Dynamic Reduced-Round TLS Extension for Energy-Saving Encryption in Wireless IoT Communications

Quentin Varo

A Thesis
in
The Department
of
Concordia Institute for Information Systems Engineering (CIISE)

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science (Information Systems Security)
Concordia University
Montréal, Québec, Canada

December 2020

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Quentin Varo**

Entitled: **Dynamic Reduced-Round TLS Extension for Energy-Saving Encryption in Wireless IoT Communications**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Information Systems Security)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

______________________ Chair
Dr. Suryadipta Majumdar

______________________ Examiner
Dr. Aiman Hanna

______________________ Examiner
Dr. Suryadipta Majumdar

______________________ Supervisor
Dr. Jun Yan

Approved by ______________________
Dr. Abdessamad Ben Hamza, Graduate Program Director

December 2020 ______________________
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Dynamic Reduced-Round TLS Extension for Energy-Saving Encryption in Wireless IoT Communications

Quentin Varo

Securing the wireless Internet of Things (IoT) is a complex challenge due to devices' computing capacity limitations, battery restrictions or insufficient power supply. Reaching 30 billion connected devices in 2020, the IoT sector is booming. According to marketing studies, by 2025, the global IoT market is expected to reach $34.4 billion and the global IoT battery market is estimated to growth to $15.8 billion. Nevertheless, the smart city, connected healthcare, Industry 4.0 and home security, representing over 75% of the IoT market, raise critical cybersecurity and energy consumption issues. The battery lifespan of specific devices such as Wireless Sensor Networks (WSNs), Wearable or Implantable Medical Devices (WMDs, IMDs) can then be drastically impacted. To meet emerging demands, new solution to provide both cybersecurity and energy efficiency must be developed. Hence, this thesis research tried to develop a dynamic and secure solution to balances communication security and power consumption according to the IoT device's current battery level and the reduced-round cryptography. The contributions are as follow: (1) the security and power consumption evaluation of reduced-round cryptography on different lightweight ciphers; (2) the design, and implementation of a dynamic mechanism to control the battery discharge by adjusting the communication encryption cipher reduced-round value; (3) the design, integration and evaluation of our dynamic reduced-round mechanism integrated within TLS protocol version 1.2 and 1.3. The results of the two experiments confirm the efficiency of the reduced-round cryptography and of our dynamic round-reduced TLS extension to achieve a trade-off between IoT's communications security level and energy savings.

Acknowledgments

I would like to thank all those that participated and supported me during the development of my thesis.

First of all, I would like to sincerely thank my supervisor, Dr. Jun Yan from Concordia Institute for Information Systems Engineering, for giving me the opportunity to join his research laboratory, for believing in my potential and for his investment in my research project. Through this experience I have discovered and greatly improved my understanding of my field of research.

Secondly, I would like to thank Concordia Institute for Information Systems Engineering for all the technologies that I needed and which contributed to the success of this research. This project is funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants RGPIN-2018-06724 and DGECR-2018-00022 and the Fonds de Recherche du Québec-Nature et Technologies (FRQNT) under grant 2019-NC-254971.

Thirdly, I would like to thank my friends. To my roommate and work colleague, William Lardier, for his invaluable assistance during my thesis and for the many hours of exciting work and discussion and the pleasant moments we shared. And to my high school friend and data science engineer, Lucas Sterna, for his kind support and advice during the implementation phase of my project.

Finally, to my parents, my sister, and my girlfriend Marina, who always offered encouragement, moral support and believed in me. For everything you do for me, I want to thank you deeply and I love you all endlessly.

Contents

List of Figures **ix**

List of Tables **x**

1 Introduction **1**

1.1 Motivations . . . 1

1.2 Contributions . . . 2

1.3 Thesis Outline . . . 3

2 Background **6**

2.1 The Internet of Things . . . 6

2.1.1 IoT Systems . . . 7

2.1.2 IoT Applications . . . 8

2.1.3 First Step to Smart Home, Smart Grid and Smart City . . . 10

2.1.4 Battery-Powered & Ressource-Constrained IoT devices . . . 10

2.2 Transport Layer Security Protocol . . . 12

2.2.1 SSL & TLS evolution . . . 12

2.2.2 TLS Structure: Handshake & Application Data subprotocols . 15

2.2.2.1 Handshake Negotiation . . . 16

2.2.2.2 Application Data . . . 18

2.2.3 Porting TLS on IoT devices . . . 18

2.3 IoT Energy and Security Challenges . . . 20

2.3.1 IoT Energy Challenges . . . 22

2.3.1.1 IoT Lightweight Protocols . . . 22

2.3.1.2 Strengths & Limits of these Protocols . . . 26

2.3.2 IoT security Challenges . . . 26

2.3.2.1 Lightweight Encryption Algorithms 27
2.3.2.2 Strengths & Limits of the Lightweight Cryptography 29

3 State of The Art 31
3.1 Lightweight Protocol Improvements 31
3.1.1 Application protocol 31
3.1.2 TLS protocol 32
3.2 Lightweight Encryption Improvements 34
3.2.1 Symmetric-key Encryption 34
3.2.2 Public-key Encryption 35
3.3 Green IoT Guideline 36

4 Reduced-Round Cryptography 38
4.1 Objectives & Security Requirements 38
4.1.1 Encryption Round 38
4.1.2 Reduced-Round Objectives 39
4.1.3 Encryption Cipher Security 39
4.1.4 Security Levels Suggestions 41
4.2 Lightweight Ciphers & Cryptanalysis Overview 42
4.2.1 SPN, FN Block Ciphers & Stream Cipher Definition 43
4.2.2 SPN-Based Ciphers 43
4.2.2.1 AES and AES-like 43
4.2.2.2 Bit-sliced S-Boxes 44
4.2.2.3 ARX 45
4.2.2.4 Others 45
4.2.3 FN-Based Ciphers 46
4.2.3.1 Two-Branched FN 46
4.2.3.2 Generalized FN 46
4.2.3.3 ARX 48
4.2.4 Stream Ciphers 53
4.2.4.1 ARX 53
4.2.4.2 Bivium-like 53
4.3 Benchmark & Results 54
4.3.1 Experiment 54

4.3.2 Implementation 55
4.3.3 Results 55
4.3.3.1 Encryption Power Consumption 55
4.3.3.2 Battery Savings 56
4.4 Discussion 59

5 Dynamic Reduced-Round TLS extension 61
5.1 Reduced-Round TLS extension 61
5.1.1 The Handshake Protocol 62
5.1.1.1 Inside *ClientHello* 62
5.1.1.2 Inside *ServerHello* 63
5.1.2 The Application Data Protocol 63
5.1.2.1 Dynamic Round Update 65
5.1.2.2 Round Configuration Update 66
5.1.2.3 Policy Update 67
5.1.2.4 UpdateResponse Message 69
5.2 Dynamic Round Number Mechanism 70
5.2.1 Single Fixed Policy ($0x0$) 70
5.2.2 Proportional Policy ($0x1$) 70
5.2.3 Multiple Fixed Policy ($0x2$) 71
5.3 PoC Implementation 72
5.3.1 WolfSSL Framework Overview 72
5.3.2 TLS Protocol Implementation 73
5.3.2.1 Handshake 73
5.3.2.2 Application Data 77
5.3.3 Dynamic Reduced-Round Mechanism 82
5.3.3.1 AES Reduced-Round Encryption 82
5.3.3.2 Operator Policies 83
5.4 Scenarios & Results 86
5.4.1 Experiments 86
5.4.2 Results 86
5.4.2.1 Power Consumption within TLS 86
5.4.2.2 Battery Savings 87
5.5 Discussion 88

6 Discussions **90**

7 Conclusion **93**

List of Figures

1 Thesis Organization 5

2 TLS Scheme 19

3 Power consumption comparison of lightweight ciphers in (a) full and (b) minimum rounds. 57

4 Power gain comparison between lightweight ciphers full and minimum round for 4KB/s data encryption. 58

5 Reduced-round Handshake protocol based on TLS 1.3 64

6 Reduced-round Dynamic Round Update on TLS 1.3. 66

7 Reduced-round Round Configuration Update on TLS 1.3. 67

8 Reduced-round Known Policy Update on TLS 1.3. 68

9 Reduced-round Unknown Policy Update on TLS 1.3. 69

7 Wireshark Capture of the *ClientHello* message 74

8 Server Instance *ClientHello* message reception and extension parsing 75

9 Wireshark Capture of the *ServerHello* message 76

10 Client Instance *ServerHello* message reception and extension parsing 76

11 Update Round functionality Example 78

12 Update Minimum Round functionality Example 80

13 Update Policy functionality Example 82

14 AES Reduced-Round Encryption debugging example 83

15 Proportional Policy round computation example 84

16 Update Policy functionality Example 85

17 Power consumption of Reduced-Round TLS Extensions Scenarios with AES-128 on TM4C1294 Connected Launchpad. 87

18 Power gain between Reduced-Round TLS Extensions Scenarios with AES-128 compared to full AES-128. 88

List of Tables

1 IETF RFC7228 Constrained Devices Classification 11
2 TLS Libraries for Embedded Devices Comparison 21
3 Security suggested in relation with the IoT category 41
4 Lightweight Ciphers Parameters 42
5 SPN Ciphers Cryptanalysis 47
6 GFN & Two Branched FN Ciphers Cryptanalysis 49
7 ARX FN Ciphers Cryptanalysis (part1) 51
8 ARX FN Ciphers Cryptanalysis (part2) 52
9 Stream Ciphers Cryptanalysis Summary 54
10 FELICS' Ciphers version used 56
11 Reduced-Round Benchmark Results 60
12 *UpdateRound* Message Format 77
13 *UpdateMinRound* Message Format 79
14 *UpdatePolicy* Message Format 81

Chapter 1

Introduction

1.1 Motivations

Known as one of the major evolutions of the Internet, the Internet of Things (IoT) has become an indispensable technology for professionals and individuals since 2008 [1]. Business of connected objects is flourishing; embedded with sensors these devices can effectively monitor systems and collect data. IoT devices are present in a wide variety of sectors such as industry, transportation, agriculture, security, smart grids or health. The development of smart homes, smart girds and smart cities around the world is based on the interconnection of sensors, meters and actuators communicating data and remotely controllable via the Internet.

Nevertheless, IoT equipment has a wide variety of designs, most of which are miniaturized, wirelessly connected and battery-powered systems. These low consumption systems are also resource-limited: computing power, memory, storage, and battery capacity. As a result, traditional security systems, which require too much capacity, cannot be directly applied to the connected objects. Connected devices must deal with threats that affect their data security and users' privacy. Despite the NISTIR 8259 guideline published by the National Institute of Standards and Technologies (NIST) [2], security analyses report that the majority of IoT devices have vulnerabilities and are not secure [3].

The rapid growth of IoT around the world is an important preoccupation. Despite their low individual footprint, IoT devices have a significant impact on their global power consumption. The International Electricity Agency (IEA) produced a report on Energy efficiency of the Internet of Thing in which it estimates an alarming annual increase in IoT consumption of 20% [4].

Offering solutions to secure and reduce the energy consumption of IoT devices is a key issue. However, the diversity of IoT devices (architecture, uses, accessibility) requires finding flexible solutions that can be integrated within these systems. In an effort to address these concerns, this work aims to propose a new solution to secure IoT communications based on a dynamic and lightweight mechanism balancing the security level according to the battery level of the IoT device.

1.2 Contributions

How to enable the security of IoT communications in flexible ways adapted to the variety of connected objects while ensuring energy and resource savings? Through this research, we will try to answer this problematic by introducing a dynamic and lightweight mechanism based on the IoT battery and an operator-chosen policy to balance security and power consumption for IoT communications.

This thesis project aims to propose a new dynamic security mechanism for battery-powered IoT devices embedded in the Transport Layer Security (TLS) as an extension varying the encryption round number of the encryption algorithm used depending on the current battery level and on an operator-selected policy. This solution allows connected objects to secure their communications using the standard security protocol RFC 8446 [5], TLS version 1.3, while minimizing the energy and resource consumption during the encryption/decryption process.

The contributions of this research are the following:

- We suggest adapting the security level according to the IoT device type and its application. We conducted a security study of 23 lightweight ciphers and 52 different versions based on the most up-to-date cryptanalysis to determine the security of these reduced-round encryption algorithms depending on their robustness. For each variant of an encryption algorithm studied, the minimum number of reduced-round ciphers is determined according to its computational complexity and its encryption key size.

