed in 2012 and KECCAK was announced as the winner of the competition.

KECCAK uses a sponge construction structure. The SHA-3 family has 4 hash and 2 extendable-output functions (XOF). The hash functions are SHA3-224, SHA3-256, SHA3-384 and SHA3-512, whereas the extendable-output functions (XOF) are SHAKE-128 and SHAKE-256. XOF's output message digest can be expanded to different lengths. SHAKE functions' numbers (128 and 256) illustrate the security strengths that provides (Dworkin, 2015).

3.2.5. Keyed-Hashing for Message Authentication (HMAC)

Integrity is one of the three information security goals CIA (Confidentiality, Integrity and Availability). HMAC provides the integrity feature with help of a secret key and a hash algorithm. HMAC is a protocol for calculating an authentication code over the given data with hash function and a secret key. Its protocol uses bunch of bitwise XOR operation, concatenation and hash functions. HMAC's main strength is the simplicity of producing output from input and difficulty of producing input from output (Krawczyk, 1997).



CHAPTER 4

EVALUATION METHODOLOGY

In this chapter, we present our research methodology for evaluation of lightweight cryptographic algorithms. We explain our choice of evaluation platform, algorithms and benchmarking criteria. We also give implementation details.

4.1. Choice of Implementation Platform

One of the main objectives of our thesis is to determine the feasibility of using LWC on the resource-constrained devices that is working on LLNs. For this, we wanted to use a very low-cost and easily obtainable platform with sufficient resources and computational capacity. After consideration of several platforms, we decided to realize our implementations on the Arduino platform, which is widely used in several embedded applications and preferred as a target development platform for its easy programming and open source platform. There are countless resources for Arduino for bugs and errors. For cost purposes, we chose the cheapest Arduino board, i.e. Arduino Uno, which offers a decent price-performance ratio.

4.1.1. Arduino Platform

Arduino is an open-source platform used to read input and outputs with the help of sensors and use these for activating something or publish it someone. It is developed as an easy tool in Ivrea Interaction Design Institute for the students and faculty who have no programming or electronics background. It has three main elements: Arduino board, Arduino programming language and Arduino Integrated Development Environment (IDE).

Arduino's main advantages are simple programming, inexpensive hardware, cross-platform support, open source and extensible software and hardware. The open-source software and hardware developed by community have already made it the platform of choice for resource constrained devices, IoT applications, 3D printing and wearables (What is Arduino?, 2019).

4.1.2. Arduino Uno Board

Arduino Uno is a microcontroller board based on ATmega328. It has 6 analog input pins, 14 digital I/O pins (of which 6 provide PWM output), 32 KB (of which 0.5 KB used by bootloader) Flash memory, 2 KB SRAM, 1 KB EEPROM, 16 MHz clock speed, a USB connection, a power jack, an ICSP header and a reset button. Its supply voltage is between 4.5-5.5 V. The ATmega328 uses a Harvard architecture. The board can be connected to a computer with a USB cable, which also powers the board. Alternatively, it can be powered using an external adapter or even by a battery. Uno is the best starter board for Arduino platforms. (Arduino Uno Rev3, 2019).

4.2 Bench-marking criteria for Block Ciphers

There have been various studies on the evaluation of lightweight ciphers. Hosseinzadeh et al made a comprehensive survey in this area. They used cost, speed and efficiency as the benchmarking criteria to evaluate the performance of lightweight ciphers. For hardware implementation, they measured the cost in terms of the gate count, whereas memory usage and code size were used as the criteria for software implementation. The authors put the speed criteria of algorithms as the number of clock cycles per block and the number of cycles per byte encryption for both software and hardware implementations. Their speed bench-marking criteria are throughput and low latency (Hosseinzadeh J. &., 2016).

In a similar study made by Dinu et al, a bench-marking framework named as Fair Evaluation of Lightweight Cryptographic Systems (FELICS) was presented. It targeted lightweight block and stream ciphers on 8-bit AVR, 16-bit MSP and 32-bit ARM micro-controllers. The framework has a pure interface. It is free, open-source and flexible. The bench-marking criteria are code size, RAM consumption, and the execution time (number of CPU clock cycles spent on executing a given operation (Dinu D. A., 2015).

Rinne et al also made a performance analysis research. Their research was done on an 8-bit AVR micro-controller with ATmega128 MCU. Their performance criteria was memory usage (total code size in flash) and encryption and decryption in measured CPU cycles (Rinne, 2007).

In our study, we used a combination of the previous works and decided to use the following bench-marking criteria:

1. **Code size:** Size of the all code uploaded to the Arduino board. The code size is taken from Arduino IDE. Binary sketch size parameter is equal to the code size of all total functions (Key schedule() + Encryption() + Decryption()) .
2. **RAM usage:** SRAM usage of an algorithm is important as these algorithms work on resource-constrained environments. Exceeding of the SRAM's capacity results in malfunction of the program. We determine the used SRAM with the help of a function that calculate the size between the heap and the stack.
3. **Execution Time:** It presents the total time for “encryption+decryption+key schedule” for one block of data. We measure this by subtracting the time after the operation from the time before the operation.
4. **Throughput:** It is calculated using the following formula:

$$\text{Throughput} = \text{Data (in bytes)} / (\text{end time} - \text{start time})$$

4.3. Selection of Cryptographic Algorithms

AES is the de-facto standard block cipher algorithm. Most of the protocol suites used by governmental, commercial and private organizations utilize it to provide security to their information systems. It is the logical choice as the reference benchmarking algorithm. As a result, it was the first algorithm we implemented on Arduino Uno in our study.

