Efficient Security Algorithm for Provisioning Constrained Internet of Things (IoT) Devices

by

Joseph Nannim Mamvong

PhD Thesis

School of Electronic Engineering and Computer Science
Queen Mary University of London
United Kingdom

January 2023

DECLARATION

I, Joseph Nannim Mamvong, confirms that the research included within this thesis is my own or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below, and my contribution indicated. Previously published material is also acknowledged.

I attest that I have exercised reasonable care to ensure that the work is original and does not to the best of my knowledge break any UK law, infringe any third-party copyright or other intellectual property right, or contain any confidential material. I accept that the college has the right to use plagiarism software to check the electronic version of the thesis. I confirm that the thesis has not previously submitted for the award of a degree by this or any other university. The copy right of this thesis rests with the author and no quotation from it or information derived from it may be published without prior written consent of the author.

Signature: Joseph Nannim Mamvong

Date: 22/12/2022

DEDICATION

This work is specially dedicated to the following: Napoleon Nanfa Mamvong (The Legend), Patricia Napoleon Mamvong, Ramya Blessed Mamvong (Cha mi!), Selbyen Mamvong, Nnander Mamvong, Dinci Sophia Mamvong, Nanna Zoe Mamvong and Chalya Mamvong.

Abstract

Addressing the security concerns of constrained Internet of Things (IoT) devices, such as client-side encryption and secure provisioning remains a work in progress. IoT devices characterized by low power and processing capabilities do not exactly fit into the provisions of existing security schemes, as classical security algorithms are built on complex cryptographic functions that are too complex for constrained IoT devices. Consequently, the option for constrained IoT devices lies in either developing new security schemes or modifying existing ones as lightweight. This work presents an improved version of the Advanced Encryption Standard (AES) known as the Efficient Security Algorithm for Power-constrained IoT devices, which addressed some of the security concerns of constrained Internet of Things (IoT) devices, such as client-side encryption and secure provisioning. With cloud computing being the key enabler for the massive provisioning of IoT devices, encryption of data generated by IoT devices before onward transmission to cloud platforms of choice is being advocated via client-side encryption. However, coping with trade-offs remain a notable challenge with Lightweight algorithms, making the innovation of cheaper security schemes without compromise to security a high desirable in the secure provisioning of IoT devices. A cryptanalytic overview of the consequence of complexity reduction with mathematical justification, while using a Secure Element (ATECC608A) as a trade-off is given. The extent of constraint of a typical IoT device is investigated by comparing the Laptop/SAMG55 implementations of the Efficient algorithm for constrained IoT devices. An analysis of the implementation and comparison of the Algorithm to lightweight algorithms is given. Based on experimentation results, resource constrain impacts a 657% increase in the encryption completion time on the IoT device in comparison to the laptop implementation; of the Efficient algorithm for Constrained IoT devices, which is 0.9 times cheaper than CLEFIA and 35% cheaper than the AES in terms of the encryption completion times, compared to current results in literature at 26%, and with a 93% of avalanche effect rate, well above a recommended 50% in literature. The algorithm is utilised for client-side encryption to provision the device onto AWS IoT core.

Acknowledgments

I would like to acknowledge and express my sincere gratitude to:

- my supervisors: Prof. Yue Gao and Dr. Gokop Goteng for guidance, reading my various reports and raising questions which led to deeper investigation of the findings that span this thesis. Many thanks for your mentorship!
- The Tertiary Education Trust Fund (TETFUND) for funding this research project.
- My stage1 and stage2 examinar and independent assessor, Dr. Zhijin Qin. Thank you for all the helpful feedback.
- The staff of Electronic Engineering and Computer Science Department at Queen Mary University of London (QMUL). These include, but not limited to: Dr. James Kelly, Dr. Zhijin Qin, Dr. Shaker M.M Alkarari and Melissa Yeo.
- All of the Antennas and Electromagnetics Group Members.
- All of the friends I made in QMUL, especially: Hadeel Saleh Alrubayyi, Habiba Akter, Georgios Zoumpourlis, Janosch Haber, Arthur Americo Passos De Rezende, Jiadong Yu and Xiaolan Liu to mention a few.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ASCII	American Standard Code for Information Interchange
IoT	Internet of Things
LPWA	Low-Power Wide-Area
NB-IoT	Narrow Band
RFID	Radio Frequency Identification
WSN	Wireless Sensor Network
LoRaWANs	Long Range Wireless Area Networks
CPU	Central Processing Unit
M2M	Machine-to-Machine
CPS	Cyber-Physical Systems
TRNG	True Random Number Generator
PRNG	Pseudo Random Number Generators
CPRNG	Cryptographically Secure Pseudo Random Number Generators
DLP	Discrete Logarithm Problem
AES	Advanced Encryption Standard
CSI	Channel State Information
MAC	Media Access Control
DES	Data Encryption Standard
3DES	Triple Data Encryption Standard
OTP	One-Time Pad
RSA	Rivest, Shamir and Adleman
ECC	Elliptic Curve Cryptography
NIST	National Institute of Standard and Technology
NSA	National Security Agency
IBM	International Business Machines
NBS	National Bureau of Standards
XOR	Exclusive Or
AFA	Algebraic Fault Analysis
SPA	Simple power Analysis

TDMA	Time-Division Multiple Access
ECB	Electronic Code Book
CAM	Content Addressable Memory
S-Box	Substitution Box
GF	Galois Field
NVIC	Nested Vectored Interrupt Controller
MPU	Memory Protection Unit
FPU	Floating-Point Unit
TRM	Technical Reference Manual
OS	Operating System
IDE	Integrated Development Environment
BLE	Bluetooth Low Energy
MQTT	Message Queueing Telemetry Transport
SE	Secure Element
IEEE	Institute of Electrical and Electronic Engineers
NIC	Network Interface Cards
SPN	Substitution Permutation Network
AWS	Amazon Web Services
GCP	Google Cloud Platform
PAP	Password Authentication Protocol
CHAP	Challenge Handshake Authentication Protocol
PPP	Point-to-Point Protocol
TLS	Transport Layer Security
IP	Internet Protocol
IPv6	Internet Protocol version six
DTLS	Datagram Transport Layer Security
6LoWPAN	IPv6 over Low Power Personal Area Networks.

Chapter 1

Introduction

1.1 Internet of Things (IoT)

The Internet of Things, otherwise known as a network of connected things is a technology topic which has in the recent past, transitioned from being theoretical discussions to a realistic actualization and cutting across several other technology topics including: Low-power Wide Area Networks, mobile devices, embedded and ubiquitous communication, cloud computing, data analytics and artificial intelligence to mention a few [1], while enabling the provision of a wealth of intelligence in planning, management and decision making. IoT devices have been estimated in several scholarly articles to be in the range of twenty to fifty billion devices by the year 2025 [2, 3]. The applications of the IoT technology cuts across a wide range of sectors of economies, enabling and enhancing rapid transformations in almost every part of human life as we know it. With it's applications aiding rapid advancement in artificial intelligence as detailed in [4] for example, the IoT technology is shown to have alleviated some of the negative effects of the COVID-19 pandemic's impact on economic development. The projection of the IoT technology is to give every real object a virtual reality, bringing about an unprecedented connectivity of "Things", more than ever before. In the years ahead, the IoT will have major impact on business models [5–9], agriculture [10, 11], transportation [12–14], automated industrial processes [15–17], homes [18–20], infrastructure, security, trade standards, and much more. However, as interesting and promising as the projection of the complete actualization of the IoT technology sounds, the

advancement is closely accompanied by myriads of challenges -including security. Security and privacy with respect to the IoT technology encompasses the means of safeguarding both the data generated and transmitted by IoT devices, as well as the the identity of users in the IoT landscape and ethical use of the technology among other issues. As rightly put by [21], such extreme interconnection will bring unprecedented convenience and economy, but it will also require novel approaches to ensure its safe and ethical use. In [22], while acknowledging the emergence of the IoT in redefining convenience in the lives and education of children through emerging applications on mobile phones, it was identified that these apps also make illegal and inappropriate contents such as pornography, violence and drugs -to mention a few, become more accessible to children and thus, negatively impacting the growth of minors. In [23], an analysis of the security challenges for IoT according to the various layers of the IoT architecture was presented. Some of these challenges include node reputation, information interception, access control, terminal security, privacy, heterogeneous technology and network security to mention but a few.

Generally, the security framework of wireless networks follows the Open System Interconnection (OSI) model, which comprises of the three upper layers (Application, presentation and session) known as the application layer, the transport, network, datalink/MAC and the physical layer. Most existing security techniques follow this layered approach in trying to address the security issues in wireless communication [24]. However, typical deployments of the traditional networked systems have most of the connected devices as stationary devices with stable sources of power; whether directly connected or using power over ethernet (PoE) technologies etc. Additionally, these devices are also of considerably proficient processing capabilities and the combination of the duo of stable/continuous power supply and high (in contrast to typical IoT devices) processing power was a huge support base for developing robust encryption schemes for communication security. The emergence of the IoT has however, caused a great shift in the practical reality of secure communication based on these provisions. This is because the reality of the IoT technology is the unprecedented eruption of widely distributed battery-powered, low processing power-communication devices, enabled by the massive deployment of low power wide area networks. As rightly put by [25], Low- Power Wide Area (LPWA) technologies complement and sometimes supersede the conventional cellular and short range wireless technologies in performance for various emerging smart city and machine-to-machine (M2M) applications. According to [26], The IoT protocol architecture is logically structured in three layers: Perception (also

called the physical) layer, transmission (network) layer and the application layer. According to [24], the security threats and vulnerability associated with each of these protocol layers are protected separately at each layer to meet the basic security requirements of authentication, confidentiality, integrity and availability. In a systematic literature review of security challenges facing wireless communication, with a focus to outline specific types of wireless network attacks and the existing counter measures against those attacks. They classified these attacks with respect to the OSI protocol architecture. However, according to [27], ensuring end-end privacy across the three layers of IoT remains a grand challenge and it is not easy to implement sufficient cryptographic functions on these devices due to their limitations. [28].

This thesis include; the development, implementation and analysis of an efficient security algorithm for a constrained category of the IoT devices, following a defined set of objectives as contained in section 1.2. This chapter introduces as sections, various topics relating to constrained IoT devices, which are detailed in later chapters. The IoT, together with associated security issues, the constrained nature of IoT devices with respect to security algorithms are all briefly introduced from sections 1.1 through to 1.4. Cloud computing as a key enabler for the provisioning of IoT devices is introduced in 1.5 and 1.5.1, with the chapter rounding on the motivation and the major contributions of this piece of work. Chapter 2 presents a detailed review of relevant literature on the sections introduced in chapter 1, furthering into a review of lightweight and classical security algorithms. A detailed analysis of the issues of the complexities with classical security algorithm in provisioning constrained IoT devices is presented in chapter 3; giving the mathematical background and basic algebraic properties of the security algorithms, a look at the experimental setup towards complexity reduction of the most widely used security algorithm in the narrative of constrained IoT devices, as well as trade-off and cryptanalytic overview of the consequence of complexity reduction. Chapter 4 hinges on the detailed analysis done in chapter 3 and presents an efficient security algorithm for constrained IoT devices. The efficient security algorithm is then utilized for the purpose of client-side encryption and secure provisioning of a sample constrained IoT device, the SAMG55 microprocessor on an IoT platform of a leading cloud services provider, the Amazon Web Services (AWS-IoT core). The results of various experimentation and analysis of the results is presented in chapter 6.

1.2 Aim and Objectives

The aim of this research is to develop an efficient security algorithm for provisioning IoT devices that are constrained in power and processing capabilities. The specific objectives towards achieving this aim include:

1. Review of security challenges and the security algorithms in use within the IoT landscape.
2. A review of the complexity of classical security algorithms, complexity reduction and cryptanalytic analysis of the consequences of complexity reduction with respect to the narrative of constrained IoT devices.
3. Trade-off, Mathematical analysis and justification of complexity reduction.
4. Implementation and comparison of the Efficient Algorithm for provisioning constrained IoT devices, between a laptop computer and a sample IoT device, the SAMG55 microprocessor
5. Implementation and comparison of the Efficient Algorithm for Constrained IoT devices with a compatible lightweight algorithm (CLEFIA)
6. Client-side encryption and secure provisioning of the SAMG55 IoT device onto AWS IoT core, using the Efficient algorithm for provisioning constrained IoT devices.

1.3 Security Challenges of the IoT

According to [29], since IoT communication protocols and technologies differ from traditional IT realms, their security solutions ought to take this difference into account. Security of conventional IT infrastructure is achieved using classical cryptographic protocols and algorithms whereas, applying classical cryptographic methods for IoT security is not efficient as those methods were not ideally designed for these kind of systems [23]. Consequently, the option for IoT in terms of security lies either in the development of new schemes or the modification of existing security algorithms to carter for the peculiar needs of constrained IoT devices in the wide and long ranged network of connected things. According to [26], applying classical cryptographic methods for IoT security is not efficient as those methods were not ideally designed for these kind of systems. They advocated for hybrid light weight models for improving security situation in IoT. Thus, the need for investigating what components of the existing algorithms make them expen-

sive and explore how they can be improved. According to [30], It will be particularly important to explore new techniques of jointly defending against multiple types of wireless attacks, which may be termed as mixed wireless attacks. Traditionally, the protocol layers have been protected separately to meet their individual communications security requirements. However, these traditional layered security mechanisms are potentially inefficient, since each protocol layer introduces additional computational complexity and latency. As a result, the need of addressing security challenges in more than one layer of the IoT protocol architecture is also advocated. In [30], it was said the classic Diffie–Hellman key agreement protocol is traditionally used to achieve the key exchange between the source and destination and requires a trusted key management centre. However, the key management is challenging in certain wireless networks operating without a fixed infrastructure. Since this is the case with IoT devices, the choice of a security algorithm for the constrained IoT devices should take into considerations the limited resources available to these devices in terms of power and processing capabilities. Consequently, there is the need for research to develop secure communication algorithms suitable for constrained IoT devices, considering their constrained nature and the fact that they constitute a part of a lager network of connected things. This would entail the identification of the existing security challenges and defence mechanisms of IoT implementations through the information available in literature, and the determination of the security mechanism or combination of mechanisms viable for developing an efficient security algorithm for IoT devices with respect to their constrained nature. Secure authentication of IoT devices that are constrained in power, memory and processing resources is an ongoing challenge desiring novel solutions. Usually, these tiny IoT devices run by battery power, making it a daunting task to design security mechanism which is a best fit [31]. According to [31], some of the challenging problems in the implementation of the IoT include: key management, device authentication, user access control, privacy preservation and identity management to mention but a few. Many attacks including eavesdropping, Denial of Service (DoS), Man-in-the-Middle and certificate manipulation among others is a serious threat to the authentication of IoT devices and the constrained nature of the devices has imposed a serious challenge in designing counter measures to combat these attacks [31]. Securing the IoT therefore, is a necessary milestone towards expediting the deployment of its applications and services[32]. According to [33], As smart home systems get more and more popular recently, the security protection of smart home systems has become an important problem. Architecting IoT focused security solutions must however, take into considerations the unique circumstance of power constrained IoT devices

as according to [34], reaping the benefits of the IoT is contingent upon developing IoT-specific security and privacy solutions.

1.4 Resource Constrained IoT

In this section, the category of IoT devices facing the most challenging security situations are introduced, with respect to the bigger picture of an efficient security algorithm for provisioning such devices. Generally, the IoT technology incorporates existing network deployments and the massive deployments of newly low powered or mobile devices. While some of these devices are quite capable in terms of requirements to deploy existing security algorithms for safe communication within the IoT networks, there are most others which are constrained in terms of the requirements to successfully implement the available security algorithms, and these categories of devices are considered for investigation. The investigation aims to identify the specific challenges and requirements for which these devices are constrained. According to [35], the problem of power consumption in microprocessors and Digital Signal Processing (DSP) devices has continued to emerge, requiring solutions in order to maximize the battery life of power constrained devices. IoT devices characterised by low power and low processing capabilities do not exactly fit into the security provision of existing security techniques, due to their constrained nature. According to [36], due to the simplicity of the devices and components at the perception (physical) layer, their resource-constrained nature, and their low computational and storage capabilities, securing the IoT transmissions is notoriously challenging. In realizing the vision of the IoT, Low-Power Wide Area (LPWA) technologies complement and sometimes supersede the conventional cellular and short range wireless technologies in performance for various emerging smart city and machine-to-machine (M2M) applications [25]. In order to realize sustainable IoT networks, one of the challenges is maximizing the battery lifetime of power-constrained IoT devices [37]. In [38], a parametric analysis of security threats and prospective solutions for the IoT was presented. They classified different types of security attacks on the overall IoT infrastructure into: Low, intermediate and high-level security issues, with a strategic reference to the IoT protocol stack architecture. One of the immediate consequences of the IoT being a technological evolution that incorporates traditional networks and emerging paradigms like the Wireless sensor networks (WSN) and cyber-physical systems (CPS) in general is that; it inherits some of the challenges in these different networks as well. A critical issue in the inherited problem is that classical security

methods such as cryptography for communication security are not efficient for deployment in the IoT, due to the low power and processing capability of the IoT devices. Therefore, the critical need for considering peculiar security techniques for a rather constrained infrastructure.

1.5 Cloud Computing and IoT Provisioning

The National Institute of Standards and Technology (NIST)'s definition of cloud computing is given as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [39]. While IoT devices/deployments generates massive data in petabyte scale, storage and leveraging these data is hugely assisted by possibilities availed by cloud computing platforms. Cloud computing technologies continue to roll out robust cloud infrastructure in service to the massive connectivity promised by the actualization of the IoT. Leading cloud services providers including but not limited to Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, The Things Networks (TTN) provide robust cloud infrastructure in service to the massive deployments of the IoT for a variety of use-cases. However, the unprecedented deployment of devices in the IoT ecosystem is accompanied by the burden to secure the devices to ensure the safe use of the technology. As majority of IoT devices are constrained devices which are mostly deployed for use cases involving capturing/monitoring real-time situations and pushing the associated data onto safe storage in cloud platforms, the cost of encrypting IoT data and the need to ensure safe transportation of the data to the cloud pose a challenge. With the IoT requiring novel solutions for it's safe use, ensuring that the data generated by these devices is safe and securely transported to a choice cloud platform is a huge security concern. According to [39], Secure authentication unto cloud platforms is one of the challenges in cloud adoption. This is made worse by the constrained nature of the IoT devices in resources including power and processing capabilities. Consequently, client-side encryption which means that data is encrypted at the end of the device before onward transfer unto a cloud platform for storage or processing is advocated. According to [40], with client-side encryption, data is encrypted before it is transferred to a cloud platform to ensure that content is transferred and stored in encrypted format and that only clients with the appropriate decryption keys have access to non-encrypted information. Furthermore, the efficient use of classical encryption schemes for appropriate client-

side encryption before cloud storage is an additional challenge given the scarcity of power and computing resources available to the constrained IoT device. Provisioning an IoT device onto a cloud platform entails the process of creating a unique identity on the cloud via requisite and secure authentication credentials to make it available, while client-side encryption is used to achieve encryption of data at the end of the device before it is sent unto cloud storage.

1.5.1 IoT Provisioning

While cloud computing technologies are rapidly enabling widespread IoT deployments, this development is also closely accompanied by challenges on the process of provisioning the IoT devices onto the the cloud platforms. [41] observed that although billions of IoT devices are estimated to be deployed in the nearest future, very little to no information is present on ease of device provisioning and according to [42], the integration of IoT devices and cloud servers is highly dependent on how security issues such as authentication and data privacy are handled. On this note, factoring in the constrained nature of IoT devices while considering IoT security as well as IoT device provisioning is strongly advocated. On another front, encryption-before-outsourcing of data on IoT devices to cloud platforms is a widely recommended method to guarantee the confidentiality of user data [43]. The consequent need of empowering these devices with client-side encryption capabilities in order to preserve the privacy of data generated and outsourced to cloud storage systems brings on another layer of burden on the devices, given the scarcity of resources. In order to protect the security of the outsourced data, an intuitive way is to encrypt the data before outsourcing it to the cloud [44], and ensuring that the encryption algorithms for this purpose takes into the cognizance, the scarcity of resources on these devices is paramount to secure and efficient provisioning of constrained IoT devices.

1.6 Motivation and Contribution

Motivated by the challenges in literature, ranging from; the security challenges in the IoT, the unsuitability of the usage of conventional cryptographic algorithms for security in the IoT landscape, resource constraint in IoT, finding secure trade-offs in effort to address the challenges of complexity in IoT deployments, secure convergence of the IoT and cloud computing, as well as the consequent challenges of IoT provisioning as introduced in sections 1.3 through to 1.5.1, this work proposed an efficient security algorithm for power constrained IoT devices which aimed to

reduce complexity of the currently used security algorithm: The Advanced Encryption Standard, in the IoT landscape. Furthermore, resource constrain on a sample IoT device (SAMG55 microprocessor) is investigated and the efficient algorithm is implemented as a low-cost client-side encryption solution for data encryption as advocated in [44], and leverages the ATECC608A in effort to addresses the challenges of key management and device authentication as highlighted in [31], by securely provisioning the device on an IoT cloud services platform. The main contributions of this work are thus, summarized as follows:

- A cryptanalytic overview and analysis of the consequences of reducing the complexity of the AES, which is the currently used encryption algorithm in the IoT landscape is presented.
- A comparison and analysis of the algebraic structures of CLEFIA to that of the AES and the reduced round algorithm based on the AES.
- A mathematical justification of reducing the complexity of the standard AES-128 algorithm, using the core algebraic properties of the standard algorithm is presented. This is followed by provisioning a secure element: the ATECC608A to aid authentication and guard against implementation attacks in line with our analysis of the consequence of round reduction as per item 1 a above.
- An implementation of safely reduced round versions (four rounds and two rounds) of the AES-128 algorithm, based on the the structure of the AES in order to reduce complexity (measured by the time it takes to complete the encryption of 16bytes of plain text).
- A comparison of the reduced round algorithm and the standard AES-algorithm, with results showing up to 35% of the time it takes to complete the encryption of a single byte of plaintext, saved.
- Experimentation and analysis of resource constraint in IoT devices by comparing a PC and SAMG55 implementations of the efficient algorithm for provisioning constrained IoT devices to the standard AES128.
- Secure provisioning of a sample IoT device (SAMg55 microprocessor) on AWS IoT core using the Amazon Web Services (AWS) Command Line Interface (CLI) programmatic

access tools.

- Implementation and comparison a Low-cost algorithm (Based on the AES) to lightweight CLEFIA, experimentation of the avalanche effect test on the low-cost algorithm and using it as client-side encryption solution in provisioning of a sample IoT device (SAMg55 microprocessor) on AWS IoT core using the AWS Command Line Interface (CLI) programmatic access tools.

With respect to the categories of security challenges in the IoT and cyber-physical systems landscape as outlined in [23], this research work aimed to address the bit of privacy and access control, guarding against implementation attacks on the associated IoT device, authentication key management as highlighted in [31], challenges of trade-off in lightweight algorithms as observed in [45] and IoT device provisioning challenges as highlighted in [41].

1.7 Chapter Summary

In this chapter, topics and sub-topics that are considered as relevant introductions to this research work were briefly introduced. An introduction of the IoT technology with respect to the thesis title and as per work done is presented in section 1.1, following which a general introduction to the security issues plaguing IoT deployments was given in 1.3. Resource constrained IoT as a section of the larger topic of the IoT, and which is central to the idea and work done in this research is presented in section 1.4. The nexus between the security challenges introduced in 1.3, the constrained category of IoT devices as contained in 1.4 and the burden imposed on constrained IoT devices by classical security algorithms is also subtly introduced in this section. Cloud computing, IoT provisioning and the need for rethinking inefficient security algorithms used by constrained IoT devices was introduced in section 1.5. The motivation and main contributions of the work done are given in section 1.6, following which a conclusion of the introductory chapter is presented in section 1.7

Chapter 2

Review of Literature

2.1 Introduction

In order to put the narrative of constrained IoT devices in a suitable perspective for this work, a detailed review of relevant literature is deemed essential. Reviews on the IoT technology and its applications, as well as the security challenges in the IoT landscape are considered to be plausible topics. A review that seeks to juxtapose the the constrained nature of IoT devices against the known issues of using classical security algorithms in the IoT landscape is also presented. Other major topics adjudged as worthy of review in detail are identified as including: lightweight security algorithms, cloud computing, authentication, IoT provisioning and the reduction of complexities of classical security algorithms. This chapter details the review of the aforementioned with respect to the narrative of constrained IoT and presented in sections 2.1 through to 2.9

2.2 The IoT Technology and Applications

Elaborate discussions on the emergence and continuous evolution of the IoT technology present insights into how the subject of IoT is transforming the world as we know it, with the technology promising a massive connection of devices in billions. Several scholarly articles including [3, 46, 47] support the claim of this estimate with elaborate discussions touching on the potential of the technology towards transforming many areas of human life. As rightly put by the authors in [48], applications of IoT offer services covering all aspects of human life, including home and

building automation, smart cities, health monitoring, emergency and surveillance services, city waste management, smart grid, smart health, intelligent traffic management, supply chain, retail, smart industry, etc. Similarly, [49] observed that in the coming years, the IoT will have major effects on business models, infrastructure, security, trade standards, healthcare, agriculture and many other sectors.The application of the IoT technology is also aiding rapid advancement in artificial intelligence, as detailed in [4] and shown to have alleviated some of the negative effects of the COVID-19 pandemic's impact on economic development. On a baseline, the IoT technology projects to give every real object a virtual reality and thus, bringing about an unprecedented connectivity of things, more than ever before. Already, the International Data Company (IDC) has predicted the market growth rate of the IoT from 2017 to 2021 at about 14% [50], which has resulted to leading technology industry players including Google, Amazon, Cisco among others already rolling out devices and technology standards relating to the IoT. In one instance, the IoT technology promises great prospects of global food security through the application of IoT is revolutionizing the field of agriculture via the concept of smart farming as discussed in [51, 52], as well as the enhancement of agricultural product value through the deployment of sensors and actuators to optimize the production chain in agricultural activities [53, 54], which in turn holds the promise of boosting the gross domestic products (GDPs) of developing economies whose teeming populations hugely rely on it agricultural products [55]. Through this technology of interconnected things, a great number of small/medium scale and large scale farmers stand the change of boosting production while at the same time, enjoying business transformation of electronic commerce (ecommerce), occasioned by another front of the application of the IoT. In ecommerce, the IoT is revolutionizing the concept of doing business through the introduction of new business models such as person-to-machine (P2M) and machine-to-machine(M2M) as against the traditional person-to-person business models [56–58] and thus, expanding potentials in ways hitherto possible. This is made possible through IoT deployments, enhancing business monitoring systems [59] and thus, providing economic opportunities for worldwide profitability.

The IoT revolution has also introduced cutting edge advancement in the delivery of health services with many applications ranging from deployment of health monitoring systems and wearable body sensors, analysis of specific health conditions, delivery of efficient telemedical services, health facility sharing and collaboration to mention but a few [60–62]. The application of the IoT technology in health in one example demonstrates how mortality rates of accidents

and critical illnesses patients have be shown to be improved by leveraging the IoT, as detailed in [63]. Coming onto the evolution of smart cities and how human lives are transforming in unprecedented ways, there is the evolution of intelligent vehicular and transportation systems occasioned by the emergence of the IoT, which promises a paradigm shift on the nature of vehicles and transport systems as we know it, into vehicles equipped with communication capabilities aided by sensors and actuators, computing units and Internet-Protocol based connectivity [13, 64, 65]. This is aimed at transforming the traditional transport systems as we know it into intelligent transport systems. Furthermore on how the IoT technology is changing the world, factory and industrial automation processes aided by the developments of the IoT is exploding exponentially, revolutionizing the face of industry with pay offs ranging from better business insights to improved productions, industrial monitoring and aiding human-robot interaction as discussed in [66**?** , 67]. The IoT promises to transform home appliances such as refrigerators, light bulbs, doors, curtains, showers -to mention but a few, into smart devices through the use of sensors and actuators [68–70] and thereby, bringing about smart homes equipped with Internet based communication capabilities and cooperation in far reaching dimensions.

However, as interesting and promising as the projection of the complete actualization of the IoT technology sounds, this advancement is closely accompanied by myriads of challenges -including security. According to [**?**], such extreme interconnection will bring unprecedented convenience and economy, but it will also require novel approaches to ensure its safe and ethical use. According to [71], securing the IoT is a necessary milestone towards expediting the deployment of its applications and services. According to [72], as smart home systems get more and more popular recently, the security protection of smart home systems has become an important problem. According to [73], reaping the benefits of the IoT is contingent upon developing IoT-specific security and privacy solutions. According to [26], The IoT protocol architecture is logically structured in three layers: Perception (also called the physical) layer, transmission (network) layer and the application layer. However, [74] posited that complex management of mass amount of sensor nodes, heterogeneity, and lack of agreed upon standards, protocols makes IoT system infrastructure easy to target for unauthorized access and eves dropping on personal data. According to [24], the security threats and vulnerability associated with each of these protocol layers are protected separately at each layer to meet the basic security requirements of authentication, confidentiality, integrity and availability, whereby ensuring end-to-end privacy across

the three layers of IoT is a grand challenge [27]. According to [36], since IoT communication protocols and technologies differ from traditional IT realms, their security solutions ought to take this difference into account. Security of conventional IT infrastructure is achieved using classical cryptographic protocols and algorithms but according to [26], applying classical cryptographic methods for IoT security is not efficient as those methods were not ideally designed for these kind of systems. Consequently, the hope for IoT in terms of security lies either in the development of new schemes or the modification of existing security schemes to meet the need of the constrained devices. The latter however, must be done while ensuring that the metrics of security of the existing schemes remain preserved.

2.3 IoT Security Challenges

Holding that the application of classical cryptographic methods for IoT security is not efficient as those methods were not ideally designed for these kind of systems, the authors in [26] advocated for hybrid light weight models for improving security situation in IoT. Thus, the need for investigating what components of the existing algorithms make them expensive and explore how they can be improved. According to [30], It will be of particularly importance to explore new techniques of jointly defending against multiple types of wireless attacks, which may be termed as mixed wireless attacks. Traditionally, the protocol layers have been protected separately to meet their individual communications security requirements. However, these traditional layered security mechanisms are potentially inefficient, since each protocol layer introduces additional computational complexity and latency. As a result, the need of addressing security challenges in more than one layer of the IoT protocol architecture is also advocated. In [30], it was said the classic Diffie–Hellman key agreement protocol is traditionally used to achieve the key exchange between the source and destination and requires a trusted key management centre. However, the key management is challenging in certain wireless networks operating without a fixed infrastructure, which is characteristic of IoT networks or deployments. Therefore, the choice of a security algorithm for the constrained IoT devices should take into considerations the limited resources available to these devices in terms of power and processing capabilities. Consequently, there is the need for research to develop secure communication algorithms suitable for constrained IoT devices, considering their constrained nature and the fact that they constitute a part of a lager network of connected things. This would entail the identification of the existing security

challenges and defence mechanisms of IoT implementations through the information available in literature, and the determination of the security mechanism or combination of mechanisms viable for developing an efficient security algorithm for IoT devices with respect to their constrained nature.

2.4 Resource Constraint in the IoT

IoT devices characterised by low power and low processing capabilities do not exactly fit into the security provision of existing security techniques, due to their constrained nature. As briefly introduced in section 1.1, the IoT architecture provides that the physical/perception layer is dominated by very simple devices that are mostly used for sensing and transmission of data onto storage systems, provided by on-premises/conventional information technology data centers or cloud computing platforms. Ensuring the secure transmission of these data generated at the perception layer however, requires novel contributions as authenticating the IoT devices onto these networked systems raises data security and privacy concerns. In [36], a deep learning study based on the use of dynamic watermarking techniques for secure authentication of IoT devices asserts that due to the simplicity of the devices and components at the perception (physical) layer, their resource-constrained nature, and their low computational and storage capabilities, securing the IoT transmissions is notoriously challenging. In general, the said challenge imposed by this resource-constrained nature of IoT devices has necessitated research towards enhancing the the capabilities of such devices in many fronts, including the maximization of the battery life of the battery-powered devices, development of lighter applications, development of light and compatible operating systems, algorithms, etc in tandem with the scarcity of resources occasioned by the nature of these devices. In order to realize sustainable IoT networks, one of the challenges is maximizing the battery lifetime of power-constrained IoT devices, being the major actors in low-power wide area networks as detailed in [37]. In a related study which reviewed open challenges and solutions as detailed in [38], a parametric analysis of security threats and prospective solutions for the IoT was presented. This study classified different types of security attacks on the overall IoT infrastructure into: Low, intermediate and high-level security issues, with a strategic reference to the IoT protocol stack architecture. As deduced, one of the immediate consequences of the IoT being a technological evolution that incorporates traditional networks and emerging paradigms like the Wireless sensor networks (WSN) and cyber-physical systems

(CPS) in general is that; it inherits most, if not all, of known challenges in these conventional networks as well. A critical issue in the inherited problem is that classical security methods such as cryptography methods for communication security are not efficient for deployment in the IoT, due to the low power and processing capability of the IoT devices. Consequently, the critical need for considering peculiar security techniques for a rather constrained infrastructure.

In what follows, a review which centers on investigating resource constrained IoT in terms of security algorithms is pursued. The review presents the notion of constraints, as imposed by means of encryption key generation, as well as constraints imposed by the associated ciphers, being the two high level components of security algorithms in general as shown in Fig. 2.1.

2.4.1 Encryption Key Generation and Processing Constraints

Consider a high-level view of a symmetric key algorithm such as the AES, as shown in figure 2.1:

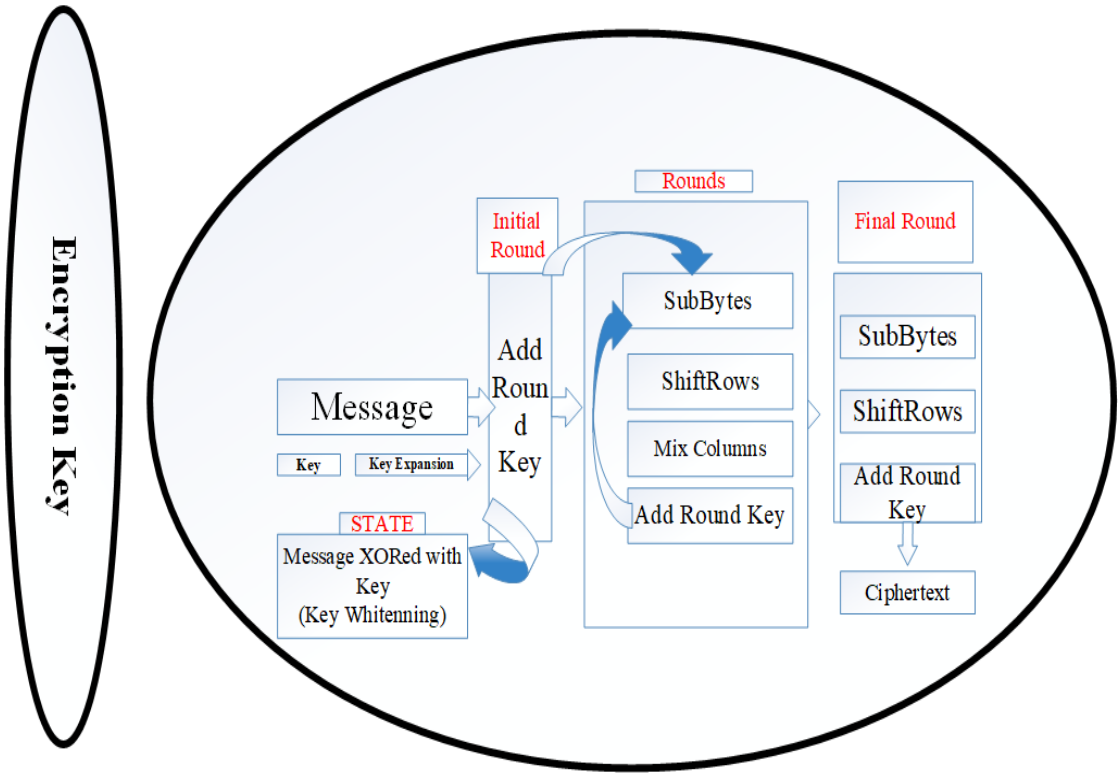

Figure 2.1: Diagram showing Key and Cipher parts of an algorithm

The challenge of complexity as typically presented by an encryption algorithm with respect to constrained IoT devices is diagnosed, based on the two major parts of an encryption algo-

rithm namely: the encryption KEY and the CIPHER. The widely discussed security of the AES algorithm for instance, is based on the key length of the algorithm and the associated algebraic construction of the cipher. While encryption process is largely done by the cipher, it is noteworthy that the process would be incomplete without the key. As such, while the cipher holds on one hand, a notion of complexity based on the constructions and operations that defines it, the source of the key used for encryption and how they are generated holds on the other hand, another notion of complexity. Hence, the probe: how are the keys generated? what are the available options? what is the cost and security implication of these options with respect to constrained IoT narrative? Known methods for generating encryption keys include: True Random Number Generators (TRNGs), Pseudo Random Number Generators (PRNGs), Cryptographically secure Pseudo Random Number Generators (CPRNGs) to mention but a few. However, in addition to whether or not there is enough randomness in the generated keys to guarantee security, there is the inherent cost of key generation in terms of computing resources, challenges of how the keys are stored and the means of exchange between the communication entities in networked systems. The obtainable situation in conventional networked systems is that these keys are generated and stored in dedicated key management systems as part of the overall network infrastructure and accessed via appropriate protocols by authorized devices when the need for authentication and encryption arises. However, the fundamental question remains the source of the encryption keys to be used for authentication, encryption/decryption as needed. This question has opened up diverse research directions into investigating the viability of key generation techniques in the paradigm of new technologies such as the IoT. In what follows, some of the foremost exploration in this regard are reviewed in order to set the narrative of finding cheaper security mechanisms in the IoT landscape. The review reveals that there are three major sources through which keys can be generated for use with an encryption algorithm. Firstly, Fig. 2.2 shows a rather compressed summary of the leading key generation methods reviewed, together with the associated notions of complexities as related to the narrative of constrained IoT, with the details of same presented in sections 2.4.1.1 through to 2.4.1.3

2.4.1.1 Centralized Key Generation and Management Systems

A fundamental requirement of symmetric algorithms such as the AES, is that the secret key used for encrypting messages is the same key that is used for decryption of the same message. However, this requirement has the underlying problem of key distribution to contend with at

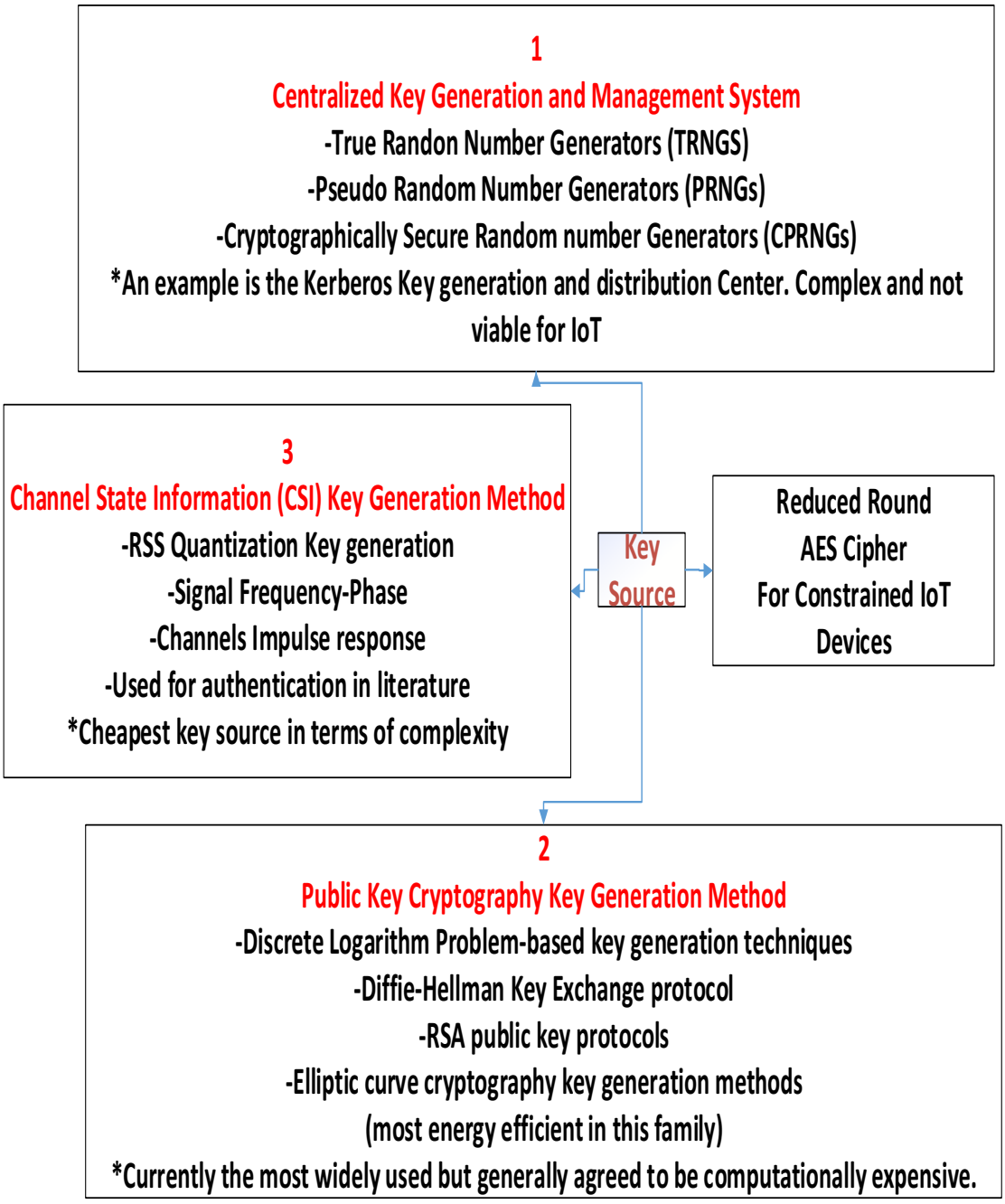

Figure 2.2: Encryption Key generation sources

point of implementation. Secure key distribution among communicating devices intending to encrypt with an algorithm in this regard is a critical part of the overall security of the networked systems using the algorithm. To this end, the aforementioned key generation techniques as contained in section 2.4.1 is consolidated by the provisioning of dedicated key management

infrastructure as a component of the overall networked systems which would be utilizing the keys for secure communication. According to [75] these approaches often require a centralized trusted third party which generates, maintains, and distributes shared keys to communicators, noting that such schemes introduce a high key management complexity for large scale wireless networks, which usually involve a large size of key pool and require intensive key distribution to support the key establishment between every pair of nodes. The problem pointed out here clearly does not suit the IoT narrative which is characterised by wireless networks in very large scale. Instead, it introduces new directions for the exploration of cheaper methods of generating and maintaining encryption keys that are in tandem with the constrained nature of IoT devices. According to [26], key management which includes generating, distributing, storing and destroying the secret keys is identified as a security challenge in wireless sensor networks, a category within the lager body of the IoT. In a related study on the techniques for preserving privacy as detailed in [76], key agreement in IoT is non-trivial due to resource constraints. Importantly for the set narrative, this work observed that many key agreement schemes used in general networks, such as Kerberos and RSA, may not be suitable for IoT because there is usually no trusted infrastructure in IoT. Furthermore, pre-distribution of secret keys for all pairs of nodes is not viable due to the large amount of memory used when the network size is large.