- To measure the energy consumption and energy savings of each reduced-round lightweight cipher, we performed a benchmark on a constrained IoT platform using the Fair Evaluation of Lightweight Cryptographic Systems (FELICS) project [6]. All the cipher algorithms studied were run in full and minimum round on the Texas Instrument launchpad MSP430FR2355 to evaluate the possible gains in data encryption and decryption. Results were compared individually and among all ciphers to determine the most energy-efficient algorithms.

- We present our TLS dynamic reduced-round extension design. First, we detail the implementation and functioning of the reduced-round extension within the handshake negotiation and the application data within TLS protocol. Secondly, we describe the dynamic round number mechanism and define the round determination algorithms according to the established policy.

- We implemented the first proof of concept (PoC) of our dynamic reduced-round TLS extension and embedded our solution on a real world wireless IoT device. Using WolfSSL embedded library we developed the reduced-round TLS extension suitable for TLS versions 1.2 and 1.3 on the Texas Instrument Tiva C TM4C1294 connected launchpad. We developed different usage scenarios for our reduced-round extension between our IoT platform (client) and a server (operator) using AES-128 bits and TLS1.3 and communicating over the Wi-Fi 802.11 network. We measured the IoT device consumption for each scenario and analysed the associated battery gains.

Co-authored with William Lardier, our first dynamic reduced-round mechanism has been published at the 2020 IEEE International Conference on Communications (ICC) [7]. The cryptanalysis overview of the lightweight cipher, described in the Chapter 4, the improvements of the reduced-round mechanism and the reduced-round TLS extension, described in the Chapter 5, have been submitted to the IEEE Transactions on Green Communications Networking (TGCN).

1.3 Thesis Outline

The thesis report is organized in seven chapters. Chapter 1 introduces the subject and outlines the thesis elements. Chapter 2 describes the Internet of Things, TLS protocol, the IoT energy and security challenges and details the technical components covered throughout this work. Chapter 3 presents the state of the art of both lightweight protocols and encryption solutions and provides a guideline for green IoT solutions. Chapter 4 presents reduced-round cryptography with an overview of lightweight ciphers cryptanalysis and a performance analysis of these encryption algorithms through a benchmark on a constrained IoT device. Chapter 5 presents our dynamic reduced-round TLS extension solution, details its design and implementation within the TLS protocol (version 1.2 and 1.3) using secure data update procedures, describes the policy based dynamic round number mechanism and evaluates the benefits of this solution through embedded scenarios on a real world IoT device. Chapter 6 will discuss the limits of this research project and define future work axes. Lastly, Chapter 7 will conclude and summarize the contributions of this thesis. Figure 1 illustrates the organization of the thesis.

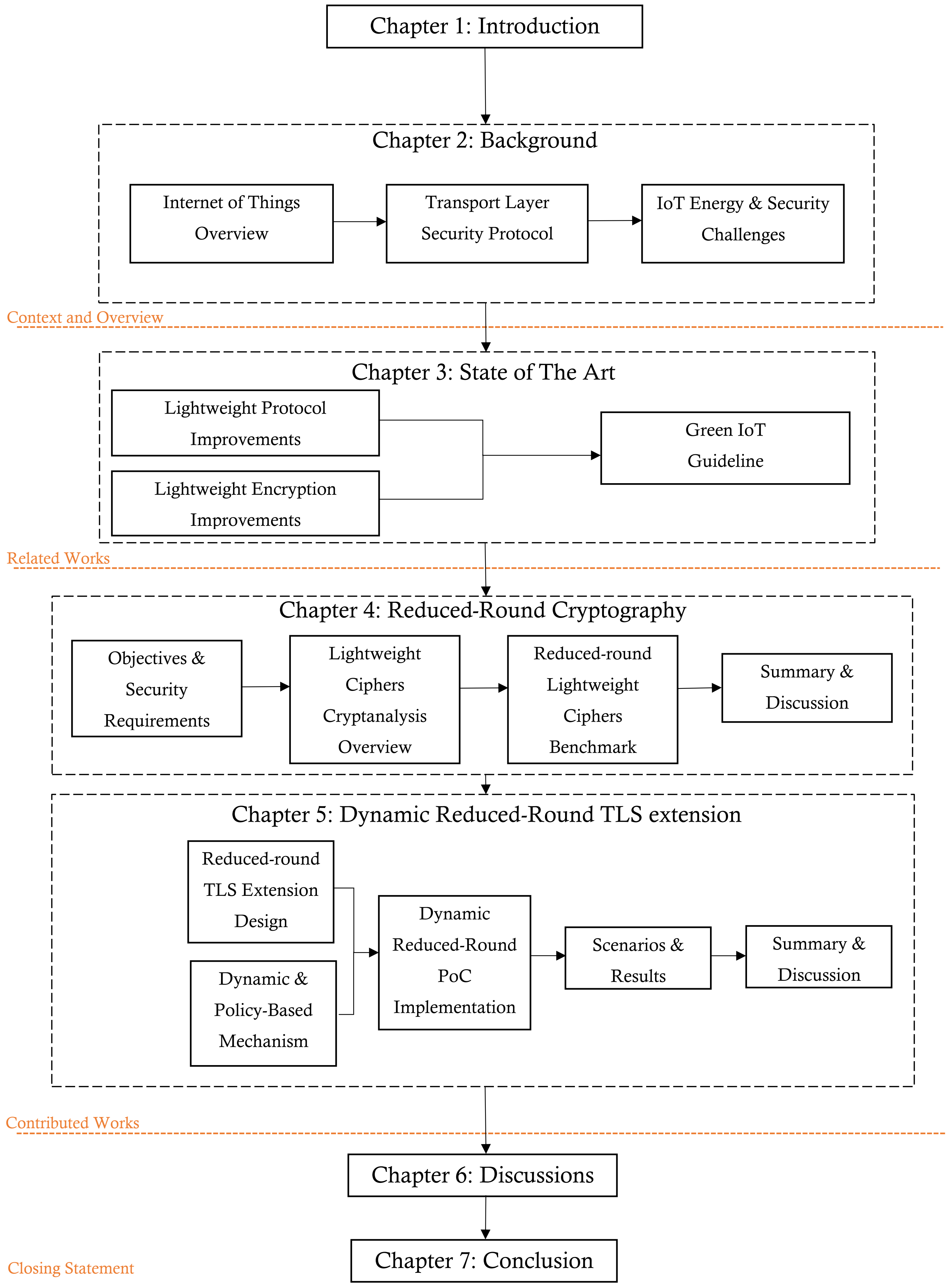

Chapter 1: Introduction
Chapter 2: Background
Internet of Things Overview
Transport Layer Security Protocol
IoT Energy & Security Challenges
Context and Overview
Chapter 3: State of The Art
Lightweight Protocol Improvements
Lightweight Encryption Improvements
Green IoT Guideline
Related Works
Chapter 4: Reduced-Round Cryptography
Objectives & Security Requirements
Lightweight Ciphers Cryptanalysis Overview
Reduced-round Lightweight Ciphers Benchmark
Summary & Discussion
Chapter 5: Dynamic Reduced-Round TLS extension
Reduced-round TLS Extension Design
Dynamic & Policy-Based Mechanism
Dynamic Reduced-Round PoC Implementation
Scenarios & Results
Summary & Discussion
Contributed Works
Chapter 6: Discussions
Closing Statement
Chapter 7: Conclusion

Figure 1: Thesis Organization

Chapter 2

Background

2.1 The Internet of Things

The Internet of Things (IoT) is defined as the network of connected objects. Initially, the Internet of Things is intended to allow physical objects to communicate automatically their states via the Internet network. Using this capability, IoT devices can monitor a system or environment using sensors and communicate their information on small data packets. Thanks to the evolution of machine learning and embedded electronics, smart objects are able to understand events (awareness), use applications (representation) and interact with users or other machines (interaction) [8]. In 2020, about 31 billion devices are connected to the Internet and the International Data Corporation (IDC) predicts an increase in the number of devices to 41.8 billion by 2025 [9].

The data digitization and task automation has increased the implementation of Industrial IoT (IIoT) devices in companies motivated by the Industrial Internet Consortium's (IIC) and Industry 4.0 [10, 11]. IIoT devices are capable of communicating between machines using Machine-to-Machine (M2M) technologies, forming a Wireless Sensor Network (WSN), using Radio Frequency Identification (RFID) technologies or constituting a Supervisory Control and data acquisition (SCADA) environment for an Industrial Control System (ICS).

For companies and manufacturers, IoT and IIoT equipments offer numerous optimizations:

- Connectivity, availability, redundancy and resilience;

- Data management, providing the efficient data analysis autonomous action in real time;
- Optimize decision-making procedures with the participation of machine learning systems;
- Reduced power consumption thanks to IoT platforms' miniaturization.

2.1.1 IoT Systems

The IoT environment is complex and diversified. However, it can be divided into 4 major systems:

- The M2M inter-machine communication technologies allowing connected objects to communicate automatically with each other and without human interaction. Mostly wireless M2M communications are based on short-range protocols such as RFID, Bluetooth, medium-range protocols such as Wi-Fi or long-range protocols such as the Global System for Mobile Communications (GSM) 3G, 4G and 5G protocols.
- The RFID system allows passive devices called tags composed of a chip and an antenna to transmit data on short range (1 to 15 meters) and without contact to a reader. These systems are mostly used to secure and control physical access and to ensure the traceability of unconnected objects.
- The WSNs consisting of a wireless ad hoc network (WANET) of low power consumption battery-powered micro sensors capable of autonomously exchanging data to collection servers. The wireless communication standards used by WSNs are mostly Bluetooth and Zigbee protocols (described in Section 2.3.1).
- The SCADA system allows companies to monitor and control industrial data on location and remotely. SCADA systems define a software and hardware package consisting of Human-machine Interface (HMI), remote terminal units (RTU), distributed Control Systems (DCS), programmable logic controller (PLC) and Cyber-Physical System (CPS).

2.1.2 IoT Applications

Nowadays IoT is used in almost every field: professional, private and military. This equipment performs various tasks of data processing, task automation, predictive analysis, monitoring and surveillance. In this subsection we will make a non-exhaustive list of some application areas and give examples of IoT devices usage.

Industry & Manufacturing: IoT's development in the industrial sector permitted improvements in predictive analysis for machine maintenance and production estimation. With the help of real time asset feedback, remote monitoring and control and M2M communications from IoT devices, information systems are able to predict events in advance and automatically optimize supply chain management. In order to assist companies to achieve their digital transformation and to implement connected objects, Microsoft has developed a management platform called Microsoft Azure IoT Platform [12].

Logistic: Embedded on cargo ships and trucks, connected devices allow fleet management to track and protect the merchandise during the transportation. The major functionalities introduced by the IoT devices in Logistics are :

- GPS real-time tracking fleet;
- Monitoring the merchandise weight;
- Inventory control in the warehouses;
- Autonomous Drone delivery.

Agriculture: The deployment of WSN and IoT devices in fields and farms permitted the optimization and monitoring of harvests and prediction and automation of farm processes. The main features brought by IoT in Agriculture are :

- Monitoring of temperature, moisture and mould in soils;
- Predict water and nutrient requirements and predict optimum harvest time to optimize crops and reduce losses;
- Automate watering and soil fertilization.

Automotive & Transport: Connected vehicle development through IoT offers opportunities to provide vehicles and means of transport capable of diagnosing their condition, communicating from vehicle to vehicle (V2V communication) to alert and assist drivers and to drive autonomously. V2V communication also called vehicular ad hoc network (VANET) allows inter-vehicle communication to obtain real-time traffic information and enable emergency systems such as Emergency Electronic Brake Light Warning to alert drivers when a vehicle ahead suddenly brakes.

Building & Home Security: Through sensors, IP camera and RFID tag reader, connected devices are capable of providing protection and surveillance systems and intrusion or natural disaster warning for homes and buildings. Key functionalities provided by the IoT for building security are :

- Real-time monitoring and motion alert through IP Camera;
- Access control and restrictions using RFID door lock system;
- Door, window and shutter security using electromagnetic, ultrasonic and laser sensors able to raise real-time alerts by SMS or mail;
- Protection against natural disasters using fire and water leakage sensors able to alert the fire department automatically.