We can briefly summarize our reasoning for choice of other lightweight block cipher algorithms as follows:

- **Simon and Speck:** For being the NSA lightweight block ciphers,
- **Roadrunner:** For being a new family of ciphers, efficient in both software and hardware implementations, especially *developed by Turkish researchers*,
- **Present:** For being one of the frontiers of lightweight block ciphers, the de-facto standard as the ultra-lightweight block cipher, and the official ISO lightweight block cipher standard,
- **Rectangle:** For being the contributor to the lightweight block cipher community in the adaptation of bit-slice techniques and being both hardware and software friendly,
- **Pride:** For being a cipher specifically designed for efficient software implementation on 8-bit micro-controllers,
- **SparX:** For being the first cipher designed with newly presented “long trail design strategy” (LTS) instead of “wide trail design strategy” (WTS),
- **RC5:** For being the block cipher algorithm used in Third Generation of wireless mobile telecommunications technology (3G),

- **LED:** For its similarity to AES with a lightweight twist,
- **Lblock:** For having efficient software implementation on 8-bit micro-controllers and assertion of enough security margin for known attacks,
- **Fantomas:** For providing security especially against side-channel attacks and having efficient software implementation on 8-bit micro-controllers.
- **Skinny:** For its competition with Simon.

We also implemented the hash algorithms; MD5, SHA-1, SHA-256 and SHA3 (Keccak), HMAC algorithms; HMAC-MD5, HMAC-SHA1, HMAC-SHA256. These are the algorithms used in standard security solutions such as 802.15.4 security services, IPsec and DTLS. We wanted to demonstrate feasibility (or lack thereof) of implementing these algorithms on Arduino Uno.

4.4. Code Implementation

The high number of algorithms we wanted to implement on Arduino Uno board made it impractical for us to implement all codes from scratch. Instead, we opted to use existing libraries where possible.

Out of the two prominent alternatives, the libraries written by AVR-Crypto-Lib (AVR-Crypto-Lib, 2019) included hash, HMAC and some of the block cipher algorithms we wanted to evaluate. However, several of our target block cipher algorithms' were missing.

The other alternative was the FELICS project libraries. FELICS is a project of CryptoLUX research group of the University of Luxembourg. It consists of a broad choice of algorithm implementations. Furthermore, the libraries have different versions tailored for different environments and hardware. Some algorithms have up to 35 different versions. The codes are written both in C and ASM. All block cipher algorithms we chose in our thesis have already been included in the project. We used the library versions which is written in C and not tailored for performance improvement (Welcome to the CryptoLUX Wiki, 2019).

The FELICS libraries were better organized and reportedly more code-efficient. Hence, we decided to go with a hybrid approach. We used the block cipher codes from FELICS libraries, and hash and HMAC codes from AVR-Crypto-Lib libraries.

We then adapted those codes to Arduino platform. In doing so, we had to write some codes almost from scratch. Most of the time, we had to go into low-level details in order to obtain optimized code. In our conversion process, we utilized the Arduino IDE, which also allowed us to measure our benchmarking criteria via its built-in functions.

CHAPTER 5

5 EVALUATION RESULTS

In this chapter we present the evaluation results of our implementations on Arduino Uno. We follow the criteria and organization presented in the previous chapter. We start with the block ciphers and continue with hash functions and HMAC.

5.1. Evaluation of Lightweight Block Ciphers

In this section, we present our evaluation results for lightweight ciphers with respect to code size, RAM usage, execution time and throughput.

5.1.1. Evaluation of Code Sizes

Table 3 depicts the code sizes of algorithms. The ciphers we have evaluated use 10-19 percent of Arduino Uno's 32256 bytes of flash memory. Speck has the smallest code size with 3330 bytes (10% flash memory usage). It is followed by Simon and Pride. Fantomas has the largest code size use with 6000 bytes (19% flash memory usage).

Surprisingly, AES's code size is 4760 bytes (15%), very close to the minimum figures used by lightweight ciphers. This, in fact, shows how well it has been designed to be suitable for both hardware and software platforms.

When we look at the whole picture, we see that, even with the largest code size use, there is still enough memory space left for other applications on Arduino Uno. In fact, it is safe to conclude that Arduino Uno flash memory capacity is sufficient for much more complex operations, e.g. an IoT protocol stack.

We present our results graphically in Figure 5.