2.4.1.2 Public Cryptography Key Generation Methods

As an alternative to the dedicated key management infrastructures, there is the public key cryptography. This option of secret key generation for communication security hugely eliminates the security concerns associated with the various methods of key generations presented in 2.4.1.1, but also comes with its own associated challenges. According to [75], using public key cryptographic approaches to generate secret keys normally require the communication entities to be equipped with desired computing chips or modules, and they have been long regarded as expensive in terms of computational complexity. Foremost among the public key cryptography key agreement algorithms is the Diffie-Hellman key agreement protocol. The Diffie-Hellman key exchange algorithm for secure distribution of encryption keys is built around a "computationally hard" mathematical problem, otherwise known as a trapdoor function. According to [77], the Diffie-Hellman public key distribution system is based on the computationally infeasible Discrete Logarithm Problem (DLP), and is also has the nice property that a block enciphering scheme

such as DES or AES can then be used for encryption and decryption of messages following the secure distribution of the encryption key, using Diffie-Hellman public key algorithm. The Diffie-Hellman algorithm accounts for both "generation" and "secure distribution" of the encryption keys by communication parties. In summary the Diffie-Hellman key exchange protocol presents the following scenario: A large prime number say p and a primitive element of a cyclic group say g are made public (that is known and available to every device that has the potential to participate in the key exchange process):

- Node A chooses a random number $a \in 1, 2, \ldots, p-1$, computes $g^a \ modp$, and sends to node B
- Node B chooses a random number say $b \in 1, 2, \ldots, p-1$, computes $g^b \ modp$, and sends to node A
- Both Nodes A and Node B compute $g^a b \ modp$.

In [78], this value is shown to be the same at both node A and B, which guarantees that a secret key can be reliably extracted from the value of this computational exchange, and which is the case in the practical implementation of the Diffie-Hellman key generation. More precisely, the communicating nodes can agree on which bits of this figure to use as an encryption key for a symmetric cipher such as the AES which is relatively faster in comparison to the public key computational complexity. Due to the largeness of the prime p and the primitive element g, it is noteworthy that the resulting value of $g^a b \ modp$ after computation is also a reasonably large number of at least 1024 bits according to [78] and hence, a very good key sources for the AES algorithm in the sense that whether it is the 128, 192 or 256 bit key length that is required for encryption, it can be reliably obtained from this figure. A required key length of 128 bits for instance could be the first 128 bits of the value of $g^a b \ modp$, which is the same at both node A and node B, and a perfect secret-key for AES encryption is established. On the alternative, any arbitrary 128 bits key-length can be encrypted and sent as a message following the establishment of a secret key using $g^a bmodp$ between two nodes, and the key can then be securely used for AES encryption and decryption subsequently. The computationally infeasible problem for an attacker given the above scenario is the problem of computing $g^a bmodp$ without the knowledge of a and b. According to [77], the given problem from an attacker's standpoint is computationally infeasible. According to [79], the time and cost of computing this value given

the situation of an intruder would be too great for it to be practical. It is therefore, on the bases of this computational infeasibility that the Diffie-Hellman key exchange protocol is secure. However, there exist a reasonable consensus in literature that this secure key generation protocol is computationally expensive, most especially from the perspective of constrained IoT devices. According to [75], this approach of encryption key generation has been long regarded as expensive in terms of computational complexity. A related study on a deep learning technique for signal authentication and security in the massive deployment of IoT as detailed in [71], emphasized that securing the IoT devices has become challenging due to the simplicity, resource-constrained nature and low storage capabilities of IoT devices at the perception layer and thus, buttressing the unsuitability of the constrained devices as candidates for participating in such expensive key generation and exchange mechanisms like the Diffie-Hellman's. According to [73], complex security techniques such as cryptographic protocols cannot be easily implemented on IoT devices due to the memory and computational resources required in deploying such algorithms.

2.4.1.3 Channel State Information Key Generation Method

As an alternative to the use of dedicated key management infrastructure or public key cryptography for encryption key generation as detailed in 2.4.1.1 and 2.4.1.2 respectively, there is the option of low-power-friendly means of generating these encryption keys through Channel State Information (CSI) estimation, for achieving authentication at the physical layer of the IoT network infrastructure. In [30] a discussion of the physical-layer authentication and cryptography solutions conceived for wireless networks was presented, which pointed out that the MAC addresses of network nodes have conventionally been used for authentication, which is however, insecure to MAC spoofing attacks and can be arbitrarily changed for the sake of impersonating another network node. consequently, the growing advocacy for new physical-layer authentication approaches such as the propagation properties of wireless channels. A related survey on lightweight ciphers for IoT devices as detailed in [80] reveals, there is an increasing concern about achieving fast and efficient key establishment via exploiting physical layer characteristics such as the reciprocity of wireless channels, whereby the receiver and the transmitter of one wireless link observe the same channel simultaneously. In [81], a physical layer security method based the property of wireless channel randomness and quantization threshold selection was proposed. Similarly, physical characteristics of wireless communication channels for secrete key establishment as detailed in [82] presents a comprehensive review of existing "state-of-the-art"

quantization schemes for physical layer security derived from the intrinsic characteristics of the communication media for key generation, sharing and randomness extraction. This survey study held that these protocols always seek to exhibit both low computational complexity and energy efficiency, whilst also maintain unconditionally secure communications. In [75], another survey on key establishment techniques based on channel reciprocity held that centralized key management approach introduces an undesired complexity, and that the exponential operations on large numbers in popular cryptographic protocols (such as detailed in 2.4.1.2) also imposes a challenge on constrained devices due to computing requirements. Hence, they presented a review of different types of existing techniques on quantization of wireless channel properties to form secret keys. A survey study detailing secure key design approaches using entropy harvesting as detailed in [83] presented several works done on secure key generation, randomness extraction and sharing of keys involving different mechanisms using properties of wireless communication channels. Consequently, although having it's own drawbacks, the extraction of Channel State Information (CSI) of wireless channels and using it as a source for generating keys for encryption would eliminate significantly, the complexity of generating keys through cryptographic means. At the same time, this would eliminate the challenge of using centralized key management infrastructures which generate, store and distribute encryption keys through the aforementioned key generation methods. Although according to [84], one of the main challenge in this domain is to increase the secret key length that is extracted from the shared channel coefficients between two nodes, and not compromising randomness and uniformity.

2.5 Classical Security Algorithms

This section reviews the classical security algorithms which are adjudged to be too complex for the constrained IoT devices. While there are many security algorithms which fall in this category, only a selected few are investigated. These include the Data Encryption Standard (DES), the triple Data Encryption Standard (3DES) and the Advanced Encryption Standard (AES). The investigations of these algorithms were done in line with the narrative of the constrained IoT devices and so, includes the details of the design, security and a cursory look at these properties in the perspective of constrained IoT devices. The choice of focusing on the selected algorithms is motivated by this review finding in [25], wherein the investigation on the security algorithm adopted by various LPWA standards was presented. Particularly the Advanced Encryption

Standard is found to be the most widely used security algorithm in the current IoT landcape and so, the need to investigate the AES together with it's predecessor, the DES in detail, as well as the notion of security and complexity of these algorithms with respect to constrained IoT devices. This review section therefore presents the details of each of the classical algorithm by its background, design, the algorithm in the IoT narrative and related security analysis. This is aimed to leave a background to the notion of reducing the complexity of classical security algorithms to suit the narrative of constrained IoT devices.

2.5.1 Background of Security Algorithms

According to [85], information security which is an integral part of all types of networks (inclusive of IoT) can be ensured by employing both cryptographic and non-cryptographic solutions, whereby the cryptographic solutions proffer protection against confidentiality, integrity and authentication challenges. The constructions of security algorithms are based in the enriched field of cryptography. According to [78], cryptography is generally classified into the families of symmetric algorithms, asymmetric algorithms and protocols. Symmetric algorithms are basically cryptographic algorithms that use the same key for encryption and decryption [77]. Popular algorithms in this family of cryptographic algorithms include but not limited to: the DES, the 3DES, the One-Time Pad (OTP) and the Advanced Encryption Standard (AES). Asymmetric algorithms on the other hand are cryptographic algorithms that use a different key for decryption of a given cipher text than the key that was used for encryption. There are generally three families of asymmetric algorithms namely: The Rivest, Shamir and Adleman (RSA), the Discrete Logarithm Problem (DLP) and the Elliptic Curve Cryptography (ECC) families of asymmetric key algorithms.

Symmetric algorithms are generally adjudged to be faster algorithms in terms of speed of encryption and decryption, as compared to their asymmetric counterparts. This translate to symmetric algorithms being of less complexity when juxtaposed with the limited resources as in power and processing capabilities in the domain of constrained IoT devices. As a result, symmetric algorithms and precisely the block ciphers in this family of algorithms becomes very potent for IoT security deployments. The choice of a security algorithm for the constrained IoT devices should take into considerations; the limited resources available to these devices in terms of power and processing capabilities. At the same time, the need for the development of security

algorithms in the first place, to ensure communication security remains of paramount importance. Consequently, security algorithms must evolve as the landscape of use evolves, raising the topics of trade-offs, compromise and balance, as has become the case for the IoT landscape and opening new frontiers for security algorithms that are targeted towards the need of constrained IoT. According to [86], the classical algorithms that have been depended on have become vulnerable to various types of attacks and unsuitable in certain domains, hence the demand for either new algorithms or a kind of modification on the classical ones to withstand evolving attacks and other security challenges. To this end, a careful review of the detailed makeup of some of these classical algorithms is deemed necessary, as a bases for new developments or the modification.

2.5.2 The Data Encryption Standard

The DES is known to be the first modern encryption algorithm. DES emerged in 1976 as the choice data encryption algorithm for standardization when the International Business Machines (IBM) submitted a proposal (in 1974), following the request for a standardized cipher by the United States' National Institute of Standard and Technology (NIST) -formerly known as the US National Bureau of Standards (NBS) in 1972. According to [78], the NIST requested the help of the National Security Agency (NSA) to investigate the security of submissions to this request, following which some changes were made to the original submissions of the IBM, and then named and standardized as the DES. According to [87], the National Security Agency (NSA) provided technical advice to the IBM development team to take advantage of the knowledge of certain cryptanalytic techniques in the design, following which DES emerged as a national standard in 1977. According to [88], the Data Encryption Standard is a block cipher that was selected by the US National Bureau of Standards in 1976 for data encryption, which uses a 56-bit key (represented on 64 bits including 8 parity check bits) and operates on 64-bit blocks of messages. The DES is arguably, the most studied algorithm for data encryption. According to [85], DES was extensively studied in the last thirty years since it was the first standard symmetric encryption algorithm. According to [78], DES is by the far the best studied symmetric algorithm, which design principles have inspired the design principle of so many modern algorithms, including the ones that are considered to be more secure than DES nowadays.

2.5.2.1 DES DESIGN

According to [78], the design of DES was based on Lucifer, which is a family of ciphers developed by Horst Feistel in the 1960s and was one of the first instances of block ciphers to operate on digital data, which encrypts blocks of 64bits of data using a key size of 128bits. Although the Lucifer family of ciphers are originally designed based on key length of 128bits, the technical advice given to the IBM by the NSA however is believed to be the reason DES key length did not emerge as the 128bits as it is in the lucifer family of ciphers and thus, leading to a standardization of the DES algorithm with a key length of 56-bits.

According to [77], the DES algorithm uses a product transformation of transpositions, substitutions and non-linear operations which are applied in sixteen iterations to each block of 64bits plaintext. Normally, the plaintext is split into blocks of 64 bits and for encryption, the encryption key of fifty-six (56) bits which are taken from an original key of 64bits is applied on each message block using the DES process. The remaining eight of these 64 bits (of the key) are used for parity check. A parity bit is normally a bit that can be appended to string of bits to make the total number of 1-bits either even or odd. For decryption of a DES-encrypted message block, the encryption process is applied in reverse to carryout out a decryption of blocks of ciphertexts.

DES has an iterative structure which applies the same round transformation f on 64-bit blocks of data. The cipher is made up of 16 rounds which are preceded by bit-permutation and followed by an inverse bit-permutation. Each round in the 16 rounds are preceded by an Initial bit-Permutation (IP) and ends by the inverse bit-permutation (IP^{-1}), otherwise known as the final permutation. Each of the round has a 48bits round key which is derived from the master key of 56bits, which in turn is a part of the 64bits key after dropping eight for parity check. According to [88], Each round of the algorithm is parameterized by a 48-bit round key that is derived from the master key. According to [89], arithmetic operation of the DES algorithm is based on the exclusive OR (XOR) logic operations, and the basic features of the algorithm is in seven steps with steps fourth to sixth repeated sixteen times (the 16 rounds of the DES algorithm). Figure 2.3 shows a diagrammatic process flow of the encryption of a message block (64bits) using the DES algorithm:

Following the breaking down of the message into 64bits blocks, the DES encryption process starts by splitting a message block into two equal halves of 32bits each as L_i (Left 32bits) and R_i

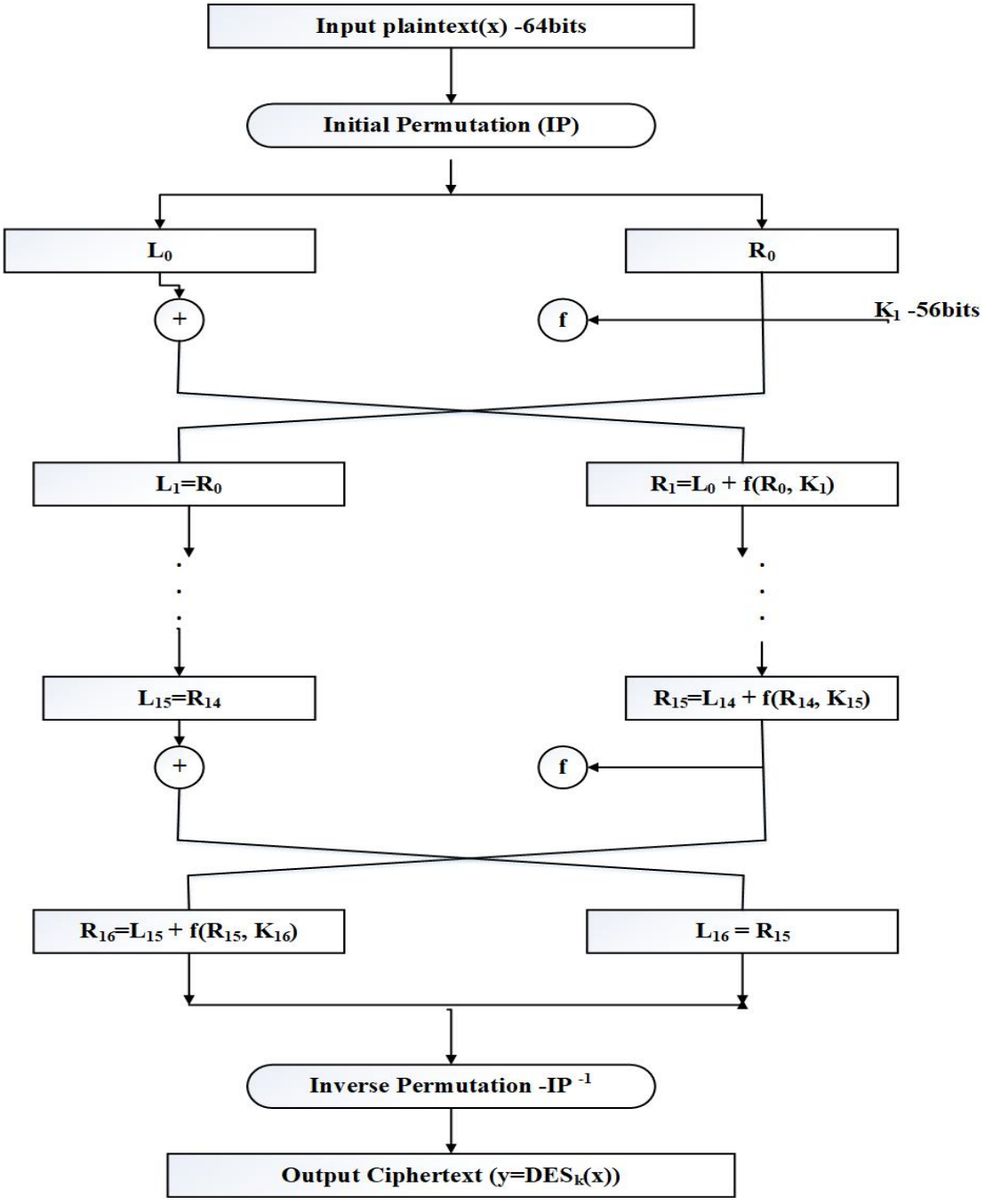

Figure 2.3: Structure of the DES cipher

(right 32bits), after which the following function is applied to L_i and R_i respectively, for sixteen iterations:

$$L_i = R_{i-1} \quad \& \quad R_i = L_{i-1} \bigoplus (R_{i-1}, K_i), \quad for \quad i = 1, 2, \ldots, 16 \tag{2.1}$$

The symbol $\bigoplus$ denotes the XOR operation while the function f takes the 32bits right half of the 64bits block and operates it with 48bits of the 56bits of the key to produce a 32bits, which is then XORed with the left 32bits (L_i) and serves as a next 32bits half of the message block in the next iteration. According to [90], the 64-bit data block is divided into two halves: the left half, L, and the right half, R, to be handled independently. In each round, a main function f operates on the right half of the data using a round sub-key (k_i) of 48 bits. This process is repeated for 16 iterative rounds after which the inverse of the initial permutation is applied on the sixteenth round to obtain a cipher text. The decryption of an encrypted block of message is achieved by reversing the encryption process. Thus, DES encryption of a message block say x is defined by: $y = E_k(x)$ while the DES decryption of a ciphertext block say y is defined by: $x = D_k(y)$.

2.5.2.2 DES in Internet of Things

Clearly, the Data Encryption Standard is an algorithm which predates the birth of the IoT technology. However, According to [91], although the DES is nowadays considered to be insecure against a determined attacker, it is still being widely used in legacy applications. Furthermore, pertinent questions on the security and usability of DES in the IoT domain would bother on whether; DES is efficient for use in the widespread deployments of IoT. What will be the security implications bearing in mind the known design limitations of the algorithm and the known attacks that have been successfully staged against the cipher. As power is also a limited resource in the IoT landscape, it is also worthy to query what the complexity of DES would be in terms of the limited resource of power consumption in the IoT, memory requirements, and taking into cognizance the light weight operating systems that run on low powered IoT devices. According to [92], DES has a high speed on hardware and it has a slow pace on software due to many permutations of bits, which makes it not so viable for IoT applications.

2.5.2.3 DES SECURITY

According to [78], one of the changes that was believed to be made by the NSA before the release of DES was the specific design aimed at making the algorithm to be resistant to differential cryptanalysis attack, an attack which was not known to the public until 1990. This is widely believed to be the reason behind the reduction of the encryption key length of 128bits as originally proposed by IBM in line with Lucifer cipher-design to a key length of 64bits, which in turn uses 56bits as the actual key length. According to [85], although the secret key consists of eight

bytes (64bits), one bit of each byte is dropped, which means that the key size is effectively 56 bits. This differential cryptanalysis defense minded design creates a brute-force vulnerability to the structure of DES, in that the key space is greatly reduced to 2^{56} as compared to 2^{128}, as the notion of breakability (by brute-force) of an algorithm is often analysed in the context of the size of the key-space. This is usually expressed in bits, where n-bits security means that the attacker would have to perform 2^n operations to break it and thus, weakening 2^{128} to 2^{56} in the case of DES. According to [78], despite very extensive cryptanalysis over the lifetime of DES, analytical attacks on it are not very efficient but the exhaustive key search (Brute-force) is relatively easier due to the size of the key space: 2^{56}. Consequently, even though regular computers are not efficient in staging this exhaustive key search of the 2^{56} key space, special purpose computers can be very efficient in staging the attack, thus making DES breakable. In many other instances, research experiments have demonstrated the somewhat insecurity or breakability of the DES algorithm. In [88], only one fault injection was shown to be required to recover the secret keys of DES using Algebraic Fault Analysis (AFA). [88] proposed an optimized hardware trojan based attack on DES and exploited the recovery of the secret key. [93], showed that Given enough known plaintext–ciphertext pairs, it is possible to recover the correct 56-bit key involved in the first 16 rounds of the DES function. [92] used Simple power Analysis (SPA) to visualize the sixteen rounds of DES operations and showed that it is possible to determine a correlation between power trace and sensitive data. According to [86], DES security was a highly contentious and controversial discourse until DES became an insecure algorithm in the year 1999. According to [78], in the year 1998, the Electronic Frontier Foundation (EFF) built a hardware machine called "deep crack" which performed a brute-force attack on DES in 56 hours. In the year 2006, a team of researchers from the university of Bochum and Kiel in Germany built a code-optimized parallel code-breaker (COPACOBANA) based of commercial integrated circuits, with which DES is said to be breakable with an average search time of seven days. According to [78], the summary of the security of DES is that the 56bits key is now considered to be too short for encrypting confidential information, and thus, standard DES or single DES should only be used for applications requiring short time security -say a few hours, while the triple DES variant of DES (3DES) is still considered to be secure.

2.5.3 The Triple-Data Encryption Standard (3DES)

The short key length has therefore been long established as a weakness of the DES algorithm and the 3DES being a viable alternative to the standard DES. According to [23], the triple 3DES was mainly proposed to achieve a higher security where DES had failed, but unfortunately introduced an important overhead, which is the cost of triple encryption/decryption, which tripled the requirements of resources and increases latency. In addition, 3DES was not suitable for real-time applications or to be used on tiny devices, a situation which is not in the best interest of the IoT when juxtaposed with its reality of constrained properties of devices.

2.5.3.1 Tripple DES (3DES) Design

The triple DES (3DES) algorithm is a multiple encryption algorithm. According to [77], multiple encryption is a combination technique aimed at improving the security of a block algorithm. With the multiple encryption technique, the same message block is encrypted repeatedly using multiple keys. The 3DES is one such algorithm where the standard DES algorithm is used to encrypt message blocks three times with multiple keys, thereby resulting in an enhanced security. According to [94], 3DES employs three DES operations on 64-bit data with three 56-bits keys. The diagrammatic structure of the 3DES algorithm is therefore the same as that of the standard DES above, only the encryption process is done three times with multiple keys, resulting in the key length of the 3DES being up to 168bits as against the standard DES key length of 56bits due to the definition of the multiple encryption function as defined in what follows:

According to [77], Each 3DES encryption and decryption operations is a compound operation of DES encryption and decryption operations defined as follows:
Encryption of a message block say x is given by: $y = E_{k_3}(D_{k_2}(E_{k_1}(E_{k_1}(x))))$
Decryption of a message block say y is given by: $x = D_{k_1}(E_{k_2}(E_{k_1}(D_{k_3}(y))))$
Where K_1, K_2 & K_3 are independent encryption keys, OR K_1, K_2 are independent keys and $K_3 = K_1$, OR $K_1 = K_2 = K_3$

According to [95], the 3DES algorithm expands the key space by repeating the procedure of single DES three times using two or three 56-bit different keys, which therefore resulting in the effective key length of up to 168 bits.

2.5.3.2 3DES in Internet of Things

Because the triple DES algorithm is essentially an extension of the standard DES algorithm that repeats the enciphering of the data blocks, the triple DES is mostly plagued by most of the drawbacks of the standard DES algorithm and even more. This implies that the drawbacks of the standard DES algorithm with regards to IoT considerations as discussed in 2.5.2.2 also come to bear as inherited drawbacks in the triple DES (3DES) algorithm, hinged on fact that the 3DES inherits the same round function and other core design properties of the standard DES algorithm. According to [85], the triple encryption/decryption also triples the cost required resources and latency as a corresponding consequence. Also according to [95], 3DES is slower than other cipher algorithms. This again when juxtaposed with the constrained properties of IoT devices in addition to the inherent drawbacks of the standard DES algorithm makes the triple DES not a very viable option for the IoT scenario. According to [96], the running time of the algorithm imposes a constraint on its applicability in several domain. They proposed that an extension of DES into Galois fields of $GF(16)$ with a 256-bits key might be a good alternative to the advanced encryption standard if the technology is sufficiently developed to run fast enough.

2.5.3.3 3DES Security

Although DES is still adjudged to be reasonably secure in the sense that staging an exhaustive key search over the key space of 2^{56} is still challenging for normal computers, it has become very clear that this level of security is threatened by the rate of growth of computing resources. To address this security vulnerability, the triple DES (3DES) which encrypts message blocks three times of the DES function is used to harden security in this regard. According to [89], 3DES uses three times 64 bit key which equals 192bits key, but actually uses 168 bits and 24 bits are used for parity check and 3DES is strengthened to perform billion of times more secure than the DES against brute force/exhaustive key search attacks. The perspective of insecurity in DES which is mainly the key length of 56bits is well covered in 3DES, owing to the possibility of having up 168 bits of key length as seen above in the encryption function of 3DES. According to [89], the security of triple DES is billion of times higher or stronger than that of DES, even though this comes with a corresponding consequence of the triple DES (3DES) being three times slower than standard DES as it essentially applies the DES algorithm three times on each block of the data.

2.5.4 The Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a block cipher cryptographic algorithm defined by the National Institute of Standards and Technology (NIST) in 2001, following the concern of the insecurity that plagued the DES and 3DES algorithms and calls for new algorithms to be standardized. It is a block enciphering scheme which takes in messages in blocks of 128bits and supports three distinct key lengths of 128, 192 and 256bits with corresponding cipher rounds of ten, twelve and fourteen respectively.

2.5.5 The AES Design

Unlike the DES and 3DES algorithms which follow the design architecture of Feistel networks, the Advanced Encryption Standard design is based on Rijndael cipher. According to [78], although the Rijndael block and key size vary between 128, 192 and 256 bits, the AES standard only called for a block size of 128 bits and hence, only Rijndael with a block length of 128 bits is known as the AES algorithm. The AES algorithm takes inputs (messages) in a fixed block size of sixteen bytes (128bits), perform an encryption on it and outputs a ciphertext of the same size, as shown in Fig 2.1

The decryption process of the encrypted message is also done on fixed block sizes as in the encryption process, only this time reverses the operations of the encryption process. The process flow and stages of the round function with which the AES performs encryption of blocks of 128bits of messages is as diagrammatically presented in Fig. **??**

STATE/Initialization (Key Whitening)

The result of the State/initialization stage which is also known as the key whitening stage is basically having the message bits XORed with the key bits: Message (16bytes) XORed Key(16bytes).

The Rounds Function

The SubBytes Stage: In the SubBytes Stage, each of the bytes in the stage above is replaced by another byte, depending on the key. These substitutions usually follow the presentation of the Rijndael S-box look up tables. According to [97], the S-box maps an 8-bit input c to an 8-bit output $S(c)$, where both the input and output are interpreted as polynomials over $GF(2)$. The round function of the AES cipher executes in four stages, the process of encryption and corresponding decryption.

The ShiftRows Stage: The ShiftRows operation is done on the result of the STATE produced after the SubBytes operation. Visualized as a matrix, the ShiftRow is an element-wise rotational operation with elements of the first Row being shifted by zero to the left (or not being shifted at all), the second row being shifted by one element, the third element shifted by two elements and so on until the last Row is shifted.

The MixColumns Stage: The MixColumn operation is done on the STATE, having undergone the shiftRow operation. The operations involve multiplying each column of the data block with an irreducible polynomial in the Galois Field $GF(2^8)$.

Add Round Key Stage: During the key expansion, the original key is used to generate round keys that are used in the rounds. This means the key in each round is different, although all generated from the same key during key schedule. In the add round key stage, the STATE bytes from the MixColumn stage are XORed with the bytes of the sub-key for that round. Depending on the key length which determines the number of rounds, each round of the AES encryption process except for the last round; performs the process of the round function above. The last round ends with the MixColumns stage and outputs the ciphertext.

2.5.6 AES in Internet of Things

Although a classical security algorithm, the AES algorithm is currently the most widely used security algorithm in IoT deployments and services. In [25], a survey of the technical specification of various Low Power Wide Area technologies shows that the AES is currently the standard algorithm for authentication and encryption and thus, crowning the AES as the algorithm currently used for communication security even in constrained IoT standards. An immediate consequence of this reality is the fact that all the aforementioned challenges of constrained IoT devices with classical security algorithms therefore abound in the current IoT landscape, thereby creating a leeway for the exploration of newer security schemes or the modification of existing ones such as the AES to salvage the challenging situation constrained IoT devices.

2.5.7 AES Security

With a simple and elegant design [77], the security of the algorithm is rooted in the mathematical complexity of computing polynomial operations and the associated difficulties in reversing same. However, while the complexity of the cryptographic functions are vital to ensuring the security

of the algorithm, according to [28], it is not easy to implement sufficient cryptographic functions on devices that are constrained (such as IoT) devices due to their limitations in processing capabilities and also in terms of power. The security which is a function of the mathematical complexity of the algorithm is however, consequent upon the underlying hardness of computations of polynomial elements in finite fields. A detailed analysis of the mathematical constructions of the AES algorithm is presented later in chapter3.2

2.5.8 AES Complexity

[98] implemented the Advances Encryption Standard (AES) on the MATLAB environment, whereby a block of plaintext was fed into the algorithm for encryption and the resulting ciphertext was re-inputted to demonstrate the encryption and decryption process. They utilized the tic toc clocking functionality of MATLAB to capture the respective times of the key setup, one round, complete encryption and decryption processes. Experimental result of their work showed that the key setup process takes about 0.45ms, a round completion took 12.56ms (for the first round), following which the time lapse in completing the AES encryption and decryption processes was detailed. However, these results show the implementation of only the AES 128-bit key length, while taking inputs as different message blocks although with a fixed block size of 128 bits (16 bytes). Also, this implementation was aimed at demonstrating AES security for data protection in several applications in areas such as: video processing, image processing, control systems and communications. In a related work, [99] implemented the Advanced Encryption Standard (AES) with respect to analyzing the encryption calculation requirement and power consumption in different payload lengths. They considered different implementation types which include: software-based AES-ECB (Electronic code book mode), hardware-based AES-ECB and hardware-based AES-CCM (counter with CBC-MAC mode) on the LAUNCHXL-CC1310 as an experimental platform. They noted that the AES algorithm consumes many Central Processing Unit (CPU) cycles, an action which may lead to an unwanted degree of power consumption. Starting with the non-AES to the other modes listed above, they measured the power consumption of the different encryption types on two sets of LAUNCHXL-CC1310, where the RF transmitter encrypts data before transmission, and the one equipped with the RF receiver decrypts data after receiving it and while utilizing the Time-Division Multiple Access (TDMA) mechanism in the experimentation. Extra waste rates were calculated with respect to when payload (message) length was less than or more than the specified AES block size of sixteen bytes. The

results of their experimentation were presented in terms of the change in the message length after encryption of the AES modes considered, and the encryption power consumption of the different encryption types. The results of this work reveals a correlation between power consumption and payload lengths, although did not consider these implementations for the three different key sizes of AES, and a measure in the difference of power consumption of the different AES types and the non AES emphasizes the cost of having to run this encryption scheme on device, -a point, not negligible in the discussion of devices constrained in power such as IoT devices.

2.5.8.1 AES Complexity in Low Power Environment

Long range Wide Area Networks reduce communication power by setting different transmission latencies for different end-devices. It adopts a widely used data encryption method, the Advanced Encryption Standard (AES), developed based on powerful algebra operations and multiple encryption cycles to ensure its communication security. however, AES does not consider its end device's encryption power [100]. Battery power must be conserved on low powered devices; thus, it is a major goal for their cryptographic implementations to minimize power consumption. This consideration has been paramount in the design of symmetric cryptographic primitives for constrained devices[101]. Furthermore, the consequence of using complex encryptions such as the AES on IoT devices is that it impacts on the performance of the network. [100] proposed a Secure Low Power Communication (SeLPC) method to further reduce end-devices' data encryption power by reducing encryption cycles of AES, so that power consumption consequent upon the complexity of the AES can be further reduced. With a focus on striking a balance between security and power consumption of low powered devices in a low-power environment, their proposed algorithm reduced the encryption cycles of AES-128-bit key length, from the default 10 cycles to 5 cycles. This is aimed to demonstrate that a less complex security is more suitable for low powered devices with respect to their constrained nature. They carried out a power consumption analysis and comparison of the traditional AES-128 and their version of the simplified algorithm. Their results showed that the simplified algorithm could reduce about 26% of the encryption power of the traditional AES-128. The results of the above-shown implementation and power consumption analysis of the improved algorithm was carried out using ARM Cortex-M4 processor and low power content addressable memory (CAM) architecture to simulate encryption process and lookup table, respectively.

2.6 Lightweight Security Algorithms

Resource constrain is a critical problem in the deployment of IoT devices for reasons of complexity associated with classical security algorithms. This is largely due to the fact that classical cryptography predates the IoT era and the design considerations of the classical algorithms did not broadly include the plethora of constrained IoT devices which have become an integral part of the technology industry and in fact, the future. According to [102], Lightweight cryptography is generally defined as the cryptography for resource constrained devices, for which Radio Frequency Identification (RFID) tags are mentioned as examples. Consequent upon the constraints on low power devices in terms of area, processing capabilities, memory and scarce power resources, Light weight cryptography emerged in efforts to ensure the security of these devices in the digital communication space. As rightly put by [28], Lightweight Cryptography (LWC) or protocols are tailored for implementations in constrained environments including RFID tags, sensors, contact less smart cards, health care devices and so on. The development of lightweight cryptographic protocols therefore, is chiefly anchored on the need to develop cheaper security schemes that are compatible with the constrained nature of these devices and without compromise to security as according to [103], the running time of the algorithm imposes a constraint on its applicability in several domain. In [104], the performance analysis of two lightweight ciphers: CLEFIA and PRESENT was presented with respect to security strengths, throughput and resource utilization, which had the latter outperforming the former in terms of memory usage, security, and the former outperforms the latter in terms of throughput. PRESENT is a lightweight algorithm with a Substitution Permutation Network (SPN) structure, and utilizes thirty one rounds of:XOR RoundKey, S-box layer and P-layer, with a final (32nd) round which XORs the STATE produced from the first thirty one rounds and the round key. It carries out message encryption in sixty four (64) bits blocks and and supports key lengths of sixty four (64)bits and one hundred and twenty eight (128)bits. The efficiency of a lightweight cipher (CLEFIA) in comparison to other conventional ciphers such as the AES, Camelia and Seed is detailed in [28], wherein the efficiency of lightweight ciphers defined as a ratio of throughput and gate size is presented with respect to energy consumption.

[104] reveals that the multiple lightweight methods developed so far have their respective merits and demerits, and determining a right choice to secure data, depends on various factors including the nature of the application or usage. [45] noted that the biggest challenge in the design of these lightweight algorithms is on how to cope with trade-offs between cost: in terms of computing resources required to run the algorithms, performance, and the security of the algorithms. Even with the plethora of lightweight ciphers aimed at addressing the problems of resource constrains in devices with limited capabilities, the lightweight algorithms are still limited in terms of design and applicability, in addressing the many security issues in the IoT landscape. [45] noted that the biggest challenge in the design of these lightweight algorithms is on how to cope with trade-offs between cost: in terms of computing resources required to run the algorithms, performance, and the security of the algorithms. To this end, the work in [105] leverages the provision of a tamper-proof secure element as a one of such desirable trade-offs aimed at providing an efficient algorithm of reduced complexity that is compatible with resource constrained devices, without compromising the security. According to [31], some of the challenging problems in the implementation of the IoT include: key management, device authentication, user access control, privacy preservation and identity management to mention but a few.

The growing need to develop less-complex algorithms that are suitable for deployment in the IoT scenario have also led to advances in lightweight ciphers which are targeted specifically at these constrained devices. According to [80], The existing cryptographic systems need some area for implementation, which is quite a challenging requirement in embedded systems and thus, lightweight ciphers which can be easily implemented in resource constrained devices were introduced. Popular among these IoT-suitable category of algorithms include but not limited to: Clefia, present, Katan, Simon, Trivium, Rectangle, Chacha, and espresso. However, advances in this direction is yet to produce results that enjoy the endorsement, standardization, and wide spread use-cases as are the cases with the standardized/classical algorithms. Rather, this development has also been closely identified with known challenges. The software implementation of PRESENT for instance, according to [80] is found to be inefficient as the cipher consumes too much energy and have low throughput -which are features that are in themselves unsuitable for the IoT narrative. A survey of comparative study of this category of lightweight ciphers according to [80] reveals that most of the lightweight ciphers are application specific and thus having

very limiting usability in a wide range and consequently, the best for the IoT landscape in terms of less-complex security algorithms remains to be seen and continues to attract more attention in terms of research.

2.7 Reduction of Complexities of Classical Security Algorithms

According to [106], IoT devices are known for their limited memory space and computational capabilities, and conventional solutions such as encryption methods are inadequate to solve many privacy concerns. According to [103], the running time of the algorithm imposes a constraint on its applicability in several domain. In favor of the narrative of constrained IoT devices, they proposed that an extension of DES into Galois fields of GF (16) with a 256-bits key might be a good alternative to the advanced encryption standard if the technology is sufficiently developed to run fast enough. In [107], A low power algorithm aimed at improving on the power consumption by classical algorithms for IoT was proposed. Observing that the power cost of transmission and reception of data typically outweighs the cost of the cryptographic algorithms themselves, they proposed a method called Authenticated Encryption with Replay protection (AERO), which shows to significantly reduce overheads even when used in higher-layer protocols above the link layer. This work suggests that the cost of transmission and reception of messages could be reduced by up to 30%, which translates into improving the limited resource of power in constrained IoT devices, although [108] observed that significant power can be saved by this method but the security of the method needs to be confirmed. The authors in [109] proposed a lightweight enhanced Distributed Low-rate Attack Mitigating (eDLAM) mechanism in tandem with the constrained resource narrative of IoT devices, which aims to mitigate DDoS attacks and obtain maximum utility in IoT deployments. In [108], an AES-128 based Secure Low Power Communication (SeLPC) algorithm for an IoT environment, which details in two phases namely: the key generation phase and data encryption phase was proposed. The algorithm aimed at significantly improving on AES to meet low powered devices security constrains, detailing that in the standard AES encryption process, the SubBytes stage typically looks up S-Box to encrypt and decrypt data stream but as the contents of the S-Box in AES are fixed, this greatly reduces its security level since the only nonlinear component of this block ciphering technique is the manipulation on S-Box. To enhance AES's cryptographic strength, an encryp-

tion key that generates the corresponding dynamic box (D-Box) to substitute for the primary substitution box(S-Box) was derived. Following this, the simplified standard AES-128 encryption process of 10 cycles down to 5 cycles with the aim to reduce computational complexity and save power consumed by end devices in an IoT environment, although the reason or rationale for simplifying to specifically 5 rounds was not stated. However, the SelPC algorithms utilizes an Enhanced Dynamic Accumulated Shifting Substitution (EDASS) algorithm which leverages high input sensitivity and randomness to harden the security of the D-Box against attacks. According to [108], if a hacker would like to decrypt an application-layer message, the attacker needs to know the 128-bit AppSKey and D-Box. As n-bit security is defined by 2^n; where n = number of bits, the possibility of the AppSKey and D-Box combination multiplies to $2^{128} \times 256!$ and thus, enhancing the security of the standard AES-128 algorithm with the default n-bit security of 2^{128} bits. Results of their method shows that the SeLPC algorithm can save 26% of power consumption in comparison to the traditional security algorithm based on the standard AES algorithm in a low-power environment and thus, improving the security of constrained IoT devices.

2.8 Cloud Computing and IoT Provisioning

According to [39], cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing deployment models range from: a private cloud deployment where the cloud services are provided for private use of a single organization with many users, a community cloud which is often deployed in the use-case of many organizations coming together to form a community of shared, required cloud services, a public cloud which is open for public use in contrast to the private and community deployments models and often deployed by government or private business owners and finally, a hybrid deployment model which is often a combination of any two or all of the aforementioned deployment models. In [110, 111], cloud service models are discussed to range from the provision of software as a service,to platform as a service and infrastructure as a service in cloud deployments; where Software as a Service (SaaS) provides room for users to rent software on the cloud instead of high spending in buying the software, Platform as a Service (PaaS) provides development platform services

and Infrastructure as a Service (IaaS) provides compute resources including network devices, memory and storage among others [111]. Through the provision of these cloud services models, the need for building a software for instance, can be met by application developers leveraging the Platform as a Service (PaaS) as a coding platform, through which virtual servers can be accessed in a matter of seconds via the Infrastructure as a service (IaaS) while paying only for resources used [112]. According to [39], cloud computing has gone beyond being just a technical solution to also include a business model through which computing power can be sold and rented. From a public cloud deployment stand point as owned by businesses, Cloud computing is defined as the on-demand delivery of IT resources and applications via the Internet with a pay as you go pricing model in [113]. While IoT generally refers to the plethora of devices on one front, the notion of an IoT cloud on the other hand, refers to a cloud computing platform which gathers data from these physical devices, processes it, and facilitate sharing of these data among other components in the IoT ecosystem, thereby providing the needed platform for the development of the IoT. According to [114], Internet of Things connects different devices in order to collect and exchange useful data, while Cloud IoT solutions enable control and remote device monitoring and one of the main purposes of an IoT cloud is to collect data, process it and exchange it with other system components. A review of various IoT cloud platforms from a security perspective is detailed in [115], covering IoT platforms anchored on cloud computing including: ThinksWorx IoT platform, Microsoft Azure IoT suite, Google cloud IoT platform, AWS IoT platform, Cisco IoT cloud connect, Oracle IoT platform, Thingspeak IoT platform and GE Predix IoT platforms among the leading ones of the rapidly developing area of cloud enabling IoT, offering many services towards the actualisation of the full potential of the IoT technology and to converge the cloud and IoT ecosystem. The benefits of cloud computing as a technology innovation becomes immediately glaring on comparison with traditional, fixed, on-premises IT infrastructure for networked devices, offering many benefits over the traditional networked digital communication environments which have become unsuitable for the distributed nature of the IoT. As low hanging fruits in the paradigm of cloud computing, capital expense associated with budgeting, purchasing and deployment of the fixed infrastructure is transformed into variable expense, whereby resources are purchased on a need-to-use bases while spending in line with the workloads for which the infrastructure is required. Economies of scale, efficient planning of capacity while eliminating resource redundancy associated to the purchase of fixed infrastructure, speed and agility in expansion and decommissioning of networked systems which was hitherto impossible. Strategic

spending via elimination of running data centers where resource utilization is sub-optimal base on expansion and contraction of organizational workloads, has also become possible. Furthermore, Ease of deployment enabling the provisioning of globally available resources within minutes as opposed to the constraint of long procurement cycles, succor for delays and installation times of physical infrastructure as obtainable in traditional on-premises networked systems, have all been availed by the cloud computing. Cloud computing also promises: highly available, fault tolerant and scalable infrastructure of elastic nature. These features empower cloud-based networked systems with benefits of instantaneous recovery of failed systems in events of disaster while having an almost zero downtime in terms of its availability for the indented purpose, in addition to the agility of scaling out with growing workloads and scaling in with reduced workloads within seconds. Although a walk away from traditionally fixed, on-premises infrastructure, cloud computing platforms are not devoid of security challenges. An analytical review of the security of data in a cloud environment is summarized in [116], whereby security issues associated with the distinct cloud services models are outlined, including: highlighting the challenges of disaster recovery in traditional systems as compared to the ease of drawing up a recovery plan and executing same in a cloud computing setting, the rigidity of using multiple encryption schemes in traditional systems as it compares to the flexibility of executing multiple encryption techniques on the cloud where resources are almost at unlimited scale, the manpower demand for managing on premises/ traditional systems as opposed to the seamless, reduced manpower demands of spinning up resources for use in a cloud computing setting towards meeting an organisation's IT needs, the long procurement cycles, cost and delays associated with device purchase and installation as opposed to the on-demand access to cloud resources on a need-to-use bases thereby saving cost. Cloud computing therefore serves, and continue to serve as the key technology innovation enabling the deployment of IoT devices, with leading cloud services providers including the AWS, GCP, Microsoft Azure, etc [116], and recent advances in this direction including the standardization of a universal cloud interface towards facilitating the integration of IoT clouds according to [117]. The massive roll out of resources occasioned by cloud computing has led to the emergence of various cloud platforms as not just the choice for provisioning IoT devices, but a defector enabler towards the complete actualization of ubiquitous deployments of the IoT as according to [43], Cloud-assisted IoT has become an increasingly popular technological trend, as the performance of IoT applications can be greatly improved by delegating the cloud to manage massive IoT data. To this end, a leeway has been created for the massive provisioning of the IoT, with

the constrained devices outsourcing among other things; storage of generated data on to the cloud. However, even with the deployment of state of the art protocols such as the Transport Layer Security (TLS) for securing remote communications between the IoT devices and the cloud as a countermeasure, vulnerabilities remain and attacks can occur between an IoT device and the cloud during translation protocol of secure transport protocol in Constrained Application Protocol (CoAP) [118]. More so, traditional security architecture is said to be broken as the user does not own the cloud [39] and so, creating the need for innovations in effectively harnessing the advantages of the cloud as utilized by IoT devices in outsourcing data. Consequently, the need for encrypting data at the device-end before pushing it to the cloud. Data stored in the cloud can be encrypted at rest or in flight, with encryption at rest being either at the end of the IoT device (Client-side Encryption) or on the cloud (Server-Side Encryption).