Healthcare: Known as e-health, digital health refers to the development of connected objects to treat, cure and monitor diseases. Implantable medical devices (IMD) are IoT devices totally or partially introduced into a patient's body through a surgical intervention. Among the different IMDs we can denote :

- Pacemaker and implantable cardioverter defibrillator (ICD) place in the chest to regulate heartbeat frequency by sending electric pulses. These cardiac devices are used to treat tachycardia;
- Drug delivery system (DDS) placed under the patient's skin to automatically inject a dose of medication into the patient's bloodstream. These medical systems are used to treat cancer, HIV, diabetes and other chronic diseases;
- Implantable neurostimulator electrodes are placed on the spinal cord, the pelvic nerve and the stomach to send low amplitude signal into the patient body. These electrodes are used to treat cases of tumors, epilepsy, parking and other chronic diseases.

Sport & Wellness: Many of the sports and wellness connected objects have been developed as wearable devices. These small, wireless, battery-powered smart devices are worn like bracelets and watches (smart watch and smart bands), clothes (e-textile), shoes or accessories (caps, glasses, jewelry). This equipment allows users to follow their physical activity, analyze their efforts, track their movements and monitor the quality of their sleep through interactive applications.

2.1.3 First Step to Smart Home, Smart Grid and Smart City

IoT is a key component in the creation of intelligent infrastructure, allowing remote monitoring and control of systems, autonomous prediction and response to events using massive data analysis. Thus, IoT technology is the basis for the development of complex systems such as Smart Home, Smart Grid and Smart City.

Smart Home: Refers to a house or building equipped with connected objects capable of communicating with each other, automating tasks in the home, remotely analysing information and predicting events. Smart Home aims to improve the inhabitants' quality of life and to optimize household tasks such as cooking and housework; improve physical security; reduce water, electricity and heating consumption.

Smart Grid: Refers to the development of connected equipment capable of monitoring and controlling the power grids responsible for the energy delivery from the power plant to the end user. The Smart Grid model allows to connect the power network and the IT network together on a two-way secure communication. Smart Grid is designed to improve power distribution, response to incidents and ensure the power grid's resilience.

Smart City: Refers to the development of Smart Devices, Smart Homes and Smart Grid communications at the city scale. Smart City's design encompasses all application domains of the IoT to improve the functioning of a city. Smart Cities aim at optimizing traffic and transportation management, reducing pollution, energy and water consumption and improving the quality of urban services.

2.1.4 Battery-Powered & Ressource-Constrained IoT devices

Compared to the traditional workstation, IoT devices are designed to perform efficiently a specific task. Typically, IoT devices architecture has been miniaturized and are battery powered to be portable and mobile in order to facilitate their implementation in complex environments such as IMD, wearable and WSN devices. According to an IAE report over 33% of IoT devices are battery powered, consuming 7 billion batteries in 2020 [4]. Battery powered IoT equipments are becoming an ever-growing design standard. The IoT battery market, which reached \$9.2 billion in 2020, is estimated at \$15.9 billion for 2025 according to a market report [13].

Furthermore, the miniaturization of the connected objects, the power consumption reduction and their single-tack function have been optimized to operate on limited capacity architectures. Limited connected devices are designed with limited computing power, memory and storage space. In order to identify most resource-constrained devices the Internet Engineering Task Force (IETF) standardized and classified the constrained devices within the RFC 7228 [14]. This document defines the constrained devices, node and network, describes the challenges associated with these network constraints and classifies the constrained devices through three classes. Each class defines a type of constrained device according to its memory and storage space:

- *Class 0:* Representing sensors with very few resources (such as WSN) that are difficult to secure and do not support traditional network protocols;
- *Class 1:* Representing constrained devices that can use network and security protocols but require attention to their capacity;
- *Class 2:* Representing the less constrained devices that can be deployed using the same protocols as servers.

The Table 1 represents the classification of constrained IoT devices performed by the IETF in RFC 7228.

Table 1: IETF RFC7228 Constrained Devices Classification

Class Name	**Data Size**	**Code Size**
Class 0 (C0)	$<$ 10 KiB	$<$ 100 KiB
Class 1 (C1)	$\sim$ 10 KiB	$\sim$ 100 KiB
Class 2 (C2)	$\sim$ 50 KiB	$\sim$ 250 KiB

2.2 Transport Layer Security Protocol

The Transport Layer Security (TLS) and formerly the Secure Sockets Layer (SSL) protocols is responsible of the communication security over the IT network. TLS and SSL are the standard protocols that secure Internet connections, email and voice over IP (VoIP). TLS (and SSL) protocol is placed between transport and application layers of the Open Systems Interconnection (OSI) model and runs over Transport Control Protocol (TCP). An adaptation of TLS over User Datagram Protocol (UDP) transport protocol has been developed under the name Datagram Transport Layer security abbreviated by DTLS. DTLS ensures secure communication over UDP and must resolve packet loss and reordering. Based on a client-server model TLS (and SSL) protects both parties against eavesdropping, data tampering or hijacking. TLS protocols provides the following security:

- *Confidentiality* of transmitted data through symmetrical data encryption of communications and robust encryption algorithms;
- *Integrity* of the communicated data through message digest calculation and hashing algorithms;
- Server and client *Authentication* (optional) through public key encryption involving Certification Authorities (CA) (as a third party) issuing certificates to prove the authenticity of both parties. SSL and TLS evolution;
- Session *Forward Secrecy* guaranteeing that both communications made and the session key cannot be retrieved if an adversary compromises one of the private keys in the future. Forward secrecy is assured by the Diffie-Hellman ephemeral key-exchange function.

2.2.1 SSL & TLS evolution

This subsection details the evolution of SSL and its successor TLS through version release upgrades.

SSL was created by Netscape under the direction of T. Elgamal since 1994.

SSL1.0 (1994): First version of SSL that was never publicly revealed because of major security flaws: no data integrity protection and vulnerable to replay attack (no nonce).

SSL2.0 (1995): First version of SSL publicly released and described in the Internet draft [15]. However, SSL2.0 contains security holes. Like the unprotected negotiation phase (called handshake) between client and server. Or the use of the end-of-connection message from the TCP protocol to close the communication instead of creating an SSL-specific alert message. Because of these flaws, the SSL2.0 protocol is vulnerable to truncation attack forcing the illegitimate SSL communications closure. Support of SSL2.0 was prohibited in 2011 by the IETF under RFC 6176 [16].

SSL3.0 (1966): The latest version of SSL published in RFC 6101 [17]. This 3rd version of SSL adds the SHA-1 hash function used with the MD5 algorithm to strengthen data integrity control. SSL3.0 generalizes the key exchange protocol (Diffie-Hellman and Fortezza key exchange and Rivest–Shamir–Adleman (RSA) no certificate. SSL3.0 adds the chain of trust that establishes a hierarchy of certificate authorities to reinforce authentication analysis security. In addition, closing alert messages are added to protect against truncation attacks. However, the structure of SSL3.0 is obsolete and vulnerable to Browser Exploit Against SSL/TLS (BEAST) and Padding Oracle On Downgraded Encryption (POODLE) attacks.

BEAST attack: BEAST exploits a vulnerability in the vector initialization (IV) within the block cipher in Cipher Block Chaining (CBC) mode. IV initialization allow to randomize identical messages encrypted with the same key using a seed (robust against chosen plaintext attack). However, in SSL3.0, IV value is the last block of the previously encrypted message (not random). Anyone who has intercepted an encrypted message knows the IV of the next message and can guess the plaintext through the record splitting attack.

POODLE attack: POODLE attack shows that CBC mode is vulnerable to block padding. This security flaw allows attackers to decrypt messages byte by byte without knowing the encryption key or the cryptographic primitive used.

As a result of these attacks, IETF decided to prohibit SSL3.0 published under RFC 7568 [18].

TLS1.0 (1999): The first version of SSL's successor, TLS is developed by the IETF and TLS1.0 is published in RFC 2246 [19]. Based on SSL3.0 model, it keeps the same structure and improves some features such as :

- Added key derivation function (KDF): Diffie-Hellman key exchange with digital signature standard (DH-DSS);

- Message Authentication Code (MAC) improvement: Added key-hashing for message authentication code (HMAC);
- Added more TLS alert messages;
- Support of 3DES (Triple DES) encryption algorithm.

Nevertheless, security reports have shown TLS1.0 is vulnerable to downgraded attack forcing the use of weak version of SSL [20]. As with SSL3.0, TLS1.0 is vulnerable to POODLE and BEAST attacks.

TLS1.1 (2006): Defined in RFC 4346 [21], TLS version 1.1 improves the security of the protocol against BEAST and POODLE attacks by randomizing the value of IVs and suppressing block padding error messages in CBC mode and exiting the session. TLS1.1 adds Internet Assigned Numbers Authority (IANA) registers for TLS parameters. However, TLS1.1 even without an error message the protocol remains vulnerable to POODLE attack. Adversaries can perform timing attacks: even if the client re-instantiate a new session after each block padding error, the secret remains on the same location inside messages.

In 2018, Microsoft, Google, Apple and Mozilla agreed to avoid and deactivated TLS1.1 and prior version compatibility by 2020. The IETF has already written a draft to announce the TLS 1.0 and TLS1.1 deactivation [22].

TLS1.2(2008): Detailed in RFC 5246 [23], TLS1.2 includes major improvements in both functionality and security :

- Added cipher suite: AES Galois/Counter Mode (GCM) and Counter with CBC-MAC (CCM), Camellia GCM and Aria GCM block cipher and Chacha20 stream cipher with Poly1305 MAC;
- Removed insecure cipher suites: IDEA CBC and 3DES CBC;
- Use of a specified cipher suite (using SHA-256) pseudorandom function (PRF) instead of MD5 and SHA-1;
- Digital signed element are signed with a single hash chosen during the handshake instead of MD5 and SHA-1;

- Added TLS extensions.

CRIME attack: In 2012 researchers found a vulnerability on the data compression inside TLS protocol version 1.2 and below. Compression Ratio Info-leak Made Easy (CRIME) is a client side exploit that allow attackers to recover authentication cookies and perform session hijacking over HTTPS communications. Using chosen plaintext, a man-in-the-middle (MITM) attacker can forge cookies with random bits to compare the compression size with the encrypted victim cookie.

TLS1.3(2018): Last update of the protocol, TLS 1.3 is published in RFC 8446. Major updates include :

- Removal of all obsolete encryption algorithms. Only AES GCM and CCM block ciphers and Chacha20-POLY1305 stream cipher have been kept;
- Data integrity is checked only with authenticated encryption with associated data (AEAD) algorithms. Removal of HMAC-MD5 and HMAC-SHA-1;
- Messages Encryption inside handshake negotiation after sending ServerHello message;
- Keep only ephemeral KDFs to ensure forward secrecy;
- Performance improvement: TLS 1.3 handshake use 1 round trip (1-RTT) instead of 2-RTT for TLS1.2. An optional 0-RTT mode can be used;
- Removed the DEFLATE compression function (against CRIME attack).

2.2.2 TLS Structure: Handshake & Application Data subprotocols

The design of TLS protocol can be divided into two distinct phases. First, the handshake phase, responsible of both parties authentication and the security settings configuration. Second, the data application record phase responsible for the secure transport of data between the client and the server using the security parameters defined during the handshake.

2.2.2.1 Handshake Negotiation

Once a TLS connection is initiated, the handshake process between the client and the server is started. During this phase, the client and server authenticate to each other and decide which version of TLS, encryption algorithm and MAC will be applied and generate the symmetric encryption keys used to encrypt/decrypt communications. The TLS handshake procedure follows these steps:

1. ***ClientHello* message:** The client sends a first message to the server. This message contains the highest TLS version supported, the supported cipher suites list and a random value client. Optionally it contains the list of supported compression functions (TLS1.2 and below), a session identifier and a list of extensions to use (TLS1.2 and above).

2. ***ServerHello* message:** When the server receives the *ClientHello* message it replies with a *ServerHello* message. This message contains the protocol version chosen, the cipher suite and MAC algorithms selected, a session identifier and a server random data. Optionally, the message contains the chosen compression method (TLS1.2 and below) and the list of extensions to use (TLS1.2 and after).

3. ***ServerCertificate* message:** The server sends its certificate to the client.

4. ***ServerKeyExchange* message (optional):** If the key exchange method use is DHE_DSS, DHE_RSA or DH_anon the server sends a *ServerKeyExchange* message. The server generates the public key from its private key and sends it to the client.

5. **Server *CertificateRequest* message (optional):** The Server request a certificate from the client for authentication.