Algorithm	Block Size (bits)	Key Size (bits)	Total Code Size (bytes)	Remaining Flash (bytes)	Usage Percentage (%)
AES	128	128	4766	27490	15
Simon	64	96	3780	28476	12
Speck	64	96	3330	28926	10
Roadrunner	64	80	3918	28338	12
Present	64	80	5010	27246	16
Rectangle	64	80	3916	28340	12
Pride	64	128	3830	28426	12
SparX	64	128	4504	27752	14
RC5	64	128	4190	28066	13
LED	64	80	4620	27636	14
Lblock	64	80	4500	27756	14
Fantomas	128	128	6000	26256	19
Skinny	64	128	4550	27706	14
Average	-	-	4378	27878	14

Table 3. Code Size of Lightweight Block Ciphers



Figure 5. Block Cipher Code Sizes in Bytes on Arduino Uno

5.1.2. Evaluation of SRAM Usage

In Table 4, it can be seen that the usage percentage of SRAM fluctuates between 21-30%. LED and Speck, with its 21 percent usage is the least SRAM using algorithms. Roadrunner and Fantomas are the second and third best, respectively. AES, Present, Rectangle and Lblock are 30%, making them the most SRAM consuming algorithms. The graphical representation of the SRAM usage of algorithms are presented in Figure 6.

Algorithm	Block Size (bit)	Key Size (bit)	Used SRAM (bytes)	Remaining Space (bytes)	Usage Percentage (%)
AES	128	128	622	1426	30
Simon	64	96	495	1553	24
Speck	64	96	428	1620	21
Roadrunner	64	80	441	1607	22
Present	64	80	604	1444	30
Rectangle	64	80	615	1433	30
Pride	64	128	575	1473	28
SparX	64	128	617	1431	30
RC5	64	128	601	1447	29
LED	64	80	425	1623	21
Lblock	64	80	617	1431	30
Fantomas	128	128	448	1600	22
Skinny	64	128	545	1503	27
AVERAGE	-	-	541	1507	27
Table 4. SRAM Usage					

Even the highest SRAM consuming algorithms use only 30% of the whole SRAM, leaving 70-80% for other applications. As in the code size, lightweight block cipher algorithms leave more than enough space in SRAM for other applications.

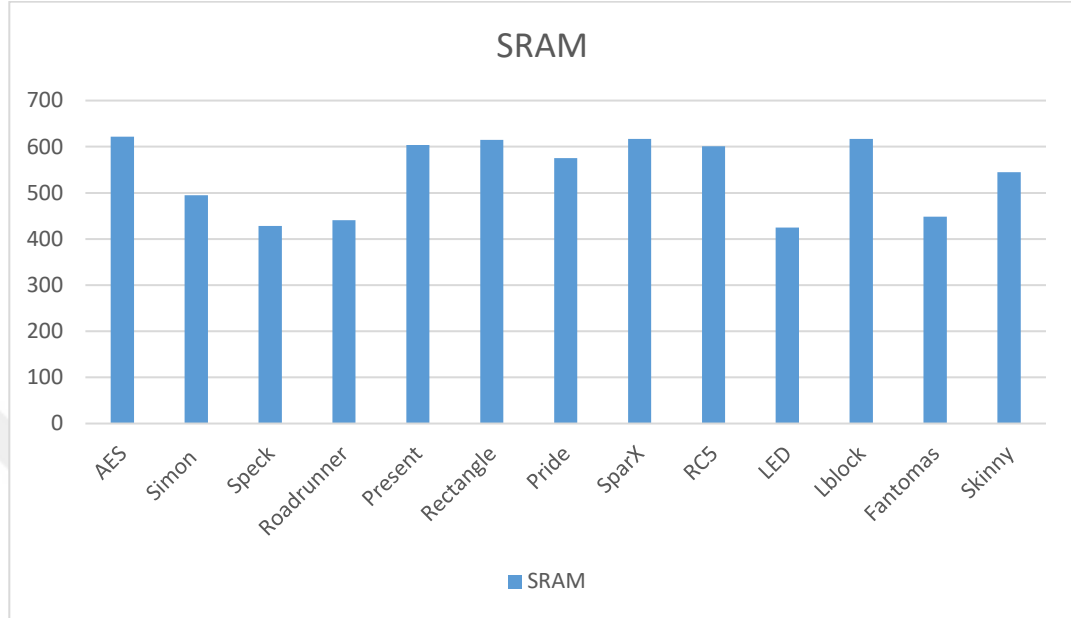


Figure 6. SRAM Usage of Lightweight Block Ciphers

5.1.3. Evaluation of Execution Time

As stated earlier, being the de-facto block cipher for information systems community, we used AES as the main benchmark for other block ciphers. Looking at the execution times given in Table 5, we see AES to be in the middle. As we mentioned in the previous sections, AES is not really designed for constrained devices. In the table, Speck is the most efficient block cipher of all with an average execution time of 0.004 ms for 1 bit data. Skinny and Fantomas are second and third lowest ones, respectively.

Present is the worst one with an execution time of 2.019 ms followed by LED with an execution time of 1.199 ms. Speck block cipher is 9 times more efficient than the standard AES algorithm, which means that confidentiality can be provided nine times more efficiently with it. While there will be other concerns for confidentiality, this is an important indicator. The graphical representation of the execution time of algorithms (excluding Present, LED and RC5) are presented in Figure 7.

There are important execution time differences between these block ciphers. The ratio between best and worst performance is $2.018 / 0.004 = 504$. Even if the two ciphers with the worst execution time – Present and LED – are excluded, the ratio is still quite high, i.e. $0.172 / 0.004 = 43$. It is important to note that shorter execution time means shorter time of operation, which directly corresponds to longer battery time and/or life time for the devices. Considering this fact, the given ratios are rather considerable. They also indicate the importance of choosing the appropriate cipher for the target application.