2.8.1 Authentication

Authentication is the process or action of verifying the identity of a user, thing or process, against what it declares itself to be, whereby a system checks credentials against a database or authentication server and authenticates the requesting entity upon successful verification of the credentials provided [112]. As an all important process in implementations of information systems, authentication methods would continue to evolve in order to curb the many security challenges in information systems. From legacy networked systems to modern networks and onto the paradigm of cloud technologies and the IoT, the principle of authentication has remained central in ensuring connectivity. Methods of authenticating users or processes in information systems range from simple password based methods using Password Authentication Protocol (PAP) or Challenge Handshake Authentication Protocol (CHAP) protocols, token based methods, certificated based method and biometrics to multi-factor methods etc. Due to it's enriched evolution from legacy networks to modern networks, the topic of authentication has gained advancement on many fronts and considerably gained complexity as networks continue to evolve. Processes and types of authentication which were hitherto not complex in the realm of fixed, on-premises infrastructures have changed dramatically with the burst of decentralized networks and applications assisted by cloud computing technologies, such as in the IoT. According to [119], authentication plays a critical role as one of the most important entry point into all kinds of information systems by ensuring the right user has access to the right system with the right identity. In a blockchain-based decentralized authentication modelling scheme in edge and IoT environments as

detailed in [119], the current traditional authentication methods are observed to be implemented as centralized schemes which are somewhat weak and prone to poor fault tolerance, reliability, and the danger of being a single point of failure. Instead, a decentralized, safe and reliable solution which uses edge devices to build blockchain nodes to form a block chain network, where all nodes can equally participate in the management of all nodes was proposed in [119]. According to [120], authentication of IoT devices is crucial as security and privacy stand as vital issues in connecting IoT devices which send information to and from users and offload information in the cloud. However, traditional security mechanisms consume too much energy and thus the much needed research of cheaper security mechanisms for constrained IoT devices[120]. The authors in [31] observed that some of the challenging problems in the implementation of the IoT include: key management, device authentication, user access control, privacy preservation and identity management to mention but a few. To this end, a smartcard-based key agreement framework for cloud computing using elliptic curve cryptography as detailed in [121] is proposed, whereby key agreement is achieved directly between the user and the cloud platform. In [122], efforts towards secure authentication have been highlighted to include the employment of bio-metric credentials such as; iris patterns, retina, face and fingerprints etc. However, these biometric means bothers on sensitive personal information and potentially open up privacy vulnerability issues. The authors in [123] observed that the noncancellability of traditional biometrics such as fingerprints for authentication increases privacy disclosure risks when the biometric templates become exposed, because users cannot create new templates at volition. The challenges of algorithm complexities as detailed in 2.5 however, is not absent in the processes of authentication in the realm of constrained IoT. As the process of authentication is facilitated through the implementation of Transport Layer Security (TLS), and the inner workings of the TLS protocols involving algorithms is worthy of exploration to examine the burden placed on constrained IoT devices in achieving secure authentication. As put by [124], security is a pre-requisite for IoT deployments and many IoT frameworks are using TLS or DTLS (Datagram Transport Layer Security) for authentication purposes. Helpful to the implementations of several IoT solutions are the many innovations around the modification of communication algorithm and protocols used in legacy systems, to suit the narrative of resource constrains in the realm of the IoT. This has seen the development of IoT targeted solutions such as the standard for low footprint IP version six (IPv6) addressing scheme for the Low Power Personal Area Networks devices (6LoWPAN), Constrained Application layer Protocol (CoAP), Light Weight Machine to Machine (LWM2M) protocol and

Message Queuing Telemetry Transport (MQTT) to mention a few, and improved authentication mechanisms with respect to IoT constraints remains an area of fertile research as the IoT technology continues to develop. Whereas simplified password based authentication mechanisms have become unsuitable for many use cases due to insufficient security an incompatibility with M2M deployments among other reasons, efforts towards ensuring secure authentication have also seen multi-factor authentication mechanisms introduced, experimented, criticized and advanced. According to [125], the role of a gateway node (GWN) in wireless sensor networks communication model as a subset of the wider category of the IoT; is explained as the medium between participating sensors and the legal user desiring to get information from the sensors, whereby the gateway node is equipped with more power and computational capabilities and handles the more complex tasks of calculations and other resource-intensive responsibilities in WSN deployments, enabling communicating nodes to reach a mutual authentication and construction of a session key which can subsequently be utilized for communication following secure authentication. Other network models such as; the user directly reaching the sensor node before the gateway which fosters communication with the node in a similar fashion to LAN switching communication has since been adjudged to be too expensive in terms of resource requirements. While multi-factor authentication considerably hardens the security of networked systems against attacks, an analytical survey for improving authentication in cloud computing systems as detailed in [112] reveals an associated level frustration by users as security levels of authentication increases. In [126], recent trends in biometric authentication based on IoT is presented, detailing the criteria for biometric authentication to include physiological biometrics such as; hand geometry, finger prints, face, iris, veins, etc, and behavioral biometrics such as; keystroke, gait, voice, signature etc. However, according to [123], traditional biometric tokens such as face recognition, voice and gait, are still naturally exposed to impostors while others, such as DNA and fingerprint, can be easily recorded by impostors without the user's knowledge. Moreover, most traditional biometrics are noncancelable and once the biometric template is exposed to impostors, it is permanently compromised because the user cannot create a new template. [126] proposed a multi-mode authentication system which combines the use of more that one model of authentication to provide a more reliable authentication process of IoT deployments. Instead of the use of non-cancellable biometric properties for authentication, [123] proposed a cancellable biometric modality based on high density surface electromyogram encoded by hand gesture passwords, for user authentication. In [122], a new low-cost, low-power tangible user-interface based authentication system is proposed, which leverages

physical vibrations to support authentication for emerging IoT devices and applications in smart access security systems. In [39], securely authenticating an IoT device onto a cloud platform is highlighted as one of the major issues affecting cloud adoption. The authors in [127] observed that as a resulting consequence of less storage capacity, memory and processing capability, many IoT devices have to be operated on lower power and hence, the security measures fail here and the devices become the victim of expensive cryptographic processes. Furthermore, a distributed algorithm to be used in the IoT structure in order to reduce the security risks confronting low-powered devices was proposed by [127]. In attempt to address issues bothering on data security in the IoT paradigm, Named Data Networking (NDN) have also been used as signature schemes to aid authentication mechanism in the IoT. However, the authors in [128] highlighted the drawbacks of applying the NDN-IoT schemes as potentially requiring additional encryption to be carried out by constrained IoT devices and thus, imposing the problems of cryptography-based authentication and vulnerability to impersonation attackers with higher processing powers that are capable of reconstructing authentication keys. They proposed a solution that integrates the lightweight and unforgeable physical-layer identity (PHY-ID) into the existing NDN signature scheme. Keys used for establishing communication sessions are traditionally negotiated using key agreement protocols of public keys and private key pairs. However, the authors in [129] observed that embedded systems are limited in their capability to implement public key encryption and client- side authentication. Proposing a solution with respect to the constrained characteristics of these devices, they proposed a protocol that seeks to remove the need for public key or certificate-based authentication by using Physical Unclonable Function-based identity. In [42], the small key size and computational efficiency of Elliptic Curve Cryptography is advocated as preferable for better security solutions over other Public Key Cryptography, with respect to the constrained resources of power, processing and memory of the devices. According to [31], authentication aims to identify the identity of a Thing, and it's detailed in a two-step process viz: the presentation of an identity and verification of the identity created, whereas one of the major challenges for constrained IoT devices remains the secure management of these authentications keys. Consequently, deploying constrained IoT devices with a more secure management of security keys which are used during authentication is highly advocated. While acknowledging the difficulty of IoT devices constrained in computation and power resources to effectively use standard security measures, the authors in [130] proposed a modified protocol for secure authentication in IoT smart homes based on a secure lightweight mutual authentication and key exchange protocol

for IoT smart home environments, which although viable in terms of lightweight, had become vulnerable to parallel session and replay attacks.

2.8.2 Client-side encryption and Constrained IoT Devices

client-side encryption is used to achieve encryption of data at the end of the IoT device before it is sent onto cloud. According to [104], a study by HP estimates that about 70% of devices don't encrypt data in communications over networks. Within the context of the IoT, this is largely due to the constrains of limited computing capabilities of the devices as according to[131], the integrated circuits (ICs) deployed in IoT based infrastructures have strong constraints in terms of size, cost, power consumption and security unlike in desktop computers, tablets, and so on, IoT devices are unable to allocate considerable memory and processing energy just for security functions [131]. The authors in [43] observed that to protect the confidentiality of data outsourced from IoT devices to the cloud, cryptographic mechanisms are usually employed to encrypt the data in such a way that only the user designated by the data owner can decrypt the data. They proposed a privacy-preserving data sharing scheme in cloud-assisted IoT which employs identity-based encryption and linear secret sharing that both preserves privacy of the IoT data pushed to the cloud as well as allow for flexible sharing of encrypted data between connected devices. The authors in [131] highlighted the main challenges identified in the deployments of IoT devices as resilience of the deployed infrastructure, confidentiality, integrity of exchanged data, user privacy and authenticity among others. According to [132], since more devices are being programmed to exchange data autonomously in IoT deployments, the importance of security and authenticity of such transmitted data is very crucial and thus requiring strong security approaches to prevent both passive and active attackers. According to [131], the insurance of privacy and data protection remains a challenge in IoT deployments desiring of solutions. They discussed the challenges, implementation and future applications of Light Weight Cryptographic (LWC) schemes in securing constrained IoT devices. Taking the constrained characteristic of the IoT devices into consideration, [133] proposed varying levels of security measures dependent on the confidentiality requirements of the data. [132] proposed a secure communication scheme for IoT devices which applies the Diffie-Hellman algorithm for authentication and uses the AES and Message Digest (MD)5 algorithms for encryption and validation respectively, of transferred data.

Hinging on the many benefits of cloud computing, the scaling number of users and the asso-

ciated challenges of cloud resources management, the authors in [134] observed the need for dynamic provisioning to aid secure and reliable provision of cloud resources. As cloud computing holds the answer for exponentially growing organizations like business and educational establishments [134], the dynamic nature of the autonomic resource provisioning approach is used to satisfy Quality of Service (Qos) of client's requirements to improve the optimization of resource management. With devices in IoT networks in billions, the need for these devices, as well as the making the cloud resources available for achieving the deployment purposes cannot be over emphasized. [134] presents a study on resource provisioning approaches in autonomic cloud computing, which is relevant to the implementation of cloud assisted IoT provisioning. While the making of IoT devices operational and the spinning of required cloud resources and services with minimal human intervention is instrumental in harnessing the full potential of the IoT technology, the process of authentication which ensures the secure connections, verification and validating the identities of the connecting entities must also be properly addressed. Accordingly, an arbitrary spectrum for enforcing security best practices, while taking into consideration, the obvious constrained nature of IoT devices is created. The high level overview of the work presented in this thesis looks at this arbitrary spectrum from the point of data encryption at the end of the constrained IoT device itself, otherwise known as client-side encryption, as well as the lightweight security consideration in the on-boarding process of a sample IoT device onto a cloud IoT platform; the AWS IoT core.

2.9 Chapter Summary

The review of major topics -relevant to the work done is presented in this chapter. Starting with a peek into the IoT technology and it's applications, a review of the security challenges in the IoT, as well as resource constraints in the IoT was presented. The review of classical security algorithms was presented, starting with a background of security algorithms in general, following which the review of specific security algorithms inclusive of the DES, the Triple 3DES was also reviewed in line with the notion of complexities when juxtaposed with the constrained nature of IoT devices. Based on available information in literature which shows the Advanced Encryption Standard (AES) as currently the most widely used algorithm in low power wide area networks as detailed in [25], a review of the AES as one of the classical algorithms was given with details into the design, the AES in the IoT, AES security and AES complexity. Hinging on

the complexities of the classical security algorithms and the consequent requirement of lighter schemes for constrained IoT, a review of lightweight security algorithms, as well as the reduction of the complexities of the classical ones to achieve lightweight was also presented. In light of cloud computing being a major enabler for provisioning IoT devices, a review of cloud computing and IoT provisioning was presented in a comparative sense of how cloud computing technologies differ from fixed, on-premises infrastructure, while focusing on the convergence of cloud computing and the IoT technologies. Consequent upon the cloud computing and provisioning review, a further review of authentication with respect to the on-boarding of IoT devices in cloud-assisted IoT platforms was also presented. This chapter concludes with a review of client-side encryption and the need for developing lightweight algorithms that are compatible with the resource constraint in IoT devices, towards secure provisioning of the devices onto the cloud-assisted IoT platforms.

Chapter 3

Analysis of Algorithm Complexities in Provisioning Constrained IoT Devices

3.1 Introduction

This chapter contains the mathematical bases and analysis upon which the research experimentation is based. First, a cryptanalytic overview and analysis of the consequence of reducing the complexity of classical algorithms is presented, and a mathematical background with respect to the basic algebraic structures of Galois fields used then follows. Also, the notion of averages, percentages and variance which are used in the processing and analysis of experimentation data in later chapters is covered as a part of the mathematical background in this chapter. Properties deemed necessary for analyzing the features of the AES algorithm and its operations are presented, as well as the underlying algebraic structures which form the crux of the mathematical operations and processes that yield the values of the S-box tables. These operations involving polynomials in the Galois Field $F = GF(p^n) = GF(2^8)$ include: polynomial addition and multiplication, together with their respective inversions. The mathematical analysis aim to show whether round reduction impacts on the outcome of the S-boxes tables of the AES algorithm,

or the security of the algorithm from the standpoint of attack types, as contained in the cryptanalytic overview and analysis. While element wise addition and its corresponding inversion of subtraction is straight forward with polynomials, the same is not the case with multiplication of polynomials and its corresponding inversion. The operations involving the polynomial elements of the Galois Field takes into consideration properties such as the cyclic nature of the field, generator elements, minimal polynomials and irreducible polynomials to mention but a few. Following an analysis of the consequence of complexity reduction, justification of the round reduction and security trade-off is provided based on the basic algebraic properties given in the mathematical background, with a sub-section 3.4.1 explaining the significance of the theorems used in the mathematical analysis.

3.2 Mathematical Background

Underpinnings of the analysis of algorithmic complexities is hinged on a select relevant topics to the work presented in this thesis. These topics include, but not limited to; the basic algebraic structures, averages, variances, percentages and finite fields. A table of useful symbols, considered relevant to the aforementioned topics is summarized in table 3.1. An algebraic structure refers to an arbitrary set on which arithmetic-like operations are defined. Operators involved in such operations on the elements of the arbitrary set could be addition, subtraction, multiplication and inversions. Section 3.2.1 details some of such structures used in the context of this work.

Table 3.1: Table of Important Symbols

Notation	Meaning
Nbr	Number of rounds executed by the round function
F	Field (as an algebraic structure)
n	An arbitrary natural number
GF	Galois Field
Deg	degree of a polynomial elements in the Galois Field
P	An arbitrary prime number
Mod	Modulo: the operation or function that returns the remainder of one number divided by another
$\mathbb{Z}_p$	Ring of integers modulo p
$n(x)$,$m(x)$, $\alpha(x)$,$\beta(x)$	Arbitrary polynomial elements of the Galois Field $GF(P^n)$
μ	Mean or average value of a set of numbers
$\sum_{i=1}^{n}$	The sum of a finite set of n numbers
σ	Variance
$\mathcal{O}$	The “big oh” notation

3.2.1 Basic Algebraic Structures and Galois Fields

GROUPS

A group (G,*) is essentially a set, endowed with the operation * such that the following axioms hold:

Closure: G is closed under the operation $*$ such that for each ordered pair $n, m \in G$, there exists a unique element $n * m \in G, for\ all\ n, m \in G$..

Associativity: the binary operation $*$ is associative such that given $n, m, z \in G$, the $(n * m) * z = n * (m * z)$, for all $n, m, z \in G$.

Identity: there exists an identity element $e \in G$ such that $n * e = e * n = n$, for all $n \in G$.

Inverse: to every $n \in G$, there is an element $n^{-1} \in G$ called an inverse of n with respect to $*$ such that $n * n^{-1} = n^{-1} * n = e \in G$.

RINGS

A set R defined with two binary operations $(R, +, *)$ is called a ring if the operations satisfy the properties of closure, associativity, identity and inversion as defined in a group above, for any arbitrary elements of R. As a ring has two operations: addition and multiplication, the requirement of the inverse and identity properties must be met with respect to the two operations defined on the ring.

FIELD

A field is an algebraic structure which incorporates the axioms of a group as defined above and those of a ring as well. As a ring, a field has two operations: Addition and multiplication, defined on it. As a group, all the elements of the field form an additive group with respect to the additive operator and with an identity element zero (0), it also forms a multiplicative group with respect to the multiplicative operator and identity element one (1), and the distributive properties holds when these two operations are joined together in an algebraic operation, such that for any arbitrary $n, m, z \in G, n(m + z) = nm + nz$.

IRREDUCIBLE POLYNOMIALS

According to [79], we can always calculate products in finite fields provided that we know the base field and the irreducible polynomial that was used to construct the finite field. According to [78], the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$ is a part of the AES specification. Given

any arbitrary finite field (also known as Galois Field), proving the existence of an irreducible polynomial is therefore, critical to guarantee the inversion of the multiplicative operator in the field, and accordingly, the resulting values of the S-Box tables in the case of the AES algorithm.

3.2.2 Averages, Percentages and Covariances

According to [135], the coding rubric for average does not have a hierarchical structure. Instead, the definitions were grouped according to the statistical measures of center: mean, median and mode. The concept of an average as applicable in this work however, is basically of representative values for data sets and are usually single values that are considered representative for experiment data. Precisely, experiments conducted in this work aimed to measure the amount of time it takes distinct algorithms or different variants of the same algorithm, to complete the processes of encryption as well as decryption, of defined message blocks. The concept of an average value thus, is applied in the instance of finding a representative value of the time it takes to complete the encryption of a single block of of plain-text, repeated in multiple instances. Defined as the sum of all such values divided by the number of values and expressed mathematically as: $\frac{1}{n}\sum_{i=i}^{n} x_i$, average encryption completion times are thus obtained for one thousand instances of message encryption as detailed in chapter 6 of this work.

Unlike the notion of averages, the concept of percentages is used to represent an amount, a number, or something with respect to being a part, of a total of one hundred. According to [136], like monetary systems based on the dollar, percentages have 100 levels that are easy to divide into increments of halves, quarters, and tenths, and are also easy to calculate and easy for most people to understand. One example of such applications of the concept of percentages in this work is in the expression of amount, of how much more time it takes to complete the encryption of message blocks between different algorithms, while holding experimental measure of the concept of averages as fixed. Furthermore, holding the sample devices such as: a laptop, a typical IoT device as constants, in addition to taking the average value of a thousand encryption completion times, larger average-values of the encryption completion times of the experimented algorithms reveal the attributes of complexity of the algorithms in comparison. The concept of percentages therefore, further details the measure of this difference. The analysis of results section in chapter 6 presents the summary of results for experiments it this work where percentages are utilized to express comparison.

Based on the computation of average values of encryption completion times, variance measures how far apart, or the numerical difference of an instance of encryption completion time from the average value. Defined as $\sigma^2 = \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}$, these values are utilized in the estimation and comparison of the complexity of distinct security algorithms as per the time it takes to complete the encryption of a fixed-sample message block. Also, an observation of the growing complexity of the algorithms considered, as the key lengths increase is observed using the the computation of co-variance values. While the variance measures the numerical value of each instance of encryption completion time from the average value for a specific variant of the algorithm, the covariance which by definition: measures the joint variability of two random variables is utilized to observe the trend of growing complexity in terms of increases in the completion times of distinct algorithms, or more than one variant of the same algorithm, with key lengths as the major factor of difference.

3.3 Cryptanalytic Overview of the Consequence of Complexity Reduction

In this chapter, a detailed cryptanalytic analysis of an algorithm is presented. The analysis seeks to expose various perspectives of the security of an algorithm from a cryptanalytic stand point, thereby laying the foundation for any reasonable modification that can be done on the algorithm. A detailed mathematical justification used as a ground for secure round reduction is also vigorously pursued in this chapter. The motivation for this investigation is that the core security structure of an algorithm should not be compromised to force a reduction in complexity to suit the constrained IoT devices narrative. In other words, there should be a reasonable security trade-off for modifying the algorithm, while the core structure of the algorithm is preserved. This is done by a detailed mathematical analysis of the underlying algebraic properties of the AES algorithm, through the use of relevant theorems and proves, following which a proposed reduced round version of the algorithm is proposed in line with the narrative of the constrained IoT devices. A framework for the implementation of the proposed algorithm is also presented, detailed as follows:

3.3.1 Cryptanalytic Overview and Analysis of Reducing the complexity of the AES

While a formal analysis of security protocols is on its own, an whole area of fertile research [137], according to [138], the security of an encryption scheme is usually measured through the application of different types of cryptanalysis methods. An analysis of the security implication of reducing the complexity of a classical algorithm: such as rounds of the Advanced Encryption Standard (AES) is thus, here presented with respect to Fig. 3.1. This includes an analysis of the families of attacks that a device is potentially vulnerable to.

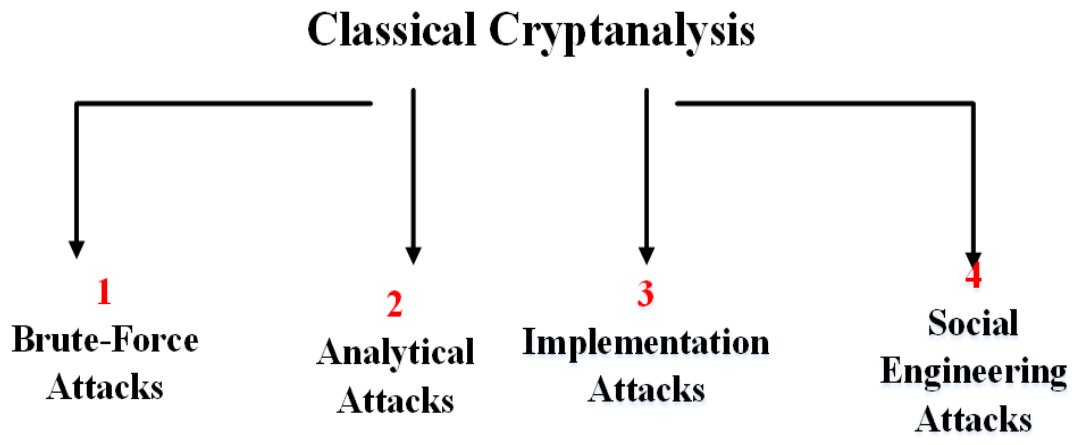

Figure 3.1: Families of Cryptanalytic Attacks

The security of the AES algorithm in terms of key length is strictly with respect to its resistance against breakability by brute-force/exhaustive key search. A common cryptanalytic attack technique on symmetric cryptographic ciphers is the exhaustive key search attack. Based on the reviewed properties of the algorithm as detailed in section 2.5.4, the notion of breakability of the algorithm is analyzed in the context of the size of the key-space. This is usually expressed in bits, where n-bits security means that an attacker would have to perform 2^n operations to break it. In the standard AES-128 algorithm, this is equals 2^{128} operations, for which finding the computing resources to achieve is challenging. In this context of exhaustive key-search attack, the reduction of the number of rounds while retaining the key length of 128bits means that the security of the algorithm is preserved. However, as shown in Fig. 3.1, an analysis of other types of attacks will suffice to detail the implication of round reduction of the classical algorithm. It is crucial to first identify that the emphasis on the security of the AES and other symmetric

algorithms such as DES and 3DES is much with respect to the resistance of the algorithms against attack types 1 and 2 as shown in Fig. 3.1 namely; the brute-force attacks and analytical attacks respectively. It is also note-worthy that the algebraic structure of the AES design majorly aims to provide resistance against these types of attacks. However, attack types 3 and 4 namely implementation attacks and social engineering attacks are attack categories that try to achieve what cannot not be achieved through brute-force attacks and analytical attacks, by attempting to manipulate the encryption algorithm at the point of hardware implementation, or through social engineering means such as bribing, blackmailing or classical espionage. According to [78], we have to choose strong algorithms and we have to ensure that social engineering and implementation attacks are not practical. In the context of [78] as cited above, Strong largely refers to ensuring design resistance against attack types 1 and 2 namely the brute-force and the analytical attacks respectively. While key length provides security in terms of n-bit security, poor storage of these encryption keys can lead to severe security vulnerabilities, as gaining access to the encryption key basically nullifies all the fortification provided by both the key length and and the robust algebraic constructions that guard against the families of brute-force and analytical attacks. Owing to the distributed nature nature of IoT networks and devices, physical access to device is a lot easier in comparison to conventional on-premises networks as detailed in chapter2.8. Consequently, the distributed nature of IoT networks and devices constitutes a major vulnerability to IoT devices in the perspective of implementation attacks as presented in Fig 3.1: as according to [78], in most internet-based attacks against remote systems, implementation attacks usually require the attacker to have physical access to the device to be able to carry out power analysis.

3.3.2 Consequence of Round Reduction and Trade-off

The multiple rounds in the encryption process of the AES is essentially to provide for as much confusion on the original message (plain text) as possible, such that decryption back to the original message is made as difficult as possible in the case of an interception. The choice of 10, 12 and 14 as the number of rounds for the key-lengths of 128,192 and 256 respectively provide similar repetitions of the encryption that is done in the first round to compound complexity for an intruder. According to [139], since there is no known attack which can break the full AES significantly faster than via exhaustive search, researchers had concentrated on attacks which can break reduced round versions of AES. On one front, such attacks are are deemed important for several reasons, one of which is the hope that it enables the development of new

attack techniques which may become potent with additional improvements. On another front, such notion of complexity reduction can be harnessed and applied in relevant use-cases such as the domain of constrained IoT. While the authors in [139] observed that there are many proposals for using reduced round AES, it is noteworthy that researching such round reduction will need to be closely accompanied by ensuring that the core security structure is preserved, as well as ensuring a secure trade-off for such improvements to avoid implementation attacks as advocated by the authors in [78]. The foremost consequence of reducing the complexity of the AES therefore, remains in the perspective of implementation attacks as presented in the cryptanalytic overview in section 3.3.1, while the core algebraic structure which guarantees the security of the algorithm with respect to attack types 1 and 2 namely brute-force and analytical attacks respectively, is shown to be preserved as detailed in section 3.4. As reviewed in chapter2.7, the modification of the standard AES algorithm in the work of [100] was hinged on the contents of the S-Box, which is a 16x16 matrix used for manipulating the STATE (Initial round/Key whitening) results, which was followed by the elimination of rounds five to nine. This notions a trade-off between one security logic and another and with respect to power consumption, whereby the S-Box was substituted by a dynamic box which logically yields a stronger security logic and serves as meaningful compensation for round reduction, which then serves as a bases reducing the number of rounds from 10 to 5. According to [100], contents of the S-Box in AES are fixed, and this can greatly impact the security of the cipher since the only nonlinear component of this block ciphering technique is the manipulation on the S-Box. To improve the cipher's security in line with this assertion, an encryption key that generates the corresponding dynamic box (D-Box) to substitute for the primary substitution box (S-Box) is derived. As a result, a logical trade-off which introduces randomness in the byte substitution stage of the cipher is achieved. The immediate advantage of such trade-off becomes the widening of the attack surface of an arbitrary eavesdropper as it translates to more uncertainty (for the attacker) being introduced in the order of byte substitution. In addition to hardening the security of the algorithm in this wise, the reduction of rounds translates into reduction of complexity of the algorithm which is beneficial to the narrative of constrained IoT devices as detailed in chapter 2.7. Fig. 6.1 shows a demonstration of how lower number of encryption rounds translate to reduction in complexity, which is measured by encryption completion time of a single block of plain text. Implementation attacks mostly aim at what could not be achieved through brute-force and analytical attacks, by attempting to manipulate the encryption algorithm, enabled by access to hardware. According

to [78], side-channel analysis can be used to obtain a secret key, for instance, by measuring the electrical power consumption of a processor through the application of signal processing techniques. Obviously, the various key-lengths of the AES algorithm make typical brute-force attacks infeasible and according to [77], if the cryptanalytic attack on a cryptosystem is infeasible, then the cryptosystem is computationally secure and computationally unbreakable. Further to the cryptanalytic overview and analysis of complexity reduction presented in section 3.3, analytical attacks on the cipher is made difficult through the Galois Fields' complex polynomial operations. As the distributed nature of IoT networks results in increased vulnerability to the end devices via common access to the device through which implementation attacks can potentially become possible, a worthy trade-off of security logic which would harden security with respect to implementation attacks would suffice in ensuring the security of the cipher in terms of brute-force, analytical and implementation attacks as shown in Fig. 3.1, in line with the overall analysis of complexity reduction presented in Fig. 3.1 and as detailed in section 3.3.1.

According to the release document of the "Microchip ATECC608A-MAHTN-T hardware crypto for LoRaWAN" [42], the "Microchip ATECC608A-MAHTN-T" is a pre-provisioned secure element also developed by ARM and the Mbed OS stack can use the secure element to automatically offload all cryptographic operations, so your keys will never be visible nor accessible, even when your device might be compromised. Hence, a proposed round reduction which appropriately justifies the preservation of the mathematical structures of the algorithm against brute-force and analytical attacks as discussed above may be suitably augmented by hardening security of the associated IoT device in the perspective of implementation attacks herein discussed, by leveraging a secure element such as the ATECC608A-MAHNT-T which has the highly desired feature to securely offload keys into a tamper-proof hardware. In addition to hardening the security of the associated IoT device in the perspective of implementation attacks as explained, this could then, serve as a worthy trade-off to complexity reduction via the reduction of rounds of the cipher, thereby sufficiently ensuring security covering: brute-force, analytical and implementation attacks, leaving social engineering attacks as per the families of cryptanalytic attacks as shown in Fig. 3.1. The bit social engineering attacks with respect to this analysis can be largely managed through the appropriate combination of regional and organizational Information Governance (IG) standards and IT policies, as security is never perfect and a continuous process. Section 3.4 details the mathematical justification for complexity reduction sequel to the analysis

in the perspective of analytical attacks. The justification sets to show that the core algebraic properties which guarantee the resilience of the cipher against analytical attacks are preserved, given the operations of the reduction of complexity via round reduction and the use of a secure element as a trade-off.

3.4 Justifying Round Reduction and Security Trade-off

In [100], a perspective of complexity reduction of the standard AES algorithm in line with the narrative of low power IoT devices was presented. While no justification for the choice of round reduction to 5 is given as per the work in [100], this section presents an analytical response to an indispensable question of the consequence of complexity reduction via the reducing the number of rounds, using a cursory examination of the core algebraic properties of the algorithm as a primary objective. Therefore, the following presents a perspective of secure round reduction of the Standard AES algorithm in line complexity reduction to suit the narrative of constrained IoT devices. Accordingly, the least complex variant of the standard AES algorithm which is the AES-128 is considered for this exercise. Starting with the brute-force attack surface and followed by the analytical attacks surface, the mathematical details of this justification using the properties of the algebraic structures defined in section 3.2.1 is presented:

The feature of the AES algorithm that guards against brute-force attack is the key length. Round reduction does not impact on the key length of the cipher and since n-bit security is defined by 2^n; where n = number of bits then

$$\Rightarrow 2^{128} = 2^{128}$$

Thus, we have that the cipher's properties which guarantee defense against brute-force attacks is preserved. Next, we want show that the Galois Fields properties of the cipher that guards against analytical attacks are preserved. It suffices to show that the algebraic operations of the cipher which render elements of the S-box hold, and thus, unaffected by round reduction. Let n and m be some arbitrary bytes of an AES message block. As n and m are bytes of an AES message, it implies that n and m are elements of the Galois Field $F = GF(2^8) = GF(P^n)$. Since the elements of a Galois Field F are non-numerical: precisely polynomials with the element

representation:

$$A(x) = x^{m-1} + ... + a_1x + a_0 \; for \; any \; (A(x) \in F = GF(p^n), (n = n(x) = a_{n-1}x^{n-1} + ... + a_1x + a_0. \tag{3.1}$$

It is pivotal to show that complexity reduction via round reduction of the AES standard algorithm does not impact the algebraic structure of the algorithm, which is essentially a bases for the security of the algorithm with respect to analytical attacks. This suffices to show that the algebraic structure which render elements of the look up tables is unaffected by round reduction. According to [78], the look up tables have special mathematical properties, which are itemized as the following:

1. The elements of the Field should form an additive group with the group operation "+" and zero (0) as the identity element.

2. Every element of the field except zero should form a multiplicative group with the multiplicative operator denoted by "x" and one (1) as the identity element.

3. When the two group operations are mixed, the distributive law should hold such that for n, m and z in $F, z(n + m) = z(x)n(x) + z(x)m(x)$.

In addition to the above properties of a field, since a field contains the properties of a group and a ring, the generality of group and ring axioms as defined above should be satisfied with respect to the operations of addition and multiplication, as well as the corresponding inversions in the field. For the additive operator, given two arbitrary polynomials n and m in F, the sum $n(x) + m(x)$ is defined by

$$z(x) = n(x) + m(x) = \sum_{i=1}^{m-1} z_i x^i \text{ where } z_i = n_i + m_i \; mod 2 \tag{3.2}$$

Elements inversion with this operation naturally gives rise to subtraction defined by

$$z(x) = n(x) - m(x) = \sum_{i=1}^{m-1} z_i x^i \text{ where } z_i = n_i - m_i \; mod 2 \tag{3.3}$$

To show the field properties of element combination and inversion with respect to the multi-

plicative operator, we consider the following theorems:

Theorem 1 (Existence of a unique polynomial)**.** *Let F be a field and let $n(x)$, $m(x)$ be polynomial elements in F, where $m(x) \neq 0$, then there exist unique polynomials $\alpha(x)$ and $r(x)$ in F such that;*

$$n(x) = m(x)\alpha(x) + r(x) \tag{3.4}$$

where either $r = 0$ or $deg(r(x)) < deg(m(x))$

proof: Firstly, we show the existence of the polynomials $\alpha(x)$ and $r(x)$ satisfying 3.4 above, together with the requirement that either $r = 0$ or $deg(r(x)) < deg(m(x))$ and then we establish the uniqueness of these polynomials. If $deg(m(x)) = 0$, then $(x) = c$ is a non-zero constant in F and by the inverse property of F, we have that:

$$n(x) = cc^{-1}n(x) \qquad \textit{by the identity in } F$$

.

$$c(c^{-1}n(x)) + 0 \qquad \textit{by associativity in } F$$

such that $m(x) = c^{-1}n(x)$ and $r(x) = 0$, satisfying 3.4. If on the other hand the degree of $m(x) \geq 1$, let Z denotes the set of all polynomials of the form $n(x) - m(x)\alpha(x)$, where $m(x) \in F$, then the degrees of all polynomials in Z form a set of non-negative integers and by the principle of the well ordering of natural numbers, there exists a smallest element of Z, say s. Let

$$r(x) = n(x) - m(x)\alpha(x) \tag{3.5}$$

be such a polynomial of degree s in Z, then:

$$deg(r(x)) < deg(m(x)) \tag{3.6}$$

for if we suppose for contradiction that $deg(r(x)) > deg(m(x))$, let $deg(m(x)) = m$ and let

$$m(x) = a_m x^m + ... + a_0, \qquad a_m \neq 0$$

$$r(x) = b_s x^s + ... + b_0, \qquad b_s \neq 0$$

As F is a field, we have that a_m has an inverse in F and so, let $r_1(x) = r(x) - b_s a^{-1} x^{s-m} m(x) \in Z$

$= b_s x^s + ... + b_0 - ([b_s a^{-1} x^{s-m} m(x)]$

$= b_s x^s + ... + b_0 - b_s a^{-1} x^{s-m} (a_m x^m + ... + a_0)$

$= b_s x^s + ... + b_0 - [b_s a^{-1} x^{s-m} \times amx^m + ... + b_s a^{-1} x^{s-m} \times ama^0]$

$= b_s x^s + ... + b_0 - [b_s x^s + b_s a^{-1} x^{s-m} a_0$. by the inverse property of F. As

$b_s x^s - b_s x^s = 0$, $deg(r_1(x) = r(x) - b_s a^{-1} x^{s-m} m(x) < deg(r(x))$. This contradicts 3.5 which requires that $r(x)$ is of the least degree in Z and thus, validating 3.6 which validates 3.4 accordingly. To show that $\alpha(x)$ and $r(x)$ are unique in the field F, let $\alpha_1(x)$ *and* $r_1(x)$ be another pair of polynomials in F satisfying 3.4, then we have that:

$$n(x) = m(x)\alpha_1(x) + r_1(x) \tag{3.7}$$

where either $r = 0$ or $deg(r_1(x)) < deg(m(x))$. By the inverse and identity property of the additive operator in F, we have that:

$0 = n(x) - n(x)$

$\Rightarrow 3.4 - 3.7$

$= m(x)\alpha(x) + r(x) - (m(x)\alpha_1(x) + r_1(x))$

$$\Rightarrow (m(x)[\alpha(x) - \alpha_1(x)] = r_1 x \tag{3.8}$$

If $\alpha(x) - \alpha_1(x) \neq 0$, then $deg(m(x)[\alpha(x)\alpha_1(x)]) < deg(m(x))$ and thus making 3.8 a contradiction of the requirements of (4), which require that $deg(r_1(x)) < deg(m(x))$. Hence,

$$\alpha(x) - \alpha_1(x) = 0 \tag{3.9}$$

Substituting 3.9 into 3.8 shows accordingly that

$$r_1 x - r(x) = 0 \tag{3.10}$$

From 3.9 and 3.10, we have that $\alpha(x) = \alpha_1(x)$, $r_1 x = r(x)$ and thus, showing the uniqueness

of $\alpha(x)$ and $r(x)$ in F and hence, 3.4. □

Theorem 2 (Monic greatest common divisor)**.** *Let F be a field, and $n(x)$ and $m(x)$ be non-zero polynomials in F, then $n(x)$ and $m(x)$ have a unique monic greatest common divisor, say $d(x)$ in F such that:*

$$d(x) = \alpha(x)n(x) + \beta(x)m(x) \tag{3.11}$$

proof: Let $d(x) = d_0 + d_1x + +d_nx^n$, $d_n \neq 0$ be a polynomial greatest common divisor of $n(x)$ and $m(x)$ in F. As a greatest common divisor is not unique, if $d_n \neq 1$, by the inverse property of F, there exists $\mathrm{d}_n^{-1} \in F$ such that $d_n^{-1}d(x)$ is also a greatest common divisor. Let $d_1(x)$ be another monic greatest common divisor of $n(x)$ and $m(x)$ in F, then $d(x)|d_1(x)$ and $d_1(x)|d(x)$.

$\Rightarrow \exists y(x), z(x) \in F$: $\mathrm{d}_1(x) = y(x)d(x)$ and $d(x) = \mathrm{z(x)d}_1(x)$

$\Rightarrow deg(d_1(x)) \geq deg(d(x))$ and $deg(d(x)) \geq deg(d_1(x))$

$\Rightarrow \deg(d_1(x)) = \deg((d(x))$

$\Rightarrow \deg(y(x)) = \deg(z(x))$

This implies that $y(x)$ and $z(x)$ are constants in F and since $d(x)$ and $d_1(x)$ are monic polynomials, then $y(x)$=$z(x) = 1$, and $d_1(x) = d(x)$. Therefore, the monic greatest common divisor $d(x)$, of $n(x)$ and $m(x)$ in F is unique and hence, Theorem 2 □

Theorem 3 (Existence of an irreducible polynomial)**.** *Let $F = GF(2^8) = GF(P^n)$ be a field, then there exists an irreducible polynomial of degree n over Z_p.*

proof: As F is a field, let g be a generator element in F. Also, let $p(x)$ be a minimal polynomial of degree n in F. By the property of the generator element g, we have that every element in F occurs as a power of g and as such, the minimal polynomial $p(x) = p_g(x)$. Let

$$deg(p_g(x)) = m \tag{3.12}$$

By Theorem 1, there exists polynomials say $n(x))$, $r(x)) \in F$ such that for any constant c, $x^c = n_c(x)p_g(x) + r_c(x)$ with either $r_c(x) = 0$ or $deg(n_c(x)) < deg(p_g(x)) = m$. Substituting g, $g^c = n_c(g)p_g(g) + r_c(g)$ and as $p_g(x)$ is a minimal polynomial, we have $g^c = \mathrm{n}_c(g).0 + r_c(g))$

$\Rightarrow g^c = r_c(g)$. But as g is a generator element in $F = GF(p^n)$, we have that $g^{p-1} = 1$ and as

such, the highest number of the distinct powers of g is $p^n - 1$ and by 3.12, we have that the highest number of the distinct powers of the generator $g = p^m - 1$, which implies that $n \leq m$. Also, as $p_g(x)$ is a minimal polynomial in F, for any two polynomials say $j(x)$ and $z(x)$ in F, $j(g) = z(g)$.

$\Rightarrow j(g)$ - $z(g) = 0$

$\Rightarrow j(x)$ - $z(x) = 0$

Since $deg(j(x) - z(x)) < deg(p_g(x))$, then the number of elements in $F = GF(p^n)$ is at least as big as the number of polynomials in F with degree less than m. This implies that $m \leq n$. Whereas, $m \leq n$ and $n \leq m$ implies that $m = n$. Therefore, $deg(p_g(x)) = n$ and since $p_g(x)$ is a minimal polynomial, $p_g(x)$ is then an irreducible polynomial in $F = GF(p^n)$. □

3.4.1 Significance of theorems

A summary of the relevance of the above theorems in this justification is here presented as an extract void of the mathematical rigors of the proves presented above:

Theorem 1 provides the algorithm for the division of polynomials. The emphasis of this theorem is that given the Galois Field $F = GF(p^n)$, such as the AES field, taking any arbitrary elements of this field; such as bytes of an AES message block, it is always possible to find polynomials (other elements) in the field with which division is made possible. This is a necessary step towards ensuring that inversion is possible in the field, even though more than this is required. Theorem 2 furnishes what more is required to guarantee the existence of a greatest common divisor between any two arbitrary elements of the Field, such as bytes of an AES message block. Akin to a feature of integers which is extended to polynomial operations, this is particularly useful in simplifying such complex polynomial operations involving the multiplicative operator. More so, the requirement of theorem 3 which is critical to accomplishing polynomial inversion and the construction of the field itself requires that the irreducible polynomial be a monic polynomial. According to [78], a monic polynomial is a polynomial whose highest coefficient is one. Theorem 3 establishes the existence of an irreducible polynomial in the field, given the Galois Field $F = GF(p^n)$ which elements are polynomials. Without the existence of an irreducible polynomial, inversion within the field will not be possible, as the product of two polynomials may lead to a polynomial of a higher degree which will not be an element of the

field. Irreducible polynomials however, enable the reduction of such products to valid polynomials elements of the given field. More so, the irreducible polynomial is fundamental to the construction of the field itself and by extension, rendering the elements of the algorithm lookup tables as in $x^8 + x^4 + x^3 + x + 1$, which is a part of the AES specification according to [78]. Also, according to [78], The advantage of using inversion in $GF(2^8)$ as the core function of the Byte Substitution layer is that it provides a high degree of non-linearity, which in turn provides optimum protection against some of the strongest known analytical attacks. With theorems 1, 2 and 3, the required operations of element addition and multiplication, together with their corresponding inversions is guaranteed in the field $F = GF(p^n)$, for any arbitrary elements n,m and $z \in F = GF(p^n)$. Accordingly, the underlying structure and operations which yield the elements of the AES look up tables hold. Consequently, the security structure of the algorithm against analytical attacks hold.