6. ***ServerHelloDone* message:** The server notifies the client it has finished sending its messages: *ServerHello*, *ServerCertificate* and *ServerKeyExchange*.

7. ***ClientCertificate* message (optional):** If the client receives the server *CertificateRequest* message, the client sends its certificate to the server.

8. ***ClientKeyExchange* message:** The client generates the pre-master key and encrypts it with the public key contained in the server certificate. The message is

then encrypted with the server's public key (received in the *ServerKeyExchange*) and sent to the server.

9. **Client *CertificateVerify* message (optional):** This message is sent if the server has sent a *CertificateRequest* message to verify the client's certificate (Except if the client has sent a certificate containing fixed Diffie-Hellman parameters).

10. **Client *ChangeCipherSpec* message:** The client generates the symmetric encryption key (called session key) using the client and server random values, its private key and the server's public key. Then, the client notifies the server it has computed the encryption key and authentication and message encryption will start.

11. **Client *HandshakeFinished* message:** Based on previous handshake messages, the client generates a hash and encrypts it with the symmetric key. The encrypted hash is sent to the server to verify the negotiation was successful and there was no message alteration.

12. **Server *ChangeCipherSpec* message:** The server generates the symmetric encryption key (called session key) from the random value server and client, its private key and the client's public key. Then the server indicates to the client it has computed the encryption key and authentication and message encryption will start.

13. **Server *HandshakeFinished* message:** The server generates a hash from the handshake messages and encrypts this signature with the symmetric key. The encrypted hash is sent to the client as a verification data.

Once these steps have been successfully completed, secure TLS communication is established between the client and the server.

Remarks :

- If an error occurs during the handshake procedure, an alert message specifying the error type is sent and the session is closed;

- Within TLS1.3, messages are encrypted in the handshake procedure following the *ServerHelloDone* message transmission. In addition, both Client and Server *ChangeCipherSpec* messages are removed from the handshake procedure.

2.2.2.2 Application Data

During this phase, client and server can exchange data as a data application using a secure session a tunnel that guarantees end-to-end security. The format of a data record message application is as follows:

- A *Record Header* containing :
  - *Data Type*, $< 0x17 >$ for Application Data;
  - *TLS protocol version*, $< 0x3\ 0x0...4 >$ for TLS1.0 to 1.3;
  - *Data Application Size*.
- The encrypted *application data*
- The *MAC* signature
- The *padding trailer* (optional)

Figure 2 illustrates TLS protocol through the handshake procedure and application data exchanges.

2.2.3 Porting TLS on IoT devices

TLS is mainly used for Internet communication security and is responsible for HTTP communication encryption with the HTTPS protocol (or HTTP over TLS). Several solutions to implement TLS protocol are proposed, most of them are free and open-source. The most widely used TLS implementation on the web is the OpenSSL library developed by The OpenSSL Project [24]. An analysis report estimates that about half of the companies in the world use OpenSSL to secure their networks and websites [25]. However, OpenSSL can only be deployed on Linux, BSD, Mac OS and Windows machines and is resource consuming. The main purpose of this library is to secure servers on the Internet and is not adapted to limited architectures used in connected objects.

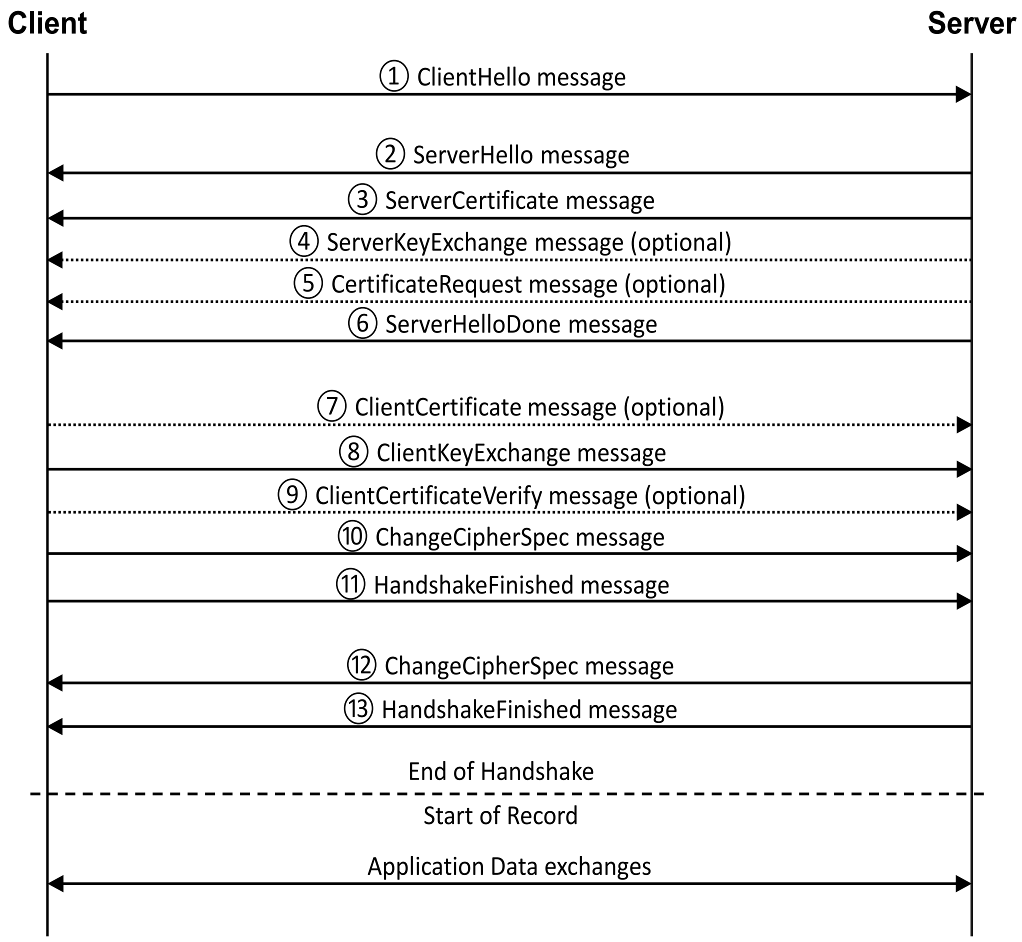

Client
Server
① ClientHello message
② ServerHello message
③ ServerCertificate message
④ ServerKeyExchange message (optional)
⑤ CertificateRequest message (optional)
⑥ ServerHelloDone message
⑦ ClientCertificate message (optional)
⑧ ClientKeyExchange message
⑨ ClientCertificateVerify message (optional)
⑩ ChangeCipherSpec message
⑪ HandshakeFinished message
⑫ ChangeCipherSpec message
⑬ HandshakeFinished message
End of Handshake
Start of Record
Application Data exchanges

Figure 2: TLS Scheme

New TLS libraries have been implemented in order to respond to the IoT devices communication security. These libraries (open-source and proprietary) constrained devices oriented try to meet the following criteria:

- Offer the most up-to-date version of TLS with certified secure implementations;
- Minimize the storage and memory footprint associated with the use of the TLS protocol;
- Support multiple IoT-oriented platforms and operating systems.

The Table 2 lists and compares the 5 main IoT based TLS libraries and OpenSSL library.

Among the five major IoT based libraries presented in the Table 2, the two most up-to-date libraries are WolfSSL and Inside Secure TLS toolkit (formerly MatrixSSL) supporting TLS1.3 (from RFC 8446) and DLTLS1.2 (from RFC 6347 [26]). These two libraries can be deployed on most IoT architectures and support a variety of OS and real-time operating systems (RTOS). RTOS systems are reliable, robust and scalable used in many smart devices. WolfSSL and Inside Secure TLS toolkit are compatible with OpenSSL allowing a more efficient transition to these libraries. WolfSSL library allocates 20 to 100kB of storage and 1 to 36 KB of memory against 65KB of storage for Inside Secure TLS toolkit. In addition, Wolcrypt (WolfSSL's cryptographic library) source code has obtained several security certifications: FIPS 140-2 level1 for cryptographic implementation [27], DO 178 DAL A for commercial and military avionics [28] and MISRA C for automotive systems [29].

2.3 IoT Energy and Security Challenges

The deployment of IoT and IIoT solutions in the professional and private sectors continues to grow over the past decade. According to the IDC study, the number of IoT devices deployed worldwide in 2025 is expected to reach 41.6 billion devices, indicating an increase of 34.2% in 5 years. A market analysis report on IoT market shows that global IoT sensor market size valued to $9.6 billion in 2018 is expected to grow to $34.4 billion by 2025 [30].

Due to economic market pressures, the security of small connected devices is not taken as priority but rather as an afterthought. NIST considers three major parameters that threaten the security of IoT devices [31]:

- Battery IoT devices rely on constrained resource and power efficient hardware to increase the battery life of the device resulting in poor security mechanism;
- Low cost IoT devices rely on low cost constrained hardware resulting in lack of security system;
- Long lifespan IoT devices increases the risk of being obsolete over time and the security mechanism used will contain critical flaws leading to vulnerabilities that cannot be fixed due to lack of update capability of these devices.

The growing development of IoT devices and market trends are facing two major challenges: IoT power consumption management and security.

Table 2: TLS Libraries for Embedded Devices Comparison

TLS Library	**Supported Platforms**	**Highest Version**	**Validation**	**Code Size & Memory**	**OpenSSL Support**	**License**
WolfSSL	Win32/64, Linux, Mac OS X, Solaris, ThreadX, FreeBSD, NetBSD, OpenBSD, embedded Linux, VxWorks, Micrium, OpenEmbedded, WinCE, OpenWRT, iOS, DevKitPro, QNX, MontaVista, NonStop, µC/OS-III, FreeRTOS, SafeRTOS, TI-RTOS, CMSIS-RTOS, NXP/Freescale MQX, Haiku, RIOT, HP/UX, TRON/ITRON/µITRON, Nucleus, TinyOS, ARC MQX, Yocto, embOS, INtime, Mbed, uT-Kernel, AIX, FROSTED, Green Hills INTEGRITY, Keil RTX, PetaLinux, Apache Mynewt, TOPPERS, uTasker, PikeOS,Deos and Azure Sphere OS	TLS 1.3 DTLS 1.2	FIPS 140-2 DO-178C DAL A MISRA-C	ROM: 20 to 100 KB RAM: 1 to 36 KB	Yes	Double License
Inside Secure TLS Toolkit (formely MatrixSSL)	Win32/64, Linux, VxWorks, Pocket PC, Mac OS X, OpenBSD, FreeRTOS, BREW eCos, VxWorks, uClinux, ThreadX, Palm, pSOS, SMX and Bare-Metal	TLS 1.3 DTLS 1.2	FIPS 140-2	ROM: ∼65KB RAM: NA	Yes	Double License
Mbed	Linux, Win32/64 , Mac OS X, OpenWrt, Android, iOS and FreeRTOS	TLS 1.2 DTLS 1.2	No validation	ROM: ∼30 KB RAM: ∼30 KB	No	Open Source
NanoSSL	Win32/64, Mac OS X, Linux, Monta Vista Linux, VxWorks, Solaris, ThreadX, (ARC) MQX, OSE, Nucleus, pSOS, and Micrium µC	TLS 1.2 No DTLS	FIPS 140-2	ROM: NA RAM: ∼16 KB	Yes	Commercial
SharkSSL	Linux, Win32/64, ThreadX, VxWorks, NTEGRITY, MQX, SMX, embOS, mbed, FreeRTOS, uCLinux, MDK-ARM, Microchip, QNX, Mediatek, lwIP, uIP and Bare-Metal	TLS 1.2 No DTLS	No validation	ROM: ∼20 KB RAM: ∼13 KB	No	Commercial
OpenSSL	Win32/64, Linux, Mac OS X, Solaris, Android, BSD	TLS 1.3 DTLS 1.2	FIPS 140-2	ROM: $\leq$ 1 MB RAM: $\leq$ 160 KB	-	Double License

2.3.1 IoT Energy Challenges

Promoted as a solution to reduce energy consumption, IoT devices consume less energy than traditional computers. As a comparison, an workstation running consumes between 50 and 200 watts while an ESP32, a common IoT wireless sensor device consumes between 528 and 848 milliwatts (in active mode). Nevertheless, the IoT device's consumption depends on the architecture used and the resources allocated. Moreover, the analysis of the power consumption of billions of connected devices has a significant impact. In this context, the IEA has produced a report about the excessive consumption of connected devices in standby mode from 2015 to 2025 (depending on the expected number of devices) [4].