Algorithm	Block Size (bit)	Encryption Time (ms)	Decryption Time (ms)	Key Schedule Time (ms)	Total Execution Time (ms)	Execution Time per bit (ms)
AES	128	1.67	2.57	0.16	4.40	0.034
Simon	64	0.75	0.73	0.74	2.22	0.035
Speck	64	0.08	0.09	0.11	0.28	0.004
Roadrunner	64	0.52	0.52	0.06	1.10	0.017
Present	64	64.00	62.8	2.44	129.24	2.019
Rectangle	64	0.44	0.39	0.63	1.46	0.023
Pride	64	0.36	0.37	0.24	0.97	0.015
SparX	64	0.40	0.37	0.20	0.97	0.015
RC5	64	0.69	0.70	9.61	11.00	0.172
LED	64	39.30	37.39	0.03	76.72	1.199
Lblock	64	0.40	0.37	0.20	0.97	0.015
Fantomas	128	0.59	0.63	0.01	1.23	0.010
Skinny	64	0.16	0.17	0.28	0.61	0.010

Table 5. Execution Time of Chosen Block Ciphers

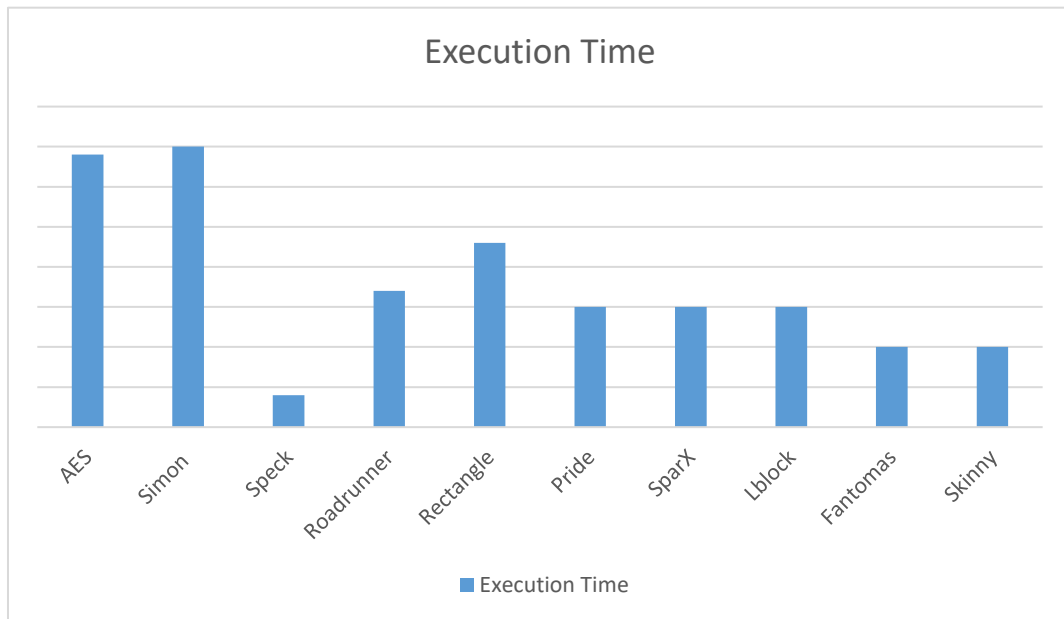


Figure 7. Execution Time (Per Bit) of Lightweight Block Ciphers

5.1.4. Evaluation of Throughput

Throughput values are directly proportional to the execution times. In Table 6, it can be seen that Speck has the highest throughput value with the 28.58 bytes/ms. Skinny and Fantomas have the second and the third highest values with 13.11 bytes/ms and 13.01 bytes/ms. Present has the lowest throughput value with 0.06 bytes/ms and is followed by LED with 0.10 bytes/ms. The graphical representation of the throughput of algorithms (excluding Present and LED) are given in Figure 8.

Algorithm	Data Size (bytes)	Total Execution Time (ms)	Throughput (bytes/ms)
AES	16	4.40	3.64
Simon	8	2.22	3.60
Speck	8	0.28	28.58
Roadrunner	8	1.10	7.27
Present	8	129.24	0.06
Rectangle	8	1.46	5.48
Pride	8	0.97	8.25
SparX	8	0.97	8.25
RC5	8	11.00	0.73
LED	8	76.72	0.10
Lblock	8	0.97	8.25
Fantomas	16	1.23	13.01
Skinny	8	0.61	13.11

Table 6. Throughput of Chosen Block Ciphers

As seen in the Table-6, the amounts of difference between the best and the worst values is considerable, i.e. 476. The ratio between de-facto standard block cipher AES and best throughput value Speck is $28.58 / 3.64 = 8$. The previous power consumption and life time discussion also holds for throughput.