As F is a field, we have that for the arbitrary elements
$n(x)$, $m(x)$ and $z(x) \in F = GF(P^n)$, $n(x) + m(x) = (a_{n-1}x^{n-1} + ... + a_1x + a_0) + (a_{m-1}x^{m-1} + ... + a_1x + a_0)$, $z(x) = n(x) + m(x) = \sum_{i=1}^{m-1} z_i x^i$, where $z_i = n_i + m_i mod 2 \in F = GF(2^8))$ $=GF(P^n)$.
Also, by the application of Theorem 1 and Theorem 3 above we have that
$n(x) \times m(x) = [(a_{n-1}x^{n-1} + ... + a_1x + a_0) \times (a_{m-1}x^{m-1} + ... + a_1x + a_0)] \; mod(x^8 + x^4 + x^3 + x + 1) = z(x) \in F = GF(2^8) = GF(P^n)$ and so, closure; with respect to the additive and multiplicative operators hold in F. Associativity also holds in F by a combination of the same theorems with the arbitrary elements on:
$(a_{n-1}x^{n-1} + ... + a_1x + a_0) + [(a_{m-1}x^{m-1} + ... + a_1x + a_0) + (a_{z-1}x^{z-1} + ... + a_1x + a_0)] = [(a_{n-1}x^{n-1} + ... + a_1x + a_0) + (a_{m-1}x^{m-1} + ... + a_1x + a_0)] + (a_{z-1}x^{z-1} + ... + a_1x + a_0)$ and
$(a_{n-1}x^{n-1} + ... + a_1x + a_0) \times [(a_{m-1}x^{m-1} + ... + a_1x + a_0) \times (a_{z-1}x^{z-1} + ... + a_1x + a_0)] = [(a_{n-1}x^{n-1} + ... + a_1x + a_0) \times (a_{m-1}x^{m-1} + ... + a_1x + a_0)] \times (a_{z-1}x^{z-1} + ... + a_1x + a_0)$ for the additive and multiplicative operators respectively. Similarly, $n(x) + (-(n(x)) = (a_{n-1}x^{n-1} + ... + a_1x + a_0) + (-(a_{n-1}x^{n-1} + ... + a_1x + a_0)) = 0$,
$n(x) \times (n(x)^{-1}) = [(a_{n-1}x^{n-1} + ... + a_1x + a_0) \times (n(x)^{-1})] \; mod(x^8 + x^4 + x^3 + x + 1) = 1$
and the identity elements and inverses exist for the additive and multiplicative operators in F accordingly. Distributivity also holds in F accordingly and so, satisfying the properties of an additive group and a multiplicative group in F. By Theorem 2, the existence of a greatest

common divisor between the arbitrary bytes of the AES message blocks: $n(x)$ and $m(x)$ is guaranteed. Division, which implies element inversion with respect to the multiplicative operator is also guaranteed in $GF(2^8)$ by Theorem 1 and by Theorem 3, as the existence of an irreducible polynomial over the message field $GF(2^8)$ is guaranteed. Moreover, the irreducible polynomial: $x^8 + x^4 + x^3 + x + 1$ is a part of the AES specification and so, the operations of addition and multiplication, together with their corresponding inversions are guaranteed given the arbitrary AES message bytes n, m and $z \in F = GF(P^n)$. As such, the underlying structures of the cipher that renders the entries of the AES S-boxes hold and thus, the core security attributes of algorithm in the context of analytical attacks is preserved with respect to round reduction. The major consequence of the complexity reduction exercise based on this analysis is therefore, with respect to implementation attacks (as shown in Fig. 3.1); considering the distributed nature of IoT devices and how prone the devices can be in terms of physical security, which enables access to IoT devices leading to an increased potential in the execution of implementation attacks as described in section 3.3. The formulation of an efficient security algorithm for power constrained IoT devices, with the trade-off for the round reduction as the introduction of a secure element: which doubles for authentication of the associated IoT device and to guard against implementation attacks as shown in Fig. 3.1 is then experimented, thereby sufficiently ensuring the IoT device security covering: brute-force, analytical and implementation attacks shown in Fig. 3.1 and leaving social engineering attacks which can be managed by policies.

3.5 Chapter Summary

In this chapter, an analysis of algorithm complexities in provisioning constrained IoT devices is detailed. Starting with an introduction of the major topics covered in the chapter in section 3.1, a mathematical background to most of the mathematical analysis in the thesis is given in 3.2, including the basic algebraic structures of Galois fields, averages, percentages and variance. A cryptanalytic overview of the consequence of reducing the complexity of classical algorithms followed in 3.3, detailing a cryptanalytic overview and analysis of the AES as an example classical algorithm and also as the most widely-used algorithm in the IoT landscape according to the work in [25], together with the trade-off for round reduction. A justification of the round reduction and trade-off is detailed in section 3.4, with the significance of relevant theorems used, presented in 3.4.1 and a conclusion of the chapter in 3.5.

Chapter 4

Efficient Security Algorithm for Power Constrained IoT Devices

4.1 Introduction

4.1.1 Light Weight Cryptography

According to [102], Lightweight cryptography is generally defined as the cryptography for resource constrained devices, for which Radio Frequency Identification (RFID) tags are mentioned as examples. Consequent upon the constraints on low power devices in terms of area, processing capabilities, memory and scarce power resources, Light weight cryptography emerged in efforts to ensure the security of these devices in the digital communication space. As rightly put by [28], Lightweight Cryptography (LWC) or protocols are tailored for implementations in constrained environments including RFID tags, sensors, contact less smart cards, health care devices and so on. The development of lightweight cryptographic protocols therefore, is chiefly anchored on the need to develop cheaper security schemes that are compatible with the constrained nature of these devices and without compromise to security. In [104], the performance analysis of two lightweight ciphers: CLEFIA and PRESENT was presented with respect to security strengths, throughput and resource utilization, which had the latter outperforming the former in terms of memory usage, security, and the former outperforms the latter in terms of throughput. PRESENT is a

lightweight algorithm with a Substitution Permutation Network (SPN) structure, and utilizes thirty one rounds of: XOR RoundKey, S-box layer and P-layer, with a final (32nd) round which XORs the STATE produced from the first thirty one rounds and the round key. It carries out message encryption in sixty four (64) bits blocks and and supports key lengths of sixty four (64)bits and one hundred and twenty eight (128)bits. The efficiency of a lightweight cipher (CLEFIA) in comparison to other conventional ciphers such as the AES, Camelia and Seed is detailed in [28], wherein the efficiency of lightweight ciphers defined as a ratio of throughput and gate size is presented with respect to energy consumption.

4.1.2 Challenges in Lightweight Cryptography

According to [35], the problem of power consumption in microprocessors and Digital Signal Processing (DSP) devices has continued to emerge, requiring solutions in order to maximize the battery life of power constrained devices. [104] reveals that the multiple lightweight methods developed so far have their respective merits and demerits, and determining a right choice to secure data, depends on various factors including the nature of the application or usage. [45] noted that the biggest challenge in the design of these lightweight algorithms is on how to cope with trade-offs between cost: in terms of computing resources required to run the algorithms, performance, and the security of the algorithms. Even with the plethora of lightweight ciphers aimed at addressing the problems of resource constrains in devices with limited capabilities, the lightweight algorithms are still limited in terms of design and applicability, in addressing the many security issues in the IoT landscape. [45] noted that the biggest challenge in the design of these lightweight algorithms is on how to cope with trade-offs between cost: in terms of computing resources required to run the algorithms, performance, and the security of the algorithms. To this end, the work in [105] leverages the provision of a tamper-proof secure element as a one of such desirable trade-offs aimed at providing an efficient algorithm of reduced complexity that is compatible with resource constrained devices, without compromising the security.

4.2 The Efficient Algorithm for Power Constrained IoT Devices

The efficient security algorithm executes a two-step process for an associated power constrained IoT device as follows:

1. Secure authentication using the ATECC608A

2. Message encryption using the reduced round cipher

Leveraging the tamper-proof secure element for secure authentication, the reduced round algorithm executes the round function in four iterations using a total of 80bytes scheduled key as against the 176bytes of the standard AES, for every block of the plaintext, following the process of key-whitening, initialization and the execution of the round function. The pseudo code of the process flow of the proposed algorithm is presented in Table 4.1.

Table 4.1: The algorithm Flow

	The Efficient Security Algorithm for Constrained IoT Devices
Step1	
*	Initializing the IoT device and the ATECC608 secure element
*	Invoking authentication using ATECC608 and generating tamper-proof security keys
Step2	
1.	Input: message, key
2.	Nbr selection: 2 or 4 and initialization of the Nbr counter
3.	Expand key to length: (block size) *Nbr + block size
4.	STATE = message XORed with Key (Key whitening)
5.	Invoke the round function: While counter is less than the selected Nbr: i. STATE = SubByte(STATE) ii. STATE = ShiftRows(STATE) iii. If counter < selected nbr: 1. STATE = MixColumn(STATE) iv. addRoundKey(STATE,NextRoundkey)
6.	Output STATE as resulting Ciphertext

4.3 Implementation Pseudocode and Comparison of the AES and the Low Cost Algorithm

4.3.1 The Encryption Process

This section presents a detailed process of encryption and decryption of a sample plain text. This process aims to detail the encryption procedure of a single block of an AES message while detailing every step of the encryption as a blue print for the implementation of the reduced round algorithm. The piece of plain text considered for this exercise is **my phd research!**. This choice is hinged on the fact that a single character makes one byte of message and so, the number of text characters in the sample plain text including all the blank spaces sum to sixteen (16) and thus, making the block size for an AES message block (16bytes). In the general cases of encryption of plain text longer than the blocksize of 16bytes, the entire plaintext is still broken down into blocks of 16bytes by padding. Message padding in this sense is used to break down larger messages sizes beyond 16bytes, as well as for rounding up message sizes that are less than 16bytes. The sole aim of message padding in this process is to ensure that every block of plain text to be encrypted sufficiently satisfies the requirement of message size of sixteen bytes.

4.3.2 Plain Text Transformation and Key Whitening

The first step of this encryption process is the transformation of the every character of the plaintext to its corresponding ASCII code values and subsequently to its binary equivalence in preparation for invoking the encryption process. The choice choice of plaintext is made specifically to construct a 16byte for illustrating the internals of the AES process. Accordingly, a simple key of 16 numbers ranging from zero (0) to fifteen (15) is considered for this exercise. The process requires a simple plaintext. The table below shows a the transformation of the sample plaintext and key to its corresponding ASCII code values and then to binary. The key whitening process executes an Exclusive OR logical operation (XOR) on the plaintext and the key to produce the STATE = (plaintext) XOR (key), which is then passed on to the round function for executing the AES encryption process. The encryption processes involving: Key Whitening, SubBytes, ShiftRows, MixColumns and AddRoundKey is diagrammatically illustrated in Fig. 2.1.

The key whitening process produces the STATE which is passed to the rounds function as shown, for the execution of bytes substitution, otherwise known as the SubBytes stage of the

Table 4.2: Plaintext Transformation and Key Whitening

Plaintext	ASCII Code	Plaintext Binary	Key	Key Binary	STATE
m	109	1101101	0	00000000	0x6d
y	121	1111001	1	00000001	0x78
	32	100000	2	00000010	0x22
p	112	1110000	3	00000011	0x73
h	104	1101000	4	00000100	0x6c
d	100	1100100	5	00000101	0x61
	32	100000	6	00000110	0x26
r	114	1110010	7	00001111	0x7d
e	101	1100101	8	00001000	0x6d
s	115	1110011	9	00001001	0x7a
e	101	1100101	10	00001010	0x6f
a	97	1100001	11	00001011	0x6a
r	114	1110010	12	00001100	0x7e
c	99	1100011	13	00001101	0x65
h	104	1101000	14	00001110	0x66
!	33	0100001	15	00001111	0x2e

cipher.

4.3.2.1 SubBytes:

The SubBytes stage is the first of the four major stages of the round function execution. Following the whitenning process of the original message and the associated round key, the STATE produced by the process explained in section 4.3.2 is used to further transform the encryption process by executing a direct substitution of the values of the STATE with its corresponding values in the Rijndael S-box table. According to [97], the S-box maps an eight bit input to an eight bit output, where both the input and output are interpreted as polynomial elements over $GF(2)$. Each element in $GF(2)$ is effectively a polynomial modulo an irreducible polynomial: $x^8+x^4+x^3+x+1$, which is a part of the AES specification [79] .

S-box =

[0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76, 0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0, 0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15, 0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75, 0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84, 0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,

0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8, 0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2, 0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73, 0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb, 0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79, 0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08, 0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a, 0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e, 0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf, 0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16][140]

For STATE = [109, 120, 34, 115,108, 97, 38, 125, 109, 122, 111, 106, 126, 101, 102,46] =[0x6d, 0x78, 0x22,0x73, 0x6c,0x61, 0x26, 0x7d, 0x6d, 0x7a, 0x6f,0x6a, 0x7e, 0x65, 0x66, 0x2e], the byte substitution executes the replacement of every byte in the block to its corresponding value in the S-box table. Consequently, a new value for the state is produced by applying the SubByte on the current STATE and so, $STATE = SubBytes(STATE)$

4.3.2.2 ShiftRows

The ShiftRows operation is done on the result of the STATE produced after the SubBytes operation. Visualizing the result as a matrix, the ShiftRow is an element-wise rotational operation with elements of the first Row being shifted by zero to the left (or not being shifted at all), the second row being shifted by one element, the third element shifted by two elements until the last row is shifted.

$$Grid - After - Shift = \begin{bmatrix} m & h & e & r \\ y & d & s & c \\ - & - & e & h \\ p & r & a & ! \end{bmatrix}$$

$$Grid - After - Shift = \begin{bmatrix} m & h & e & r \\ d & s & c & y \\ e & h & - & - \\ ! & p & r & a \end{bmatrix}$$

As the STATE is a 16byte message block, executing the shiftRow operation in the bit-wise operation results in:

tmp13 = state[13], state[13] = state[1], state[1] = state[5], state[5] = state[9], state[9] = tmp13, tmp15 = state[15], state[15] = state[11], state[11] = state[7], state[7] = state[3], state[3] = tmp15, tmp2 = state[2], state[2] = state[10], state[10] = tmp2, tmp6 = state[6], state[6] = state[14], state[14] = tmp6

The ShiftRow operation returns a new value of the STATE which would be passed on in the round function for subsequent operations of Mix columns and Add Round Key.

4.3.2.3 Mix Colums

The Mix-Column operation is the third of the round function and is executed on the STATE, having undergone the shiftRow operation. The operations of the MixColumns stage of the AES are rooted in the properties of the Galois Fields which are shown with the proves of relevant theorem in section 3.4. Again, visualizing the STATE as a 4x4 grid, the MixColumn operation multiplies the arbitrary message block by a defined AES matrix.

$$\begin{pmatrix} m & h & e & r \\ y & d & s & c \\ - & - & e & h \\ p & r & a & ! \end{pmatrix} \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 1 & 1 & 3 & 2 \end{pmatrix}$$

According to [140], the resulting values of the byte-wise multiplication of the STATE bytes by the defined AES matrix is contained in the multiplication by two (Mul2) and multiplication by three (Mul3) Rijndael look up tables. Further, the element wise multiplication of the message byte and the defined AES matrix is guided by the properties of the polynomial operations in the Galois Field where the product of two such polynomials is taken modulo $x^8 + x^4 + x^3 + x + 1$, where $x^8 + x^4 + x^3 + x + 1$ is a part of the AES specification [78]

MUL2 =

[0x00, 0x02, 0x04, 0x06, 0x08, 0x0a, 0x0c, 0x0e, 0x10, 0x12, 0x14, 0x16, 0x18, 0x1a, 0x1c, 0x1e, 0x20, 0x22, 0x24, 0x26, 0x28, 0x2a, 0x2c, 0x2e, 0x30, 0x32, 0x34, 0x36, 0x38, 0x3a, 0x3c, 0x3e, 0x40, 0x42, 0x44, 0x46, 0x48, 0x4a, 0x4c, 0x4e, 0x50, 0x52, 0x54, 0x56, 0x58, 0x5a, 0x5c, 0x5e, 0x60, 0x62, 0x64, 0x66, 0x68, 0x6a, 0x6c, 0x6e, 0x70, 0x72, 0x74, 0x76, 0x78, 0x7a, 0x7c, 0x7e, 0x80, 0x82, 0x84, 0x86, 0x88, 0x8a, 0x8c, 0x8e, 0x90, 0x92, 0x94, 0x96, 0x98, 0x9a, 0x9c, 0x9e, 0xa0, 0xa2, 0xa4, 0xa6, 0xa8, 0xaa, 0xac, 0xae, 0xb0, 0xb2, 0xb4, 0xb6, 0xb8, 0xba, 0xbc, 0xbe, 0xc0, 0xc2, 0xc4, 0xc6, 0xc8, 0xca, 0xcc, 0xce, 0xd0, 0xd2, 0xd4, 0xd6, 0xd8, 0xda, 0xdc, 0xde, 0xe0, 0xe2, 0xe4, 0xe6, 0xe8, 0xea, 0xec, 0xee, 0xf0, 0xf2, 0xf4, 0xf6, 0xf8, 0xfa, 0xfc, 0xfe, 0x1b, 0x19, 0x1f, 0x1d, 0x13, 0x11, 0x17, 0x15, 0x0b, 0x09, 0x0f, 0x0d, 0x03, 0x01, 0x07, 0x05, 0x3b, 0x39, 0x3f, 0x3d, 0x33, 0x31, 0x37, 0x35, 0x2b, 0x29, 0x2f, 0x2d, 0x23, 0x21, 0x27, 0x25, 0x5b, 0x59, 0x5f, 0x5d, 0x53, 0x51, 0x57, 0x55, 0x4b, 0x49, 0x4f, 0x4d, 0x43, 0x41, 0x47, 0x45, 0x7b, 0x79, 0x7f, 0x7d, 0x73, 0x71, 0x77, 0x75, 0x6b, 0x69, 0x6f, 0x6d, 0x63, 0x61, 0x67, 0x65, 0x9b, 0x99, 0x9f, 0x9d, 0x93, 0x91, 0x97, 0x95, 0x8b, 0x89, 0x8f, 0x8d, 0x83, 0x81, 0x87, 0x85, 0xbb, 0xb9, 0xbf, 0xbd, 0xb3, 0xb1, 0xb7, 0xb5, 0xab, 0xa9, 0xaf, 0xad, 0xa3, 0xa1, 0xa7, 0xa5, 0xdb, 0xd9, 0xdf, 0xdd, 0xd3, 0xd1, 0xd7, 0xd5, 0xcb, 0xc9, 0xcf, 0xcd, 0xc3, 0xc1, 0xc7, 0xc5, 0xfb, 0xf9, 0xff, 0xfd, 0xf3, 0xf1, 0xf7, 0xf5, 0xeb, 0xe9, 0xef, 0xed, 0xe3, 0xe1, 0xe7, 0xe5][140]

MUL3 =

[0x00, 0x03, 0x06, 0x05, 0x0c, 0x0f, 0x0a, 0x09, 0x18, 0x1b, 0x1e, 0x1d, 0x14, 0x17, 0x12, 0x11, 0x30, 0x33, 0x36, 0x35, 0x3c, 0x3f, 0x3a, 0x39, 0x28, 0x2b, 0x2e, 0x2d, 0x24, 0x27, 0x22, 0x21, 0x60, 0x63, 0x66, 0x65, 0x6c, 0x6f, 0x6a, 0x69, 0x78, 0x7b, 0x7e, 0x7d, 0x74, 0x77, 0x72, 0x71, 0x50, 0x53, 0x56, 0x55, 0x5c, 0x5f, 0x5a, 0x59, 0x48, 0x4b, 0x4e, 0x4d, 0x44, 0x47, 0x42, 0x41, 0xc0, 0xc3, 0xc6, 0xc5, 0xcc, 0xcf, 0xca, 0xc9, 0xd8, 0xdb, 0xde, 0xdd, 0xd4, 0xd7, 0xd2, 0xd1, 0xf0, 0xf3, 0xf6, 0xf5, 0xfc, 0xff, 0xfa, 0xf9, 0xe8, 0xeb, 0xee, 0xed, 0xe4, 0xe7, 0xe2, 0xe1, 0xa0, 0xa3, 0xa6, 0xa5, 0xac, 0xaf, 0xaa, 0xa9, 0xb8, 0xbb, 0xbe, 0xbd, 0xb4, 0xb7, 0xb2, 0xb1, 0x90, 0x93, 0x96, 0x95, 0x9c, 0x9f, 0x9a, 0x99, 0x88, 0x8b, 0x8e, 0x8d, 0x84, 0x87, 0x82, 0x81, 0x9b, 0x98, 0x9d, 0x9e, 0x97, 0x94, 0x91, 0x92, 0x83, 0x80, 0x85, 0x86, 0x8f, 0x8c, 0x89, 0x8a, 0xab, 0xa8, 0xad, 0xae, 0xa7, 0xa4, 0xa1, 0xa2, 0xb3, 0xb0, 0xb5, 0xb6, 0xbf, 0xbc, 0xb9, 0xba, 0xfb, 0xf8, 0xfd, 0xfe, 0xf7, 0xf4, 0xf1, 0xf2, 0xe3, 0xe0, 0xe5, 0xe6, 0xef, 0xec, 0xe9, 0xea, 0xcb, 0xc8, 0xcd, 0xce, 0xc7, 0xc4, 0xc1, 0xc2, 0xd3, 0xd0, 0xd5, 0xd6, 0xdf, 0xdc, 0xd9, 0xda, 0x5b, 0x58,

0x5d, 0x5e, 0x57, 0x54, 0x51, 0x52, 0x43, 0x40, 0x45, 0x46, 0x4f, 0x4c, 0x49, 0x4a, 0x6b, 0x68, 0x6d, 0x6e, 0x67, 0x64, 0x61, 0x62, 0x73, 0x70, 0x75, 0x76, 0x7f, 0x7c, 0x79, 0x7a, 0x3b, 0x38, 0x3d, 0x3e, 0x37, 0x34, 0x31, 0x32, 0x23, 0x20, 0x25, 0x26, 0x2f, 0x2c, 0x29, 0x2a, 0x0b, 0x08, 0x0d, 0x0e, 0x07, 0x04, 0x01, 0x02, 0x13, 0x10, 0x15, 0x16, 0x1f, 0x1c, 0x19, 0x1a][140]

4.3.2.4 AddRoundKey

During the key expansion, the original key is used to generate round keys that are used in the rounds. This means the key in each round is different, although all generated from the same key during key expansion in the add round key stage, the STATE bytes from the MixColumn stage are added with the bytes of the sub-key for that round, using the arithmetic of binary addition modulo two (2). For any arbitrary number of executions of the round function, the last round of the function is spared the execution of the MixColumns, in which case the AddRoundkey which executes an Exclusive OR (XOR) operation with the round key becomes the state and output cichertext.

4.3.3 The Decryption Process

The decryption process reverses the ciphertext back to a readable plain text using a sequence of steps. Like the enciphering process, the decryption process equally takes in chunks of message blocks of sixteen bytes and for every encryption process explained above, the decryption process reverses it towards recovering the original plaintext from which the ciphertext was obtained. The following steps outline the decryption process of the reduced round algorithm based on the standard AES algorithm.

4.3.3.1 AddRoundKey

The decryption process starts by adding the round key, which is of equal length with the message block in an Exclusive OR operation to produce the STATE to be passed down to the rest of the decryption process. Worthy of note is that the message to be XORed with the round key is ciphertext, as against the key whitenning of the encryption process. A STATE which is a 16byte block is produced as the result of this operation.

4.3.3.2 The InvMixColums

The inverse mix column stage of the decryption process takes the STATE produced from the section 4.3.3.1 as input, on which it executes the InvMixColumns operation on it. In a similar fashion to the mix column process discussed in section 4.3.2.3, the InvMixColum process executes a byte multiplication of the STATE with a unique defined Rijndael matrix to return a new STATE. The matrix multiplication of the InvMixClomns operation is given below with a sample STATE of "ciphertext block". Similar to the enciphering process, the choice of the STATE here is chosen to carefully construct a sixteen bytes (16bytes) message fit for the AES blocksize.

$$\begin{pmatrix} c & e & x & l \\ i & r & t & o \\ p & t & - & c \\ h & e & b & k \end{pmatrix} \begin{pmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{pmatrix}$$

According to [140], the resulting byte values of multiplication with the defined matrix is presented in the MUL9, MUL11, MUL13 and MUL14 tables in 4.3.3.2. where the element wise multiplication of the message byte and the defined AES matrix is guided by the properties of the polynomial operations in the Galois Field and the product of two such polynomials is taken modulo $x^8 + x^4 + x^3 + x + 1$.

MUL9 =

[0x00, 0x09, 0x12, 0x1b, 0x24, 0x2d, 0x36, 0x3f, 0x48, 0x41, 0x5a, 0x53, 0x6c, 0x65, 0x7e, 0x77, 0x90, 0x99, 0x82, 0x8b, 0xb4, 0xbd, 0xa6, 0xaf, 0xd8, 0xd1, 0xca, 0xc3, 0xfc, 0xf5, 0xee, 0xe7, 0x3b, 0x32, 0x29, 0x20, 0x1f, 0x16, 0x0d, 0x04, 0x73, 0x7a, 0x61, 0x68, 0x57, 0x5e, 0x45, 0x4c, 0xab, 0xa2, 0xb9, 0xb0, 0x8f, 0x86, 0x9d, 0x94, 0xe3, 0xea, 0xf1, 0xf8, 0xc7, 0xce, 0xd5, 0xdc, 0x76, 0x7f, 0x64, 0x6d, 0x52, 0x5b, 0x40, 0x49, 0x3e, 0x37, 0x2c, 0x25, 0x1a, 0x13, 0x08, 0x01, 0xe6, 0xef, 0xf4, 0xfd, 0xc2, 0xcb, 0xd0, 0xd9, 0xae, 0xa7, 0xbc, 0xb5, 0x8a, 0x83, 0x98, 0x91, 0x4d, 0x44, 0x5f, 0x56, 0x69, 0x60, 0x7b, 0x72, 0x05, 0x0c, 0x17, 0x1e, 0x21, 0x28, 0x33, 0x3a, 0xdd, 0xd4, 0xcf, 0xc6, 0xf9, 0xf0, 0xeb, 0xe2, 0x95, 0x9c, 0x87, 0x8e, 0xb1, 0xb8, 0xa3, 0xaa, 0xec, 0xe5, 0xfe, 0xf7, 0xc8, 0xc1, 0xda, 0xd3, 0xa4, 0xad, 0xb6, 0xbf, 0x80, 0x89, 0x92, 0x9b, 0x7c, 0x75, 0x6e, 0x67, 0x58, 0x51, 0x4a, 0x43, 0x34, 0x3d, 0x26, 0x2f, 0x10, 0x19, 0x02, 0x0b, 0xd7, 0xde, 0xc5, 0xcc, 0xf3, 0xfa, 0xe1, 0xe8, 0x9f, 0x96, 0x8d, 0x84, 0xbb, 0xb2, 0xa9, 0xa0,

0x47, 0x4e, 0x55, 0x5c, 0x63, 0x6a, 0x71, 0x78, 0x0f, 0x06, 0x1d, 0x14, 0x2b, 0x22, 0x39, 0x30, 0x9a, 0x93, 0x88, 0x81, 0xbe, 0xb7, 0xac, 0xa5, 0xd2, 0xdb, 0xc0, 0xc9, 0xf6, 0xff, 0xe4, 0xed, 0x0a, 0x03, 0x18, 0x11, 0x2e, 0x27, 0x3c, 0x35, 0x42, 0x4b, 0x50, 0x59, 0x66, 0x6f, 0x74, 0x7d, 0xa1, 0xa8, 0xb3, 0xba, 0x85, 0x8c, 0x97, 0x9e, 0xe9, 0xe0, 0xfb, 0xf2, 0xcd, 0xc4, 0xdf, 0xd6, 0x31, 0x38, 0x23, 0x2a, 0x15, 0x1c, 0x07, 0x0e, 0x79, 0x70, 0x6b, 0x62, 0x5d, 0x54, 0x4f, 0x46]

MUL11 =

[0x00, 0x0b, 0x16, 0x1d, 0x2c, 0x27, 0x3a, 0x31, 0x58, 0x53, 0x4e, 0x45, 0x74, 0x7f, 0x62, 0x69, 0xb0, 0xbb, 0xa6, 0xad, 0x9c, 0x97, 0x8a, 0x81, 0xe8, 0xe3, 0xfe, 0xf5, 0xc4, 0xcf, 0xd2, 0xd9, 0x7b, 0x70, 0x6d, 0x66, 0x57, 0x5c, 0x41, 0x4a, 0x23, 0x28, 0x35, 0x3e, 0x0f, 0x04, 0x19, 0x12, 0xcb, 0xc0, 0xdd, 0xd6, 0xe7, 0xec, 0xf1, 0xfa, 0x93, 0x98, 0x85, 0x8e, 0xbf, 0xb4, 0xa9, 0xa2, 0xf6, 0xfd, 0xe0, 0xeb, 0xda, 0xd1, 0xcc, 0xc7, 0xae, 0xa5, 0xb8, 0xb3, 0x82, 0x89, 0x94, 0x9f, 0x46, 0x4d, 0x50, 0x5b, 0x6a, 0x61, 0x7c, 0x77, 0x1e, 0x15, 0x08, 0x03, 0x32, 0x39, 0x24, 0x2f, 0x8d, 0x86, 0x9b, 0x90, 0xa1, 0xaa, 0xb7, 0xbc, 0xd5, 0xde, 0xc3, 0xc8, 0xf9, 0xf2, 0xef, 0xe4, 0x3d, 0x36, 0x2b, 0x20, 0x11, 0x1a, 0x07, 0x0c, 0x65, 0x6e, 0x73, 0x78, 0x49, 0x42, 0x5f, 0x54, 0xf7, 0xfc, 0xe1, 0xea, 0xdb, 0xd0, 0xcd, 0xc6, 0xaf, 0xa4, 0xb9, 0xb2, 0x83, 0x88, 0x95, 0x9e, 0x47, 0x4c, 0x51, 0x5a, 0x6b, 0x60, 0x7d, 0x76, 0x1f, 0x14, 0x09, 0x02, 0x33, 0x38, 0x25, 0x2e, 0x8c, 0x87, 0x9a, 0x91, 0xa0, 0xab, 0xb6, 0xbd, 0xd4, 0xdf, 0xc2, 0xc9, 0xf8, 0xf3, 0xee, 0xe5, 0x3c, 0x37, 0x2a, 0x21, 0x10, 0x1b, 0x06, 0x0d, 0x64, 0x6f, 0x72, 0x79, 0x48, 0x43, 0x5e, 0x55, 0x01, 0x0a, 0x17, 0x1c, 0x2d, 0x26, 0x3b, 0x30, 0x59, 0x52, 0x4f, 0x44, 0x75, 0x7e, 0x63, 0x68, 0xb1, 0xba, 0xa7, 0xac, 0x9d, 0x96, 0x8b, 0x80, 0xe9, 0xe2, 0xff, 0xf4, 0xc5, 0xce, 0xd3, 0xd8, 0x7a, 0x71, 0x6c, 0x67, 0x56, 0x5d, 0x40, 0x4b, 0x22, 0x29, 0x34, 0x3f, 0x0e, 0x05, 0x18, 0x13, 0xca, 0xc1, 0xdc, 0xd7, 0xe6, 0xed, 0xf0, 0xfb, 0x92, 0x99, 0x84, 0x8f, 0xbe, 0xb5, 0xa8, 0xa3][140]

MUL13 =

[0x00, 0x0d, 0x1a, 0x17, 0x34, 0x39, 0x2e, 0x23, 0x68, 0x65, 0x72, 0x7f, 0x5c, 0x51, 0x46, 0x4b, 0xd0, 0xdd, 0xca, 0xc7, 0xe4, 0xe9, 0xfe, 0xf3, 0xb8, 0xb5, 0xa2, 0xaf, 0x8c, 0x81, 0x96, 0x9b, 0xbb, 0xb6, 0xa1, 0xac, 0x8f, 0x82, 0x95, 0x98, 0xd3, 0xde, 0xc9, 0xc4, 0xe7, 0xea, 0xfd, 0xf0, 0x6b, 0x66, 0x71, 0x7c, 0x5f, 0x52, 0x45, 0x48, 0x03, 0x0e, 0x19, 0x14, 0x37, 0x3a, 0x2d, 0x20, 0x6d, 0x60, 0x77, 0x7a, 0x59, 0x54, 0x43, 0x4e, 0x05, 0x08, 0x1f, 0x12, 0x31, 0x3c, 0x2b, 0x26, 0xbd, 0xb0, 0xa7, 0xaa, 0x89, 0x84, 0x93, 0x9e, 0xd5, 0xd8, 0xcf, 0xc2, 0xe1, 0xec, 0xfb,

0xf6, 0xd6, 0xdb, 0xcc, 0xc1, 0xe2, 0xef, 0xf8, 0xf5, 0xbe, 0xb3, 0xa4, 0xa9, 0x8a, 0x87, 0x90, 0x9d, 0x06, 0x0b, 0x1c, 0x11, 0x32, 0x3f, 0x28, 0x25, 0x6e, 0x63, 0x74, 0x79, 0x5a, 0x57, 0x40, 0x4d, 0xda, 0xd7, 0xc0, 0xcd, 0xee, 0xe3, 0xf4, 0xf9, 0xb2, 0xbf, 0xa8, 0xa5, 0x86, 0x8b, 0x9c, 0x91, 0x0a, 0x07, 0x10, 0x1d, 0x3e, 0x33, 0x24, 0x29, 0x62, 0x6f, 0x78, 0x75, 0x56, 0x5b, 0x4c, 0x41, 0x61, 0x6c, 0x7b, 0x76, 0x55, 0x58, 0x4f, 0x42, 0x09, 0x04, 0x13, 0x1e, 0x3d, 0x30, 0x27, 0x2a, 0xb1, 0xbc, 0xab, 0xa6, 0x85, 0x88, 0x9f, 0x92, 0xd9, 0xd4, 0xc3, 0xce, 0xed, 0xe0, 0xf7, 0xfa, 0xb7, 0xba, 0xad, 0xa0, 0x83, 0x8e, 0x99, 0x94, 0xdf, 0xd2, 0xc5, 0xc8, 0xeb, 0xe6, 0xf1, 0xfc, 0x67, 0x6a, 0x7d, 0x70, 0x53, 0x5e, 0x49, 0x44, 0x0f, 0x02, 0x15, 0x18, 0x3b, 0x36, 0x21, 0x2c, 0x0c, 0x01, 0x16, 0x1b, 0x38, 0x35, 0x22, 0x2f, 0x64, 0x69, 0x7e, 0x73, 0x50, 0x5d, 0x4a, 0x47, 0xdc, 0xd1, 0xc6, 0xcb, 0xe8, 0xe5, 0xf2, 0xff, 0xb4, 0xb9, 0xae, 0xa3, 0x80, 0x8d, 0x9a, 0x97][140]

MUL14 =

[0x00, 0x0e, 0x1c, 0x12, 0x38, 0x36, 0x24, 0x2a, 0x70, 0x7e, 0x6c, 0x62, 0x48, 0x46, 0x54, 0x5a, 0xe0, 0xee, 0xfc, 0xf2, 0xd8, 0xd6, 0xc4, 0xca, 0x90, 0x9e, 0x8c, 0x82, 0xa8, 0xa6, 0xb4, 0xba, 0xdb, 0xd5, 0xc7, 0xc9, 0xe3, 0xed, 0xff, 0xf1, 0xab, 0xa5, 0xb7, 0xb9, 0x93, 0x9d, 0x8f, 0x81, 0x3b, 0x35, 0x27, 0x29, 0x03, 0x0d, 0x1f, 0x11, 0x4b, 0x45, 0x57, 0x59, 0x73, 0x7d, 0x6f, 0x61, 0xad, 0xa3, 0xb1, 0xbf, 0x95, 0x9b, 0x89, 0x87, 0xdd, 0xd3, 0xc1, 0xcf, 0xe5, 0xeb, 0xf9, 0xf7, 0x4d, 0x43, 0x51, 0x5f, 0x75, 0x7b, 0x69, 0x67, 0x3d, 0x33, 0x21, 0x2f, 0x05, 0x0b, 0x19, 0x17, 0x76, 0x78, 0x6a, 0x64, 0x4e, 0x40, 0x52, 0x5c, 0x06, 0x08, 0x1a, 0x14, 0x3e, 0x30, 0x22, 0x2c, 0x96, 0x98, 0x8a, 0x84, 0xae, 0xa0, 0xb2, 0xbc, 0xe6, 0xe8, 0xfa, 0xf4, 0xde, 0xd0, 0xc2, 0xcc, 0x41, 0x4f, 0x5d, 0x53, 0x79, 0x77, 0x65, 0x6b, 0x31, 0x3f, 0x2d, 0x23, 0x09, 0x07, 0x15, 0x1b, 0xa1, 0xaf, 0xbd, 0xb3, 0x99, 0x97, 0x85, 0x8b, 0xd1, 0xdf, 0xcd, 0xc3, 0xe9, 0xe7, 0xf5, 0xfb, 0x9a, 0x94, 0x86, 0x88, 0xa2, 0xac, 0xbe, 0xb0, 0xea, 0xe4, 0xf6, 0xf8, 0xd2, 0xdc, 0xce, 0xc0, 0x7a, 0x74, 0x66, 0x68, 0x42, 0x4c, 0x5e, 0x50, 0x0a, 0x04, 0x16, 0x18, 0x32, 0x3c, 0x2e, 0x20, 0xec, 0xe2, 0xf0, 0xfe, 0xd4, 0xda, 0xc8, 0xc6, 0x9c, 0x92, 0x80, 0x8e, 0xa4, 0xaa, 0xb8, 0xb6, 0x0c, 0x02, 0x10, 0x1e, 0x34, 0x3a, 0x28, 0x26, 0x7c, 0x72, 0x60, 0x6e, 0x44, 0x4a, 0x58, 0x56, 0x37, 0x39, 0x2b, 0x25, 0x0f, 0x01, 0x13, 0x1d, 0x47, 0x49, 0x5b, 0x55, 0x7f, 0x71, 0x63, 0x6d, 0xd7, 0xd9, 0xcb, 0xc5, 0xef, 0xe1, 0xf3, 0xfd, 0xa7, 0xa9, 0xbb, 0xb5, 0x9f, 0x91, 0x83, 0x8d][140]

4.3.3.3 Inverse ShiftRows

In a similar fashion to the Shift Rows operation, the inverse ShiftRows operation is executed on the result of the STATE, which is in this case obtained from the inverse Mix Columns operation as against the Shiftrows in section 4.3.2.2 which takes its STATE from the SubBytes operation defined in 4.3.2.1. Visualizing the operation as a matrix, the Inverse Shift Rows operation is also an element-wise rotational operation with elements of the first Row being shifted by zero to the left (or not being shifted at all), the second row being shifted by one element, the third element shifted by two elements until the last row is shifted as shown:

$$Grid - Before - InvShift = \begin{bmatrix} c & e & x & l \\ i & r & t & o \\ p & t & - & c \\ h & e & b & k \end{bmatrix}$$

$$Grid - After - InvShift = \begin{bmatrix} c & e & x & l \\ r & t & o & i \\ - & c & p & t \\ k & h & e & b \end{bmatrix}$$

4.3.3.4 Inverse SubBytes

The inverse byte substitution (Inv SubBytes) operation is the last decryption step of the round function.It takes is input as the STATE output from the inverse Mix Columns operation and performs a look up of the Rijindael inverse substitution box table to derive the corresponding byte values of the STATE. An ASCII transformation of its bytes returns the values into the original plaintext before the key whitening process explained in section 4.3.2.

InvS-box =

[0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb, 0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb, 0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e, 0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25, 0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,

0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84, 0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06, 0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b, 0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73, 0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e, 0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b, 0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4, 0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f, 0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef, 0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61, 0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d] [140]

4.4 Comparison of the AES, Clefia & the Reduced Rounds Algorithm

4.4.1 Algebraic Structures and Constructions

Instrumental to the implementations of both Clefia and the Efficient Algorithm is the Galois field operations involving polynomial elements and the unique roles of irreducible polynomials in the constructions that yield the look up table values. Sequel to the cryptanalytic overview of the consequence of AES round reduction and the corresponding justification of same as detailed in sections 3.3 and 3.4 respectively, the efficient algorithm is based on the AES which underlying cryptographic structure is based on the Substitution Permutation Network (SPN), while the fundamental structure of Clefia is a Generalized Feistel Structure (GFN) consisting of 4 data lines, in which there are two 32-bit F-functions per one round [141]. Based on theorem 3 (detailed in section 3.4) and according to according to [142], several algorithms in coding theory and cryptography make use of irreducible polynomials of degree n over finite fields. By definition, a polynomial $f(x) \in R[x]$ with $deg(f(x)) >= 1$ is said to be irreducible if it satisfies the following condition:

$$f(x) = g(x)h(x),\ where\ g(x)\ or\ h(x) \in R[x] \tag{4.1}$$

$\implies$ *either* $g(x)$ *or* $h(x)$ *is a constant* [79]. The multiplication of polynomial elements modulo irreducible polynomials in $GF(2^8)$ forms the crux of the algebraic operations which give rise to the lookup tables for implementing both Clefia and the AES –and by extension, the Efficient algorithm for constrained IoT devices. The algebraic operations for both Clefia and the AES happen in $GF(2^8)$ and the existence of the requisite irreducible polynomials for the algebraic constructions for both algorithms is guaranteed by theorem 3. In the case of the Efficient algorithm, an arbitrary STATE (message block) in the Mix-column stage of the round function undergoes the element-wise multiplication of the STATE bytes and the defined AES matrix as exampled in 4.2

$$\texttt{Lookup Table} = \begin{pmatrix} a & e & i & n \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{pmatrix} \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 1 & 1 & 3 & 2 \end{pmatrix} \; Mod \;\; (x^8 + x^4 + x^3 + x + 1) \tag{4.2}$$

The AES S-box maps an eight bit input to an eight bit output, where both the input and output are interpreted as polynomial elements over $GF(2)$ According to [97], and each element in $GF(2^8)$ effectively being a polynomial modulo 4.3 -which is a part of the AES specification [79].

$$I(x) = x^8 + x^4 + x^3 + x + 1 \tag{4.3}$$

In a similar fashion to the Galois Field operations that characterize the Mix-Columns stage of the AES round function; such as the multiplication of polynomial elements in $GF(2^8)$ as in 4.7, element-wise multiplication between the Hadamard-type matrices in the definitions of the F-functions of CLEFIA are also performed over $GF(2^8)$. However, in comparison to the AES which uses a single substitution box, Clefia uses two non-linear 8bits S-boxes and different diffusion matrices in 4.4, using a process known the Diffusion Switching Mechanism (DMS) to enhance immunity against a family of attacks and a considerable reduction in the number of

rounds [141].

$$M1 = \begin{pmatrix} 0x01 & 0x02 & 0x04 & 0x06 \\ 0x02 & 0x01 & 0x06 & 0x04 \\ 0x04 & 0x06 & 0x01 & 0x02 \\ 0x06 & 0x04 & 0x02 & 0x01 \end{pmatrix}, M2 = \begin{pmatrix} 0x01 & 0x08 & 0x02 & 0x0a \\ 0x08 & 0x01 & 0x0a & 0x02 \\ 0x02 & 0x0a & 0x01 & 0x08 \\ 0x0a & 0x02 & 0x08 & 0x01 \end{pmatrix} \tag{4.4}$$

Consequent upon the two distinct 32-bits F-functions occasioned by the GFN structure of CLEFIA, the requirements of the existence of the irreducible polynomials guaranteed by Theorem 3 for GF operations involving the matrices defined in 4.4 give rise to the two distinct irreducible polynomials defined in 4.5 and 4.6

$$I(x) = x^4 + x + 1 \tag{4.5}$$

$$I(x) = x^8 + x^4 + x^3 + x^2 + 1 \tag{4.6}$$

,

used for the constructions of the S-boxes: S_0 and S_1 in the Clefia F-functions: F_0 and F_1 respectively. As a result, the reduction of the number of rounds of the round function is made possible without compromise to the general security of the cipher in comparison to the AES with higher number of rounds. Among other considerations, the authors in [141] anchored in part, the novelty of the design consideration of Clefia on the need to find a secure trade-off to the high number of rounds as found in the preceding block ciphers such as the AES and hence, consolidating the motivation for finding a secure trade-off to the complexity imposed by the round function as detailed in [105].

$$\begin{pmatrix} c & e & x & l \\ i & r & t & o \\ p & t & - & c \\ h & e & b & k \end{pmatrix} \begin{pmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{pmatrix} \tag{4.7}$$

For an arbitrary finite field with q elements F_q with $P_{n,q}$ as the set of irreducible polynomials of degree n over F, any arbitrary element in P is a plausible candidate for the construction of the S-box as obtainable with 4.3 and 4.6 in the algebraic constructions of the AES and Clefia respectively. The authors in [142] highlighted the concept of enumeration with respect to the number of irreducible polynomials over a finite field, and went on to give a detailed analysis of deterministic and randomized algorithms that enumerate any number of irreducible polynomials of degree n over a finite field in quasi-linear time cost per element. With the plethora of use-cases of algorithms characterised by such complexities in the deployments of IoT devices that are constrained in power and processing capabilities, investigation of complexity reductions of such algorithms which are widely in use in the IoT landscape remains highly desirable; such that multiplication of any two arbitrary elements in $GF(2^8)$ is done modulo $(x^8 + x^4 + x^3 + x + 1)$ for the AES lookup table constructions. Similar to the AES, multiplication of matrices and vectors towards the construction of the s-boxes for Clefia is also done in $GF(2^8)$ but modulo 4.6, where 4.6 is in fact, a primitive polynomial in $GF(2^8)$ and which existence is guaranteed in an arbitrary field by Theorem 3 as detailed in section 3.4. Moreover, a primitive polynomial p in an arbitrary field F is an irreducible polynomial with the additional properties of being the generator element.