According to their study, IoT devices in standby mode consumed 7.5 TWh in 2015, rising over 20 TWh in 2020 (an evolution of 167% in 5 years) and should reach about 46 TWh in 2025. With 36 TWh consumed only by smart home devices, 7 TWh by smart appliances and 3 TWh by smart lights.

Another energy and pollution concern about IoT is for the battery-powered connect devices. The battery-powered solution is widely used to address IoT deployment needs in restricted environments (underwater, fields and space), wearable devices (IMD and sport devices) or embedded systems (vehicle, drone and robots). According to the IEA, battery-powered IoT device annually consumed 2 billions batteries in 2015, reaching 7 billions in 2020 and will rise to 12 billions batteries in 2025.

Numerous efforts are deployed to reduce the IoT devices' energy consumption such as lightweight communication protocols, deep sleep operation modes, low power memory optimization with ferroelectric RAM (FRAM) or approximate computing for energy-efficient operations. The following section will focus on the development of lightweight protocols for IoT-based communications.

2.3.1.1 IoT Lightweight Protocols

With the objective to extend new resource and energy saving solutions to the variety of IoT devices, new protocols have been proposed on several OSI model layers: on the data link, network and application layers.

1. Data Link Layer

Responsible of the network transport management with lightweight protocols RFID, Bleutooth Low energy (BLE), Zigbee, Z-wave, NB-IoT, LoRaWAN and SigFox. Communication protocol selection depends on the coverage area (from less than one meter to several hundred kilometers), the required data rate (from 8 Kbps to 10 Gbps) and energy considerations (energy supply, energy consumption and IoT device lifespan).

a. ***On Wireless Body Area Network (WBAN):***

Covering less than one meter of distance, it includes the Low Frequency and High Frequency RFID (LF and HF RFID).

LF & HF RFID: The LF RFID operates on the frequency bands 120-150KHz and can transmit up to 8 Kbps over 30 cm range. While HF RFID operates on the 13.56 MHz frequency and can transmit up to about 800 Kbps over 1 meter. LF and HF RFID are passive track tags that are powered only when a nearby RFID reader tries to access the tag [32].

b. ***On Wireless Personal Area Network (WPAN):***

Covering up to ten meters, it includes the ultra-high frequency (UHF) passive and active RFID, BLE, Zigbee and Z-Wave protocols.

UHD RFID: Passive UHD RFID uses the 860 to 960 MHz frequency bands and has a data transfer range of 15 meters. Active UHF rely on 433MHz and 2.45 GHz frequencies and have a data transfer range of up to 150 meters. Both UHF RFID can transfer up to 640 Kbps data.

BLE: Commercially known as Bleutooth Smart, BLE is the energy reduction solution for IoT communication using Bluetooth protocol [33]. Compared to traditional Bleutooth, BLE uses the same 2.4 GHz frequency band, however it reduces the latency and connection time from 100ms to 3 ms and lower the data transmission rate from 3 Mbps to 1Mbps. BLE uses a deep sleep mode allowing the device to reduce its power consumption and resource allocation when the prototype is activated but not in use (standby). Since 2017, BLE enables device-to-device communication through mesh network to enable message routing between network nodes.

Zigbee: Zigbee is an open-source, low-power communication protocol based on the LR WAN protocol (IEEE 802.15.4) developed by the ZigBee Alliance community specialized in home automation integration [34]. This protocol uses the 868 MHz, 915 MHz and 2.4 GHz frequency bands to transmit 20 to 250 Kbps over 10 meters that can be extended using the mesh network. In addition, Zigbee uses 128-bit AES 128-bit algorithm encryption in CCM mode to secure its communications.

Z-Wave: Z-Wave is a proprietary lightweight communication protocol designed by the Z-Wave Alliance for home automation applications [35]. Z-wave protocol uses the 865-926 MHz frequency bands and transmits between 9 and 100 Kbps of data over 30 to 40 meters that can be extended using the mesh network. Optionally, users can use AES-128 encryption algorithm to encrypt their communications.

c. *On Wireless Wide Aera Network (WWAN):*

Covering up to several hundred kilometers, we distinguish between protocol types, high speed protocols such as the upcoming 5G focus on high data transfer and low latency; and Low power wide area network (LP WAN) protocols such as SigFox, LoRaWAN and NB-IoT.

5G: 5G is the latest cellular network introduced in 2018 and currently under worldwide deployment. 5G is optimized to offer 10 Gbps data rates, reduce latency to 1 ms, connect 100 times more devices and reduce power consumption by 90% compared to the 4G network. 5G also promises to improve the management, maintenance, monitoring, performance, security and power consumption of the IoT devices [36].

SigFox: SigFox is a low energy long distance communication protocol for small message transfer up to 12 bytes developed by the telecommunications company SigFox. The protocol uses the 868 MHz frequency band and the Utra Narrow Band (UNB) technology: using only 192KHz of the frequency band, each message takes 100Hz and the transfer data rate is 100 bps for a distance of 10 to 40 km. SigFox provides MAC integrity check and optional end to end encryption [37].

LoRaWAN: LoRaWAN is a lightweight protocol for low bandwidth and long distance communication developed by the LoRa Alliance. LoRaWAN rely on star of stars topology : an application server is connected to multiple gateways which are connected to multiple end-devices. This protocol uses the 868 MHz (Europe) and 915 MHz (North America) frequency band and can reach 50 Kbps throughput over a distance of 5 to 15 km [38].

NB-IoT: NB-IoT is a low power, long range communication protocol targeting internal communications with a high traffic concentration. Proposed by the 3rd Generation Partnership Project (3GPP) [39]. NB-IoT uses the narrow band 200 kHz from Long Term Evolution (LTE) GSM standard. This protocol allows communication rates from 20 to 250 Kbps with a less than 10 second latency across 1 to 20 km range.

2. Network Layer

Responsible of network infrastructure management with the lightweight protocol IPV6 over Low Power Wireless Personnal Aera Networks (6LoPWAN).

6LoWPAN: Developed by the 6LoWPAN Working Group, this protocol allows the connection of constrained devices over the internet (limited to 250 Kbps) based on the Low Rate Wireless Area Network (LR WPAN) communication protocol standardized by IEEE 802.15.4 [40]. 6LoWPAN protocol provides packet fragmentation and IPv6 header compression to accommodate LR WPAN support; neighbor discovery mechanism for IP auto configuration; and routing system. The most up-to-date version of the 6LoWPAN protocol is defined in RFC 8066 [41].

3. Application Layer

Responsible of the data formatting with two main lightweight protocols Constrained Application Protocol (CoAP) and Message Queuing Telemetry Transport (MQTT).

CoAP: CoAP is an application protocol standardized in RFC 7252 and specialized in constrained objects communication over the Internet [42]. CoAP protocol works over User Datagram Protocol (UDP) and integrates devices (called nodes) in the web through HTTP translation. CoAP have a Representation State Transfer (REST) architecture and client-server model. A client can request only GET, POST, PUT and DELETE actions. As an option, DTLS protocol can be enabled inside CoAP communications to ensure node communication security.

MQTT: MQTT is a lightweight application protocol based on the publish-subscribe mechanism over the Transport Control Layer (TCP) developed by the Organization for Advanced of Structured Information Standard (OASIS) and standardized in ISO/IEC 20922 [43]. Within MQTT protocol, a server acts as a message broker (or MQTT broker) that receives and analyzes messages from connected sensors (or publishers) and sends the information back to client devices (or subscribers). As a security option, TLS protocol can be enabled over MQTT communications.

2.3.1.2 Strengths & Limits of these Protocols

Through these lightweight protocols, the power consumption and hardware resources have been adapted to the constrained devices. The communication protocols use specific frequency bands and modulation techniques to minimize short and long-distance data transmission overhead. Network protocols fragment packets and compress data to reduce the size of messages on the network. Finally, application protocols allow the integration of constrained devices on the Internet using mechanisms based on a centralized data collector server. In addition, further study and project to combine these protocols and increase the performance and resources savings have been conducted such as CoAP over Zigbee [44], MQTT over Z-Wave [45] or 6LoWPAN over BLE [46].

However, data security is not enough addressed within these protocols and remains for the majority of them a non-recommended option since it is not adapted to the limited resources of constrained devices and drains battery power drastically. In addition, some protocols contain security vulnerabilities such as the Zigbee protocol which reuses known encryption keys that cannot be modified [47]. Finally, most of these protocols remain relatively unknown and limit data transfer rates. As a results, to facilitate the integration of the IoT and ensure a higher throughput than required by many IoT devices, users and manufacturers still rely on traditional network communication protocols such as Bluetooth, Wi-Fi and cellular (3G and 4G) protocols that target big data rates despite high power consumption and resource requirements.

2.3.2 IoT security Challenges

As introduced in Section 2.3 , low-cost, constrained, battery-powered and long lifespan IoT device have high risk of vulnerabilities and suffer from a lack of security mechanism adapted to their capacities and energy limits. A 2020 report published by a company specialized in Enterprise IoT Security Auditing announced that according to their analysis 83% of IoT communications are not encrypted making company and user data vulnerable to spyware, sniffing and MITM attacks [3].

To prevent these common flaws, the Open Web Application Security Project (OWASP) has listed the Top ten most common IoT security threats [48]. Among these threats, they ranked the deployment of unsecured network services as the second most common, the lack of a secure update mechanism as the fourth most common, and unsecured data communications and transfers as the seventh most common.

Compromised IoT devices carry risks that threaten data security, user privacy and also the security of private and corporate networks. Unsecured IoT devices can be used by an eavesdropper to perform data and user privacy theft, as an example Shodan web service maps insecure devices connected over the Internet [49]. Attackers can also infect millions of IoT devices to form a botnet network and launch a distributed denial-of-service attack (DDOS) attack against targeted victims. This occurred in the Mirai attack that disrupted Twitter, Reddit, Netflix, Airbnb and other services servers using a DDOS attack launched by hundreds of thousands of compromised connected objects [50]. Taking remote control of an IoT device is also a way to penetrate a computer network. Attackers can set up persistent backdoors in connected devices and attempt social engineering, lateral movement and privilege escalation techniques to take control of the enterprise network. Microsoft Threat Intelligent Center reported that in 2019 IoT devices were used to gain privileges in corporate networks [51]. Finally, with the use of wearable devices such as IMDs and smart bands another risk must be taken into account: the user safety. An attacker who can take control of such devices can harm users' health. Researchers have shown that a connected pacemaker can be compromised, and its use can be hijacked to remotely control it and send an electric shock of 830 volts and kill its owner [52].

2.3.2.1 Lightweight Encryption Algorithms

To address the security issues associated with insecure IoT communications, NIST has taken into consideration the diverse architectures and limited power capabilities of connected devices in order to adapt security mechanisms [31] . In addition, the NIST admitted that traditional cryptographic standards designed for workstations and servers are not appropriate for efficient operation on constrained devices. In this regard, NIST has published a call of paper to define new lightweight cryptographic standards [53]. Afterwards, they initiated a two-round standardization process to evaluate and compare the security and efficiency of candidate primitive cryptographic standards. In the first round of evaluation (security analysis), 56 candidates of lightweight algorithm AEAD were selected. In the second round (benchmark evaluation), the 32 most efficient candidates were selected. The process is not yet complete but the NIST plans to announce their finalists before the end of 2020. Many lightweight cryptographic algorithms have been developed to reduce the power consumption and resources needed for encryption operations. Based on traditional block ciphers, the development of lightweight primitive cryptographic algorithms relies on lighter architectures to reduce their impact. Four encryption algorithm parameters can be modified:

- The encryption block size : The size of the encryption block defines the number of bits processed by the encryption algorithm in a single block. The AES encryption standard, which is the reference for block encryption algorithms, is based on a 128-bit block. However, the size of the encryption blocks can be reduced to decrease the power consumption required to encrypt messages. This is the case of the SIMON and SPECK encryption algorithms, which proposes 32-bit block size versions [54]. Such encryption algorithms are more suitable for IoT devices such as connected sensors that send small messages smaller than 128 bits in length (e.g. temperature, pressure and humidity) at regular intervals.

- The encryption key length : The length of the encryption key defines the maximum number of operations required for decryption and ensures the maintain of the key's secrecy. For traditional systems (workstation and server) the minimum length recommended by NIST is 112 bits [55]. The AES encryption standard offers three different key lengths: 128, 192 and 256 bits. However, the key length can be shortened to reduce the number of computations required to encrypt messages and reduce resource demands. The Light Encryption Device (LED) encryption algorithm can be used with encryption keys of 64 or 80 bits in length [56].