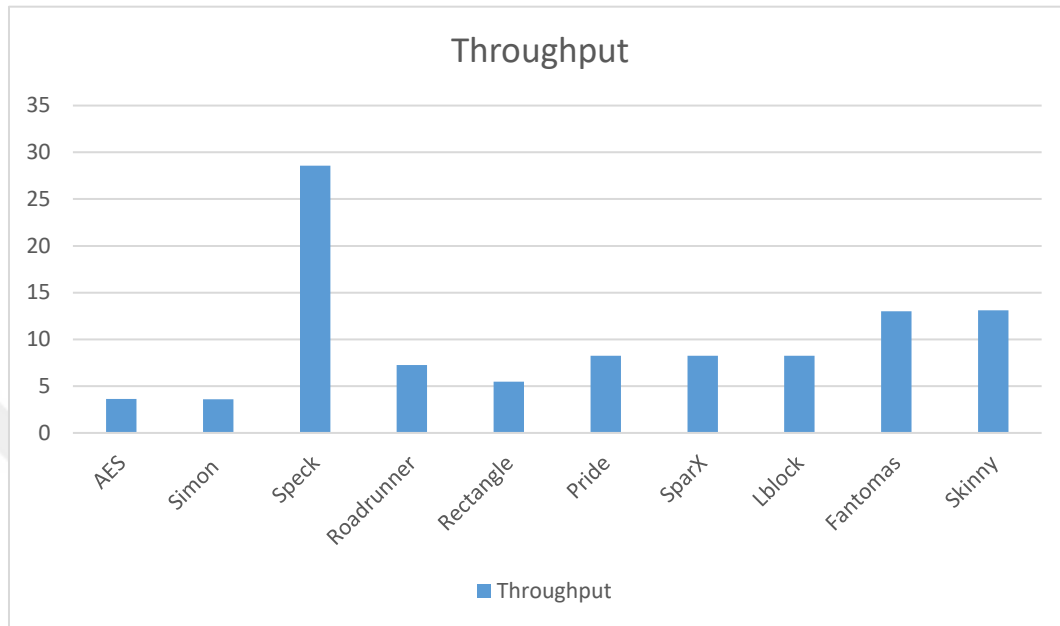


Figure 8. Throughput (bytes/ms) of Lightweight Block Ciphers

5.1.5. Overall Evaluation

Table 7 shows an overall comparison of all block ciphers with respect to all criteria. In summary, flash memory usage percentage for the total code sizes varies between 10-19%, whereas the SRAM memory usage is between 21-30%. It is safe to say that, even with the most memory consuming cipher, there will be no issues in terms of flash memory and/or SRAM usage, even on a simple platform like Arduino Uno. They do not need to be considered as number one priority.

However, the same cannot be said for the execution times and hence throughput. The ratio between the best and worst performance is larger than 43, and can be as high as 504 when worst cases are considered. The same ratios reflect to throughput as well. Clearly, throughput is a much higher priority. As noted before, it directly affects the overall operation time and hence power consumption. Therefore, choice of the right cipher for IoT applications is imperative for a longer lasting battery life.

It is no surprise that Speck block cipher has the best performance for almost all criteria (see Table 7), as it has originally been designed for lightweight software applications. Skinny has the second-best execution time; and it is also good in SRAM usage. Fantomas has the third-best execution time. However, it has the biggest code size with the 19 percent and 6000 bytes flash memory usage.

Algorithm	Total Code Size (bytes)	Code Size Percentage (%)	Used SRAM (bytes)	SRAM Percentage (%)	Execution Time (ms/bit)	Throughput (bytes/ms)
AES	4766	15	622	30	0.034	3.64
Simon	3780	12	495	24	0.035	3.60
Speck	3330	10	428	21	0.004	28.58
Roadrunner	3918	12	441	22	0.017	7.27
Present	5010	16	604	30	2.019	0.06
Rectangle	3916	12	615	30	0.023	5.48
Pride	3830	12	575	28	0.015	8.25
SparX	4504	14	617	30	0.015	8.25
RC5	4190	13	601	29	0.172	0.73
LED	4620	14	425	21	1.199	0.10
Lblock	4500	14	617	30	0.015	8.25
Fantomas	6000	19	448	22	0.010	13.01
Skinny	4550	14	545	27	0.010	13.11

Table 7. Overall Comparison of Block Ciphers

5.2. Evaluation of Hash Functions and HMAC

As we mentioned before, while block ciphers provide confidentiality, hash functions and HMAC provide integrity and authentication. Any evaluation related to IoT systems should include these functions and be targeted for constrained-devices. In our study, we have performed evaluation for the hash functions, whose general characteristics are given in Table 8.

Algorithm	Block Size (bits)	Message-Digest Size (bits)
MD5	512	128
SHA-1	512	160
SHA-256	512	256
SHA3-224	$144 \times 8 = 1152$	224
SHA3-256	$136 \times 8 = 1088$	256

Table 8. Hash Functions Characteristics

In Table 9, we can see the implementation results of the conventional hash functions and HMACs. Total code size percentage values are between 15-20%. SRAM usage percentage is between 16-30% and execution times are between 2.73–50.36 ms.

MD5 and SHA-1 hash functions do not provide dependable security according to the results of the latest attacks. Therefore, if a comparison is done, with these insecure algorithms are excluded, the execution times are considerably high with respect to those of block cipher algorithms.

It is to be expected, as these algorithms are developed for powerful computers and servers which do not have resource problems. The conventional hash functions should also be replace with lightweight ones in order to comply with the resource requirements of constrained-devices.