4.4.2 Light-weight versus the Efficient algorithm

This section aims to compare; in terms of encryption completion times, the implementation of the Efficient Algorithm for Power Constrained IoT devices and a select lightweight algorithm using some factors of compatibility as a baseline. The Efficient Security Algorithm for Constrained IoT Devices detailed in [105] is based on the AES. It securely reduces rounds of the AES round function and trades-off by leveraging a tamper proof Secure Element as detailed in the justification of round reduction and security trad-off in chapter3.4, towards reduction of complexity in tandem with resource constrain in a sample SAMG55 IoT device and without compromise on security. In a related work detailed in [28] and as graphically presented in Fig. 4.4.2, Clefia emerged a plausible lightweight cipher -in terms of requiring the least computing resources when compared to CAMELLIA, SEED and the standard AES. Furthermore, the efficient algorithm for constrained IoT devices is utilized as a client-side encryption solution for a sample IoT device, preceding the secure provisioning of the IoT device onto the AWS IoT cloud platform. A summary of this experimentation and comparison of the efficient algorithm for constrained IoT devices and the lightweight CLEFIA is presented in chapter 6.2.3

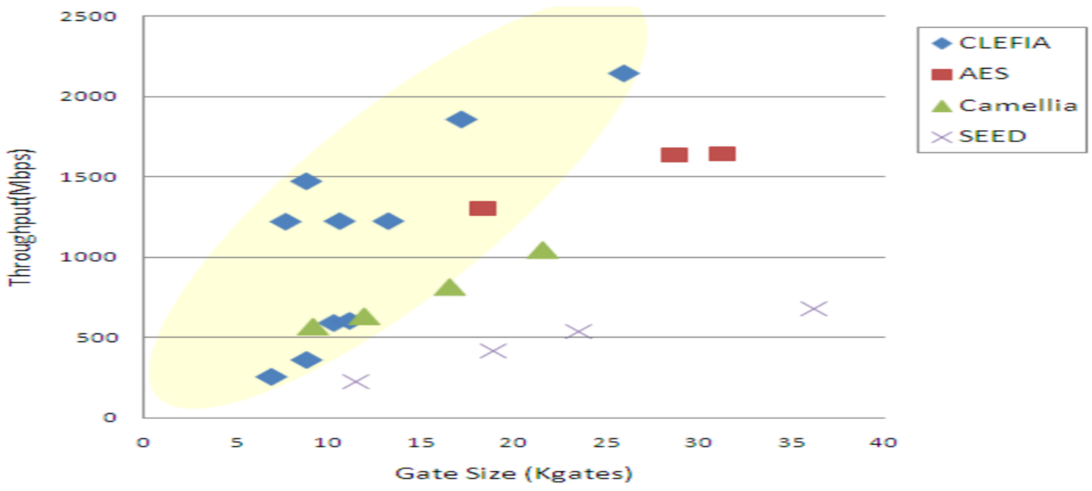

Figure 4.1: A comparison of lightweight algorithms
[28]

4.5 Experiments

Research experimentation upon which the content of this chapter is based include: the implementation of the variants of the Standard AES algorithm viz AES128, AES192 and AES256, implementation of lightweight clefia and implementation of the Efficient algorithm for Constrained IoT Devices. Following the comparison of the algorithms in terms of algebraic structures and constructions, lightweight versus the efficient algorithm, implementations of the algorithms was experimented as per the following setup in section 4.5.1.

4.5.1 Experimental Setup

Algorithm 1: Client-Side-Encryption Execution Flow

```
Message, Key
initialization of the counter i = 0 and Nbr = 2 or 4
Expand key to length: (block size) *Nbr + block size
STATE = message XORed with Key (Key whitening)
Invoke the round function:
while i < Nbr : do
    STATE = SubByte(STATE)
    STATE = ShiftRows(STATE)
    if i < Nbr : then
        STATE = MixColumn(STATE)
    end
    Invoke addRoundKey(STATE, NextRoundkey)
end
STATE as resulting Ciphertext
```

Experimentation tools used include a laptop computer and a SAMG55 microprocessor in terms of hardware. Software level components of the implementation tools include the Zerynth studio -which runs python optimized with C, and Jupyter notebook for analysis and plots. Two platforms adjudged to suffice for the aim of experimentation include a laptop computer as a resource-sufficient platform and a SAMG55 microprocessor as a resource constrained platform as well as a typical IoT device. While the goal on one hand is to utilize the notion of percentages as presented in section 3.2.2 to express the impact of resource constraint on the sample constrained device in comparison to the resource-sufficient laptop, on the other hand is to observe and compare the attributes of complexity between the aforementioned algorithms on both platforms in a mutually exclusive context. For each instance of encryption, the encryption is repeated for one thousand iterations and the average execution time of the one thousand iterations is logged. The notion of average as detailed in section 3.2.2 is applied to obtain statistically relevant values in terms of a representative number for the encryption completion time of a single message block. Furthermore, this experiment is repeated for one hundred encryption instances for the distinct key lengths: 128, 192 and 256 of the standard AES algorithm on both platforms. Following the same setup and using the same platforms of experimentation, the lightweight clefia algorithm was also implemented, and finally the Efficient security algorithm for Constrained IoT devices. Data obtained with respect to the aforementioned experiments is presented in table table 5.1. The results and analysis detailed in chapter 6 further shows these experimental data in table 4.3 and 4.4 for the laptop-resource-sufficient platform and the SAMG55 resource-constrained platform respectively. The algorithm table 4.1 details the Efficient Security Algorithm for Constrained IoT Devices as implemented, whereby the iteration number of the round function is experimentally reduced to four and two respectively as evidenced by the data in table 4.3 and 4.5. Following the completion of step1 of the Efficient algorithm for Constrained IoT Devices flow as detailed in table 4.1, the execution of the step2 as in table 4.1 utilizes the reduced round cipher as detailed in Algorithm 1, based on the outcome of the aforementioned experiments which shows the reduced round algorithm as to have the least encryption completion time of the sample message block. Also, based on the sustained security in terms of the key-length as detailed in 3.4, the key length is expanded as per the third sequence in the algorithm 1 before it is mixed with the sample message block for key whitening in the fourth sequence. The round function is finally invoked to achieve encryption following the stages of the encryption process outlined in section 4.3.1.

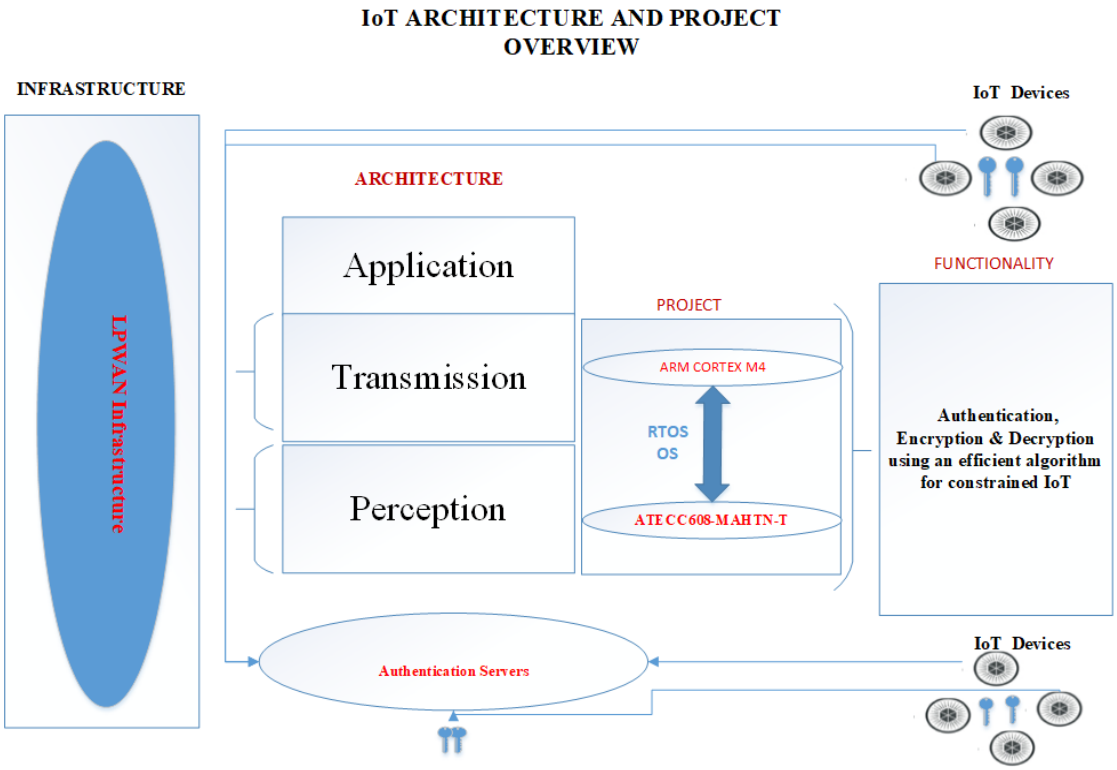

Figure 4.2: IoT Architecture and Project Overview

With respect to the IoT protocol architecture as shown in Fig. 4.2 thus, an arbitrary IoT device which in this case, is a SAMG55 microprocessor, invokes step1 of the efficient algorithm as detailed in table 4.1 to achieve secure authentication with an IoT cloud platform, followed by the execution of step2 which avails other cloud computing potentials as reviewed in 2.8; including scalable storage systems for agile storage of IoT device data. Furthermore, according to [143], It will be of particular importance to explore new techniques of jointly defending against multiple types of wireless attacks, which may be termed as mixed wireless attacks and the duo of secure authentication and encryption of data before transmission unto to cloud storage systems meet security requirements in more than one layer of the IoT protocol architecture as advocated by the authors in [143], including the perception and transmission layers of the architecture as shown in Fig. 4.2

Table 4.3: Encryption Times Data Generated from Laptop Implementations of Standard AES Variants, RR4 and RR2

Instances	AES-128	AES-192	AES-256	4Rounds	2Rounds
1	0.796481	1.063513	1.150296	0.358913	0.319948

2	0.938084	1.087956	1.231361	0.33023	0.355408
3	0.678996	0.958239	1.091734	0.774768	0.264625
4	0.840003	0.779829	0.862399	0.288513	0.257222
5	0.699007	0.820399	1.179502	0.329047	0.241581
6	0.871251	1.024502	0.926096	0.29894	0.231383
7	0.659962	0.760395	0.91801	0.399697	0.523993
8	0.647439	0.744336	0.898726	0.3847	0.196657
9	0.646284	0.745524	0.84429	0.291152	0.335179
10	0.642803	0.784104	0.847197	0.354563	0.262377
11	0.668974	0.775927	0.95745	0.383063	0.198691
12	0.656269	0.746075	0.827348	0.270903	0.199992
13	0.669492	0.786635	0.803035	0.287626	0.208207
14	0.651663	0.721784	0.788633	0.295411	0.217068
15	0.65963	0.762982	0.782648	0.335264	0.203221
16	0.644828	0.74586	0.808947	0.295549	0.206765
17	0.678491	0.746925	0.816973	0.292222	0.209141
18	0.640799	0.773456	0.814105	0.287358	0.348329
19	0.646623	0.730141	0.794316	0.28049	0.319346
20	0.652565	0.725419	0.793985	0.288851	0.266928
21	0.640252	0.764499	0.836343	0.296679	0.204343
22	0.64393	0.767146	0.814982	0.29164	0.198584
23	0.637668	0.745981	0.793461	0.295319	0.240341
24	0.633993	0.755575	0.795994	0.299369	0.196061
25	0.652754	0.732563	0.78073	0.28942	0.204323
26	0.662431	0.761634	0.842196	0.29031	0.200725
27	0.637823	0.811709	0.815935	0.297496	0.277536
28	0.65987	0.771289	0.863997	0.287387	0.215974
29	0.658937	0.734334	0.826035	0.290604	0.197776
30	0.636545	0.775253	0.791555	0.305222	0.20276
31	0.662086	0.751674	0.78919	0.296161	0.203225
32	0.644834	0.729328	0.791722	0.28568	0.200307
33	0.667148	0.786044	0.790263	0.293869	0.239411
34	0.634388	0.735546	0.801227	0.300489	0.20953
35	0.676704	0.82529	0.812097	0.292282	0.188598
36	0.635997	0.742736	0.801363	0.286407	0.20107
37	0.643702	0.71343	0.824174	0.296671	0.192198
38	0.641621	0.772996	0.801476	0.279898	0.227209
39	0.631685	0.743731	0.797509	0.280169	0.196283
40	0.639805	0.759186	0.801189	0.279618	0.210804
41	0.674406	0.751732	0.79255	0.29026	0.204457

42	0.724704	0.752392	0.784566	0.277302	0.191754
43	0.650104	0.720907	0.829447	0.284235	0.241602
44	0.67115	0.771285	0.797776	0.293654	0.192311
45	0.658444	0.720485	0.810518	0.29496	0.219324
46	0.637829	0.797668	0.805972	0.299274	0.228382
47	0.657	0.7275	0.785955	0.284178	0.190228
48	0.641977	0.861797	0.793709	0.294227	0.263724
49	0.642562	0.770741	0.793679	0.295848	0.240137
50	0.655844	0.752201	0.788024	0.287839	0.192244
51	0.640611	0.740977	0.792546	0.298359	0.205322
52	0.675508	0.849912	0.786215	0.278771	0.257448
53	0.676227	0.734257	0.807021	0.281918	0.199227
54	0.652618	0.763153	0.831031	0.294223	0.308798
55	0.651402	0.750046	0.794944	0.281681	0.217811
56	0.653366	0.786155	0.79249	0.289811	0.197847
57	0.741723	0.760848	0.812319	0.296721	0.206297
58	0.65985	0.775478	0.796919	0.280992	0.201199
59	0.650861	0.758782	0.797387	0.291227	0.20358
60	0.663025	0.772559	0.819652	0.299667	0.208102
61	0.645486	0.843913	0.797077	0.292224	0.221715
62	0.660474	0.755524	0.799229	0.293403	0.246193
63	0.647182	0.732947	0.808645	0.302071	0.21593
64	0.642671	0.741091	0.791025	0.281508	0.188439
65	0.639493	0.82436	0.834648	0.305902	0.210972
66	0.661803	0.737679	0.844368	0.288427	0.26281
67	0.647245	0.746962	0.835154	0.2906	0.203778
68	0.655512	0.724558	0.802613	0.340236	0.194663
69	0.632978	0.743145	0.828356	0.295826	0.187151
70	0.657448	0.756955	0.806119	0.286422	0.223906
71	0.660351	0.775812	0.795449	0.287188	0.346972
72	0.639	0.788759	0.819285	0.281686	0.20823
73	0.644025	0.717566	0.81187	0.276355	0.208011
74	0.686817	0.783176	0.806307	0.291758	0.194373
75	0.686989	0.74995	0.792521	0.294978	0.216359
76	0.669858	0.73884	0.808769	0.32362	0.345135
77	0.660895	0.743109	0.813774	0.289241	0.187716
78	0.991533	0.801572	0.80964	0.287427	0.213549
79	0.646495	0.760508	0.812479	0.315125	0.203112
80	0.658818	0.732363	0.802092	0.293908	0.215188
81	0.659467	0.739964	0.802328	0.288141	0.282489

82	0.655914	0.748594	0.819241	0.288273	0.193076
83	0.641507	0.850415	0.799361	0.287449	0.290985
84	0.653858	0.790232	0.813728	0.299868	0.200441
85	0.649888	0.718799	0.845874	0.284034	0.206625
86	0.66349	0.755336	0.831243	0.295317	0.226451
87	0.672976	0.760736	0.867467	0.282433	0.209834
88	0.645787	0.793376	0.850376	0.287209	0.199801
89	0.654096	0.745713	0.783989	0.290388	0.191049
90	0.650949	0.751276	0.811824	0.295241	0.211907
91	0.668338	0.741746	0.812153	0.285306	0.264901
92	0.653422	0.734143	0.78999	0.29843	0.264115
93	0.652116	0.815095	0.856662	0.308453	0.197625
94	0.648468	0.772482	0.816817	0.322918	0.198779
95	0.649995	0.781928	0.831812	0.290007	0.210326
96	0.652253	0.759551	0.787012	0.293948	0.30342
97	0.649166	0.717744	0.787099	0.299119	0.204918
98	0.673989	0.741115	0.840173	0.28418	0.203088
99	0.677502	0.825463	0.813646	0.294659	0.190189
100	0.649556	0.745381	0.797679	0.300579	0.209313

Table 4.4: Encryption Times Data Generated from SAMG55 Implementations of Standard AES Variants, RR4 AND RR2

Instances	AES-128	AES-192	AES-256	4Rounds	2Rounds
1	7.984538	4.376509	4.591796	6.363885	5.306837
2	7.089851	1.788835	1.755769	4.235857	3.558903
3	7.127803	1.721226	1.776487	3.16629	3.651014
4	7.174031	1.726693	1.85489	3.184767	3.503775
5	7.152289	1.710917	1.845072	3.178487	3.447525
6	7.104029	1.727236	1.767246	3.187219	3.493384
7	7.157279	1.696156	1.777752	3.172035	3.480428
8	7.000349	1.709376	1.774204	3.173628	3.657735
9	7.171574	1.693379	1.780848	3.175791	3.560993
10	7.163608	1.701858	1.754988	3.242606	3.674276
11	7.185225	1.740665	1.750485	3.165409	3.476701
12	7.168224	1.713398	1.772632	3.170396	3.578366
13	7.200771	1.726966	1.767801	3.237513	3.411661
14	7.034439	1.69054	1.809806	3.158723	3.587528
15	7.118135	1.86884	1.810308	3.217531	3.502104

16	7.012956	1.75641	1.752223	3.20731	3.447414
17	7.050733	1.747278	1.750933	3.225302	3.507083
18	6.932242	1.719292	1.785582	3.217135	3.551365
19	7.057883	1.741946	1.780796	3.183174	3.600353
20	7.194373	1.752254	1.766307	4.316629	3.791293
21	7.18741	1.720412	1.766582	4.997538	3.789522
22	7.009948	1.713064	1.754277	4.826296	3.566956
23	7.18193	1.748767	1.783556	5.088456	3.513176
24	7.119167	1.752333	1.803143	4.966397	3.543044
25	7.070501	1.730518	1.828147	4.91398	3.497344
26	7.27944	1.751703	1.881975	5.113627	3.48971
27	7.317773	1.737005	1.820635	5.143361	3.558582
28	7.325782	1.693597	1.760835	4.828806	3.597799
29	7.316189	1.743311	1.764166	4.996462	3.474725
30	7.135198	1.703553	1.775462	4.943059	3.485705
31	7.148603	1.707309	1.77276	4.913815	3.565918
32	7.148829	1.717427	1.780382	5.156979	3.646651
33	7.174035	1.711046	1.792778	5.419771	3.534058
34	7.145739	1.712246	1.736315	5.239681	3.500095
35	7.0777	1.715508	1.756895	4.936069	3.571149
36	7.156501	1.752385	1.765164	4.915023	3.671927
37	7.27553	1.734839	1.759506	4.919376	3.494774
38	7.192501	1.715024	1.752835	4.871579	3.75103
39	7.206351	1.686251	1.799183	5.137943	3.50964
40	7.191128	1.753334	1.791394	5.211111	3.448011
41	7.15071	1.716189	1.793412	4.854047	3.512652
42	7.127051	1.737906	1.752143	4.736589	3.524758
43	7.186883	1.704948	1.756145	4.998609	3.460276
44	7.002464	1.703144	1.760613	4.836232	3.481107
45	7.163118	1.693002	1.771976	5.055323	3.481105
46	7.064893	1.714426	1.785475	4.890475	3.482311
47	7.193802	1.704635	1.827684	4.805816	3.54122
48	7.150964	1.703566	1.833864	4.811775	3.528512
49	7.195818	1.699609	1.803228	5.223254	3.561153
50	6.876114	1.699607	1.768758	5.227791	3.466134
51	7.078675	1.711923	1.772974	5.126851	3.574118
52	7.146508	1.72394	1.778504	5.239709	3.583825
53	7.185622	1.72274	1.757705	5.345432	3.447737
54	7.194463	1.755427	1.760729	5.273161	3.487202
55	7.155756	1.695859	1.774073	5.083458	3.450933

56	7.129797	1.689589	1.775179	5.141819	3.538909
57	6.975633	1.705105	1.764888	5.305414	3.537691
58	7.194548	1.710178	1.790286	5.346198	3.517696
59	7.126028	1.692449	1.772387	5.254532	3.540433
60	7.135184	2.062214	1.739689	5.297949	3.493113
61	7.160069	1.804836	1.752984	5.246082	3.437365
62	7.178187	1.820262	1.74384	5.292363	3.585943
63	7.080727	1.817761	1.759753	5.264364	3.468655
64	7.212507	1.729478	1.789987	4.856729	3.477246
65	7.185731	1.708357	1.766227	5.28034	3.455186
66	7.138659	1.725593	1.762311	5.262072	3.452635
67	7.171036	1.743637	1.751412	5.130869	3.457152
68	7.119335	1.696275	1.758483	5.264585	3.911168
69	7.093276	1.756966	1.750662	5.027514	3.485562
70	7.043371	1.707867	1.745099	5.049326	3.483925
71	6.859634	1.709464	1.751969	5.203746	3.568834
72	7.025556	1.707596	1.778411	5.092111	3.606569
73	7.126573	1.721982	1.767941	5.221105	3.616531
74	7.149767	1.729844	1.759829	5.143913	3.447822
75	7.231375	1.728715	1.771473	5.277038	3.506609
76	7.157712	1.693133	1.791444	5.253356	3.552993
77	7.168858	1.737991	1.748703	5.227837	3.460477
78	7.184709	1.752007	1.773258	5.240417	3.467442
79	7.185987	1.739966	1.751512	5.304774	3.46825
80	7.241599	1.706745	1.783099	5.187791	3.492075
81	7.196707	1.717986	1.749307	5.344512	3.454579
82	7.148875	1.727395	1.782425	5.276219	3.48244
83	7.187938	1.721819	1.813241	5.293019	3.575745
84	7.225692	1.699264	1.760564	5.177794	3.526932
85	7.18255	1.722557	1.750845	5.276276	3.507894
86	7.105837	1.720625	1.770941	5.287	3.48867
87	7.139653	1.715209	1.787905	5.216691	3.481596
88	7.220688	1.752833	1.764806	5.318556	3.42864
89	7.197842	1.77962	1.766387	5.304845	3.525836
90	7.13766	1.712285	1.75845	5.262061	3.475132
91	7.155074	1.700417	1.767977	5.343874	3.477388
92	7.134312	1.693913	1.789865	5.301605	3.509621
93	7.176993	1.708472	1.778387	5.270608	3.430877
94	7.041505	1.713679	1.780969	5.219958	3.447758
95	7.150066	1.703254	1.791569	5.16915	3.398484

96	7.157666	1.710291	1.740554	5.117923	3.392671
97	7.182454	1.695913	1.744816	5.273776	3.446055
98	7.084174	1.713604	1.792557	5.081166	3.475504
99	7.219173	1.699406	1.767681	5.252665	3.535517
100	7.204138	1.705019	1.760351	5.309807	3.424857

Table 4.5: Encryption Times Data Generated from PC AND SAMG55 Implementations

Instances	PC AES128	SAMG55 AES128	PCRR2	SAMG55RR2
1	0.8653966	7.9845381	0.3199476	5.306837
2	0.6722173	7.0898513	0.3554082	3.558903
3	0.6078102	7.1278026	0.2646247	3.651014
4	0.8142278	7.1740311	0.2572222	3.503775
5	0.7733388	7.1522892	0.2415805	3.447525
6	0.5505474	7.1040291	0.2313827	3.493384
7	0.5367047	7.1572788	0.5239930	3.480428
8	0.9111719	7.0003486	0.1966569	3.657735
9	0.538378	7.1715739	0.33517940	3.560993
10	0.5797766	7.1636076	0.2623766	3.674276
11	0.5616452	7.1852247	0.1986907	3.476701
12	0.5526894	7.1682235	0.1999917	3.578366
13	0.5901672	7.2007712	0.2082066	3.411661
14	0.5529859	7.0344392	0.2170684	3.587528
15	0.525381	7.1181350	0.20322110	3.502104
16	0.5492386	7.0129556	0.2067645	3.447414
17	0.5311931	7.0507333	0.2091410	3.507083
18	0.5814451	6.9322417	0.3483287	3.551365
19	0.5453837	7.0578825	0.3193460	3.600353
20	0.5378147	7.1943727	0.2669280	3.791293
21	0.5528532	7.1874095	0.2043434	3.789522
22	0.5674588	7.0099481	0.1985842	3.566956
23	0.5449442	7.1819301	0.2403406	3.513176
24	0.5545544	7.1191668	0.1960610	3.543044
25	0.5924382	7.0705010	0.20432250	3.4973440
26	0.5724627	7.2794395	0.2007245	3.489710
27	0.5300381	7.3177725	0.2775364	3.558582
28	0.5526312	7.325782	0.21597360	3.597799
29	0.5449595	7.316189	0.1977761	3.474725
30	0.5474292	7.1351984	0.2027602	3.485705

31	0.5702799	7.1486028	0.2032251	3.565918
32	0.5282262	7.1488287	0.2003072	3.646651
33	0.5658316	7.174035	0.2394114	3.534058
34	0.5339959	7.1457394	0.20953	3.500095
35	0.5886807	7.0776997	0.1885983	3.571149
36	0.5565304	7.1565013	0.2010696	3.671927
37	0.5620644	7.2755302	0.1921977	3.494774
38	0.5560042	7.1925013	0.2272087	3.75103
39	0.5326382	7.2063509	0.1962833	3.50964
40	0.6217603	7.1911279	0.2108037	3.448011
41	0.5486803	7.1507099	0.204457	3.512652
42	0.5354819	7.1270511	0.1917535	3.524758
43	0.5610803	7.1868825	0.2416022	3.460276
44	0.544726	7.0024638	0.1923111	3.481107
45	0.6280711	7.1631181	0.219324	3.481105
46	0.5349875	7.0648929	0.2283822	3.482311
47	0.5944224	7.1938023	0.190228	3.54122
48	0.5295156	7.1509642	0.2637239	3.528512
49	0.5640447	7.1958179	0.2401371	3.561153
50	0.5686602	6.876114	0.1922442	3.466134
51	0.5318003	7.0786747	0.205322	3.574118
52	0.6053492	7.1465078	0.2574483	3.583825
53	0.5514164	7.1856221	0.1992267	3.447737
54	0.5522621	7.1944632	0.3087982	3.487202
55	0.5351141	7.1557555	0.2178106	3.450933
56	0.5556107	7.1297972	0.1978465	3.538909
57	0.5365631	6.9756328	0.2062973	3.537691
58	0.5505253	7.1945475	0.2011987	3.517696
59	0.5406325	7.1260282	0.2035803	3.540433
60	0.5376175	7.135184	0.2081023	3.493113
61	0.5655508	7.1600694	0.2217153	3.437365
62	0.5551825	7.1781874	0.2461931	3.585943
63	0.5491718	7.0807266	0.2159295	3.468655
64	0.595563	7.2125069	0.1884392	3.477246
65	0.5490705	7.1857306	0.2109718	3.455186
66	0.5433521	7.1386586	0.2628103	3.452635
67	0.6499065	7.1710358	0.203778	3.457152
68	0.5715895	7.1193347	0.1946626	3.911168
69	0.5486831	7.0932761	0.1871505	3.485562
70	0.5419936	7.0433712	0.2239057	3.483925

71	0.5741845	6.8596342	0.3469724	3.568834
72	0.6248896	7.0255559	0.2082296	3.606569
73	0.5728943	7.1265729	0.2080106	3.616531
74	0.5422509	7.1497671	0.194373	3.447822
75	0.5681231	7.2313748	0.2163594	3.506609
76	0.619145	7.1577117	0.3451346	3.552993
77	0.5585851	7.1688576	0.1877159	3.460477
78	0.6192985	7.1847087	0.2135492	3.467442
79	0.581996	7.1859872	0.2031121	3.46825
80	0.5696902	7.2415987	0.2151881	3.492075
81	0.5627767	7.1967067	0.2824886	3.454579
82	0.5439197	7.148875	0.1930755	3.48244
83	0.5451428	7.1879384	0.290985	3.575745
84	0.6398623	7.2256915	0.2004411	3.526932
85	0.5348342	7.1825497	0.2066249	3.507894
86	0.5525812	7.1058369	0.2264506	3.48867
87	0.5737178	7.1396526	0.2098343	3.481596
88	0.5860344	7.2206878	0.1998014	3.42864
89	0.731048	7.1978417	0.1910487	3.525836
90	0.6128936	7.1376596	0.2119074	3.475132
91	0.5570411	7.1550742	0.2649007	3.477388
92	0.5813946	7.134312	0.2641149	3.509621
93	0.6102484	7.1769933	0.1976251	3.430877
94	0.557563	7.0415045	0.1987785	3.447758
95	0.6324176	7.1500664	0.210326	3.398484
96	0.6989839	7.1576659	0.30342	3.392671
97	0.5737455	7.1824543	0.2049178	3.446055
98	0.5710415	7.0841743	0.2030884	3.475504
99	0.5861858	7.2191732	0.1901892	3.535517
100	0.5410907	7.204138	0.2093132	3.424857

4.6 Chapter Summary

This chapter detailed the Efficient Algorithm for Constrained IoT devices, starting with an introduction of lightweight cryptography and it's inherent challenges as presented in sections 4.1.1 and 4.1.2, following which an overview of the efficient algorithm was given in 4.2, highlighting the two-step process of IoT device authentication and resource efficient message encryption. The implementation pseudocode for the the efficient algorithm based on the standard AES algorithm

is presented, detailing the encryption and decryption processes as presented in 4.3.1 and 4.3.3 respectively. A sample block of message (16bytes) and encryption key (16bytes) is used to illustrate the inner workings of the round function. The Pseudocode is presented for a single round of message encryption and decryption with the specific objective to detail the process of message and key whitening, byte substitution, shiftrows, mix columns and the add round key sequences respectively, and the reverse process detailing Add Round Key, Inverse Mix Columns, Inverse Shift Rows and Inverse bytes substitution is also presented. Section 4.4 presented a comparison of the classical AES algorithm, light weight clefia and the reduced round algorithm with respect to the algebraic structures and constructions. The experiments on which the content of the chapter is based was presented in section 4.5, detailing the setup and factors considered in the experiments as presented in section 4.5.1. Data generated from the experiments done are also presented in tables: 4.3, 4.4 and 4.5 as encryption times data generated from implementations of the algorithms considered on a laptop, SAMG55 microprocessor and laptop & SAMg55 combined, respectively. Section 4.6 concludes by providing a compressed summary of the various sections of the chapter: Efficient Security Algorithm for Power Constrained IoT Devices.

Chapter 5

Low cost Client-Side Encryption and Secure IoT Provisioning

5.1 Introduction

An overview of IoT device provisioning and low cost client-side encryption is diagrammatically presented in Fig. 5.1. According to the observation of the authors in [41], although billions of IoT devices are estimated to be deployed in the nearest future, very little to no information is present on ease of device provisioning. Also, the realities of IoT device constraints when juxtaposed with IoT security requirements as introduced in chapter1.5.1 gives a bases for encryption-before-outsourcing of data on IoT devices to cloud platforms, and is a widely recommended method to guarantee the confidentiality of user data according to [43]. The tools and processes required to securely provision an IoT device are herein employed, and utilized to provision a sample IoT device, the SAMG55 microprocessor to the AWS IoT cloud platform. Following the experimentation on the complexity of related security algorithms for IoT device data encryption as detailed in chapter 4, the Efficient Security Algorithm for Power Constrained IoT devices is then employed to facilitate a cost-efficient client-side encryption following the secure provisioning of the IoT device.

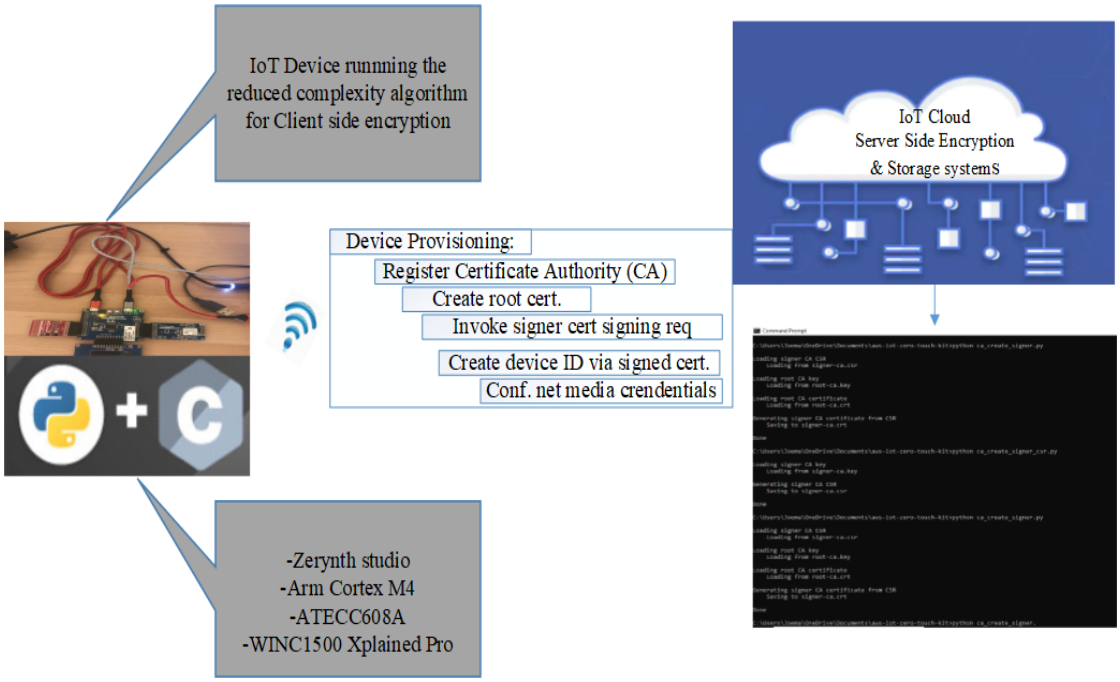

Figure 5.1: Experimental setup: Showing the SAMG55 IoT device setup using Zerynth studio. The IoT device is equiped with the ARM cortex M4, together with the Atecc608 secure element and leverages the WINC1500 Xplained pro board for wifi connectivity onto the AWS IoT Core, using the AWS CLI

5.2 Experiments

Fig. 5.1 summarizes the research experiments, as well as the setup for the experiments on which the content of this chapter is based. Broadly grouped into two as: secure IoT device provisioning and low cost encryption of the IoT device data, the device provisioning experimentation involves the process of securely connecting an IoT device to an IoT cloud platform. Hinged on the bases of cloud computing as a major enabler in the provisioning of IoT as detailed in 2.8, section 5.2.1 presents the experimentation, setup and an execution of the provisioning and client-side encryption processes.

5.2.1 Experimental Setup

The experimentation tools include: the Amazon Web Services (AWS) IoT core service, a laptop computer, a SAMG55 microprocessor tool kit, Zerynth studio, Jupyter notebooks for analysis and plots of experimental data on client-side encryption and python on Visual Studio Code -Integrated Development Environment (IDE). The Amazon Web Services (AWS) is a leading cloud services

provider which provides a plethora of cloud computing services relevant to the provisioning of IoT devices and much more. The AWS cloud service for the IoT is known and the AWS IoT core. Other IoT relevant services provided by the AWS include the Amazon Simple Storage Service (Amazon S3) -which enables scalable storage of the massive data generated either by the IoT or just traditional data, The AWS Key Management Service (KMS) -which enables the generation and management of encryption/decryption keys, which facilitates either client-side encryption or server-side encryption services of the stored data on the AWS infrastructure or on client premises, AWS lambda service which provides server-less computing service that enables running of codes on event-driven bases, the AWS Identity and Access Management (IAM) service -which offers fine-grained access control management across AWS services with respect to groups, roles and policies etc. The AWS offers various ways of accessing and communicating with these cloud services including: access through the management console, AWS command line interface (AWS CLI) and AWS Software Development Kits (SDKs) and Application Programmable Interface (API) services. Regardless of the method of accessing the platform used, the creation of an account with the AWS cloud is key, through which all required services are accessed on a pay as you go pricing model. Once the requirement of account creation is fulfilled on the AWS cloud platform, access method through the management console offers a web-application, point-and-click Graphical User Interface (GUI) comprising of many service consoles for using the AWS cloud infrastructure. Other access methods such as the AWS CLI and SDKs offer access to these range of services using a programmatic fashion. Fig. 5.2 shows the setup of the microprocessor kit which is then connected through the Universal Serial Bus (USB) ports of the laptop computer, already equipped to communicate with the AWS cloud platform through the AWS CLI and programmatically using SDKs through the installation of the AWS CLI program. The AWS SDKs used include the boto3 and botocore, which are compatible with the python programming language version 2.7 and above and used with the Visual Studio Code (VScode) IDE for scripting the requisite code snippets required for interacting with all AWS services necessary to provision the SAMG55 microprocessor. Programmatic access to AWS services using the CLI and SDKs required the retrieval of the necessary programmatic access credentials, consisting of the Access Key ID and Secret access key, which are retrieved using the AWS Identity and Access Management service console. Firstly, the AWS account is created using the management console and the IAM service is utilized to setup access rights to requisite services including the AWS IoT core and AWS Lambda services, as well as to retrieve the programmatic access credentials from the console,.

The retrieved access credentials are then utilized to establish connection to the platform using the CLI access tool. Following the establishment of a connection to the AWS cloud services platform facilitated by gaining access through the CLI, the IoT device provisioning calls are made by: registering a Certificate Authority (CA), creating a root certificate, invoking a signer certificate signing request and creation of the IoT device ID through the signed certificate -using a combination of the management console, CLI and SDK access methods.

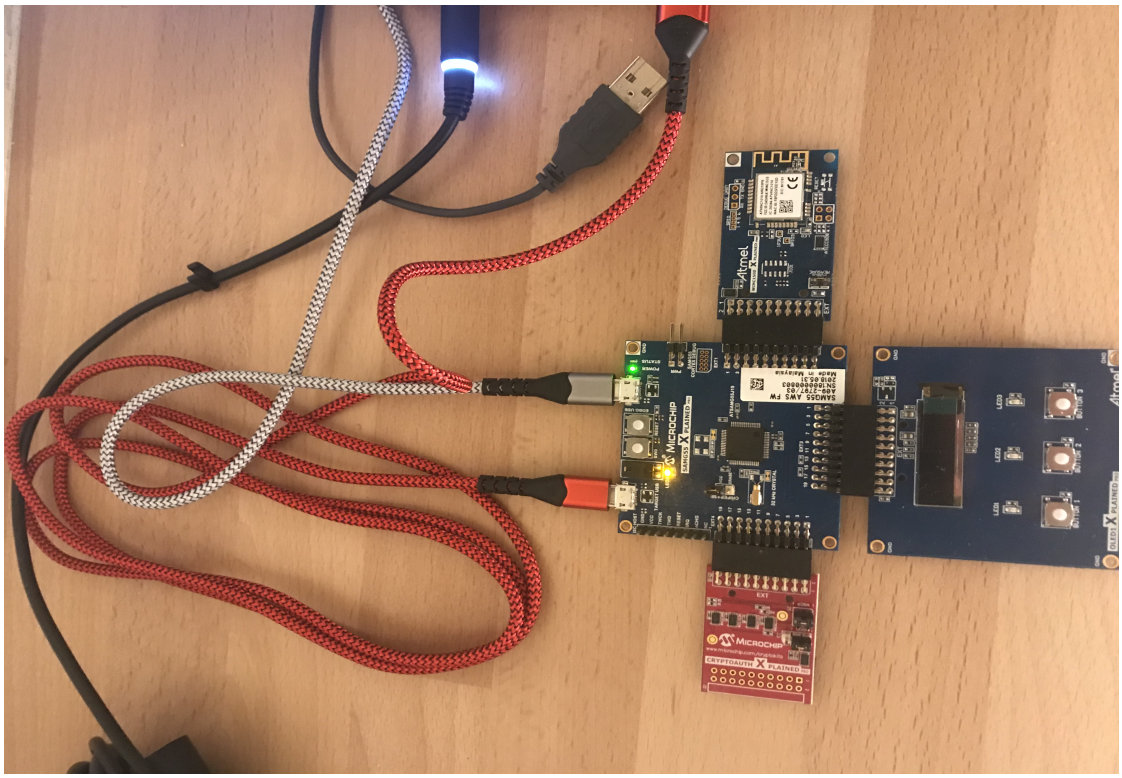

Figure 5.2: SAMG55 Provisioning experimental setup including the ARM cortex M4, the ATECC608 secure element and the WINC1500 board for enabling wifi connectivity of the IoT device.

The SAMG55 microprocessor and development kit comprise of an Xplain pro board and the ATECCX08A and ATWINC1500 extension boards which facilitate the functionalities of a hardware cryptographic accelerator and a WiFi-compatible network media module for connecting the SAMG55 microprocessor respectively. Leveraging on the functionalities of the ATECC608A as detailed in chapter5.3.1, The ATWINC1500 is utilized to facilitate the connection of the SAMG55 IoT device through a WiFi network to serve as a communication medium between the IoT device and the AWS IoT platform for onward transmission of data as needed.