- The number of encryption rounds : The number of encryption rounds defines the number of encryption cycles, i.e. the number of times an encryption block is run through a series of operations. The encryption round value is specific to the encryption algorithm and ensures robust output encrypted data. AES encryption standard defines 10 encryption rounds for 128-bit block, 12 rounds for

192-bit block and 14 rounds for 256-bit block. However, reducing the number of encryption rounds significantly reduces the number of operations to be performed and increases the number of cycles per byte resulting in power savings and reduced latency within encrypted communications. Chaskey 128-bit block cipher uses 8 rounds of encryption and a 128-bit key [57]. Conversely, other encryption algorithms rely on a high number of rounds with few operations to reduce power consumption and resource allocation. TWINE 64-bit block cipher with 80 or 128-bit key and 36 encryption rounds [58].

- Key scheduling optimization: Suggest lighter key scheduling techniques and optimized key generation algorithms to save energy and resources. As an example, Researchers proposed a new lightweight and secure method for key generation of AES with 5 encryption round [59].

A detailed description of each lightweight encryption algorithm is provided in Section 4.2.

2.3.2.2 Strengths & Limits of the Lightweight Cryptography

Thanks to these lightweight encryption algorithms, constrained devices can use ciphers with lower resource requirements such as central processing unit (CPU) computing power, Random-access memory (RAM) size and Read-only memory (ROM) storage capacity. These encryption solutions can significantly reduce the energy consumption required during the encryption process and reduce network latency. By changing the key and encryption block size and the number of rounds, the energy and resource consumption can be varied according to the level of security. Lightweight encryption techniques allow a security and power consumption balance that can be adapted to the constraints of IoT devices.

Nevertheless, lowering the security level is a dangerous risk, and using weak encryption algorithms will not guarantee the security of IoT communications. For example, using too small encryption keys will greatly reduce the number of possibilities that can lead to brute-force attacks to recover the secret key by testing all these possibilities. In addition, as described in Section 2.3.1, IoT devices use different communication protocols from one implementation to another depending on their uses and constraints. Lightweight cryptography solutions must be generalized to adapt to these protocols and allow secure IoT communications without depending on particular dependencies. Finally, dynamic mechanisms based on lightweight encryption algorithms must be proposed to adapt to the different architectures and resources available to connected devices in order to be deployed on a large number of IoT devices.

Chapter 3

State of The Art

This section details existing works to improve the security of IoT lightweight protocols, the efficiency of IoT lightweight encryption algorithms, and discuss about suggested green IoT solutions principles and these applications.

3.1 Lightweight Protocol Improvements

3.1.1 Application protocol

IoT application protocols such as CoAP and MQTT have been designed for constrained environments and are suitable for asynchronous communications. They minimize the amount of data to be communicated in order to reduce bandwidth and resource requirements dramatically. However, these protocols have not been designed to guarantee the security of IoT devices and their communications. They are compatible with the secure communication protocols TLS (for MQTT) and DTLS (for CoAP) but are not suitable for limited IoT device capacities. F. Siddiqui *et al.* have shown that the use of DTLS over CoAP secures the IoT device communications, but it has a significant impact on the IoT device and the network [60]. According to their experimental results, DTLS over CoAP increased the IoT battery consumption by 10% and the network latency increased by 100% compared to CoAP without DTLS. The researchers performed their experiment using the Texas Instrument SensorTag CC2650STK which is a Class 1 constrained device consisting of an ARM Cortex M3 MCU, a 128 KB ROM and a 20KB static RAM (SRAM). Using DTLS over CoAP on this constrained IoT device showed a 43.5% growth in storage utilization and 17.23%

increase in memory allocation. Finally, the device had only 10% of ROM space left (about 13 KB) and 8.5% of SRAM memory (about 1.7KB) free.

In order to secure the IoT communications, the IETF proposed the Object Security of CoAP (OSCoAP), a new end-to-end encryption, integrity and replay protection solution for clients and servers based on the CoAP application protocol [61]. This solution encrypts and authenticates parts of the CoAP package (payload, options and header) depending on the security level selected. For compatibility reasons OSCoAP leaves the proxies' required fields in plaintext and secures the other parts of the message. OSCoAP ensures data confidentiality and integrity, user authentication and protection against re-play attacks using AEAD block cipher suite in CCM mode using the AES encryption standard. Among the possible CCM configurations, OSCoAP uses the configuration with 64-bit IV, 64-bit authentication tag and 128-bit key AES (AES-CCM-64-128). Still under draft, OSCoAP is proposed as an option in both client and server CoAP messages. Finally, to ensure full hop-by-hop protection of CoAP messages, OSCoAP can be combined with the DTLS security protocol.

However, the CCM mode in OSCoAP is operated on the software side, which is not optimized for resource allocation and energy savings. Based on this approach, R. H. Randhawa *et al.* proposed an enhancement of OSCoAP in order to perform CCM operations on the radio chip hardware [62]. Since CoAP is an application-level protocol, the researchers proposed a cross-layer data exchange between OSCoAP and the IEEE 802.15.4 (low-rate wireless personal area networks- LR-WPAN) MAC on the link layer in order to exploit the integrated CCM mode on the IoT radio chip. This hardware optimization provides power consumption reduction and runtime efficiency compared to the CCM software implementation initially proposed by the IETF. Their experimental results on the wireless sensor Z1 mote embedding the cc2420 radio chip showed power savings of approximately 30% and an improved execution performance of 37%.

3.1.2 TLS protocol

As introduced in the Section 2.2, TLS (and DTLS) on IoT-based application protocols over TCP (and UDP) packet transport significantly increases the power consumption of IoT devices. This resource-intensive solution drains the battery of connected devices and greatly reduces their lifespan. T. Fischer *et al.* have analyzed the impact of the TLS protocol on an IoT device [63].For their experiment, they used the low-cost and low power ESP32 microcontroller unit (MCU), embedded with a 240 MHz Xtensa dual-core 32-bit LW6 CPU, 448 KB of ROM storage, 520 KB of RAM and a Wi-Fi 802.11 b/g/n antenna. They realized TLS version 1.2 and 1.3 communication between the ESP32 using the WolfSSL library and a workstation using the OpenSSL library. According to their analysis, a complete TLS session increased the consumption of ESP32 by 14 times compared to unprotected communications. On the other hand, the optimizations brought by TLS 1.3 offer energy savings compared to TLS 1.2.

In order to make the TLS protocol suitable for IoT communications, P. Li *et al.* proposed an end-to-end secure transport lightweight protocol based on the TLS and DTLS protocol version 1.3 with zero round trip time (0-RTT) [64]. Called iTLS/iDTLS, this protocol is an extension of TLS/DTLS that does not use a certificate for authentication but an identity-based authenticated key agreement. Client and server must register to a trusted authority to initialize security parameters and generate private keys based on their unique identities. In *ClientHello* and *ServerHello* messages the *identity_share* extension specifies the support for iTLS/iDTLS and gives information about their identity, trusted authority and security configuration. Using iTLS/iDTLS, client can send protected data to the server on the ClientHello message. This early data can be encrypted using the server identity (server public key) and the client's private key. Performance evaluations of iTLS/iDTLS showed that overhead traffic was reduced by 71% and the handshake latency was reduced by 59% compared to traditional TLS1.3.

K. P. Tange *et al.* proposed another extension to the TLS1.3 protocol compatible with 0-RTT session resumption [65]. Based on the double ratchet algorithm which is responsible for maintenance and regular renewal of keys combining a ratchet DH key exchange and a ratchet KDF, they propose ratchet TLS protocol (rTLS). rTLS allows to perform session resumptions reducing bandwidth overhead while ensuring the forward secrecy, protection against replays attacks and resiliency. According to their evaluations, rTLS reduces overheard traffic by a factor of 3 compared to the traditional TLS1.3 session recovery.

3.2 Lightweight Encryption Improvements

3.2.1 Symmetric-key Encryption

As a result of the lightweight cipher standardization process initiated by NIST and the critical need for energy-efficient encryption algorithms, many candidate ciphers have been proposed. These primitives are based on a simple structure and propose versions with limited block and key sizes to fit the constrained IoT systems. A description of the main lightweight ciphers is given in Section 4.2.

Once a new cipher is proposed, a performance analysis is performed on different hardware. However, comparing some performance criteria between several ciphers is not accurate if they have been evaluated on different platforms and under different conditions. In order to evaluate the performance of lightweight ciphers on the same hardware and following the same experimental conditions, D. Dinu *et al.* developed a benchmarking project called Fair Evaluation of Lightweight Cryptographic Systems (FELICS) [6]. FELICS framework is an open-source project implemented in ANSI C with inline assembly code designed for evaluating code size, RAM usage and execution time of lightweight block and stream ciphers compatible with three platforms: 8-bit Atmel, 16-bit MSP and 32-bit Arduino. The block cipher benchmark can be launched under three different scenarios:

- *Scenario 0* (or Cipher Operation) evaluates the encryption and decryption operations of a cipher block;
- *Scenario 1* (or Communication Protocol) considers the secure communication limitations and analyzes the encryption and decryption of 128 bytes of data with block ciphers in CBC mode;
- *Scenario 2* (or Challenge-Handshake Authentication Protocol) considers the authentication phase requirements and analyzes the encryption and decryption of 128 bytes of data with block ciphers in counter (CTR) mode.

Among all block ciphers assessed within scenarios 1 and 2, Chaskey, Speck and SparX performed the best energy-efficiency performance.

In order to improve lightweight ciphers efficiency, B. Rashidi modified the structure of Present, SIMON and LED primitives to improve their throughput and reduce the power consumption due to the encryption process [66]. The S-Boxes were implemented using only 14 logic gates. In addition, the proposed structures are flexible on several versions of these ciphers: Present 80-bit and 128-bit keys, LED 64-bit and 128-bit keys and SIMON 96-bit to 256-bit keys. The results obtained on the 180 nm CMOS showed that the power consumption of the SIMON block cipher was reduced by approximately 4.9%.

As introduced in Section 2.3.2, the key scheduling optimization provides energy savings and lowers IoT resource usage. In this context, Tsai *et al.* proposed a Secure Low Power Communication (SeLPC) over AES using 128-bit key with 5 rounds of encryption [59]. The SeLPC method regularly updates the encryption key and the look-up table using dynamic boxing (D-Box) on both parties. The encryption key and D-Box update procedure is generated and sent by the server to the devices. According to their analysis results on an ARM Cortex-M4 processor, the SeLPC method on AES-128 with 5 rounds reduced the power consumption of the device by about 26% compared to full AES-128.

3.2.2 Public-key Encryption

Regarding public key encryption algorithms, the Elliptic curve cryptography (ECC) is preferred to the RSA standard for IoT encryption solutions. For the same level of security ECC uses much smaller encryption keys than RSA, resulting in energy and resource savings during the encryption process. As a comparison, 112-bit security is achieved by RSA 2048-bit against an ECC curve with 224-bit key; 128-bit security is achieved by RSA 3072-bit key against 256-bit for an ECC curve. M. Suarez-Albela *et al.* compared the performance of RSA 1024, 2048 and 3072-bit key against Elliptic Curve Digital Signature Algorithm (ECDSA) using different types of ECC: prime192v1, secp224r1, secp256r1 and secp384r1 [67]. They analyzed the power consumption of each asymmetric encryption algorithm sending a 512 bytes payload about 100 times over TLS protocol on an ESP32 (using the OpenSSL library). The results obtained for secp224r1 show a power consumption reduction of 14% (18 mWh) compared to RSA 2048-bit (21mWh) with a throughput improvement.

Based on this approach, X. Yao *et al.* proposed a lightweight no-pairing ECC attribute-based encryption (ABE) to reduce the power consumption of IoT communications [68]. ABE is an asymmetric encryption primitive in which the user's encryption key is based on personal attributes. Unlike traditional ABE schemes whose security depends on Diffie-Hellman assumptions, their solution is based on the Elliptic Curve Decisional Diffie-Hellman (ECDDH) problem. Their comparative study shows that their approach leads to reduced power consumption and improved encryption rates.