Algorithm	Total Code Size (bytes)	Code Percentage (%)	Used SRAM (bytes)	SRAM Percentage (%)	Execution Time (ms)
MD5	5468	17	418	20	2.73
HMAC-MD5	6212	19	590	29	10.98
SHA-1	4864	15	373	18	5.20
HMAC-SHA-1	6060	19	624	30	20.84
SHA-256	5826	18	424	20	12.22
HMAC-SHA-256	6550	20	584	29	48.81
SHA3-224	5946	18	334	16	50.35
SHA3-256	5946	18	334	16	50.36

Table 9. HMAC and Hash Functions Results



CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we investigated the feasibility of lightweight block ciphers and hash functions on Arduino Uno. We have evaluated a chosen set of ciphers and hash functions using a predefined benchmarking criteria. While the chosen ciphers were mostly lightweight ones (except AES), the hash functions were conventional ones.

The choice of Arduino Uno was done intentionally in order to demonstrate the suitability and feasibility (or the opposite) of the chosen algorithms to provide security in IoT applications.

As can be seen in our results, none of the ciphers we have evaluated (even the standard AES) does not have any Flash and/or SRAM memory problems on the target platform. With below 20% usage, all ciphers leave enough memory space for other operations. It is safe to conclude that Arduino Uno is a suitable platform for lightweight block cipher based security applications. And choosing the suitable block cipher algorithm is very important. Because there are considerable performance differences between algorithms that effects the battery and lifetime of the devices.

However, security of an IoT application does not only rely on a block cipher, which is limited to providing only confidentiality in its pure form. Hash functions and HMAC should also be considered for applications where confidentiality and integrity services are required for the IoT device. Therefore, we have extended our investigation to hash functions. However, due to lack of lightweight libraries for these functions, we were limited to using conventional functions. Unexpectedly, we have observed huge differences in performance in the implementation of those functions compared to block ciphers.

6.1. Limitations

Our study focuses on the resource usage of lightweight block ciphers and other conventional cryptographic algorithms. We benchmarked the performance of the algorithms. But providing security to a system is not just about the best performance. It depends on the strength of a cryptographic algorithm against different attacks. In our research, we didn't make any research about the security strength of an algorithm. Instead, we relied on the security claims of the designers.

6.2. Future Works

As pointed out earlier, we used conventional hash functions. Furthermore, we did not consider any integrated solution. We only focused on standalone performance. There have already been studies to make lightweight versions of security protocols that is used to provide security to the constrained-devices working on LLNs. In the future, we plan to extend our research to cover both lightweight hash functions and lightweight security protocols specifically targeting IoT devices.

It would be not just interesting but also informative to replace conventional crypto protocols used on IPsec, DTLS and 802.15.4 security services with their lightweight counterparts and evaluate their performance on constrained devices.

We also encourage future researchers to include countermeasures against side-channel attacks and evaluate their effects on the overall performance.

REFERENCES

- A *short history of the Web*. (2019, June 01). Retrieved from <https://home.cern/science/computing/birth-web/short-history-web>
- A. Al-Fuqaha, M. G. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, 17(4), 2347 - 2376. doi:10.1109/COMST.2015.2444095
- A. Caposelle, V. C. (2015). Security as a CoAP resource: An optimized DTLS implementation for the IoT. *IEEE International Conference on Communications (ICC)*, (pp. 549-554). London.
- A., B. (2007). PRESENT: An Ultra-Lightweight Block Cipher. *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 450-466). Berlin, Heidelberg: Springer. doi:https://doi.org/10.1007/978-3-540-74735-2_31
- Albrecht M.R., D. B. (2014). Block Ciphers – Focus on the Linear Layer (feat. PRIDE). *Advances in Cryptology – CRYPTO 2014* (pp. 57-76). Berlin, Heidelberg: Springer, Berlin, Heidelberg.
- Andrew Banks, E. B. (2019, March 07). MQTT Version 5.0. *MQTT Version 5.0*. Retrieved from <https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html>
- Arduino Uno Rev3*. (2019, April 22). Retrieved from <https://store.arduino.cc/usa/arduino-uno-rev3>
- AVR-Crypto-Lib*. (2019, April 22). Retrieved from <https://wiki.das-labor.org/w/AVR-Crypto-Lib/en>
- Beierle, C., Gregor, L., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., & Sim, S. M. (2016). The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In *Advances in Cryptology -- CRYPTO 2016* (pp. 123--153). Berlin, Heidelberg: Springer Berlin Heidelberg.