Based on the experimental setup in 4.5.1, Table 5.1 details the experimentation data on the comparison of the algorithms considered and the consequent selection of the efficient algorithm as the option for low-cost client side encryption. The notion of averages and percentages as detailed in section 3.2.2 is employed to obtain statistically relevant values in terms of a representative number for the encryption completion time of a single message block, such that for each instance of encryption, the encryption is repeated for one thousand iterations and the average execution time of the one thousand iterations is logged. The experiment is repeated for one hundred encryption instances as evident by the 'Instance' column in table 5.1. As shown in table 5.1, this was done for both encryption and decryption process of the clefia lightweight algorithm and for the distinct key lengths: 128, 192 and 256, the AES128 and and RR2 experimentation data also logged in the 'AES-128' and 'RR2' columns respectively, for the comparative analysis detailed in chapter 6.

5.3 The SAMG55 Authentication and Secure Provisioning

Based on the experimental setup detailed in 5.2.1, IoT device provisioning begins with the all important process of the device authentication. Also, as detailed in the authentication and cloud assisted provisioning of IoT devices in sections 2.8.1 and 2.8 respectively, various methods of authenticating an IoT device to the cloud abound. However, the authentication of the SAMG55 microprocessor to the AWS cloud platform leverages the enriched features of the ATECC608A detailed in 5.3.1 to achieve certificate based authentication of the SAMG55 device onto the AWS IoT core.

5.3.1 The secure element (SE):

ATECC608x extension board is an integral component of the SAMG55 microprocessor, equipped as a hardware cryptographic accelerator with tamper-proof capabilities to hold sensitive data and facilitate the running of secure applications. Developed by the ARM in collaboration with Long Range Wide Area Networks and the Things industry, it holds the potential to automatically offload all cryptographic operations, such that keys will never be visible nor accessible, even when the associated IoT device is compromised [144]. The ARM Cortex M4 is a microprocessor designed for low energy efficient devices which supports IoT application development via the Atmel studio Integrated Development Platform and compatible with the IEEE 802.15.4 spec-

ification and lower power communications protocols including: Bluetooth Low Energy (BLE), Zigbee, Low Power Wide Area Network, Routing Protocol for Low power devices (RPL), Constrained application protocol (CoAP) and the Message Queuing Telemetry Transport (MQTT) messaging protocol for constrained IoT devices. The ATECC608A functionality wades against implementation attacks which aim to exploit the recovery of encryption keys through manipulating the efficient security algorithm for power constrained IoT device at the point of hardware implementation. This serves as compensation for the reduced rounds and consolidating the preserved security properties of the algorithm in the perspective of brute-force and analytical attacks as detailed in chapter3.3.2. Moreover, the ATECC608A-MAHTN-T microchip offers a cost-efficient secure authentication of the associated IoT device onto the network infrastructure; as a preceding requirement to secure message encryption and decryption hence, facilitating the aforementioned two-step process of secure authentication and client-side encryption using the efficient security algorithm for power constrained IoT as detailed in chapter 4. The algorithm table: 'Device Provisioning Process Flow' and table 3 detail the provisioning sequence and the client-side encryption execution respectively.

5.3.2 The Device Provisioning Algorithm

Algorithm 2: Device Provisioning Process Flow

```
Initializing the IoT device and the ATECC608A secure element
Invoking device-cloud authentication leveraging the ATECC608A tamper-proof security
  keys
while Creation and registration of a Certificate Authority (CA) and the IoT device's
  security credentials do
    Create a Certificate Authority's root certificate
    ← Certificate
    Invoke the IoT device's certificate signing request to a certificate signer Certificate
      Authority
    sign the certificate signing request using the root certificate
    ← Certificate
    register the device's digital identity using the signed certificate.
    ← DeviceUniqueID
end
Connect the device to the IoT cloud by Via passing the network medium credentials to
  the WINC1500
```

Leveraging the capabilities of the tamper-proof secure element as detailed in 5.3.1 for secure authentication, the algorithm table 4 shows the sequence of communication exchanges, executed using the AWS command Line Interface (CLI) according to the experimental setup in 5.2.1. An

AWS account is first setup, accessed through the management console and programmatic access credentials configured as per the the setup detailed in 5.2.1. The SAMG55 microprocessor, the ATECC608A and the ATWINC1500 extension boards are then connected to the SAMG55 Xplain pro board as shown in Fig. 5.2, and the setup connected to the USB port of a Laptop computer, which then powers the IoT device. The device initializes and the tamper-proof functionalities of the ATECC608A is leveraged for setting up sessions with the respective AWS services consoles by means of invoking commands through the AWS CLI installed on the laptop computer. The AWS Lambda service provides server-less and event-driven possibilities for automatic execution of code snippets and management of related computing resources. This is utilized to create a role to facilitate an event-driven registration of the SAMG55 device on the AWS IoT core service. This instance of the AWS lambda function enables self identification of the device to the IoT service, creates a 'Thing' on the AWS IoT core which represents the SAMG55 device, as well as activate the device certificate. A Certificate Authority (CA) root certificate creation request is invoked, which returns a created root certificate. A device certificate signing request call is then made to a signer Certificate Authority (CA), following which the device certificate is signed using the initially created root certificate. This returns a signed certificate, which is then used to register the device's digital identity, and a unique device ID is returned. The lambda function enables the establishment of the device as a 'Thing' on the AWS IoT core which also supports other capabilities such as creating a shadow of the unique device on the cloud, which facilitates the ability to work on the 'Thing' even when it is not actively connected to the platform, and a synchronizing feature when the 'Thing' eventually becomes available. Finally, the ATWINC1500 is configured with the network media credentials which facilitates the subsequent automatic re-connection of the device to the IoT core. This useful feature supports the limited resources management by enabling the device to only connect as needed, while the device shadow service backs up through the provision of a continues virtual presence of the device.

5.4 Client-Side Encryption for Constrained IoT devices

client-side encryption is used to achieve encryption of data at the end of the IoT device before it is sent onto cloud. According to[131], the integrated circuits (ICs) deployed in IoT based infrastructures have strong constraints in terms of size, cost, power consumption and security. Unlike in desktop computers, tablets, and so on, IoT devices are unable to allocate considerable

memory and processing energy just for security functions [131]. The authors in [43] observed that to protect the confidentiality of data outsourced from IoT devices to the cloud, cryptographic mechanisms are usually employed to encrypt the data in such a way that only the user designated by the data owner can decrypt the data. They proposed a privacy-preserving data sharing scheme in cloud-assisted IoT which employs identity-based encryption and linear secret sharing that both preserves privacy of the IoT data pushed to the cloud as well as allow for flexible sharing of encrypted data between connected devices. The authors in [131] highlighted the main challenges identified in the deployments of IoT devices as resilience of the deployed infrastructure, confidentiality, integrity of exchanged of exchanged data, user privacy and authenticity among others. According to [132], since more devices are being programmed to exchange data autonomously in IoT deployments, the importance of security and authenticity of such transmitted data is very crucial and thus requiring strong security approaches to prevent both passive and active attackers. According to [131], the insurance of privacy and data protection remains a challenge in IoT deployments desiring of solutions. They discussed the challenges, implementation and future applications of Light Weight Cryptographic schemes in securing constrained IoT devices. Taking the constrained characteristic of the IoT devices into consideration, [133] proposed varying levels of security measures dependent on the confidentiality requirements of the data. [132] proposed a secure communication scheme for IoT devices which applies the Diffie-Hellman algorithm for authentication and uses the AES and Message Digest (MD)5 algorithms for encryption and validation respectively, of transferred data.

5.4.1 Low-cost Client-Side Encryption

Algorithm 3: Client-Side-Encryption Execution Flow

```
Message, Key
initialization of the counter i = 0 and Nbr = 2
Expand key to length: (block size) *Nbr + block size
STATE = message XORed with Key (Key whitening)
Invoke the round function:
while i < Nbr : do
    STATE = SubByte(STATE)
    STATE = ShiftRows(STATE)
    if i < Nbr : then
        STATE = MixColumn(STATE)
    end
    Invoke addRoundKey(STATE, NextRoundkey)
end
STATE as resulting Ciphertext
```

Hinging on the analysis of consequence and justification of complexity reduction of classical algorithms in tandem with IoT resource constraint as detailed in sections 3.3 through to 3.4 respectively, leveraging the features of the ATECC608A secure element detailed in chapter5.3.1 as a trade-off towards the mitigation of implementation attacks as detailed in chapter 3.3 and the provisioning of the IoT device as detailed in 5.3.2, the reduced round algorithm, is thus is utilized for the experimentation of client-side encryption. Furthermore, based on the results published in [105] and the experimentation data presented in table 5.1, the reduced round resource-efficient algorithm (RR2) -based on the Advanced Encryption Standard (AES) was utilized for experimenting a client-side encryption of 16bytes data. Based on the AES standard cipher, the reduced round algorithm executes the round function in two iterations using a total of 48bytes scheduled key as against the 176bytes of the standard AES, for every block of the plain text, following the process of key-whitening, initialization and the execution of the round function detailed in chapter4.3.1. The diagrammatic presentation of the reduced round function is as shown in Fig. 2.1 and as detailed by the algorithm table: 'Client-Side Encryption Execution Flow'.

Table 5.1: Encryption Times Data Generated from Laptop Implementations of CLEFIA Variants, AES128 and RR2

Instances	CLF-128enc	CLF-128dec	CLF-192enc	CLF-192dec	CLF-256enc	CLF-256dec	AES-128	RR2
1	1.995802	1.994848	1.994371	1.994133	3.960848	2.991915	1994.848	1994.371
2	2.025366	1.994133	2.991676	2.992868	1.99461	2.025366	1994.133	2991.676
3	0.991106	1.175642	2.175093	2.19059	2.99263	2.222776	1175.642	2175.093
4	2.000809	0.819445	1.994371	2.795458	2.019882	2.961159	819.445	1994.371
5	1.105785	1.994848	2.992153	1.994371	2.968073	2.017736	1994.848	2992.153
6	0.997782	0.997305	1.99461	2.991915	2.991676	2.001524	997.305	1994.61
7	2.020597	1.994848	1.994133	2.173424	2.993107	2.115488	1994.848	1994.133
8	0.972271	0.997305	2.992868	1.816511	1.993656	1.998425	997.305	2992.868
9	2.027035	1.99461	1.99461	2.99263	4.023552	2.991438	1994.61	1994.61
10	0.968218	1.129389	1.994848	1.99461	2.267838	1.990557	1129.389	1994.848
11	1.99151	1.862764	2.991915	2.991438	1.995564	2.985477	1862.764	2991.915
12	0.997543	0.997305	1.99461	1.99461	2.991676	2.0051	997.305	1994.61
13	2.160072	1.99461	2.992153	1.995564	2.958059	2.987623	1994.61	2992.153
14	0.997066	0.997782	1.995087	2.992153	3.026247	1.999378	997.782	1995.087
15	1.994371	1.994371	2.992392	1.994133	2.993584	2.991676	1994.371	2992.392
16	0.99802	0.997782	1.994371	2.992392	2.994299	1.961946	997.782	1994.371
17	0.997305	1.994371	2.993822	1.993656	1.992464	3.025055	1994.371	2993.822
18	1.99461	0.997543	2.017498	2.994776	2.99263	1.991987	997.543	2017.498

19	1.249075	1.994848	1.969337	1.991987	1.992226	3.139973	1994.848	1969.337
20	2.024889	0.996828	3.002882	1.996756	2.994537	1.814842	996.828	3002.882
21	1.000643	1.995087	1.983881	2.995253	1.95837	3.02887	1995.087	1983.881
22	1.992702	0.997543	3.000021	1.990318	3.020287	2.986193	997.543	3000.021
23	1.000166	1.995087	1.992226	1.993418	1.994133	2.120256	1995.087	1992.226
24	1.974583	0.99659	2.985477	2.187729	2.999306	2.868176	996.59	2985.477
25	0.983953	1.995802	1.995087	1.801729	1.995087	1.989603	1995.802	1995.087
26	2.163172	1.995564	1.995564	1.994848	2.990007	2.978086	1995.564	1995.564
27	0.999451	0.996351	1.994371	1.995087	1.96147	1.983404	996.351	1994.371
28	2.029181	1.993656	1.99461	1.994371	3.027439	2.030611	1993.656	1994.61
29	0.992298	1.99461	2.992392	2.181768	1.996517	2.979279	1994.61	2992.392
30	1.993895	0.997543	1.994133	1.810551	2.990961	1.998186	997.543	1994.133
31	0.998974	1.994371	1.99461	2.989054	1.999855	2.996922	1994.371	1994.61
32	2.110958	2.993107	1.99461	1.995325	2.988577	1.995802	2993.107	1994.61
33	1.033068	2.990007	1.99461	1.993895	1.991272	3.001451	2990.007	1994.61
34	1.960278	2.993822	1.994848	2.992392	2.993822	1.984358	2993.822	1994.848
35	1.02973	1.027107	1.99461	1.994848	1.995564	2.958775	1027.107	1994.61
36	1.997471	1.996756	1.99461	1.994848	2.989769	2.028942	1996.756	1994.61
37	0.996828	0.995159	1.994371	2.991915	1.994371	2.991676	995.159	1994.371
38	1.994848	1.996517	1.995325	1.99461	2.993345	1.99461	1996.517	1995.325
39	1.107454	0.995636	2.001762	1.994848	1.999378	2.992392	995.636	2001.762
40	2.029896	1.994371	1.989603	1.994371	2.991676	1.998425	1994.371	1989.603
41	0.996828	1.996994	1.995087	2.992153	2.986908	2.988577	1996.994	1995.087
42	1.961708	0.965118	1.99461	1.994371	1.994133	1.995564	965.118	1994.61
43	1.994371	2.027035	3.026009	1.995325	2.991438	2.957106	2027.035	3026.009
44	0.997066	1.994848	1.997232	2.991915	1.99604	2.02775	1994.848	1997.232
45	2.092361	0.994921	1.95837	1.995325	2.991438	2.994061	994.921	1958.37
46	0.997782	1.997232	2.030611	1.993895	1.960278	1.994371	1997.232	2030.611
47	1.995325	0.997066	1.993418	2.99263	3.020763	2.990484	997.066	1993.418
48	2.024651	1.99461	2.993584	1.99461	2.005339	1.995564	1994.61	2993.584
49	1.004696	1.996517	1.99461	2.992392	2.988338	2.993107	1996.517	1994.61
50	1.957893	1.000643	1.958609	1.993895	1.995325	2.995253	1000.643	1958.609
51	0.997305	1.989841	2.029657	1.994848	1.993656	1.99008	1989.841	2029.657
52	2.130508	0.964642	2.129078	1.99461	2.956629	2.992868	964.642	2129.078
53	1.021624	2.027273	2.986431	2.992392	2.029419	1.994133	2027.273	2986.431
54	1.997948	1.995802	2.00057	1.994848	2.991199	2.991676	1995.802	2000.57
55	0.994444	0.963926	1.995564	2.992392	1.995087	1.995087	963.926	1995.564
56	2.001524	2.027035	1.994848	2.210855	2.957821	2.992868	2027.035	1994.848
57	2.014399	1.993418	2.997875	1.99461	3.127575	1.994133	1993.418	2997.875
58	1.121283	0.998497	1.989126	1.995564	1.890182	2.993345	998.497	1989.126

59	1.99461	1.995325	1.99461	2.991676	2.997398	1.992941	1995.325	1994.61
60	0.966311	1.992226	2.956629	2.180576	1.993895	2.991676	1992.226	2956.629
61	2.02632	0.999689	1.99461	1.995087	3.070593	1.996517	999.689	1994.61
62	0.966787	1.992702	2.023697	2.991199	1.914978	2.991199	1992.702	2023.697
63	2.025127	1.002312	2.994776	2.993584	2.992868	2.032518	1002.312	2994.776
64	1.116276	1.992941	1.99914	1.994371	1.998663	2.995253	1992.941	1999.14
65	2.027035	1.991034	2.116442	2.991676	3.073692	1.990557	1991.034	2116.442
66	0.966787	0.999928	1.995087	1.994371	1.910686	2.993345	999.928	1995.087
67	1.994133	1.992941	2.999783	2.991676	2.993584	1.992941	1992.941	2999.783
68	1.994848	1.994371	1.992941	1.995325	1.990318	2.99263	1994.371	1992.941
69	0.998259	0.966311	2.000093	1.994371	3.141642	2.99263	966.311	2000.093
70	1.993656	2.029419	2.95639	2.992153	1.846075	1.994371	2029.419	2956.39
71	1.162052	1.11413	2.035856	1.994371	2.992153	2.991438	1114.13	2035.856
72	1.995802	1.996279	1.984596	2.992392	1.998425	1.995087	1996.279	1984.596
73	0.998259	2.00367	2.994776	1.995087	3.06344	2.998114	2003.67	2994.776
74	1.993895	1.985312	1.99151	2.991915	2.916336	1.988411	1985.312	1991.51
75	1.99461	0.997305	1.994848	1.994133	1.994371	2.986193	997.305	1994.848
76	0.997305	2.120495	2.995491	1.99461	2.993107	2.001286	2120.495	2995.491
77	2.112865	1.896143	1.99461	2.992392	2.125502	2.993107	1896.143	1994.61
78	1.028061	0.996113	2.132654	1.994848	2.860546	1.959085	996.113	2132.654
79	1.995087	2.001762	2.958059	2.991438	1.996517	3.154516	2001.762	2958.059
80	1.70207	1.960993	2.029181	1.995087	2.989531	1.996994	1960.993	2029.181
81	0.99349	0.996351	1.961708	1.995802	2.057314	2.987146	996.351	1961.708
82	2.027512	1.994371	3.021717	2.990961	2.924681	1.997709	1994.371	3021.717
83	0.963449	0.997543	1.991272	1.994371	1.997709	2.995014	997.543	1991.272
84	2.153873	1.99461	2.001286	2.992153	2.96092	1.989126	1994.61	2001.286
85	0.997305	0.997543	2.958536	1.994848	3.027678	3.000259	997.543	2958.536
86	2.027035	1.99461	1.993179	1.995087	1.996756	1.986742	1994.61	1993.179
87	1.994848	0.997305	1.995087	2.991438	2.953053	2.992868	997.305	1995.087
88	0.99802	1.995087	3.023863	2.124548	3.022194	1.995087	1995.087	3023.863
89	2.138138	0.997066	1.996517	2.862692	3.000021	2.992153	997.066	1996.517
90	1.028061	1.99461	2.093554	1.994371	1.991272	1.993895	1994.61	2093.554
91	1.995087	1.99461	2.894878	2.991915	2.993345	2.991915	1994.61	2894.878
92	0.967979	0.997782	1.993895	2.157688	1.998186	1.960754	997.782	1993.895
93	2.036572	1.994371	1.995802	2.861738	2.987146	3.026485	1994.371	1995.802
94	1.989126	0.997305	3.078222	1.992464	2.994061	2.991915	997.305	3078.222
95	0.998497	1.994848	1.907587	2.965212	1.996756	2.131701	1994.848	1907.587
96	2.020597	1.028776	1.998901	3.021002	2.987385	1.997709	1028.776	1998.901
97	0.998974	0.999689	2.988338	1.961946	1.998186	2.989054	999.689	2988.338
98	1.993418	1.992226	1.995087	1.994371	2.988815	1.995564	1992.226	1995.087

99	2.001047	0.967979	1.993895	2.019882	1.995564	3.122568	967.979	1993.895
100	1.019716	2.026558	2.991676	1.996279	1.997948	1.985312	2026.558	2991.676

5.5 Chapter Summary

With not much information available on the ease of secure provisioning of IoT devices, couple with the inherent challenges of these devices in encrypting data before transmission to cloud platforms due to their constrained nature, this chapter details the use of an efficient algorithm based on the AES for the client-side encryption using a SAMG55 microprocessor, as well as the secure provisioning of the IoT device onto AWS IoT core, an IoT cloud platform of a leading cloud services provider know as the Amazon Web Services (AWS). Starting with an introduction of the concept of IoT device provisioning and client-side encryption in 5.1, an overview of the experiments on which the content of the chapter is based, and the experimental setup was presented in section 5.2 and 5.2.1 respectively. Section 5.3 details the authentication of the sample IoT device, the SAMG55 microprocessor. The details of the secure element which facilitates secure authentication and serves as the trade-off for complexity reduction in the formulation of the algorithm is presented in chapter5.3.1, following which the device provisioning algorithm is presented and the process detailed in 5.3.2. An overview of client-side encryption for constrained IoT devices together with the reduced round algorithm for low cost client-side encryption were presented in sections 5.4 and 5.4.1 respectively. The experimentation data detailing the comparison of a lightweight algorithm, the AES and the reduced round algorithm presented in table 5.1 and section 5.5 summarizes the various sections and content of the chapter.

Chapter 6

Analysis of Results and Contribution

6.1 Introduction

The results herein presented follow from the mathematical background of this work as presented in chapter3.2, including the application of the concepts of basic algebraic structures, Galois fields, averages, percentages and co-variances. The experiments and experimental setups presented in chapters/sections 4.5.1 and 5.2.1 hinge on this background, following which the experimentation data in tables: 4.3, 4.4, 4.5 and 5.1 are obtained. As detailed in the various experimentation setups in the preceding chapters, Jupyter notebook was utilized in the plotting of the experimentation data obtained. This chapter presents the results of the various experimentation and highlights the contributions of the research work. Broadly grouped in four sections, an introduction, evaluation of the implementation and analysis of results, the contributions of this research work and conclusion of the chapter respectively, are presented.

6.2 Implementation Evaluation and Analysis of Results

This section presents the results of the experiments which data is contained in the tables: 4.3, 4.4, 4.5 and 5.1. With the experiments aimed at implementing and comparing the efficient algo-

rithm to classical algorithms and lightweight algorithms alike, an analysis of the computational complexity of the Efficient algorithms for Constrained IoT devices is first given as follows:

6.2.1 Analysis of Computation Complexity of the Efficient Algorithm for Constrained IoT Devices

The measure of an algorithm's complexity is popularly by the use of the big $\mathcal{O}$ notation, which essentially is a mathematical notation that describes the limiting behavior of a function when the argument tends to a value or infinity [145]. According to [146], this is frequently used in the analysis of algorithms to describe an algorithm's usage of computational resources whereby: the worst case or average case running time or memory usage of an algorithm is often expressed as a function of the length of its input. The input sized of the Efficient Algorithm for Constrained IoT Devices is a fixed 16bytes block size of input, and thus of $\mathcal{O}(1)$ computational complexity in terms of the big $\mathcal{O}$ notation with respect to the input size and $\mathcal{O}(m)$, with a growing message size m, as there would be m blocks to encrypt. As reviewed in 2.5 the running time of an encryption algorithm ultimately impacts on the power and processing resources of the constrained IoT devices . In line with this consequence, an analysis of the computation complexity of the Efficient Algorithm for Constrained IoT Devices is presented, such that the execution time-difference between the standard AES and variants of the proposed algorithm for constrained IoT devices is measured.

First, a measure of the notion of growing complexity is investigated, using the variants of the standard AES algorithm. As such, the plain text which is a single 16bytes of message, the factors of experimentation, as well as the platforms of experimentation remain fixed, while the encryption completion times of using the AES128, AES192 and AES256 as detailed in table 4.3 are compared. Fig. 6.1 shows the graphical plot of these encryption completion times of the aforementioned variants of the standard AES algorithm. The x-axis plots the one hundred encryption instances considered as evidenced by the data in table 4.3, while the vertical or y-axis of the figure shows the average encryption time in one thousand iterations (as per the concept of averages detailed in 3.2.2), per encryption instance. As visually shown by the legend included in Fig. 6.1, this average encryption completion times increase with respect to increase in the requirements of the round function, in terms of key length and the number of rounds of the cipher respectively, as detailed in chapter2.4. Also, as an application of the concept of percentages as detailed in sections: 3.2.2,

the experimental setup in 4.5.1 and the experimentation data in table 4.3 reveals that the AES-128 is 10% and 16% cheaper than the AES-192 and AES-256 respectively in encrypting the single block (sixteen bytes) of plain text considered.

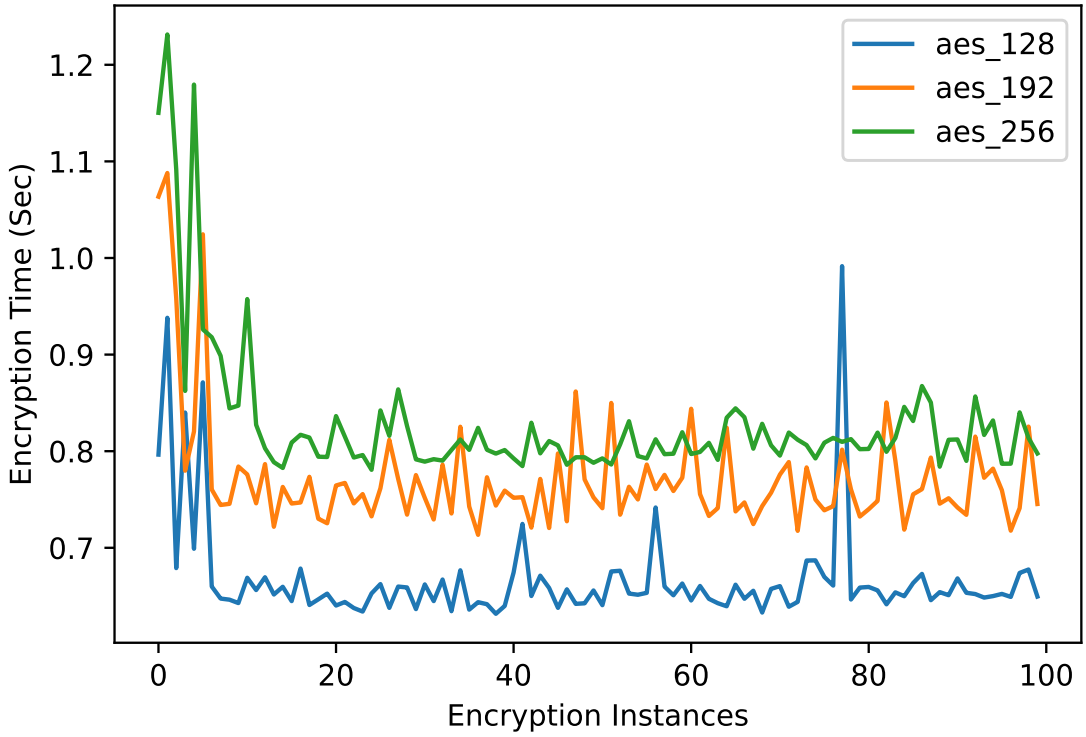

Figure 6.1: Averagely timed encryption instances of standard AES Key lengths with KB message size. Each encryption is timed at an average of 1000 iterations for the standard AES variants. The spiky encryption times around the initial encryption instances is suggestive amount of computational work that accounts for the key whitening stage of the cipher where the sub-keys for the round function ad also produced for the respective encryption iterations.

However, as indicated by the title of table 4.3, the experimentation data is obtained by use of a laptop computer for the implementation of the algorithms considered. Juxtaposing the realities of resource constrain in IoT devices as presented in chapter/chapter2.4 and based upon which the unsuitability of classical algorithms for IoT devices was presented in 2.5, a measure of the impact of this difference in resource availability between a resource sufficient device such as a laptop and a typical IoT device is necessitated. The results of this bit of the experimentation is presented as a comparative analysis of the IoT devices' constraints in section 6.2.2, such that the same experiments, on the same set of algorithms and following the same setup as detailed

in 4.5.1 was repeated using a resource a constrained device as evidenced by the experimentation in table 4.4. Also juxtaposing the varying level of constrain in different constrained IoT devices, establishing the precise case for the platform of experimentation is deemed a necessary addition. The level of resource constrains in ARM cortex A or R series of processors for example differ from the cortex M series and so, establishing and example case for the ARM cortex M series of constrained devices (as the experimentation platform) is necessitated.

6.2.2 Comparative analysis of IoT Devices' Constraints

As an experimentation of the consequence of resource constraint for the SAMG55 constrained device, a comparative analysis of the encryption completion time of the AES-128 and the reduced round cipher is presented; by comparing the implementations on a laptop computer (64-bits operating system, 8GB RAM and x64-based intel processor (core i5) and a Samg55 32-bit cortex M4 microprocessor which features 512Kbytes of Flash and 164Kbytes of RAM. Further experimentation verification was repeated on: An M1 chip MacBook computer running the macOS version 12.6.1 and Memory of 16GB and 8 core of CPU, and also on a compute optimised Elastic Compute Cloud instance (c4.4xLarge) hosted in the eu-west-2 region of the AWS cloud, running the Amazon Linux2 machine image (OS) and equipped with 30GB of memory and 16 virtual CPU cores. According to the data-sheet [18], the SAMG55 devices are general-purpose low-power micro-controllers that are suitable for wide range of IoT deployments. Firstly, the PC and SAMG55 implementations of the standard AES algorithm are compared. Fig. 6.2 shows a graphical representation of the difference in the completion time of encrypting one block of data on the laptop in comparison to the SamG55 micro-controller using the standard AES-128 algorithm. Consequent upon the constraint of computing resources of the SAMG55 device, our experimentation shows a 657% increase in the encryption completion time using the SAMG55 device in comparison to the encryption completion time using a PC, the constants being the message size, key-length, and encryption cipher while the variables are the computing resources available to the SAMG55 and the PC respectively. This represents a typical implication of the constraint imposed on an IoT device in terms of scarce computing resources and hence, the need for more efficient encryption algorithms in the IoT landscape without compromise to security.

The SAMG55 implementation of the reduced round algorithm however shows a 331% increase in the encryption completion time in comparison to the PC implementation. In contrast to the

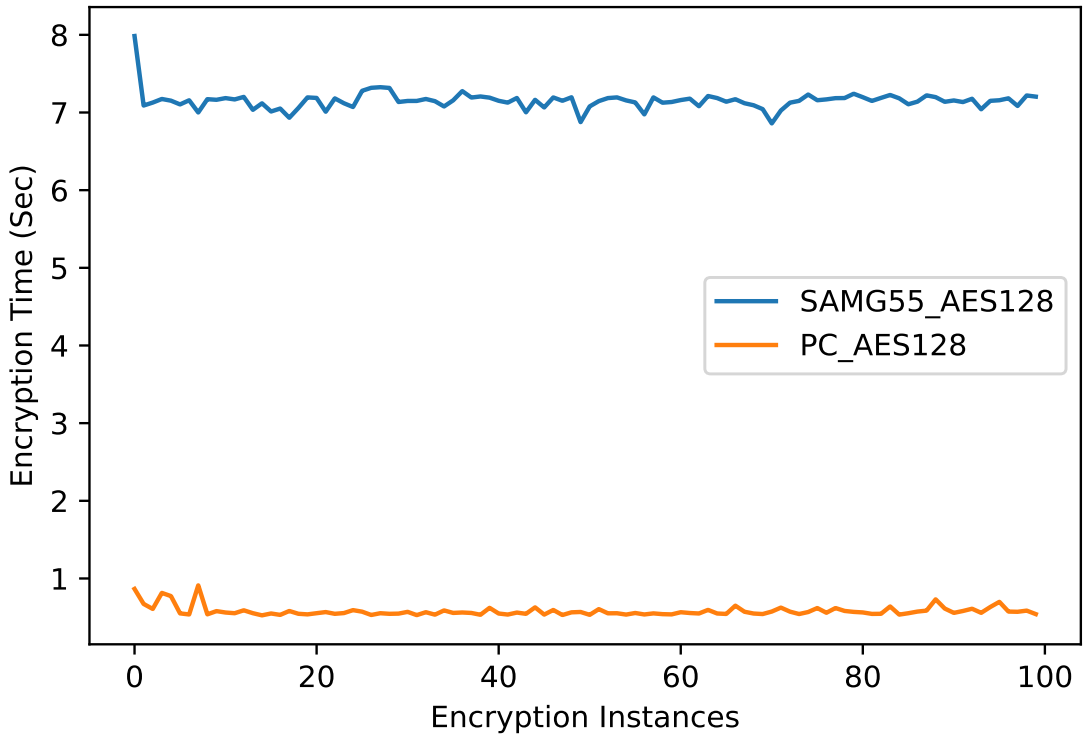

Figure 6.2: PC and SAMG55 implementation of the standard AES-128 algorithm, showing a comparison of the PC & IoT device resource differences.

PC and SAMG55 comparison of the encryption completion time, the reduced round algorithm shows a 50% reduction in the encryption completion time in favour of the resource constrain of the IoT device, by the metrics of analysis detailed in 3.2.2 and as per the experimental setup detailed in 4.5.1. Similar to the comparative analysis for the AES-128, the constants remain the message size (16bytes), key size and cipher, while the variables remain the difference in the availability of computing resources on the PC and the SAMG55 device. Fig. 6.3 shows the plot of the experimentation data.

6.2.3 Comparison and Analysis of CLEFIA, AES and the Reduced Round Algorithm

Consequent upon the impact of resource constraints on a typical IoT devices as shown by the results in section 6.2.2, further experimentation explored the viability of reducing the complexities of the known algorithms thus. This served as a bases for the reduction of the complexity of the classical AES as detailed in the cryptanalytic overview of the consequence of complexity

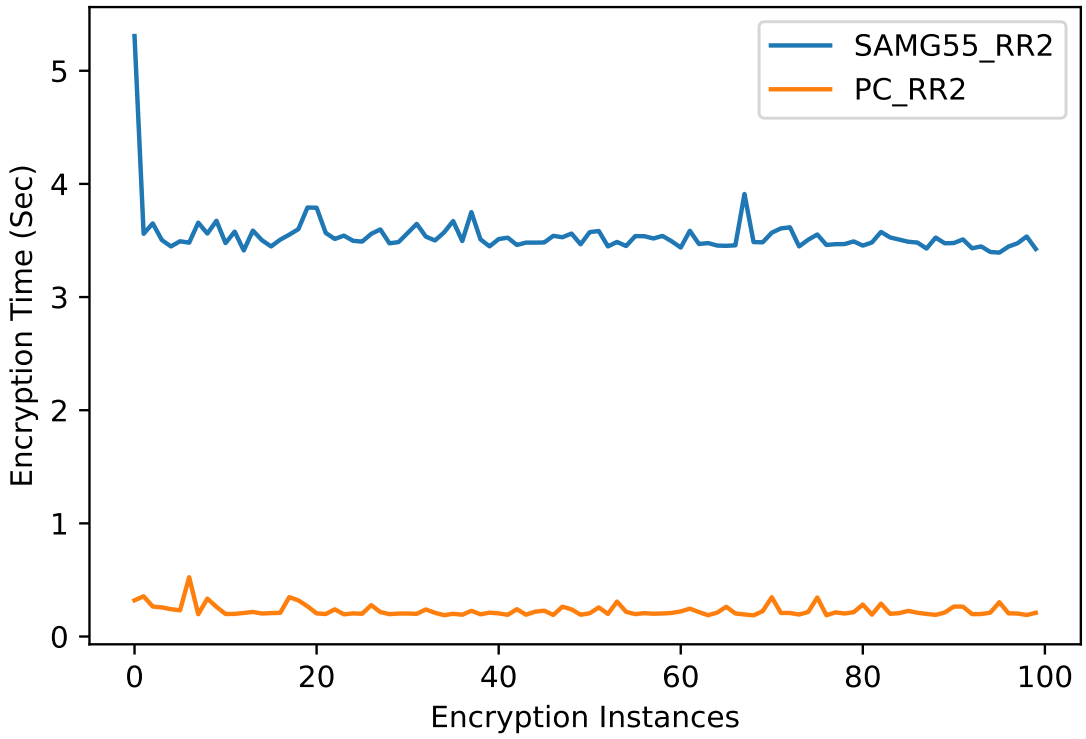

Figure 6.3: PC and SAMG55 implementations of the Reduced Round Algorithm, showing a PC and IoT device resource differences with respect to the reduced round algorithm and with a KB message size.

reduction in chapter3.3 through to 3.3.2. Accordingly, the experimentation of the reduction of the classical algorithm to four rounds and further to two rounds follows the justification for the round reduction and security trade-off detailed in chapter3.4. Both tables 4.3 and 4.4 contain data obtained from this experimentation, but on the two distinct platform: laptop computer and the SAMG55 microprocessor as detailed in section 6.2.1. In a similar fashion to the comparison of the complexities of the least variant of the AES: AES128 to those of AES192 and the most complex:AES256 as contained in section 6.2.1 and shown in Fig. 6.1, the implementations of the reduced round variant to four and two named as: RR4 and RR2 respectively, as shown in tables 4.3 and 4.4 compare the complexities of the reduced round variants to the AES128, which is the least complex in the classical consideration but most complex when compared to RR4 and RR2, as evidenced by experimental data. Fig. 6.4 shows the plot of this comparison, following the same factors of analysis detailed in 3.2.2 and as detailed in the experimental setup. While the AES-128 is 10% and 16% cheaper than the AES-192 and AES-256 respectively in encrypting a single block (sixteen bytes) of plain text, the reduced algorithms are shown to be 27% and 35%

cheaper than the standard AES-128 for the RR4 and RR2 implementations respectively.

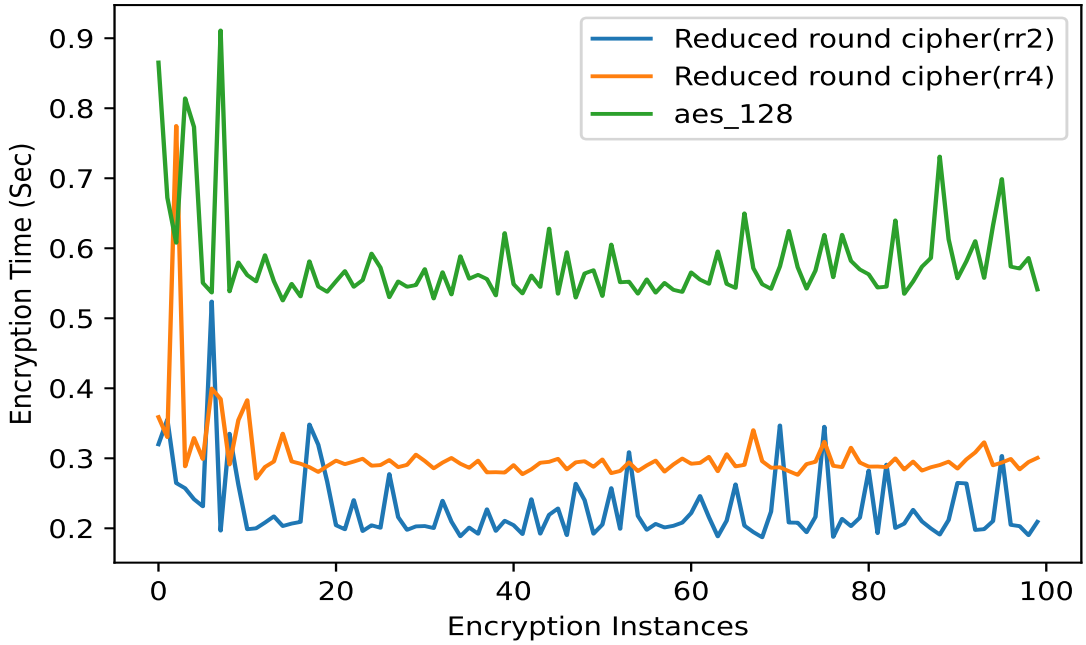

Figure 6.4: A comparison of the Reduced Round Cipher variants: (rr2 & rr4) and the standard AES-128 cipher, with a KB message size.

The AES128 serves as a reference point in the following analysis because while it is the least complex of the standard variants of the classical AES against the most complex: AES256, the RR2 which is in turn, the least complex of the reduce round variants is compared against the AES128 as the most complex, when compared with the RR2 and RR4. The analysis also reveals the rate of change in complexity between the RR2 and the AES128 to be higher than the rate of change in complexity between the AES128 and AES256. Precisely degree of complexity reduction measured by the complexity difference between the AES-128 and RR2 is obtained to be higher than the complexity difference between the AES-256 and the AES-128 by 11%, which is indicative of a higher efficiency in the latter than the former. This would be particularly useful in IoT applications and deployments that support the use of different algorithms on a need to use bases as according to [147], resources sharing mechanism in the IoT domain where resource constrained IoT devices can offload computationally intensive resources to resource-rich ones in order to achieve high quality of experience. A graphical plot of the rate of change in complexity between the RR2 and AES128, and the rate of change in complexity between the AES128 and AES256 is as shown Fig. 6.6.

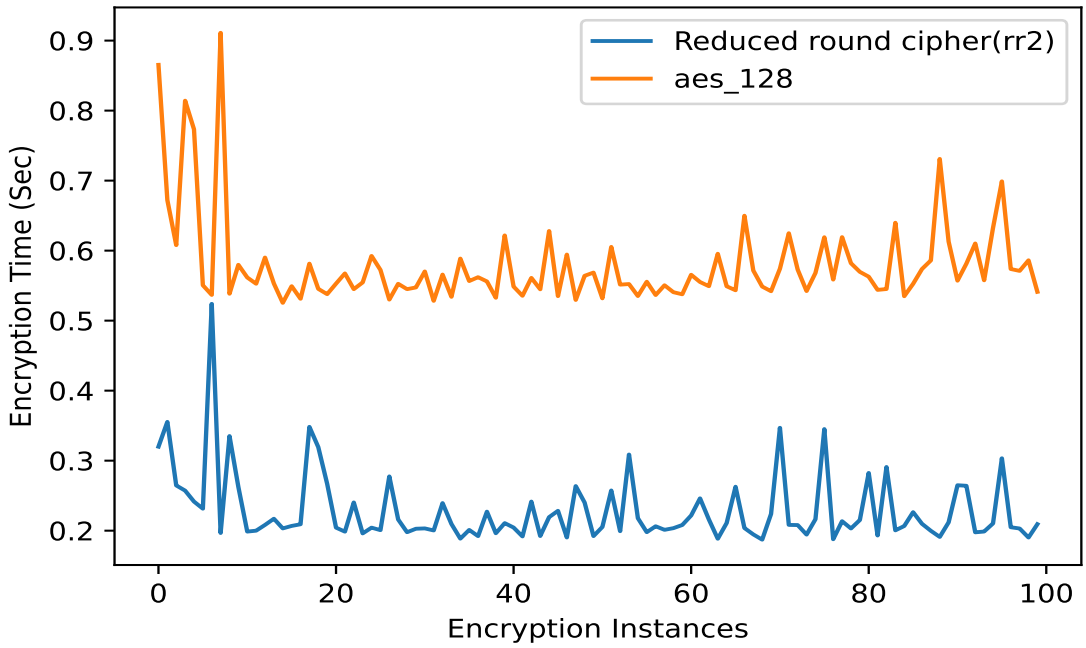

Figure 6.5: A comparison of the Reduced Round Cipher (rr2) and the standard AES-128 cipher, using a KB message size.

Furthermore, a comparison of the reduced round variants is also experimented against clefia as a choice lightweight algorithm, based on the comparison of the AES, Clefia and the reduced round algorithm detailed in chapter4.4, and in terms of the algebraic structures and constructions, as well as the lightweight versus efficient algorithm detailed in 4.4.1 and 4.4.2. A comparison in terms of common metrics, the difference between lightweight algorithm and the AES on which the Efficient algorithm is based is presented. The rate of change of complexity between the key variants of CLEFIA is compared to that of the standard AES. The encryption completion times of CLEFIA is also compared to that of the efficient algorithm.