3.3 Green IoT Guideline

In search of new energy saving solutions, R. Arshad *et al.* [69] have proposed 5 principles guideline for the Green IoT:

#1 Rely on policy making: Develop policy-based mechanisms and strategies to minimize the power consumption;

#2 Make intelligent trade-off: Determine a parameter whose variation can influence a system's energy efficiency;

#3 Use selective sensing: Focus only on the relevant data collection;

#4 Reduce the network size: Use energy-efficient network protocols and routing mechanisms;

#5 Use hybrid architecture: Combine passive and active sensors to perform specific tasks;

Based on (#1) policy making, M. V. Moreno *et al.* [70] proposed an energy-efficient automated platform for IoT management based smart buildings. Their platform was implemented within the City Explorer [71] autonomous home management system. According to their experimental results, their energy-efficient policies reduced by up to 23% the building's power consumption.

Based on (#2) intelligent trade-off, C. Karakus *et al.* [72] analyzed the WSN's power consumption during data compression and transmission. They determined the best trade-off between compression and transmission in order to optimize the power savings of WSNs. Results showed that compressive sensing-based approaches can prolong by 4 times the lifetime of WSNs compared to traditional approaches.

Based on (#3) selective sensing, C. Perera *et al.* [73] proposed an energy-efficient IoT cloud platform for selective mobile crowdsensing called context-aware mobile sensor data engine (C-MOSDEN). Experimental results, displayed as IoT applications scenarios (transport, e-health and wellness), highlighted 5 to 12% energy savings using context-aware capabilities.

Our solution respects 4 of these 5 principles: The reduced-round extension presented in Chapter 5 uses a dynamic mechanism based on (#1) a policy defined by the operator to achieve (#2) a trade-off between safety and energy consumption; the trade-off is decided by (#3) the battery level collected on the IoT device to (#4) reduce the impact on the network.

Chapter 4

Reduced-Round Cryptography

This section defines the reduced-round cryptography used in our dynamic reduced-round mechanism presented in the Chapter 5. We explain the objectives of this method and define our security limits, then we evaluate the security level of various lightweight reduced-round ciphers, and finally we perform a power consumption benchmark of these lightweight reduced-round ciphers on a level 1 constrained IoT device.

4.1 Objectives & Security Requirements

4.1.1 Encryption Round

As introduced in Section 2.3.2, many encryption algorithms and specifically block ciphers encrypt their data by performing a series of operations that they repeat in a specific number of iterations also called encryption rounds. On the security aspect, using multiple round of encryption increases confusion level responsible of the complexity to find a relationship between the ciphertext output and the encryption key used, and diffusion level responsible of the complexity to find a relationship between the ciphertext and the plaintext. On the performance aspect, encryption round allow to reduce the size of the code of a cipher block.

4.1.2 Reduced-Round Objectives

Increasing the number of rounds improve the robustness of an encryption algorithm at the expense of allocated resources and energy consumption. Conversely, lowering the number of rounds reduces the number of operations that lower the levels of confusion and diffusion in the output ciphertext, but increases the number of cycles per byte (cpb) and thus reduces power consumption and latency. Thus, encryption rounds balance the security level and resource footprint of an encryption algorithm. Cryptographic analyses of known attacks (also called cryptanalyses) on symmetric encryption algorithms round reduced show that ciphers security level decreases as the round value is reduced. Nevertheless, these cryptanalyses also provide an accurate estimation of ciphers' security level at different round values, and the results of these attacks suggest that lowering reasonably the number of rounds keeps the ciphers above the current security standards. The objectives of reduced-round cryptography are as follows:

- Determine security limits for IoT applications;
- Evaluate the security of reduced-round lightweight ciphers and determine their minimum round value according to the established security limits;
- Analyze and compare the power consumption performance of reduced-round lightweight ciphers on an IoT platform.

4.1.3 Encryption Cipher Security

Encryption cipher security is determined according to its robustness against an attack based on one of the following techniques: bruteforce, ciphertext only, know plaintext , chosen plaintext or chosen ciphertext attacks. Generally, there are four different types of cryptanalysis on cipher blocks:

- Linear Cryptanalysis: Know plaintext attack trying to simplify the analysed cipher by performing a linear approximation which is improved using known plaintext and ciphertext pairs. By increasing the number of known inputs/outputs, encryption key information can be retrieved and eventually extracted.

- Differential Cryptanalysis: Chosen plaintext attack trying to find differences between pairs of messages encrypted with the same encryption key. These differences in ciphertext are calculated in order to find statistical patterns in the distribution of the encryption algorithm. By increasing the number of chosen plaintext pairs, it allows to find biases and to identify probable keys leading to the encryption key recovery.

- Differential-Linear Cryptanalysis: Attack combining differential and linear cryptanalysis. This attack consists in performing a linear approximation based on a differential attack which is then used to achieve out a linear attack.

- Algebraic Cryptanalysis: A known plaintext attack attempting to model the encryption algorithm as equations system in order to solve it and to retrieve the encryption key.

During a cryptanalysis the security of the encryption algorithm is examined according to three types of complexities :

- Computational complexity: Representing the number of computer operations to be performed. Also called complexity in time, it defines the time it takes for the attack to be completed;

- Data complexity: Representing the amount of data to be stored. It determines the number of plaintext and/or ciphertext to be generated to accomplish the attack;

- Memory complexity: Representing the amount of RAM memory allocated to launch the attack.

Often referred to as the security level of an encryption algorithm, computational complexity is the most important criteria. Its maximum value is defined according to the encryption key length used, generally ranging from 64 to 256 bits. In general, the method used during a cryptanalysis consists in launching an attack on a reduced-round number of the targeted encryption algorithm in order to attenuate the time required (and thus its time complexity) to recover the encryption key.

4.1.4 Security Levels Suggestions

Encryption round reduction must be adapted to the latest cryptographic analyses performed on each encryption algorithm in order to determine an appropriate security level to protect the IoT communications. Unfortunately, no recommendation regarding the minimum level of security to be applied to secure IoT devices has been issued. Since 2015, NIST has updated its security recommendations for government agencies and requires that encryption algorithms use 112-bit keys (i.e., 2^{112} computational complexity) minimum instead of the current 80-bit keys (2^{80} computational complexity) [55]. However, these recommendations target storage and communication in traditional information systems such as servers and workstations with no resource and power supply constraints.

In their recent baseline requirement about IoT security for consumer and manufacturer, NIST and European Telecommunications Standards Institute (ETSI) requested to secure data from connected devices with encryption mechanisms adapted to resources, risks and usage [74, 75]. Generally, connected objects communicate at regular intervals and store only a few data that are quickly replaced (from one second to several days). Based on this analysis, the minimum acceptable security level of the IoT devices can be adapted according to the duration and sensitivity of the data. These security levels can be achieved according to the encryption key length chosen and by applying the reduced-round encryption algorithm. Table 3 shows our suggested security levels according to the data sensitivity and the type of the IoT device.

Table 3: Security suggested in relation with the IoT category

Sensitivity period	**Applications**	**Key sizes**	**Time Comp.**
[N ms - 1+ min.]	RFID, WSN	64bits 72bits	$\geqslant 2^{50}$
[N s - 1+ hour]	Wearable, Implementable Medical Devices	80bits 96bits	$\geqslant 2^{64}$
[N min. - 1+ year]	Implementable Medical Devices, Transportation	72bits 128bits	$\geqslant 2^{80}$
$+\infty$	Military, Banks, Medical Power Management	192bits 256bits	$\geqslant 2^{128}$

4.2 Lightweight Ciphers & Cryptanalysis Overview

Based on existing cryptographic analyses performed on various lightweight ciphers, we performed a security level study of 23 encryption algorithms, including a total of 52 different versions. The following subsections present each cryptographic primitive and give an overview of the reduced-round security of each cipher, classified according to the encryption algorithm structure: substitution–permutation network (SPN), fiestel network (FN) Block ciphers or Stream ciphers. The Table 4 shows the list of ciphers evaluated and their parameters.

Table 4: Lightweight Ciphers Parameters

Cipher	***Structure***	***Year***	***Block***	***Key***	***Rounds***
AES	AES-like SPN	1998	128	128	10
LED		2011	64	64/128	32/48
Fantomas	Bit-Sliced S-Boxes SPN	2014	128	128	12
Pride		2014	64	128	20
Robin		2014	128	128	16
SPARX	ARX SPN	2016	64/128	128	24
ICEBERG	Other SPN	2004	64	128	16
PRESENT		2007	64	80/128	31
PRINCE		2012	64	128	12
XTEA	ARX FN	1997	64	128	64
Hight		2006	64	128	32
LEA		2013	128	128/192/256	24/28/32
SIMON		2013	32 to 128	64 to 256	32 to 72
SPECK		2013	32 to 128	64 to 256	22 to 34
Chaskey		2014	128	128	8
PICCOLO	GFN	2011	64	80/128	25/31
TWINE		2012	64	80/128	36
Lilliput		2016	64	80	30
LBlock	Two-branched FN	2011	64	80	32
RoadRunneR		2015	64	128	12
SALSA20	ARX stream cipher	2005	512	128/256	20
CHACHA		2008	512	128/256	20
KATAN	Bivium-like	2009	32/48/64	80	254

4.2.1 SPN, FN Block Ciphers & Stream Cipher Definition

Symmetric cryptography is divided into two categories of encryption algorithms: block ciphers and stream ciphers.

On the one hand, the block cipher encrypts and decrypts the data by block of fixed size at a time. The size of the block depends on the design of the encryption algorithm used and is generally between 32 and 512 bits. Among the block cipher algorithms there are two major architectures: the substitution -permutation network (SPN) and the Feistel network (FN). SPN based block cipher distributes the input data in several small blocks and transforms plaintext into ciphertext using substitution boxes (S-boxes) and permutation boxes (P-boxes) and operation with the encryption key which are applied during several encryption rounds. While the FN divides the input into Left and Right blocks (or branches) of identical size which transform plaintext into ciphertext by swapping places and performing operations between branches and with the encryption key during several encryption rounds.

On the other hand, the stream cipher encrypts and decrypts data one bit at a time. The security of stream encryption algorithms is based on pseudo-random number generators (PRNG) to encrypt their input data.

4.2.2 SPN-Based Ciphers

4.2.2.1 AES and AES-like

AES: Developed as the Rijndael 128-bit block cipher, this encryption algorithm was standardized Advanced Encryption Standard (AES) by the NIST in 2001 [76] and approved by the National Security Agency in 2003 [77]. This cipher is based on a 128-bit block with a 128, 192 or 256-bit encryption key. AES encrypts its data by performing SubBytes, ShiftRows, MixColumns, and AddRoundKey operations on a 4x4-byte array of 128-bit blocks. AES-128 repeats this series of operations over 10 rounds. The most successful attack on a reduced-round version of the AES-128 is the meet-in-the-middle attack on 7 rounds P. Derbez *et al.* This attack reduces the algorithmic complexity of AES-128 to 2^{99} with a data complexity of 2^{97} [78].

LED: Presented at the international conference on cryptographic hardware and embedded systems (CHES) in 2011, LED is a 64-bit cipher block based on minimal hardware implementation to reduce power consumption and resource requirements

[56]. Two versions of LED are offered depending on the encryption key size, a 64-bit key version with 32 encryption rounds and a 128-bit key version with 48 rounds. LED block cipher architecture is based on AES architecture. A 128-bit data block is input as a 4x4 matrix and perform the 4 operations: AddConstants, SubCells, ShiftRows, and MixColumnsSerial. I. Nikolic *et al.* performed a multi-collision differential attack on the two versions of reduced-round LED [79]. Over 16 and 20 rounds of 64-bit key LED to obtain respectively a computational complexity of $2^{33.5}$ and $2^{60.2}$ with 2^{32} and $2^{61.5}$ data complexity. Over 32 and 40 rounds of LED-128 to obtain respectively $2^{33.5}$ and $2^{60.3}$ computational complexity with 2^{32} and 2^{60} data complexity.

4.2.2.2 Bit-sliced S-Boxes

The Bit-Sliced S-Boxes architecture concerns block ciphers using non-linear permutation boxes performing several small non-linear functions formed from bit-by-bit operators such as AND, OR and XOR logic gates. Bit-Sliced S-Boxes optimize the performance and robustness of the encryption algorithms at the software level.