- Bose, A. U. (2014). Lightweight security scheme for IoT applications using CoAP. *International Journal of Pervasive Computing and Communications*, Vol. 10, 372-392.
- Cambridge Dictionary*. (2019, June 01). Retrieved from <https://dictionary.cambridge.org/>
- Chovatiya, F. &. (2017). A Research Direction on Data Mining with IOT. *2nd International Conference on ICT for Intelligent System*. Ahmedabad : Springer Proceedings.
- D. Eastlake, 3. ., (2001). US Secure Hash Algorithm 1 (SHA1). *RFC 3174*. IETF.
- De Rubertis, A. &. (2013). Performance evaluation of end-to-end security protocols in an Internet of Things. *21st International Conference on Software, Telecommunications and Computer Networks*. Primosten: IEEE.
- Dinu, D. &. (2016). Design Strategies for ARX with Provable Bounds: Sparx and LAX. *Advances in Cryptology – ASIACRYPT 2016* (pp. 484-513). Berlin, Heidelberg: Springer.
- Dinu, D. A. (2015). FELICS - Fair Evaluation of Lightweight Cryptographic Systems. *NIST Workshop on Lightweight Cryptography 2015*. NIST.
- Dworkin, M. J. (2015, August). SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. (*NIST FIPS*) - 202. Federal Inf. Process. Stds.
- Eastlake 3rd, D. a. (2006, July). US Secure Hash Algorithms (SHA and HMAC-SHA). *RFC 4634*. IETF.
- G. Montenegro, N. K. (2007, September). Transmission of IPv6 Packets over IEEE 802.15.4 Networks. "*Transmission of IPv6 Packets over IEEE 802.15.4 Networks*", *RFC4944*. IETF. Retrieved from , <https://tools.ietf.org/html/rfc4944>
- Getting Started with the Alexa Skills Kit*. (2019, June 02). Retrieved from <https://developer.amazon.com/en-US/alexa/alexa-skills-kit/start>
- Glissa, G. &. (2018). 6LoWPSec: An end-to-end security protocol for 6LoWPAN. *Ad Hoc Networks*, 100-112. doi:<https://doi.org/10.1016/j.adhoc.2018.01.013>.
- Guo, J. P. (2011). The LED block cipher. In *Cryptographic Hardware and Embedded Systems–CHES 2011* (pp. 326-341). Berlin, Heidelberg: Springer.
- History of IEEE*. (2019, April 08). Retrieved from <https://www.ieee.org/about/ieee-history.html>

- Hosseinzadeh, J. &. (2016). A Comprehensive Survey on Evaluation of Lightweight Symmetric Ciphers: Hardware and Software Implementation. *Advances in Computer Science : an International Journal*, 31-41.
- Hosseinzadeh, J. H. (2016). A Comprehensive Survey on Evaluation of Lightweight Symmetric Ciphers: Hardware and Software Implementation. *Advances in Computer Science : an International Journal*.
- IDC Forecasts Worldwide Technology Spending on the Internet of Things to Reach \$1.2 Trillion in 2022. (2019, April 15). Retrieved from <https://www.idc.com/getdoc.jsp?containerId=prUS43994118>
- IEEE Standard for Low-Rate Wireless Networks. (2016, April 22). (1-709). IEEE.
- Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions). (2019, April 15). Retrieved from <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>
- K.Seo, S. a. (2005, December). Security Architecture for the Internet Protocol. "Security Architecture for the Internet Protocol", RFC4301. IETF. Retrieved from <https://tools.ietf.org/html/rfc4301>
- Kramp T., v. K. (2013). Introduction to the Internet of Things. In v. K. Kramp T., *Enabling Things to Talk* (pp. 1-10). Berlin, Heidelberg: Springer.
- Krawczyk, H. B. (1997, February). HMAC: Keyed-Hashing for Message Authentication. RFC 2104. IETF.
- Lightweight Cryptography. (2019, June 02). Retrieved from <https://csrc.nist.gov/projects/lightweight-cryptography>
- Lim C.H., K. T. (2006). A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. *Springer*.
- Lin, H. (2014). A Practical Approach to Provide Security in IEEE 802.15.4 Wireless Sensor Network. *Applied Mechanics and Materials*, 568-570.
- Marc Stevens (CWI Amsterdam), E. B. (2019, April 20). *Announcing the first SHA1 collision*. Retrieved from <https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html>
- Mulligan, G. (2007). The 6LoWPAN architecture. In *Proceedings of the 4th workshop on Embedded networked sensors (EmNets '07)* (pp. 78-82). New York: ACM. doi:<http://dx.doi.org/10.1145/1278972.1278992>

- N.Modadugu, E. a. (2012, January). Datagram Transport Layer Security Version 1.2. "Datagram Transport Layer Security Version 1.2", *RFC6347*. IETF. Retrieved from <https://tools.ietf.org/html/rfc6347>
- Negi, S. U. (2017). Internet of Things and RPL routing protocol: A study and evaluation. *2017 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-7). Coimbatore: IEEE.
- Nemati, A., Soheil, F., Arash, A., Saeed, H., Majid, A., & Shahpour, A. (2015). An efficient hardware implementation of few lightweight block cipher. In *2015 The International Symposium on Artificial Intelligence and Signal Processing (AISP)* (pp. 273-278). Mashhad.
- NIST. (2001). NIST FIPS: Advanced Encryption Standard (AES). Federal Information Processing Standards Publication, 197.
- ONeill, N. H. (2012). Hardware Comparison of the ISO/IEC 29192-2 Block Ciphers. *2012 IEEE Computer Society Annual Symposium on VLSI* (pp. 57-62). Amherst: MA.
- R. Beaulieu, S. T.-C. (2015). The SIMON and SPECK lightweight block ciphers. *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, (pp. 1-6). San Francisco.
- R.L., R. (1995). The RC5 encryption algorithm. *Lecture Notes in Computer Science* (pp. 86-96). Berlin, Heidelberg: Springer.
- Ray, P. P. (2016). A Survey of IoT Cloud Platforms. *Future Computing and Informatics Journal*, 135-46.
- Raza, S. (2013). Lightweight Security Solutions for the Internet of Things. *Malardalen University Press Dissertations*, 139.
- Rinne, S. E. (2007). Performance Analysis of Contemporary Lightweight Block Ciphers on 8-bit Microcontrollers.
- Rivest, R. (1992, April). The MD5 Message-Digest Algorithm, . *RFC1321*. IETF.
- S. Raza, H. S. (2013). Lithe: Lightweight Secure CoAP for the Internet of Things. *IEEE Sensors Journal*, 3711-3720.
- S. Raza, S. D. (2011). Securing communication in 6LoWPAN with compressed IPsec. *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, (pp. 1-8). Barcelona.
- S.Kent. (2005, December). IP Authentication Header. "IP Authentication Header", *RFC4302*. Retrieved from <https://tools.ietf.org/html/rfc4302>