Bench-marking the AES which is currently the most widely-used algorithm in the IoT landscape according to the investigations detailed in [105], an implementation and analysis of the lightweight algorithm and the efficient security algorithm with respect to the characteristics of the AES is hereby presented. Juxtaposing the challenges of directly comparing different technologies, this was done by considering the algorithms with higher compatibility, instead of the ones adjudged to be of lower compatibility. The compatibility factor was determined in terms of the structure of the cipher: being widely of either a Feistel structure as in the case of DES and CLEFIA, or of Substitution Permutation Network (SPN) structure as in the case of AES,

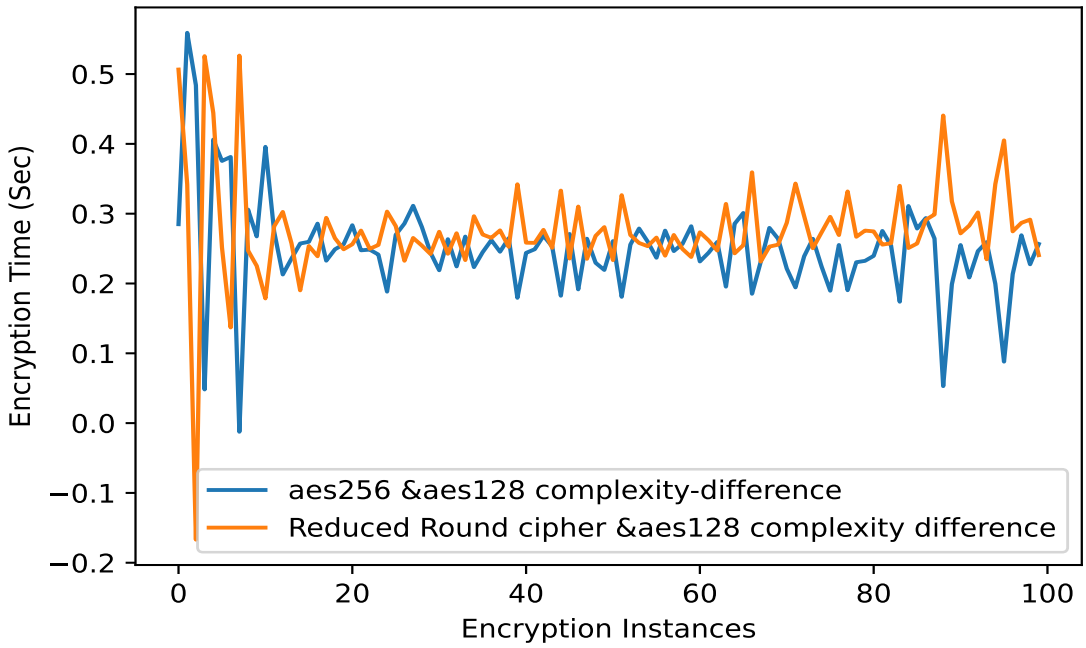

Figure 6.6: A comparison of the complexity difference (with respect to encryption completion time) between AES-256 AES-128; AES-128 the reduced round algorithm.

the key-length variants of the cipher: in terms of n-bits security defined by 2^n where n is the number of bits of the key variant, the sample message size: adjudged by block size for block ciphers as in the cases of DES, AES and CLEFIA. With respect to the aforementioned factors of compatibility, CLEFIA holds a higher compatibility with the AES in terms of key lengths; such that both the AES and CLEFIA have three key variants of 128, 192 and 256 bits. Although both the AES and CLEFIA are similar in the message block-sizes being of 16bytes in addition to the compatibility with respect to key lengths, they differ in the structure of the round functions -which constitute the algebraic constructions and security of the algorithms. While the AES operates 10, 12 and 14 rounds for the key lengths 128, 192 and 256 bits respectively, the number of rounds for CLEFIA are: 18, 22 and 26 for key lengths 128, 192 and 256 respectively and thus, the selection of CLEFIA for the comparative analysis with the Efficient Algorithm. Since the compatibility of the AES and CLEFIA extends to the efficient algorithm, the encryption completion times of the efficient algorithm was experimented and compared to that of CLEFIA. Following the experimental setup as detailed in chapter 4.5.1, the implementation of the three key variants of CLEFIA is first experimented as did the variants of the AES in section 6.2.1. Fig. 6.7 shows plots of the CLEFIA standard variants, while Fig 6.8 shows how CLEFIA compares

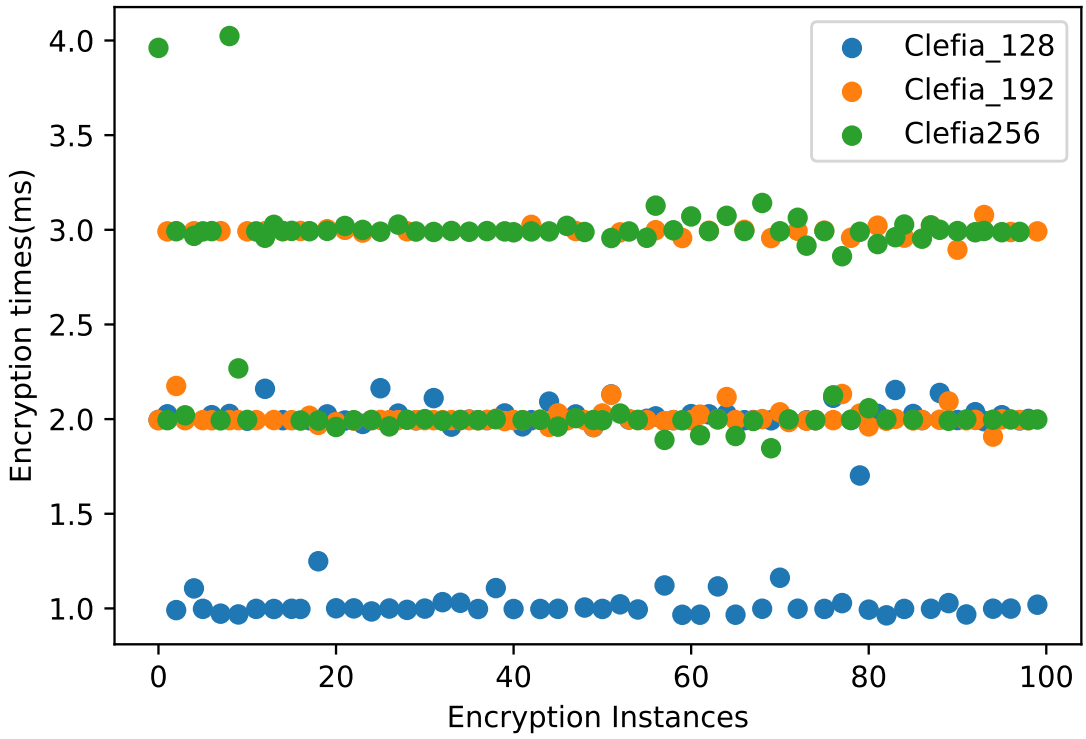

Figure 6.7: Averagely timed encryption instances of standard Clefia Key lengths with KB message size. Each encryption is timed at an average of 1000 iterations for the standard Clefia variants

to the reduced efficient algorithm.

The rate of change of complexity between the least complex variant of CLEFIA and most complex variant as per CLEFIA128 and CLEFIA256 shown in Fig. 6.7 was also experimented and compared against the rate of change of complexity in the classical AES variants. Similar to the comparison of the rate of change in the efficient algorithm versus the AES as shown in 6.6, the rate of change in complexity of CLEFIA is also higher than the rate of change in the standard AES variants as shown in Fig. 6.9.

6.2.4 Low Cost Client-side Encryption

Hinging on the review of related work as presented in 2.8.2 and the work on client-side encryption for constrained IoT devices as presented in chapter 5.4, low cost client-side encryption aims to utilize the efficient algorithm, made of the reduced round cipher for encrypting IoT device data before onward transmission for storage, following the provisioning process of the associated IoT device. The comparison and analysis of experimentation results for clefia, the AES and reduced

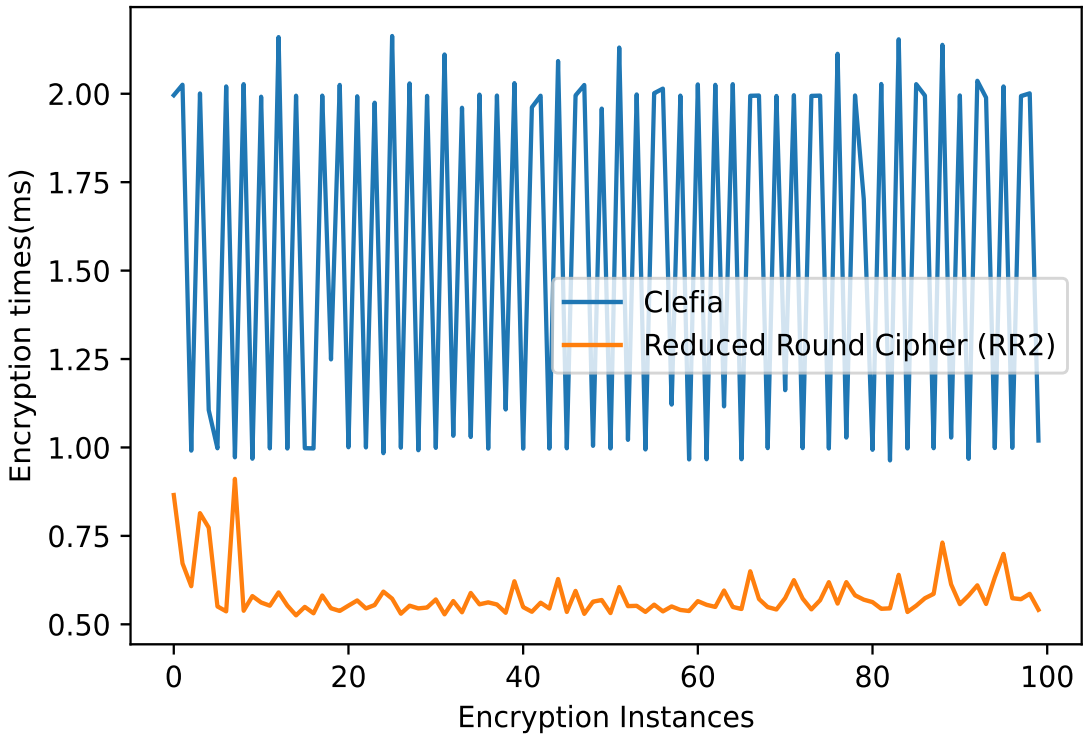

Figure 6.8: Comparison of lightweight CLEFIA and the Reduced Round Cipher (RR2)

round algorithm is as detailed in 6.2.3. These results form the bases for the reduced round cipher as the choice for low cost client-side encryption thus, and section 6.2.4.1 details the client-side encryption execution flow using the reduced round cipher and subsequently, the results and analysis of its usage for the experimentation of low cost client-side encryption and secure provisioning.

6.2.4.1 The Reduced Round Cipher

The algorithm table 3 presented in chapter 5.4.1 together with the diagrammatic representation of the reduced round cipher as presented in Fig. 6.10 outlines the client-side encryption sequence. In [133], varying levels of security measures dependent on the confidentiality requirements of the data is proposed. Accordingly, and as detailed in chapter5.4.1 the reduced round cipher considered the number of rounds: four and two, as presented in Fig. 6.10 and the algorithm table 3 respectively, which are independently usable for client-side encryption of IoT device data in line with the IoT security model proposed in [133]. Also as evidenced by the experimental data presented in tables 4.3 and 4.4, the encryption completion times of the aforementioned

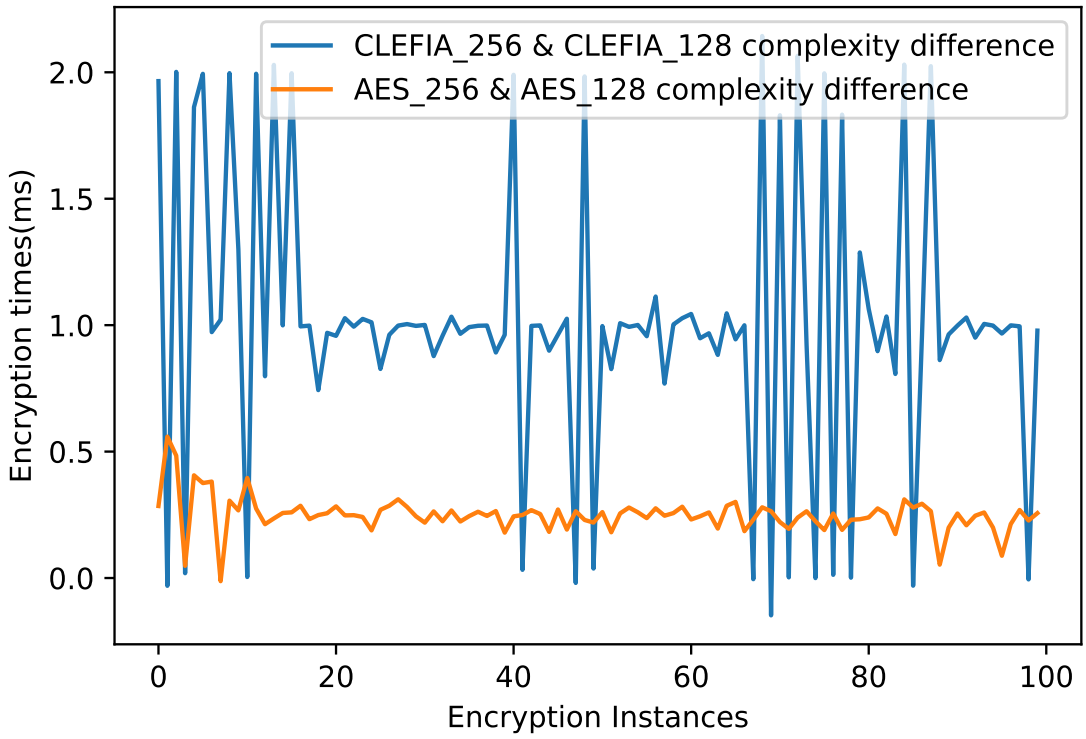

Figure 6.9: A measure of the rate of complexity between AES256 & AES128, vs CLEFIA256 & CLEFIA128

rounds were logged for comparative analysis of computational complexity as detailed in section 6.2.1, through to section 6.2.3. The results, as presented in section 6.2.3 shows variants of the reduced round cipher to be 27% and 35% cheaper than the standard AES-128 for the RR4 and RR2 implementations respectively, and thus leveraged for low cost client-side encryption and in compatibility with the security model proposed by the authors in [133].

Depending on the security requirement juxtaposed by the required number of rounds, the third sequence in the client-side encryption execution flow as shown in the algorithm table 3 is executed, whereby; the length of the encryption key is expanded accordingly to equal the product of the message block size and the choice number of rounds, summed with the length of the message block size. Moreover, the standard AES from which the reduced round cipher is derived, the lightweight clefia and both RR2 and RR4 have a fixed sixteen bytes of message block size as detailed in chapter 4.3.1, consequently yielding key lengths of 80bytes and 48bytes for RR4 and RR2 respectively, while contributing to efficiency with respect to the analysis of key related complexity as reviewed in 2.4.1. As detailed in the client-side encryption execution

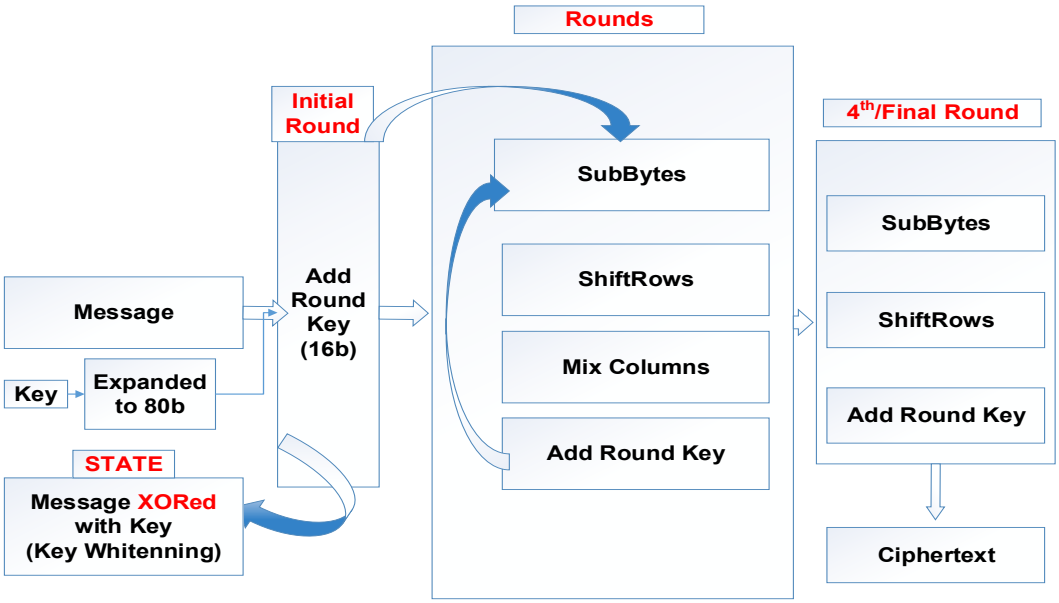

Figure 6.10: Diagrammatic structure of the Reduced Round Algorithm

flow and algorithm table 3, the round function is invoked following the required number of round selection and corresponding key expansion, and the encryption process in 4.3.1 is accomplished.

While 27% and 35% account for the percentage efficiency of the variants of the reduced round cipher as detailed in 6.2.3, an analysis of co-variance of the reduced round variants, together with an Avalanche effect analysis is deemed a necessary consolidation for the attributes that make the reduced round cipher a plausible candidate for the low cost client side encryption. Section 6.2.5 presents the results of the covariance and Avalanche effect analysis accordingly.

6.2.5 Covariance Table and Avalanche Effect

Further to the experimentation results of encryption completion times and based on which the efficient algorithm is utilized for client-side encryption, a covariance analysis of the experimentation data consolidates the theory of complexity associated with encryption keys and the structure of the cipher, with positive value results, is presented in this section. This was done through the computation of the covariance matrices of pairs of key lengths to buttress the notion of growing complexities associated with the key lengths as discussed in chapter2.4.1, and demonstrated by the result of positive covariance values as shown in table 6.1, in addition to the reduced algorithm

being 27% and 35% cheaper than the standard AES as shown in Fig. 6.4.

Table 6.1: Covariances Comparison of the Reduced Round Algorithm & the Standard AES key Lengths

Cov(AES-128, AES-192)	Cov(AES-192, AES256)	Cov(AES-128 AES256)	Cov(rr4-AES128, AES128)	Cov(rr2-AES128, rr4-AES128)
0.00202746	0.0033184	0.00202	0.00065489	0.00055592

The avalanche effect properties of a cipher is measured in terms of the rate of change in the cipher text with respect to changes in the plain text. According to [104], the ideal avalanche effect for a good cipher should be at least 50%, which means that a single change in the plain text should alter the outcome of the cipher text by at least 50% when encrypted by the cipher. First, 6.2 shows the sample plain-text and cipher-text for the distinct variants of the low-cost algorithm, with which the results detailed in tables 6.3, 6.4 and 6.5 are obtained.

Table 6.2: Reduced Round Algorithm Plain-text and corresponding Cipher-text

RR Variant	Message	Ciphertext
RR2	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 108, 63]	[138, 243, 252, 95, 61, 75, 45, 57, 161, 83, 125, 88, 154, 94, 31, 120]
RR3	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 108, 63]	[209, 22, 245, 185, 40, 157, 157, 106, 74, 172, 60, 74, 114, 85, 56, 185]
RR4	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 108, 63]	[118, 142, 29, 129, 217, 196, 51, 217, 1, 187, 202, 118, 155, 183, 109, 5]

For each of 6.3, 6.4 and 6.5, a byte level flip is performed on the block of the sample plain-text, and the observed avalanche effect is recorded. The block bytes which reoccur in the output cipher-text block when contrasted against the corresponding original cipher-text in 6.2 is presented in red in tables 6.3, 6.4 and 6.5. The arithmetic average of the five instances of avalanche effect experiment per variant of the low-cost algorithm is then computed using $\sum_{i=1}^{5} x_i$, where x is an instance of the avalanche effect experiment, which yields 23%, 98% and 93%for 6.3, 6.4 and 6.5 respectively for the distinct variants of the low-cost algorithm.

Table 6.3: RR2 Avalanche Effect Log for the Five Byte Flip Instances

Byte flip	Message	Ciphertext
1	[87, 120, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 108, 63]	[138, 243, 252, 60, 61, 75, 136, 57, 161, 142, 125, 88, 107, 94, 31, 120]
2	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 120, 63]	[138, 15, 252, 95, 115, 75, 45, 57, 161, 83, 125, 75, 154, 94, 153, 120]
3	[87, 104, 97, 120, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 108, 63]	[120138, 108, 252, 95, 28, 75, 45, 57, 161, 83, 125, 85, 154, 94, 14, 120]
4	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 120, 114, 101, 101, 108, 63]	[254, 243, 252, 95, 61, 75, 45, 193, 161, 83, 200, 88, 154, 26, 31, 120]
5	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 120, 101, 108, 63]	[138, 243, 252, 2, 61, 75, 233, 57, 161, 204, 125, 88, 221, 94, 31, 120]

Table 6.4: RR3 Avalanche Effect Log for the Five Byte Flip Instances

Byte flip	Message	Ciphertext
1	[87, 120, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 108, 63]	[79, 63, 93, 127, 26, 253, 55, 48, 163, 29, 153, 10, 70, 194, 85, 170]
2	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 120, 63]	[239, 249, 61, 174, 22, 74, 78, 132, 104, 215, 179, 185, 163, 112, 246, 135]
3	[87, 104, 97, 120, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 108, 63]	[81, 139, 48, 212, 99, 101, 87, 90, 93, 66, 132, 69, 118, 59, 160, 78]
4	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 120, 114, 101, 101, 108, 63]	[46, 130, 149, 228, 28, 203, 159, 26, 252, 79, 28, 97, 42, 189, 38, 97]
5	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 120, 101, 108, 63]	[237, 154, 223, 145, 162, 240, 61, 201, 130, 93, 211, 210, 8, 109, 9, 178]

Table 6.5: RR4 Avalanche Effect Log for the Five Byte Flip Instances

Byte flip	Message	Ciphertext
1	[87, 120, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 108, 63]	[219, 111, 183, 151, 108, 67, 54, 0, 146, 174, 27, 187, 120, 11, 96, 1]
2	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 120, 63]	[27, 190, 233, 46, 102, 113, 27, 196, 217, 148, 33, 61, 27, 132, 76, 228]
3	[87, 104, 97, 120, 32, 105, 115, 32, 97, 32, 99, 114, 101, 101, 108, 63]	[186, 45, 240, 227, 6, 249, 217, 248, 40, 237, 101, 103, 146, 33, 198, 157]
4	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 120, 114, 101, 101, 108, 63]	[150, 116, 145, 14, 239, 30, 192, 50, 243, 47, 167, 253, 185, 115, 60, 40]
5	[87, 104, 97, 116, 32, 105, 115, 32, 97, 32, 99, 114, 120, 101, 108, 63]	[254, 171, 72, 30, 27, 247, 181, 235, 160, 163, 205, 215, 1, 94, 21, 117]

6.2.6 Secure Provisioning

Algorithm 4: Device Provisioning Process Flow

1 Initializing the IoT device and the ATECC608A secure element

2 Invoking device-cloud authentication leveraging the ATECC608A tamper-proof security keys

3 **while** *Creation and registration of a Certificate Authority (CA) and the IoT device's security credentials* **do**

4 Create a Certificate Authority's root certificate

5 $\leftarrow Certificate$

6 Invoke the IoT device's certificate signing request to a certificate signer Certificate Authority

7 sign the certificate signing request using the root certificate

8 $\leftarrow Certificate$

9 register the device's digital identity using the signed certificate.

10 $\leftarrow DeviceUniqueID$

11 **end**

12 Connect the device to the IoT cloud by Via passing the network medium credentials to the WINC1500

The experimental setup in chapter 5.2.1 highlighted the tools utilized for the provisioning of the IoT device, as well as the details of the experiment. Fig. 6.11 shows a graphic picture of this

setup, including the Laptop computer -equipped with the AWS CLI tool, Zerynth Studio, Atmel Studio, conected with the sample IoT device, while other components of the setup such as the creation of an AWS account on the AWS cloud are virtually integrated. The steps for provisioning the sample IoT device is presented in the algorithm table 4. Beginning with the initialization of the IoT device equipped with the ATECC608A, the device-to-cloud authentication mechanism is invoked leveraging the tamper-proof security keys in the ATECC608A device.

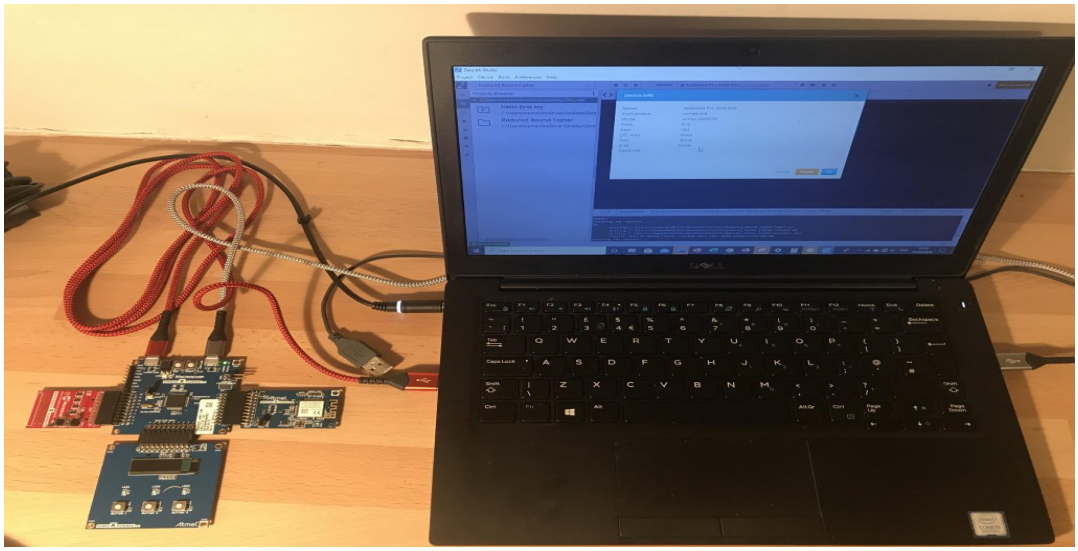

Figure 6.11: Setup of experimentation platforms: PC the SAMG55 IoT device

As detailed in the experimental setup in 5.2.1, the third step of the provisioning algorithm presented in table 4 is the creation and registration of a Certificate Authority, using the AWS CLI tool. The result of making this call reads the created root certificate, sets a certificate registration from the cloud end of the session, generates a signer certificate authority verification certificate and registers the certificate authority with the AWS IoT service, the AWS IoT core as shown in Fig. 6.12.

Integrated as an authentication security functionality of the ATECC608A is a security architecture based of the Elliptic Curve Cryptography (ECC), which is primarily a variant of the Elgamal algorithm. By leveraging the secure hardware accelerator in the ATECC608A, the SAMG55 microprocessor is securely authenticated to the AWS IoT core via utilizing it's tamper-proof keys. The SAMG55 is thus, provisioned on the cloud via utilizing the AWS CLI tool and executing the the process flow outlined in the algorithm table 4. This enforces the challenge of

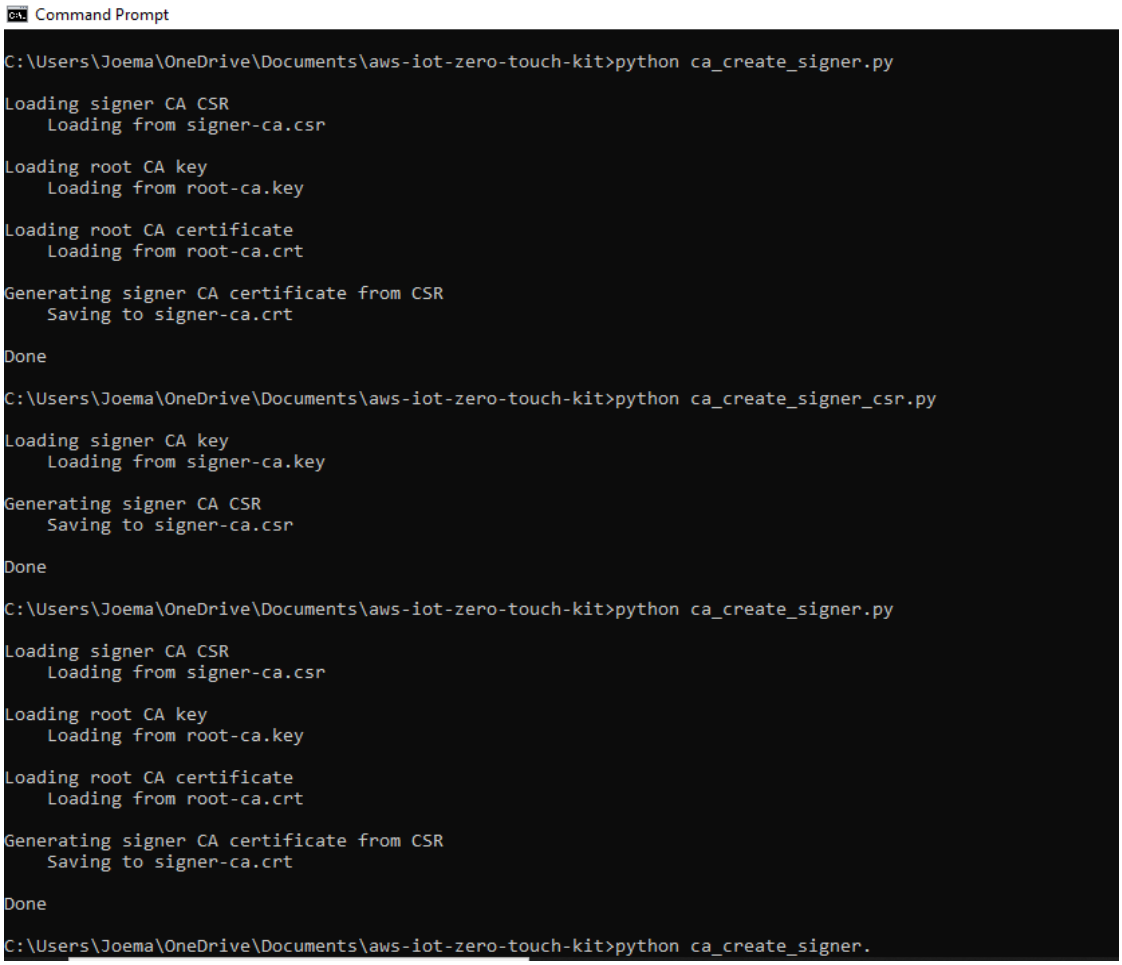

Figure 6.12: Certificate Authority Registration and Creation of a Unique Device ID on the Cloud

increasing the diameter of vulnerabilities associated with the secure handling of keys, from social engineering issues to third party mistrust. By design, The ATECC608A holds tamper-proof keys usable for invoking an authentication process that rids the challenges of social engineering detailed in chapter3.3 and mistrust in the handling of authentication keys. More so, these keys are completely isolated from the associated IoT device and the software vulnerabilities, owing to its taper-proof architecture. The AWS IoT core offers a bi-directional communication service between IoT devices and the AWS cloud, allowing for the registration of a device, associating it with certificates as shown in 6.12, custom attributes and requires a connecting IoT device to have a mutual authentication compatibility with Just-In-Time-Registration and Transport Layer Security (TLS1.2).

Following the creation of a unique ID for the device on the AWS IoT cloud, the just-In-

Figure 6.13: Configuration of the IoT Network Medium Credentials

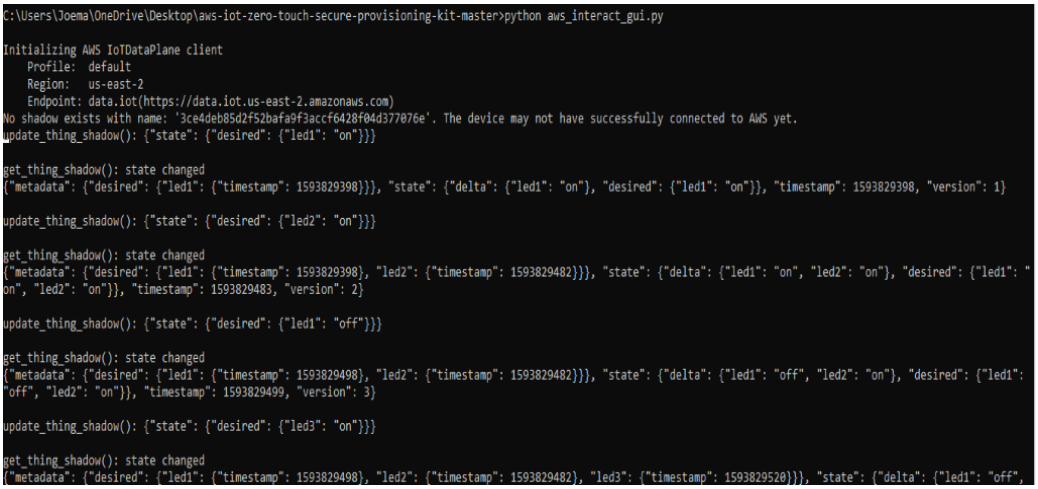

Figure 6.14: IoT Device Interaction with the Device Shadow on the AWS IoT Core

Time registration functionality facilitated through and AWS lambda function ensures that an IoT device connecting to the cloud automatically invokes a process of self registration at the time of connection. Network medium credentials are then passed onto the device by means of

configuring the ATWINC1500. The ATWINC1500 is a low-power 802.11b/g/n that enables a network interface extension to the SAMG55 microprocessor to aid the process of connecting the device to a network medium. For this process, a WIFI network equipped with Wireless Protected Access (WPA2) security is used. The WIFI medium credentials viz: the Service Set Identifier (SSID) and password of the WIFI is then successfully passed onto the ATWINC1500 using the AWS CLI programmatic access tool as shown in 6.13

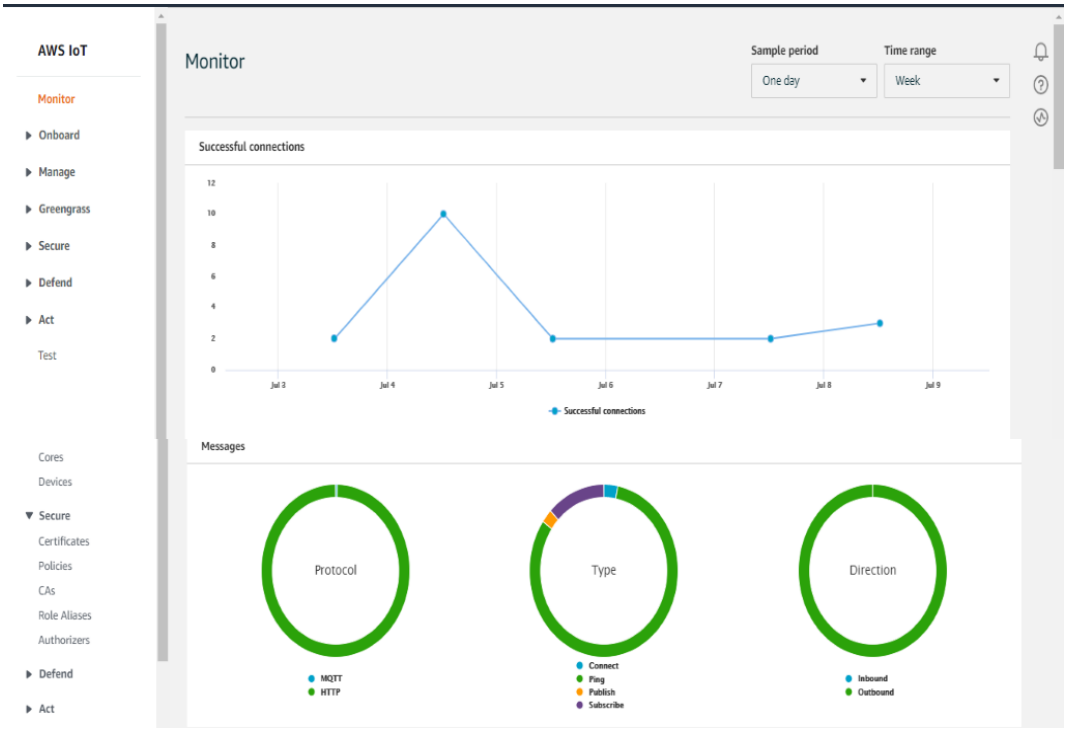

Figure 6.15: Cloud-end View of Provisioned IoT Device and Connection Records

Further to it's low-power compatibility with constrained IoT devices is the provision of a device shadow service which makes it possible to adopt the constrained IoT devices' power management strategy of devices going into sleep mode when it's not processing or transmitting information, allowing for the status of the device to be updated from the cloud end and the IoT device synchronizes the updates on the shadow upon re-connection. Fig. 6.15 shows the AWS dashboard with a record of the number of times of times of re-connection of the IoT device on the need to transmit bases, while optimizing the management of constrained power resource using the device shadow service.

6.3 Comparative Analysis with Results in Literature and Contribution

This section aims to compare the work done in this research with a few of similar works in literature and most importantly, to highlight the bit of differences found in the results and furnish a bases for the main contributions of this research, which is summarily captured in chapter 1.6.

Consequent upon resource constrains which greatly limits the processing capabilities of constrained IoT devices, the work in [100] in efforts to balance power consumption and security of IoT devices in low-power environments proposed a low power consumption communication scheme called the Secure Low Power Communication method (SeLPC), which lowers the power consumed by end devices for data encryption. The formulation of the SeLPC also leverage the idea of reducing the complexity of the classical AES algorithm which is found to be the defector standard for encryption in ensuring secure communication in low-power environments. Although the work in In [100] did not define a rationale for reducing the number of AES rounds to 5, according the results reported in [100], comparing with the traditional AES, analysis show that the SeLPC can reduce 26% of encryption power, and effectively able to resist known key attacks and replay attacks on associated IoT devices. In contrast, the formulation of the efficient algorithm for power constrained IoT devices hinged on experimental results of an estimation of the precise degree of constraint on an example IoT device as detailed in 6.2.2, to furnish requirements for the reduction of complexities of classical security algorithms for IoT devices' use-cases. Also as detailed in chapter 3.4, justification for round reduction of the classical algorithm as well as the details of security trade-off is provided with an examination of the core algebraic properties of the cipher, as a result of which results show a 27% and 35% respectively for reduction to rounds four and further to two as detailed in section 6.2.3.

In [104], the notion of developing optional security algorithms for different use-cases in IoT environments was presented. Following the evaluation of the software implementation of two lightweight algorithms: CLEFIA and PRESENT which are both in line with the narrative of availing less expensive algorithms for constrained IoT devices. Based on results of experiments detailed in [104], the memory footprints of PRESENT are less than that of CLEFIA as CLEFIA uses two S-boxes, but PRESENT is found to consume more time and CPU cycles for encrypting the same amount of data. Consequently, they opined that for time sensitive applications,

CLEFIA could be used, while advocating the use of PRESENT as a better algorithm for security-sensitive applications. In contrast, the efficient algorithm for constrained IoT devices presents the reduced round variants of the low cost algorithm as options for navigating trade-off requirement in a similar lightweight manner, through the investigation of the encryption completion times of different reduced rounds of the cipher to encrypt the same amount of data. Also, the efficient algorithm compares the encryption of completion time of encrypting the data to that of lightweight CLEFIA, as detailed in section 6.2.3, such that the RR2 could be prioritised for time-sensitive applications while the RR3 and RR4 can be prioritized for security sensitive requirement specification scenarios. Furthermore, the Avalanche effect test experimentation which results are detailed in 6.2.5 presents insights into the usability requirements of the reduced-round variants on a need to use bases as it shows 93%, 98% and 23% for reduced rounds to four, three and two respectively. Moreover, the authors in [104] opined that the ideal avalanche effect for a good cipher should be at least 50%, which means that a single change in the plain text should alter the outcome of the cipher text by at least 50% when encrypted by the cipher.

Building on the narrative that classical cryptographic algorithms are mainly designed for the desktop computing era and consequently unsuitable for the IoT, the authors in [45], observed that the biggest challenge of designing lightweight ciphers is that of coping with trade-offs between performance, cost and security. Noting the impossibility of providing the trio together to constrained IoT devices, the work therefore presented an ultra-lightweight security algorithm called SLIM, designed for power and area constraint efficiency without compromise on the security constructions of the cipher. According to [45], SLIM uses 4x4 S-boxes that work as nonlinear components of the cipher for performing non-linear operations on 16bits words and a key length of 80 to achieve immunity against exhaustive key search as per NIST recommendation. Hinging on the same narrative of developing lightweight algorithms by means of trading off factors of complexity as did the authors in [45], the development of the efficient algorithm for provisioning constrained IoT devices leveraged the design of the ATECC608x secure element as a trade-off for round reduction in the classical AES algorithm, similar to the case in [100] earlier presented. In the case of the efficient algorithm detailed through the chapters of this work, the tamper-proof capabilities of the provisioned secure element which guarantee the security of keys, together with the mathematical justification and the avalanche effect analysis presented in sections 3.4 and 6.2.5 respectively provides the bases for trading the number of rounds which account for complexity

in the classical AES, to make lightweight, the efficient algorithm.

In [148], an evaluation of the amount of time it takes to provision IoT devices using automated Zero Touch Provisioning (ZTP) is given, wherein the authors opined that considering the large number of devices to be deployed in the next generation of IoT networks, the amount of time required for manual provisioning of the devices is capable of significantly delaying the provisioning process, in addition to the likelihood of human-induced failures and errors associated with the manual provisioning processes. Although the IoT device provisioning process outlined in chapter 5.3 demonstrated the ZTP process in part, the ZTP solutions experimented in [148] found that soft-AP and Bluetooth-based ZTP solutions out-perform manual provisioning with about 154% and 313% when compared to expert provisioning and with about 434% and 880% when compared to non-expert provisioning in terms of the time it takes to provision the IoT device. While the device provisioning algorithm detailed in the algorithm table 4 utilize the AWS rules engine and lambda functions to facilitate the process of Just In Time Registration (JITR) of the device as detailed in 5.3.2, a plausible direction for future work could leverage the approach of the ZTP presented in [148], such that in furtherance to the concept of load aware provisioning of IoT services detailed in [149], the classification of IoT applications and services with respect to time-sensitivity or security sensitivity as detailed in [104] can uniquely leverage the results of the ZTP solutions in [148] to consolidate on the utilization of the variants of the reduced round algorithms for client-side encryption, based on the sensitivity requirements for client-side encryption, thereby improving the time it takes to provision IoT devices.

6.4 Chapter Summary

This chapter presented the analysis of the results of experiments conducted in this work, together with an itemization of the contributions. An introduction of the chapter explaining how the content is organized is first given in 6.1, which also highlighted key sections of previous chapters upon which the content of the chapter is based. Section 6.2 presents the implementation evaluation of the experiments done and analysis of results, wherein an analysis of computational complexity of the efficient algorithm for constrained IoT devices and comparative analysis of IoT devices' constraints are detailed in sections 6.2.1 and 6.2.2 respectively. The comparison and analysis of CLEFIA, AES and variants of the reduced round algorithms is presented in section 6.2.3, while 6.2.4 presents the results and analysis of low cost client-side encryption for constrained IoT

provisioning. The results of an Avalanche effect investigation and covariance computation from the experimental data is detailed in 6.2.5. The implementation evaluation and analysis of results section concludes with results and analysis of secure provisioning of a sample IoT device in 6.2.6. Sequel to the implementation and analysis of results presented in section 6.2, a comparative analysis of the results therein with results in literature is presented in 6.3, with respect to the contributions of the work in chapter 1.6 and the conclusion of the chapter in section 6.4.