- S.Kent. (2005, December). IP Encapsulating Security Payload. Retrieved from <https://tools.ietf.org/html/rfc4303>
- S.Kent. (2014, October). "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC7296. IETF. Retrieved from <https://tools.ietf.org/html/rfc7296>
- Sehrawat, D. a. (2018). Lightweight Block Ciphers for IoT based applications : A Review. *International Journal of Applied Engineering Research*, 2258-2270.
- Shahid Raza, S. D. (2012). Secure Communication for the Internet of Things - A Comparison of Link-Layer Security and IPsec for 6LoWPAN. *Journal of Security and Communication Networks*.
- Shelby, Z. H. (2014, June). The Constrained Application Protocol (CoAP). Internet Engineering Task Force (IETF). doi:10.17487/RFC7252
- Star Trek (film)*. (2019, June 01). Retrieved from [https://en.wikipedia.org/wiki/Star_Trek_\(film\)](https://en.wikipedia.org/wiki/Star_Trek_(film))
- Şahin, A. B. (2015). RoadRunneR: A Small and Fast Bitslice Block Cipher for Low Cost 8-Bit Processors. *In Revised Selected Papers of the 4th International Workshop on Lightweight Cryptography for Security and Privacy - Volume 9542*.
- T. Winter, P. T. (2012, March). RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. "*RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*", RFC6550. IETF. Retrieved from <https://tools.ietf.org/html/rfc6550>
- The Internet of Things*. (2019, April 08). Retrieved from www.ietf.org: <https://www.ietf.org/topics/iot/>
- The Internet of Things*. (2019, April 08). Retrieved from <https://www.gartner.com/it-glossary/internet-of-things/>
- Towards a definition of the Internet of Things (IoT)*. (2019, April 08). Retrieved from https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf
- Vincent Grosso, G. L.-X. (2014). LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations. *Fast Software Encryption - 20th International Workshop*. London.
- Wang X., Y. H. (2005). How to Break MD5 and Other Hash Functions. *Advances in Cryptology – EUROCRYPT 2005*. Berlin, Heidelberg: Springer.
- Welcome to the CryptoLUX Wiki*. (2019, April 22). Retrieved from <https://www.cryptolux.org/index.php/Homes>

What is Arduino? (2019, April 22). Retrieved from <https://www.arduino.cc/en/Guide/Introduction>

William J. Buchanan, S. L. (2017). Lightweight cryptography methods. *Journal of Cyber Security Technology*, 187-201.

Wu W., Z. L. (2011). LBlock: A Lightweight Block Cipher. *Applied Cryptography and Network Security. ACNS 2011* (pp. 327-344). Berlin, Heidelberg: Springer.

Zhang, W. &. (2015). - RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *Science China Information Sciences*, 1-15.



TEZ İZİN FORMU / THESIS PERMISSION FORM

ENSTİTÜ / INSTITUTE

Fen Bilimleri Enstitüsü / Graduate School of Natural Sciences

Sosyal Bilimler Enstitüsü / Graduate School of Social Sciences

Uygulamalı Matematik Enstitüsü / Graduate School of Mathematics

Enformatik Enstitüsü / Graduate School of Informatics

Deniz Bilimleri Enstitüsü / Graduate School of Marine Sciences

YAZARIN / AUTHOR

Soyadı / Surname :

Adı / Name :

Bölümü/Department :

TEZİN ADI / TITLE OF THE THESIS (**İngilizce / English**) :

.....

.....

TEZİN TÜRÜ **Yüksek Lisans / Master**

Doktora / PhD

1. **Tezin tamamı dünya çapında erişime açılacaktır.** / Release the entire work immediately for access worldwide.

2. **Tez iki yıl süreyle erişime kapalı olacaktır.** / Secure the entire work for patent and/or proprietary purposes for a period of **two year**. *

3. **Tez altı ay süreyle erişime kapalı olacaktır.** / Secure the entire work for period of **six months**. *

** Enstitü Yönetim Kurulu Kararının basılı kopyası tezle birlikte kütüphaneye teslim edilecektir. A copy of the Decision of the Institute Administrative Committee will be delivered to the library together with the printed thesis.*

Yazarın imzası / Signature

Tarih / Date