Chapter 7

Summary, Conclusion and Future Work

7.1 Introduction

Classical cryptographic algorithms such as: DES, 3DES and the AES -to mention a few, are not well-suited for use in constrained Internet of Things (IoT) devices due to their computational and storage requirements. These algorithms often require significant processing power and memory to encrypt and decrypt data, making them too resource-intensive for many IoT devices, which often have limited computing resources. Classical cryptographic algorithms also rely on long key sizes to ensure security, but this can further increase the storage requirements for these algorithms on IoT devices, which often have limited storage capacity. The use of classical cryptographic algorithms can also have significant power consumption implications for constrained IoT devices, which often rely on battery power and have limited energy budgets. The frequent use of these algorithms can quickly drain battery resources, making them impractical for use in many IoT scenarios. Consequently, while classical cryptographic algorithms may be effective in securing data in certain contexts, they are not a good fit for use in constrained IoT environments due to their resource and power consumption requirements. Alternative cryptographic approaches, such as lightweight cryptography, may be more suitable for use in these types of devices.

With cloud computing being a key enabler of massive IoT deployments in several use-cases, the IoT provisioning process highlights the need for effective planning and management strategies in order to ensure the successful deployment and operation of these devices. Provisioning is an important aspect of IoT deployment and management, as it ensures that devices are properly configured and able to function as intended within the larger IoT ecosystem. This includes the need to maintain the security and integrity of the provisioning process itself, such as: protecting against attacks like man-in-the-middle attacks and ensuring that only authorized devices are able to join the network.

Client-side encryption in the context of IoT provisioning is the process of encrypting data on the IoT device, before it is transmitted over a network or stored in the cloud. This is useful for protecting sensitive data such as financial information, personal identity information and other types of confidential data, and typically used in conjunction with cloud-based storage and other types of networked services to help ensure the security and privacy of data in transit and at rest. In this model, the IoT device is responsible for generating and managing the encryption keys used to protect the data, rather than relying on a third-party server to provide this security. This increases the security and privacy of IoT data, as encryption keys are stored and managed on the IoT device, and are less vulnerable to theft or interception by external parties and providing protection against against man-in-the-middle attacks and other forms of network-based surveillance.

This thesis presents and Efficient Security Algorithm for provisioning Constrained IoT devices based on the AES. The efficient security algorithm is leveraged for client-side encryption before the secure provisioning of the IoT device onto AWS IoT Core platform. The summary of work done, as contained from chapter 1, through to chapter 7 is presented in the section that follows. In a chronological manner, section 7.2 presents the specific summaries of the respective chapters which opens with an introductory chapter of the thesis as contained in 1, a summary of the review of relevant literature in chapter 2, onto the summary of analysis of algorithm complexities in provisioning constrained IoT device and the efficient security algorithm for power constrained IoT devices as contained from 3.1 and 4.1 respectively, while concluding with the summaries of low cost client-side encryption and secure IoT provisioning and analysis of results and contributions as presented from 5.1 and 6.1 respectively.

7.2 Summary

In chapter 1, topics and sub-topics that are considered as relevant introductions to this research work were briefly introduced. An introduction of the IoT technology with respect to the thesis title and as per work done is presented in chapter 1.1, following which a general introduction to the security issues plaguing IoT deployments was given in 1.3. Resource constrained IoT as a section of the larger topic of the IoT, and which is central to the idea and work done in this research is presented in chapter 1.4. The nexus between the security challenges introduced in 1.3, the constrained category of IoT devices as contained in 1.4 and the burden imposed on constrained IoT devices by classical security algorithms is also subtly introduced in this section. Cloud computing, IoT provisioning and the need for rethinking inefficient security algorithms used by constrained IoT devices was introduced in chapter 1.5. The motivation and main contributions of the work done are given in chapter 1.6, following which a conclusion of the introductory chapter is presented in chapter 1.7

Chapter 2 presented a review of major topics -relevant to the research work done. Starting with a peek into the IoT technology and it's applications, a review of the security challenges in the IoT, as well as resource constraints in the IoT was presented. The review of classical security algorithms was presented, starting with a background of security algorithms in general, following which the review of specific security algorithms inclusive of the DES, the Triple 3DES was also reviewed in line with the notion of complexities when juxtaposed with the constrained nature of IoT devices. Based on available information in literature which shows the Advanced Encryption Standard (AES) as currently the most widely used algorithm in low power wide area networks as detailed in [25], a review of the AES as one of the classical algorithms was given with details into the design, the AES in the IoT, AES security and AES complexity. Hinging on the complexities of the classical security algorithms and the consequent requirement of lighter schemes for constrained IoT, a review of lightweight security algorithms, as well as the reduction of the complexities of the classical ones to achieve lightweight was also presented. In light of cloud computing being a major enabler for provisioning IoT devices, a review of cloud computing and IoT provisioning was presented in a comparative sense of how cloud computing technologies differ from fixed, on-premises infrastructure, while focusing on the convergence of cloud computing and the IoT technologies. Consequent upon the cloud computing and provisioning review, a further review of authentication with respect to the on-boarding of IoT devices in cloud-assisted IoT

platforms was also presented. Chapter 2 concludes with a review of client-side encryption and the need for developing lightweight algorithms that are compatible with the resource constraint in IoT devices, towards secure provisioning of the devices onto the cloud-assisted IoT platforms.

In chapter 3, an analysis of algorithm complexities in provisioning constrained IoT devices is detailed. Starting with an introduction of the major topics covered in the chapter in chapter 3.1, a mathematical background to most of the mathematical analysis in the thesis is given in 3.2, including the basic algebraic structures of Galois fields, averages, percentages and variance. A cryptanalytic overview of the consequence of reducing the complexity of classical algorithms followed in 3.3, detailing a cryptanalytic overview and analysis of the AES as an example classical algorithm and also as the most widely-used algorithm in the IoT landscape according to the work in [25], together with the trade-off for round reduction. A justification of the round reduction and trade-off is detailed in chapter 3.4, with the significance of relevant theorems used, presented in 3.4.1 and a conclusion of the chapter in 3.5.

Chapter 4 detailed the Efficient Algorithm for Constrained IoT devices, starting with an introduction of lightweight cryptography and it's inherent challenges as presented in sections 4.1.1 and 4.1.2, following which an overview of the efficient algorithm was given in 4.2, highlighting the two-step process of IoT device authentication and resource efficient message encryption. The implementation pseudo code for the the efficient algorithm based on the standard AES algorithm is presented, detailing the encryption and decryption processes as presented in 4.3.1 and 4.3.3 respectively. A sample block of message (16bytes) and encryption key (16bytes) is used to illustrate the inner workings of the round function. The Pseudo code is presented for a single round of message encryption and decryption with the specific objective to detail the process of message and key whitening, byte substitution, shiftrows, mix columns and the add round key sequences respectively, and the reverse process detailing Add Round Key, Inverse Mix Columns, Inverse Shift Rows and Inverse bytes substitution is also presented. chapter 4.4 presented a comparison of the classical AES algorithm, light weight clefia and the reduced round algorithm with respect to the algebraic structures and constructions. The experiments, which is based on the content of this chapter was presented in chapter 4.5, detailing the setup and factors considered in the experiments as presented in chapter 4.5.1. Data generated from the experiments done are also presented in tables: 4.3, 4.4 and 4.5 as encryption times data generated from implementations of the algorithms considered on a laptop, SAMG55 microprocessor and laptop & SAMg55 combined,

respectively. chapter 4.6 concludes by providing a compressed summary of the various sections of the chapter: Efficient Security Algorithm for Power Constrained IoT Devices.

With not much information available on the ease of secure provisioning of IoT devices, couple with the inherent challenges of these devices in encrypting data before transmission to cloud platforms due to their constrained nature, chapter 5 detailed the use of an efficient algorithm based on the AES for the client-side encryption using a SAMG55 microprocessor, as well as the secure provisioning of the IoT device onto AWS IoT core, an IoT cloud platform of a leading cloud services provider know as the Amazon Web Services (AWS). Starting with an introduction of the concept of IoT device provisioning and client-side encryption in 5.1, an overview of the experiments on which the content of the chapter is based, and the experimental setup was presented in chapter 5.2 and 5.2.1 respectively. chapter 5.3 detailed the authentication of the sample IoT device, the SAMG55 microprocessor. The details of the secure element which facilitates secure authentication and serves the trade-off for complexity reduction in the formulation of the algorithm is presented in chapter 5.3.1, following which the device provisioning algorithm is presented and the process detailed in 5.3.2. An overview of client-side encryption for constrained IoT devices together with the reduced round algorithm for low cost client-side encryption were presented in sections 5.4 and 5.4.1 respectively. The experimentation data detailing the comparison of a lightweight algorithm, the AES and the reduced round algorithm presented in table 5.1 and chapter 5.5 summarizes the various sections and content of the chapter.

In chapter 6, the analysis of the results of experiments conducted in this research is presented, together with an itemization of the specific contributions. An introduction of the chapter explaining how the content is organized is first given in 6.1, which also highlighted key sections of previous chapters upon which the content of the chapter is based. chapter 6.2 presents the implementation evaluation of the experiments done and analysis of results, wherein an analysis of computational complexity of the efficient algorithm for constrained IoT devices and comparative analysis of IoT devices' constraints are detailed in sections 6.2.1 and 6.2.2 respectively. The comparison and analysis of CLEFIA, AES and variants of the reduced round algorithms is presented in chapter 6.2.3, while chapter 6.2.4 presents the results and analysis of low cost client-side encryption fo constrained IoT provisioning. The results of an Avalanche effect investigation and covariance computation from the experimental data is detailed in 6.2.5. The implementation evaluation and analysis of results section concludes with results and analysis of secure provi-

sioning of a sample IoT device in 6.2.6. Sequel to the implementation and analysis of results presented in chapter 6.2, a comparative analysis of the results therein with results in literature is presented in 6.3, together with an itemization of the major contributions of the work as per chapter 1.6 and the conclusion of the chapter in chapter 6.4.

7.3 Conclusion

The IoT technology is impacting norms in several sectors of the economies, including but not limited to; transportation [13, 64, 65] businesses [55–59], healthcare, agriculture [51, 52] and homes [68, 70], facilitated through massive deployments of IoT devices and applications. The deployments of plethora of devices and applications however, is accompanied by a corresponding plethora of challenges in ensuring the security and safe use of the IoT technology, and the work in this thesis is motivated by the existing need to tackle some of the identified challenges. As rightly put by [32], securing the IoT is a necessary milestone towards expediting the deployment of its applications and services. According to [33], As smart home systems get more and more popular recently, the security protection of smart home systems has become an important problem. Architecting IoT focused security solutions must however, take into considerations the unique circumstance of power constrained IoT devices as according to [34], reaping the benefits of the IoT is contingent upon developing IoT-specific security and privacy solutions. Since IoT communication protocols and technologies differ from traditional IT realms, their security solutions ought to take this difference into account [29]. The security of conventional IT infrastructure is achieved using classical cryptographic protocols and algorithms whereas, applying classical cryptographic methods for IoT security is not efficient as those methods were not ideally designed for these kind of systems [23]. Consequently, the option for IoT in terms of security becomes either in the development of new schemes or the modification of existing ones to fit the nature of the constrained category of IoT devices. Further to the existing problems identified through the review of relevant literature, the authors in [41] observed that although it is estimated that IoT devices are being deployed in billions, very little to no information is present on ease of device provisioning IoT devices. Secure authentication of IoT devices that are constrained in power, memory and processing resources is an ongoing challenge desiring novel solutions as usu-

ally, these tiny IoT devices run by battery power, making it a daunting task to design security mechanism which is a best fit [31]. Also according to [31], some of the challenging problems in the implementation of the IoT include: key management, device authentication, user access control, privacy preservation and identity management to mention but a few. Many attacks including eavesdropping, Denial of Service (DoS), Man-in-the-Middle and certificate manipulation among others is a serious threat to the authentication of IoT devices and the constrained nature of the devices has imposed a serious challenge in designing counter measures to combat these attacks [31]. These challenges constitute a motivation for the work done in this thesis with the aim and specific objectives as detailed in 1.2. Also, encryption-before-outsourcing of data is a widely recommended method to guarantee the confidentiality of user data [43] and the consequent need of architecting devices with client-side encryption capabilities in order to preserve the privacy of data generated and outsourced to cloud storage systems brings on another layer of burden on the devices, given the scarcity of resources. In order to protect the security of the outsourced data, an intuitive way is to encrypt the data before outsourcing it to the cloud [44] and according to [42] , the integration of IoT devices and cloud servers is highly dependent on how security issues such as authentication and data privacy are handled. Thus, provisioning these IoT devices with low-cost encryption algorithms and without compromise to secure provisioning is advocated.

Motivated by the aforementioned works which summarize the unsuitability of the usage of conventional cryptographic algorithms for deploying security solutions in the IoT landscape, the work in this thesis focused on the development of an efficient security algorithm for power constrained IoT devices and the utilization of the efficient algorithm to experiment client-side encryption and secure provisioning of a sample constrained IoT device, the SAMG55 micro-processor. With the AES identified as currently, the predominantly used algorithm in the IoT landscape based on the work in [25], this work aimed to reduce complexity of the AES into a more efficient algorithm suitable for the constrained category of IoT devices. The questions that follow thus are that of: what the measure of complexity of the AES is, what components of the security algorithm accounts for what is currently adjudged to be its complexity and what the consequence of reducing the complexity is. The contributions of this work with respect to the aforementioned problems and novelty of a solution lead to the specific contributions itemized in chapter 1.6. of the work. A cryptanalytic overview and analysis of the consequences of reducing the complexity of the AES, as the currently used encryption algorithm in the IoT landscape is

first presented, together with a mathematical justification of reducing the complexity of the standard AES-128 algorithm, using the core algebraic properties of the standard algorithm, which is followed by provisioning a secure element: the ATECC608A to aid authentication and guard against implementation attacks in line with our analysis of the consequence of round reduction of the AES-128. This work implemented a safely reduced round versions (four rounds and two rounds) of the AES-128 algorithm, based on the the structure of the AES in order to reduce complexity (measured by the time it takes to complete the encryption of 16bytes of plain text). Results of the comparison of the reduced round algorithm and the standard AES-algorithm show that up to 35% of the time it takes to complete the encryption of a single byte of plain-text is saved, as detailed in chapter 6 of this work and published in 4. An investigation to ascertain the measure of resource constraint on the sample IoT device was conducted. Juxtaposing the varying level of constrain in different constrained IoT devices, this work considered establishing the precise case for the platform of experimentation and thus, the experimentation and analysis of resource constrain comparing a PC and SAMG55 implementations of the efficient algorithm to the standard AES128. Also, the implementation and comparison of the efficient algorithm (Based on the AES) to lightweight CLEFIA, experimentation of the avalanche effect test on the low-cost algorithm and using it as client-side encryption solution in provisioning the SAMG55 microprocessor as detailed in 5.4 was conducted, following which the sample constrained IoT device was provisioned on Amazon Web Services (AWS) IoT core via the use of Command Line Interface (CLI) programmatic access tools. As detailed in chapter 6, experimentation result shows an increase of up to 657% in the encryption completion time on the IoT device in comparison to the PC due to resource constrain. The low-cost algorithm shows up to 50% reduction in the aforementioned encryption completion time and so, was utilized for experimenting low cost client-side encryption and the device provisioned to the cloud.

With respect to the categories of security challenges in the IoT and cyber-physical systems landscape as outlined in [23], this work addresses the bit of privacy and access control, through message encryption and secure authentication respectively, and also to guard against implementation attacks on the associated IoT device in terms of the cryptanalytic overview detailed in [105] and chapter 3.3. The efficient algorithm is utilized for implementing a low-cost client-side encryption algorithm for data encryption as advocated in [44], while leveraging the ATECC608A addresses the challenges of key management and device authentication as highlighted in [31], by

securely provisioning the device onto the AWS IoT core.

7.4 Furure Work

Based on the results and findings in this thesis and with respect to other works and results available in literature, the following items include plausible directions that can be explored to build on the work contained in this thesis:

- With respect to the results on Zero Touch Provisioning, the work in [148] found that soft-AP and Bluetooth-based ZTP solutions out-perform manual provisioning with about 154% and 313% when compared to expert provisioning and with about 434% and 880% when compared to non-expert provisioning in terms of the time it takes to provision IoT devices and as such, a plausible direction for future work could leverage the approach of the ZTP presented in [148], such that in furtherance to the concept of load aware provisioning of IoT services detailed in [149], the classification of IoT applications and services with respect to time-sensitivity or security sensitivity as detailed in [104] can uniquely leverage the results of the ZTP solutions in [148] to consolidate on the utilization of the variants of the reduced round algorithms for client-side encryption, based on the sensitivity requirements for client-side encryption, thereby improving the time it takes to provision IoT devices.

- Resources sharing mechanism in the IoT domain where resource constrained IoT devices can offload computationally intensive resources to resource-rich ones in order to achieve high quality of experience is encouraged in [147]. Accordingly, more complex scenarios of the use case of the proposed power efficient algorithm can be explored to leverage these efforts, and use the algorithm for resource constrained devices while adapting to the standard algorithm for the resource rich scenarios. Leveraging the efficient security algorithm for constrained IoT devices in dynamic resource sharing environments where computationally intensive tasks are off-loaded to the resource-rich ones for assisted processing, and comparing method efficiency.

- Leveraging measurement techniques and the use of Probability Distribution Functions on the experimented instances of encryption and analyse the difference in the distributivity. An interesting investigation of this direction would also include probing the big $\mathcal{O}$ complexity of block ciphers with respect to the distributions of these encryption instances.

References

[1] K. K. Patel, S. M. Patel *et al.*, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 2016.

[2] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things," ser. HotNets-XIV. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2834050.2834095

[3] I. E. Etim and J. Lota, "Iecon 2016 - 42nd annual conference of the ieee industrial electronics society," 2016, pp. 4701–4705.

[4] C. Zhang, "Intelligent internet of things service based on artificial intelligence technology," in *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, 2021, pp. 731–734.

[5] C. Liu, Y. Xiao, V. Javangula, Q. Hu, S. Wang, and X. Cheng, "Normachain: A blockchain-based normalized autonomous transaction settlement system for iot-based e-commerce," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4680–4693, 2019.

[6] O. Sohaib, H. Lu, and W. Hussain, "Internet of things (iot) in e-commerce: For people with disabilities," in *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2017, pp. 419–423.

[7] H. Yu and X. Zhang, "Research on the application of iot in e-commerce," in *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, vol. 2, 2017, pp. 434–436.

[8] H. Desai, D. Guruvayurappan, M. Merchant, S. Somaiya, and H. Mundra, "Iot based grocery monitoring system," in *2017 Fourteenth International Conference on Wireless and Optical Communications Networks (WOCN)*, 2017, pp. 1–4.

[9] P. Gyeltshen and K. Osathanunkul, "Linking small-scale farmers to market using ict," in *2018 International Conference on Digital Arts, Media and Technology (ICDAMT)*, 2018, pp. 120–125.

[10] R. Dagar, S. Som, and S. K. Khatri, "Smart farming – iot in agriculture," in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2018, pp. 1052–1056.

[11] K. T. E. Keerthana, S. Karpagavalli, and A. M. Posonia, "Smart system monitoring agricultural land using iot," in *2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR)*, 2018, pp. 1–7.

[12] K. Guan, D. He, B. Ai, D. W. Matolak, Q. Wang, Z. Zhong, and T. Kürner, "5-ghz obstructed vehicle-to-vehicle channel characterization for internet of intelligent vehicles," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 100–110, 2019.

[13] X. Luo, H. Zhang, Z. Zhang, Y. Yu, and K. Li, "A new framework of intelligent public transportation system based on the internet of things," *IEEE Access*, vol. 7, pp. 55 290–55 304, 2019.

[14] R. Silva and R. Iqbal, "Ethical implications of social internet of vehicles systems," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 517–531, 2019.

[15] K. Al-Gumaei, K. Schuba, A. Friesen, S. Heymann, C. Pieper, F. Pethig, and S. Schriegel, "A survey of internet of things and big data integrated solutions for industrie 4.0," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2018, pp. 1417–1424.

[16] W. Sun, J. Liu, Y. Yue, and Y. Jiang, "Social-aware incentive mechanisms for d2d resource sharing in iiot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5517–5526, 2019.

[17] D. C. Trancă, D. Rosner, R. Curatu, A. Surpăteanu, M. Mocanu, Ş. Pardău, and A. V. Pălăcean, "Industrial wsn node extension and measurement systems for air, water and environmental monitoring: Iot enabled environment monitoring using ni wsn nodes," in *2017 16th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2017, pp. 1–6.

[18] L. C. Souza, J. J. P. C. Rodrigues, G. D. Scarpioni, D. A. A. Santos, V. H. C. de Albuquerque, and S. K. Dhurandher, "An iot automated curtain system for smart homes," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2018, pp. 249–253.

[19] K. Hayashi and H. Suzuki, "Cooperation between heterogeneous iot devices using ihac hub," in *2019 IEEE International Conference on Consumer Electronics (ICCE)*, 2019, pp. 1–2.

[20] R. Pavithra, M. Karthigha, A. Hema, K. R. Bharathi, and A. Madhumitha, "Cloud based smart pantry system using iot," in *2021 6th International Conference on Inventive Computation Technologies (ICICT)*. IEEE, 2021, pp. 224–228.

[21] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.

[22] Q. Luo, J. Liu, J. Wang, Y. Tan, Y. Cao, and N. Kato, "Automatic content inspection and forensics for children android apps," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[23] Y. Ashibani and Q. H. Mahmoud, "Cyber physical systems security: Analysis, challenges and solutions," *Comput. Secur.*, vol. 68, pp. 81–97, 2017.

[24] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1727–1765, 2016.

[25] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *ieee communications surveys & tutorials*, vol. 19, no. 2, pp. 855–873, 2017.

[26] Y. Ashibani and Q. H. Mahmoud, "Cyber physical systems security: Analysis, challenges and solutions," *Computers & Security*, vol. 68, pp. 81–97, 2017.

[27] P. P. Jayaraman, X. Yang, A. Yavari, D. Georgakopoulos, and X. Yi, "Privacy preserving internet of things: From privacy techniques to a blueprint architecture and efficient implementation," *Future Generation Computer Systems*, vol. 76, pp. 540–549, 2017.

[28] M. Katagi, S. Moriai *et al.*, "Lightweight cryptography for the internet of things," *Sony Corporation*, vol. 2008, pp. 7–10, 2008.

[29] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[30] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1727–1765, 2016.

[31] A. K. Sahu, S. Sharma, S. S. Tripathi, and K. N. Singh, "2019 international conference on information technology (icit)," 2019, pp. 217–221.

[32] A. Ferdowsi and W. Saad, "Deep learning for signal authentication and security in massive internet-of-things systems," *IEEE Transactions on Communications*, vol. 67, no. 2, pp. 1371–1387, 2019.

[33] E. Ruiz, R. Avelar, and X. Wang, "2018 ieee/acm 40th international conference on software engineering: Companion (icse-companion)," 2018, pp. 212–213.

[34] Y. Sharaf-Dabbagh and W. Saad, "2016 ieee 17th international symposium on a world of

wireless, mobile and multimedia networks (wowmom)," 2016, pp. 1–3.

[35] Z. Bekli and W. Ouda, "Energy monitoring of the cortex-m4 core, embedded in the atmel sam g55 microcontroller," 2017.

[36] A. Ferdowsi and W. Saad, "Deep learning-based dynamic watermarking for secure signal authentication in the internet of things," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[37] Z. Qin, F. Y. Li, G. Y. Li, J. A. McCann, and Q. Ni, "Low-power wide-area networks for sustainable iot," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 140–145, 2019.

[38] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[39] S. Aldossary and W. Allen, "Data security, privacy, availability and integrity in cloud computing: Issues and current solutions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 4, 2016. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2016.070464

[40] E. Henziger and N. Carlsson, "Delta encoding overhead analysis of cloud storage systems using client-side encryption," in *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2019, pp. 183–190.

[41] O. Ali, M. K. Ishak, L. Wuttisittikulkij, and T. Z. B. Maung, "Iot devices and edge gateway provisioning, realtime analytics for simulated and virtually emulated devices," in *2020 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, 2020, pp. 1–5.

[42] A. Sahoo, S. S. Sahoo, S. Sahoo, B. Sahoo, and A. K. Turuk, "2020 international conference on communication systems networks (comsnets)," 2020, pp. 419–426.

[43] H. Deng, Z. Qin, L. Sha, and H. Yin, "A flexible privacy-preserving data sharing scheme in cloud-assisted iot," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[44] L. Liu, H. Wang, and Y. Zhang, "Secure iot data outsourcing with aggregate statistics and fine-grained access control," *IEEE Access*, vol. 8, pp. 95 057–95 067, 2020.

[45] B. Aboushosha, R. A. Ramadan, A. D. Dwivedi, A. El-Sayed, and M. M. Dessouky, "Slim: A lightweight block cipher for internet of health things," *IEEE Access*, vol. 8, pp. 203 747–203 757, 2020.

[46] R. Gurunath, M. Agarwal, A. Nandi, and D. Samanta, "An overview: security issue in iot network," in *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-*

SMAC), 2018 2nd International Conference on. IEEE, 2018, pp. 104–107.

[47] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things," ser. HotNets-XIV. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2834050.2834095

[48] M. A. Iqbal, S. Hussain, H. Xing, and M. A. Imran, *IoT Cloud and Fog Computing*, 2021, pp. 127–145.

[49] E. P. Yadav, E. A. Mittal, and H. Yadav, "Iot: Challenges and issues in indian perspective," in *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU).* IEEE, 2018, pp. 1–5.

[50] J.-Y. Yu and Y.-G. Kim, "Analysis of iot platform security: A survey," in *2019 International Conference on Platform Technology and Service (PlatCon).* IEEE, 2019, pp. 1–5.

[51] R. Dagar, S. Som, and S. K. Khatri, "2018 international conference on inventive research in computing applications (icirca)," 2018, pp. 1052–1056.

[52] K. T. E. Keerthana, S. Karpagavalli, and A. M. Posonia, "2018 international conference on emerging trends and innovations in engineering and technological research (icetietr)," 2018, pp. 1–7.

[53] P. Corista, D. Ferreira, J. Gião, J. Sarraipa, and R. J. Gonçalves, "An iot agriculture system using fiware," in *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC).* IEEE, 2018, pp. 1–6.

[54] D. Davcev, K. Mitreski, S. Trajkovic, V. Nikolovski, and N. Koteli, "Iot agriculture system based on lorawan," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS).* IEEE, 2018, pp. 1–4.

[55] P. Gyeltshen and K. Osathanunkul, "2018 international conference on digital arts, media and technology (icdamt)," 2018, pp. 120–125.

[56] C. Liu, Y. Xiao, V. Javangula, Q. Hu, S. Wang, and X. Cheng, "Normachain a blockchain-based normalized autonomous transaction settlement system for iot-based e-commerce," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4680–4693, 2019.

[57] O. Sohaib, H. Lu, and W. Hussain, "2017 12th ieee conference on industrial electronics and applications (iciea)," 2017, pp. 419–423.

[58] H. Yu and X. Zhang, "2017 ieee international conference on computational science and engineering (cse) and ieee international conference on embedded and ubiquitous computing (euc)," vol. 2, 2017, pp. 434–436.

[59] H. Desai, D. Guruvayurappan, M. Merchant, S. Somaiya, and H. Mundra, "2017 fourteenth international conference on wireless and optical communications networks (wocn)," 2017, pp. 1–4.

[60] T. T. Chhowa, M. A. Rahman, A. K. Paul, and R. Ahmmed, "A narrative analysis on deep learning in iot based medical big data analysis with future perspectives," in *2019 International conference on electrical, computer and communication engineering (ECCE)*. IEEE, 2019, pp. 1–6.

[61] M. F. Mubin, F. Ahmed, M. Islam *et al.*, "Iot based health monitoring system for elderly people," Ph.D. dissertation, BRAC University, 2017.

[62] S. Ananth, P. Sathya, and P. M. Mohan, "Smart health monitoring system through iot," in *2019 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2019, pp. 0968–0970.

[63] A. Zameer, M. Saqib, V. R. Naidu, and I. Ahmed, "Iot and big data for decreasing mortality rate in accidents and critical illnesses," in *2019 4th MEC International Conference on Big Data and Smart City (ICBDSC)*. IEEE, 2019, pp. 1–5.

[64] K. Guan, D. He, B. Ai, D. W. Matolak, Q. Wang, Z. Zhong, and T. Kürner, "5-ghz obstructed vehicle-to-vehicle channel characterization for internet of intelligent vehicles," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 100–110, 2019.

[65] R. Silva and R. Iqbal, "Ethical implications of social internet of vehicles systems," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 517–531, 2019.

[66] W. Lee, "3d machine vision in iot for factory and building automation," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–1.

[67] D. C. Trancă, D. Rosner, R. Curatu, A. Surpăteanu, M. Mocanu, Ş. Pardău, and A. V. Pălăcean, "Industrial wsn node extension and measurement systems for air, water and environmental monitoring: Iot enabled environment monitoring using ni wsn nodes," in *2017 16th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2017, pp. 1–6.

[68] L. C. Souza, J. J. P. C. Rodrigues, G. D. Scarpioni, D. A. A. Santos, V. H. C. de Albuquerque, and S. K. Dhurandher, "2018 international conference on advances in computing, communications and informatics (icacci)," 2018, pp. 249–253.

[69] A. Banerjee, F. Sufyanf, M. S. Nayel, and S. Sagar, "Centralized framework for controlling heterogeneous appliances in a smart home environment," in *2018 International Conference on Information and Computer Technologies (ICICT)*. IEEE, 2018, pp. 78–82.

[70] K. Hayashi and H. Suzuki, "2019 ieee international conference on consumer electronics (icce)," 2019, pp. 1–2.

[71] A. Ferdowsi and W. Saad, "Deep learning for signal authentication and security in massive internet-of-things systems," *IEEE Transactions on Communications*, vol. 67, no. 2, pp. 1371–1387, 2019.

[72] E. Ruiz, R. Avelar, and X. Wang, "Protecting remote controlling apps of smart-home-oriented iot devices," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, 2018, pp. 212–213.

[73] Y. Sharaf-Dabbagh and W. Saad, "On the authentication of devices in the internet of things," in *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2016, pp. 1–3.

[74] M. S. Rahman and M. Hossam-E-Haider, "Quantum iot: A quantum approach in iot security maintenance," in *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*. IEEE, 2019, pp. 269–272.

[75] T. Wang, Y. Liu, and A. V. Vasilakos, "Survey on channel reciprocity based key establishment techniques for wireless systems," *Wireless Networks*, vol. 21, no. 6, pp. 1835–1846, 2015.

[76] P. P. Jayaraman, X. Yang, A. Yavari, D. Georgakopoulos, and X. Yi, "Privacy preserving internet of things: From privacy techniques to a blueprint architecture and efficient implementation," *Future Generation Computer Systems*, vol. 76, pp. 540–549, 2017.

[77] S. Y. Yan, *Computational Number Theory and Modern Cryptography*. John Wiley & Sons, 2013.

[78] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[79] N. R. Reilly, *Introduction to applied algebraic systems*. Oxford University Press, 2009.

[80] M. A. Philip *et al.*, "A survey on lightweight ciphers for iot devices," in *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*. IEEE, 2017, pp. 1–4.

[81] J. M. Hamamreh, H. M. Furqan, Z. Ali, and G. A. S. Sidhu, "An efficient security method based on exploiting channel state information (csi)," in *2017 International Conference on Frontiers of Information Technology (FIT)*. IEEE, 2017, pp. 288–293.

[82] M. Bottarelli, G. Epiphaniou, D. K. B. Ismail, P. Karadimas, and H. Al-Khateeb, "Physical characteristics of wireless communication channels for secret key establishment: A survey

of the research," *Computers & Security*, vol. 78, pp. 454–476, 2018.

[83] A. Ghosal, S. Halder, and S. Chessa, "Secure key design approaches using entropy harvesting in wireless sensor network: A survey," *Journal of Network and Computer Applications*, vol. 78, pp. 216–230, 2017.

[84] H. M. Furqan, J. M. Hamamreh, and H. Arslan, "Secret key generation using channel quantization with svd for reciprocal mimo channels," in *2016 International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2016, pp. 597–602.

[85] M. Noura, H. N. Noura, A. Chehab, M. M. Mansour, and R. Couturier, "S-des: An efficient & secure des variant," in *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*. IEEE, 2018, pp. 1–6.

[86] M. A. S. Eldeen, A. A. Elkouny, and S. Elramly, "Des algorithm security fortification using elliptic curve cryptography," in *2015 Tenth International Conference on Computer Engineering & Systems (ICCES)*. IEEE, 2015, pp. 335–340.

[87] D. Coppersmith, "The data encryption standard (des) and its strength against attacks," *IBM journal of research and development*, vol. 38, no. 3, pp. 243–250, 1994.

[88] F. Zhang, Y. Zhang, S. Shi, S. Guo, Z. Liang, S. Qureshi, and C. Xu, "Optimized lightweight hardware trojan-based fault attack on des," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2018, pp. 654–661.

[89] S. Mitra, B. Jana, and J. Poray, "Implementation of a novel security technique using triple des in cashless transaction," in *2017 International Conference on Computer, Electrical & Communication Engineering (ICCECE)*. IEEE, 2017, pp. 1–6.

[90] S. Amic, K. S. Soyjaudah, H. Mohabeer, and G. Ramsawock, "Cryptanalysis of des-16 using binary firefly algorithm," in *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*. IEEE, 2016, pp. 94–99.

[91] S. Oukili and S. Bri, "Fpga implementation of data encryption standard using time variable permutations," in *2015 27th International Conference on Microelectronics (ICM)*. IEEE, 2015, pp. 126–129.

[92] S. D. Putra, A. S. Ahmad, and S. Sutikno, "Power analysis attack on implementation of des," in *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*. IEEE, 2016, pp. 1–6.

[93] W.-S. Yap, S.-H. Heng, and B.-M. Goi, "Security analysis of m-des and key-based coded permutation ciphers in wireless channels," *IET Communications*, vol. 12, no. 10, pp. 1230–

1235, 2018.

[94] T. Fu and S. Li, "A 3des asic implementation with feedback path in the cbc mode," in *2017 International Conference on Electron Devices and Solid-State Circuits (EDSSC)*. IEEE, 2017, pp. 1–2.

[95] Y. He and S. Li, "A 3des implementation especially for cbc feedback loop mode," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[96] L. Scripcariu, P.-D. Mătăsaru, and F. Diaconu, "Extended des algorithm to galois fields," in *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*. IEEE, 2017, pp. 1–4.

[97] Wikipedia, "Rijndael s-box," https://en.wikipedia.org/wiki/Rijndael_S-box, 2020.

[98] D. Reddy and S. Jilani, "Implementation of 128-bit aes algorithm in matlab," *Int. J. Eng. Trends Technol.(IJETT)*, vol. 33, no. 3, pp. 126–129, 2016.

[99] C.-W. Hung and W.-T. Hsu, "Power consumption and calculation requirement analysis of aes for wsn iot," *Sensors*, vol. 18, no. 6, p. 1675, 2018.

[100] K.-L. Tsai, Y.-L. Huang, F.-Y. Leu, I. You, Y.-L. Huang, and C.-H. Tsai, "Aes-128 based secure low power communication for lorawan iot environments," *Ieee Access*, vol. 6, pp. 45 325–45 334, 2018.

[101] D. McGrew, "Low power wireless scenarios and techniques for saving bandwidth without sacrificing security," in *NIST Lightweight Cryptography Workshop*, vol. 2015, 2015.

[102] S. B. Sadkhan and A. O. Salman, "A survey on lightweight-cryptography status and future challenges," in *2018 International Conference on Advance of Sustainable Engineering and its Application (ICASEA)*, 2018, pp. 105–108.

[103] L. Scripcariu, P. Mătăsaru, and F. Diaconu, "2017 international symposium on signals, circuits and systems (isscs)," 2017, pp. 1–4.

[104] M. Jangra and B. Singh, "Performance analysis of clefia and present lightweight block ciphers," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 8, pp. 1489–1499, 2019.

[105] J. N. Mamvong, G. L. Goteng, B. Zhou, and Y. Gao, "Efficient security algorithm for power-constrained iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5498–5509, 2021.

[106] A. Riahi Sfar, Y. Challal, P. Moyal, and E. Natalizio, "A game theoretic approach for privacy preserving model in iot-based transportation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4405–4414, 2019.

[107] D. McGrew, "Low power wireless scenarios and techniques for saving bandwidth without sacrificing security," 2015.

[108] K. Tsai, Y. Huang, F. Leu, I. You, Y. Huang, and C. Tsai, "Aes-128 based secure low power communication for lorawan iot environments," *IEEE Access*, vol. 6, pp. 45 325–45 334, 2018.

[109] G. Liu, W. Quan, N. Cheng, H. Zhang, and S. Yu, "Efficient ddos attacks mitigation for stateful forwarding in internet of things," *Journal of Network and Computer Applications*, vol. 130, pp. 1–13, 03 2019.

[110] I. Nowrin and F. Khanam, "2019 international conference on applied machine learning (icaml)," 2019, pp. 183–186.

[111] T. L. Mokgetse and R. Sridaran, "2019 6th international conference on computing for sustainable global development (indiacom)," 2019, pp. 1218–1222.

[112] D. Patil and N. Mahajan, "An analytical survey for improving authentication levels in cloud computing," in *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE, 2021, pp. 6–8.

[113] J. Baron, H. Baz, T. Bixler, B. Gaut, K. E. Kelly, S. Senior, and J. Stamper, *AWS certified solutions architect official study guide: associate exam*. John Wiley & Sons, 2016.

[114] M. Matic, M. Antic, P. Istvan, and S. Ivanovic, "Optimization of mqtt communication between microservices in the iot cloud," in *2021 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2021, pp. 1–3.

[115] P. Mishra, S. Kumar, U. Garg, E. S. Pilli, and R. Joshi, "Security perspectives of various iot cloud platforms: A review & case study," in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. IEEE, 2021, pp. 727–731.

[116] M. Joshi, S. Budhani, N. Tewari, and S. Prakash, "Analytical review of data security in cloud computing," in *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*. IEEE, 2021, pp. 362–366.

[117] T. Jun, M. Kim, J. Kim, D. Kim, K. Kim, and D. Kang, "Facilitating integration of iot ecosystems via universal cloud interface," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021, pp. 1–4.

[118] H. Boujezza, H. Kaffel-Ben Ayed, and L. A. Saïdane, "2017 13th international wireless communications and mobile computing conference (iwcmc)," 2017, pp. 423–428.

[119] M. Zhaofeng, M. Jialin, W. Jihui, and S. Zhiguang, "Blockchain-based decentralized authentication modeling scheme in edge and iot environment," *IEEE Internet of Things Journal*,

vol. 8, no. 4, pp. 2116–2123, 2020.

[120] S. Sahoo, S. S. Sahoo, P. Maiti, B. Sahoo, and A. K. Turuk, "2019 6th international conference on signal processing and integrated networks (spin)," 2019, pp. 1024–1029.

[121] A. Kumari, M. Y. Abbasi, and M. Alam, "A smartcard-based key agreement framework for cloud computing using ecc," in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*. IEEE, 2021, pp. 43–48.

[122] X. Yang, S. Yang, J. Liu, C. Wang, Y. Chen, and N. Saxena, "Enabling finger-touch-based mobile user authentication via physical vibrations on iot devices," *IEEE Transactions on Mobile Computing*, 2021.

[123] X. Jiang, X. Liu, J. Fan, X. Ye, C. Dai, E. A. Clancy, D. Farina, and W. Chen, "Enhancing iot security via cancelable hd-semg-based biometric authentication password, encoded by gesture," *IEEE Internet of Things Journal*, 2021.

[124] P. Urien, "Innovative tls 1.3 identity module for trusted iot device," in *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, 2021, pp. 1–4.

[125] F. Wu, X. Li, L. Xu, P. Vijayakumar, and N. Kumar, "A novel three-factor authentication protocol for wireless sensor networks with iot notion," *IEEE Systems Journal*, vol. 15, no. 1, pp. 1120–1129, 2020.

[126] B. M. Alsellami and P. D. Deshmukh, "The recent trends in biometric traits authentication based on internet of things (iot)," in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*. IEEE, 2021, pp. 1359–1365.

[127] A. S. Rachini and R. Khatoun, "2020 sixth international conference on mobile and secure services (mobisecserv)," 2020, pp. 1–5.

[128] P. Hao and X. Wang, "Integrating phy security into ndn-iot networks by exploiting mec: Authentication efficiency, robustness, and accuracy enhancement," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 4, pp. 792–806, 2019.

[129] M. S. E. Quadir and J. A. Chandy, "2020 ieee international conference on consumer electronics (icce)," 2020, pp. 1–6.

[130] P. Varsha, K. Hemanth, and A. Raut, "Modified protocol for secure mutual authentication in iot smart homes," in *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2021, pp. 398–405.

[131] N. A. Gunathilake, W. J. Buchanan, and R. Asif, "2019 ieee 5th world forum on internet of things (wf-iot)," 2019, pp. 707–710.

[132] K. Quist-Aphetsi and M. C. Xenya, "2019 international conference on cyber security and internet of things icsiot," 2019, pp. 88–92.

[133] S. Rajashree, P. Gajkumar Shah, and S. Murali, "2018 ieee international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata)," 2018, pp. 219–221.

[134] A. Sheshasaayee and R. Megala, "A study on resource provisioning approaches in autonomic cloud computing," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 2017, pp. 141–144.

[135] J. Kaplan, D. G. Fisher, and N. T. Rogness, "Lexical ambiguity in statistics: how students use and define the words: association, average, confidence, random and spread," *Journal of Statistics Education*, vol. 18, no. 2, 2010.

[136] T. R. Guskey, "The case against percentage grades," *Educational Leadership*, vol. 71, no. 1, p. 68, 2013.

[137] D. Gollmann, "Analysing security protocols," in *Formal Aspects of Security*. Springer, 2002, pp. 71–80.

[138] K. Jithendra and T. Shahana, "New results in related key impossible differential cryptanalysis on reduced round aes-192," in *2018 International Conference On Advances in Communication and Computing Technology (ICACCT)*. IEEE, 2018, pp. 1–5.

[139] A. Bar-On, O. Dunkelman, N. Keller, E. Ronen, and A. Shamir, "Improved key recovery attacks on reduced-round aes with practical data and memory complexities," *Journal of Cryptology*, vol. 33, no. 3, pp. 1003–1043, 2020.

[140] Cryptography.com, "Rijndael mix columns," https://cryptography.fandom.com/wiki/Rijndael_mix_columns, 2020.

[141] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata, "The 128-bit blockcipher clefia," in *International workshop on fast software encryption*. Springer, 2007, pp. 181–195.

[142] N. H. Bshouty, N. Diab, S. R. Kawar, and R. J. Shahla, "Enumerating all the irreducible polynomials over finite field," *arXiv preprint arXiv:1602.05032*, 2016.

[143] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1727–1765, 2016.

[144] h. y. microchip.com, title=Network and Accessories secure authentication.

[145] h. y. wikipedia.org, title=Big O notation.

[146] S. Gayathri Devi, K. Selvam, and S. P. Rajagopalan, "An abstract to calculate big o factors of time and space complexity of machine code," in *International Conference on Sustainable Energy and Intelligent Systems (SEISCON 2011)*, 2011, pp. 844–847.

[147] W. Sun, J. Liu, Y. Yue, and Y. Jiang, "Social-aware incentive mechanisms for d2d resource sharing in iiot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5517–5526, 2020.

[148] I. Boškov, H. Yetgin, M. Vučnik, C. Fortuna, and M. Mohorčič, "Time-to-provision evaluation of iot devices using automated zero-touch provisioning," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–7.

[149] B. Donassolo, I. Fajjari, A. Legrand, and P. Mertikopoulos, "Load aware provisioning of iot services on fog computing platform," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

Appendix A

Author's publications

1. **J. N. Mamvong**, G. L. Goteng, B. Zhou, and Y. Gao, "Efficient Security Algorithm for power-constrained IoT devices," IEEE Internet of Things Journal, vol. 8, no. 7, pp. 5498–5509, 2021

2. **J.N.Mamvong**., G.Goteng, Y.Gao, "Low-cost Client-side Encryption for Secure Internet of Things Provisioning" Frontiers of Computer Science Journal https://doi.org/10.1007/s11704-022-1256-9, 2022.