Fantomas & Robin: Proposed at the Fast Software Encryption (FSE) conference in 2014, Fantomas and Robin are two 128-bit block and key encryption algorithms [80]. Both Fantomas and Robin use a LS-design structure defined by an SxL matrix with an 8-bit S-Box bit-slice on each column and a 16-bit L-Box on each row of the matrix. However, Robin uses involutive L-Boxes (respecting the involution property) while Fantomas uses non-involutive L-Boxes. Robin encrypts his data over 16 rounds against 12 for Fantomas. Dwivendi *et al.* performed a 5-round linear cryptanalysis of Robin and the complexity obtained is 2^{114} in time and 2^{82} in data [81]. G. Leander *et al.* performed a subspace invariant attack on 5-round Robin that decreased the complexities to 2^{64} and 2^{32} in time and data [82].

Pride: Presented at the CRYPTO conference in 2014, Pride is a 64-bit cipher block with a 128-bit key specialized in 8-bit microcontrollers [83]. Pride is a lightweight algorithm combining involutive and bit-slice S-Box to minimize its implementation and the number of instructions required for its operation. A differential attack over 19 rounds (over 20 rounds) of Pride performed by Q. Yang *et al.* reduced the computational complexity to 2^{63} with complexity of 2^{62} in data and 2^{71} in memory [84].

4.2.2.3 ARX

ARX architecture is a paradigm allowing only three types of operations: modular addition, rotation and XOR (ARX). The advantages of using an ARX-based encryption algorithm are simple implementation, speed of execution and low resource consumption.

SparX: Unveiled at the ASIACRYPT conference in 2016, SparX is a block cipher using a 128-bit encryption key [85]. SparX block cipher is specialized in software implementations and optimizes ROM and RAM space with minimal code size and memory allocation. SparX is available with 64-bit bock and 24 rounds of encryption or 128-bit block cipher and 32 rounds of encryption. The security of this algorithm is based on the Long Trail Strategy (LTS): favouring the number of rounds on ARX S-Boxes rather than widely distributed S-Boxes. Ankele and E. List have realized a truncated differential attack on 16 rounds of 64-bit block SparX which reduced the algorithmic complexity to 2^{93} with 2^{32} and 2^{61} complexity in data and memory [86]. While by M. Tolba *et al.* performed a Zero-Correlation attack on 22 rounds of 128-bit SparX block which was reduced to $2^{117.38}$ with $2^{116.2}$ data [87].

4.2.2.4 Others

Iceberg : Presented at the FSE conference in 2004, Iceberg is a 64-bit block cipher with a 128-bit encryption key specialized in reconfigurable hardware implementation [88]. Iceberg uses few resources, offers high encryption rates and is scalable for different types of hardware devices including Field Programmable Gate Array (FPGA) technologies. This encryption algorithm is based on an involutional structure that runs over 16 rounds. Y. Sun *et al.* realized a 7 round Iceberg structural attack that can reduce the security of Iceberg 7 rounds to $2^{90.28}$ in time, 2^{57} in data and 2^{48} in memory [89].

Present : Unveiled at CHES 2007, Present is a 64-bit block cipher specialized on low resource allocation hardware implementations [90]. Present exists in two versions depending on the encryption key, 80 or 128-bit, and performs 31 encryption rounds. Present-128 has been standardized ISO/IEC-29192-2 as a lightweight block cipher standard. J. Y. Cho performed a multi-dimensional linear attack on 25 rounds of Present 80-bit key reducing its complexity to 2^{65} in time, $2^{62.4}$ in data and 2^{34} in memory [91]. J. Nakahara *et al.* performed a Linear Hulls attack on 26 rounds of

Present 128 key bits reducing security to $2^{98.62}$ with 2^{64} data and 2^{40} memory [92].

Prince: Proposed at the ASIACRYPT conference in 2012, Prince is a low latency 64-bit block cipher with 128-bit key running on 12 cycles targeting hardware implementations [93]. Prince does not make a key schedule and derives 3 keys from its 128-bit master key using an FX structure allowing to increase the size of the encryption key using whitening keys. A. Canteaut *et al.* perfomed a multiple differential attack on 10 rounds (out of 12) of Prince, dropping its security to $2^{60.62}$ with $2^{57.94}$ data and $2^{61.52}$ memory [94].

The Table 5 details the up-to-date cryptanalyses on reduced-round SPN based block ciphers.

4.2.3 FN-Based Ciphers

4.2.3.1 Two-Branched FN

LBlock: Presented at the international conference on applied cryptography and network security (ACNS) in 2011, LBlock is a 64-bit block cipher with an 80-bit key [105]. LBlock is a lightweight, low power consumption and low throughput cipher specialized on 8 to 32-bit hardware platforms. It is composed of two 32-bit branches (two branched FN) and performs 32 rounds of encryption. M. Xie and Q. Zeng performed an impossible boomerang attack on 22 rounds of LBlock, reducing to $2^{68.76}$ in computational complexity with 2^{58} data [106].

RoadRunneR: Presented at the LightSec workshop in 2015, RoadRunneR is a 64-bit block cipher and a 128-bit key specialized for software implementation on 8-bit processors [107]. RoadRunneR implementation offers low costs memory and storage usage. Its architecture is based on an SPN structure with a bits-sliced S-Box as a function of its Feistel Network and performs 12 encryption rounds. Q. Yang *et al.* realized a meet-in-the-middle attack on 7 rounds decreasing the robustness of the cipher to 2^{121} in time, 2^{55} in data and 68 bits of memory [108].

4.2.3.2 Generalized FN

Generalized Feistel Network (GFN) is the generalization of the classical architecture of two branched feistel network at 2^n branches (with $n \in \mathbb{N}$, n > 1) of equal block size for efficient diffusion.

Table 5: SPN Ciphers Cryptanalysis

Round	**Time** (2^x)	**Data** (2^x)	**Memory** (2^x)	**Cryptanalysis / Attack**
LED: 64 bits Block & 64 bits Key				
16	33.5	32	-	Differential Multicollision [79]
20	60.2	61.5	-	
32 (full)	63.58	8	11	Biclique [95]
PRESENT: 64 bits Block & 80 bits Key				
16	20	36	16	Statistical Saturation [96]
25	65	62.4	34	Multidimensional Linear [91]
26	72	64	34	
31 (full)	79.76	23	-	Biclique [97]
AES: 128 bits Block & 128 bits Key				
7	110.2	110.2	-	Impossible Differential [98]
	99	97	-	Meet-in-the-Middle [78]
8	125.3	88	-	Biclique [99]
10 (full)	126.18	88	-	
Fantomas: 128 bits Block & 128 bits Key				
5	114	82	-	Linear [81]
ICEBERG: 64 bits Block & 128 bits Key				
3	34	26	small	Bit-Pattern [100]
4	46.7	27.4	small	
7	90.28	57	48	Structure [89]
8	96	63	32	Multiple Differential [89]
LED: 64 bits Block & 128 bits Key				
32	33.5	32	-	Differential Multicollision [79]
40	60.3	60	-	Differential Distinguisher [79]
44	126.92	64	8	Biclique [95]
48 (full)	127.23	64	8	
PRESENT: 64 bits Block & 128 bits Key				
26	98.62	64	40	Linear Hulls [92]
31 (full)	127.81	19	-	Biclique [97]
PRINCE: 64 bits Block & 128 bits Key				
4	11	7	4	Integral Attack [101]
6	33.7	16	31.9	Meet-in-the-middle [102]
9	51.21	46.89	52.21	Multiple Differential Attack [94]
10	60.62	57.94	61.52	
12 (full)	125.14	1	small	Exhaustive Search [103]
Pride: 64 bits Block & 128 bits Key				
18	63	61	35	Differential [104]
19	63	62	71	Differential [84]
Robin: 64 bits Block & 128 bits Key				
5	64	32	-	Invariant Subspace [82]
7	106	98	-	Linear [81]
SPARX: 64 bits Block & 128 bits Key				
16	93	32	61	Truncated Differential [86]
SPARX: 128 bits Block & 128 bits Key				
22	117.38	116.2	-	Zero-Correlation [87]

Piccolo: Presented at the CHES conference in 2011, Piccolo is a lightweight, four 16-bit branches (64-bit block) GFN encryption algorithm specialized for limited hardware implementations for its low resource cost [109]. Piccolo uses an 80-bit encryption key and performs 25 rounds of encryption or a 128-bit key and performs 31 rounds. Piccolo uses only 8 logic gates: 4 NOR, 3 XOR and 1 XNOR gates. M. Z. Ahangarkolaei *et al.* performed a Zero-Correlation attack over 13 rounds of Piccolo 80-bit key and reduced complexity in time, data and memory to $2^{67.4}$, 2^{58} and 2^{56} [110]. M. Minier performed an impossible differential attack on 21 rounds of Piccolo 128-bit key reducing time and data complexities to $2^{117.77}$ [111].

TWINE: Proposed at the workshop on Lightweight Crypto in 2011, TWINE is a four 16-bit branches (64-bit block) GFN cipher focusing on low hardware consumption and low ROM and RAM resources usage for software application. TWINE uses an 80-bit or 128-bit encryption key and performs 36 encryption rounds. T. Suzaki *et al.* performed a saturation attack over 22 rounds of TWINE 80-bit key and reduced its security to $2^{68.43}$ with a data complexity of 2^{62} and memory complexity of $2^{68.43}$. They performed the same attack on 23 rounds of TWINE 128-bit key reducing the security of the cipher to $2^{106.14}$ with data and memory complexity of $2^{62.81}$ and 2^{103} [58].

Lilliput: Published in 2015, Lilliput is a 64-bit block Extended Generalized Feistel Network (EGFN) divided into 16 nibbles (4 bits) and 80 key bits operating in 30 rounds. This encryption algorithm introduces EGFN as an optimization of the GFN structure adding a diffusion layer and consumption footprint reductions. T. P. Berger *et al.* performed an Integral attack on Lilliput 13 rounds reducing the cipher security to 2^{73} with 2^{62} data [112].

The results of the most-relevant cryptanalyses on reduced-round GFN and Two Branched FN block ciphers are detailed in the Table 6.

4.2.3.3 ARX

XTEA: Published in 1997, Extended TEA (XTEA) is a security enhancement of the TEA block cipher (having critical flaws) which is software-oriented due to its small resource footprint [124]. XTEA is a 64-bit FN block cipher with a 128-bit key and running over 64 encryption rounds. D. Moon *et al.* realized an impossible differential attack on 14 rounds of XTEA reducing time complexity to 2^{85} with $2^{62.5}$ data [125].

Table 6: GFN & Two Branched FN Ciphers Cryptanalysis

Round	**Time** (2^x)	**Data** (2^x)	**Memory** (2^x)	**Cryptanalysis / Attack**
LBlock: 64 bits Block & 80 bits Key				
20	63.7	63.7	-	Integral [105]
22	68.76	58	-	Impossible Boomerang [106]
23	74.06	59.6	74.6	Impossible Differential [113]
Lilliput: 64 bits Block & 80 bits Key				
13	73	62	-	Integral [112]
17	77	63	-	Division Property [114]
PICCOLO: 64 bits Block & 80 bits Key				
2	14	14	-	Conditional linear [115]
12	51.4	58.2	50	Zero-Correlation [110]
13	67.4	58.2	56	
14	68	62.2	-	Impossible Differential [116]
25 (full)	78.95	48	-	Biclique [117]
TWINE: 64 bits Block & 80 bits Key				
22	68.43	62	68.43	Saturation [58]
23	72.15	62.1	60	Zero-Correlation [118]
36 (full)	79.1	60	8	Biclique [119]
PICCOLO: 64 bits Block & 128 bits Key				
21	121	64	9	Meet-in-the-middle [120]
	117.77	117.77	-	Impossible Differential [111]
28	126.79	24	-	Biclique [117]
31 (full)	127.12	4	-	Biclique [121]
TWINE: 64 bits Block & 128 bits Key				
23	106.14	62.81	103	Saturation [58]
27	119.5	62.95	60	Key-Difference Invariant Bias [122]
36 (full)	125.75	60	8	Biclique [123]
	126.16	8	8	
RoadRunneR: 64 bits Block & 128 bits Key				
7	121	55	6.09	Meet-in-the-middle [108]

Hight: Presented at the CHES conference in 2006, Hight is a 64-bit FN block cipher with a 128-bit key focusing on low-cost implementation for constrained platforms [126]. Hight cipher performs only XOR operations, bitwise rotation and modulo addition and subtraction 2^8 over 32 rounds. Y. Sasaki and L. Wang performed a meet-in-the-middle attack on Hight 22 round-reduced reducing the time, data and memory complexities respectively to $2^{102.35}$, 2^{62} and 2^{64} [127].

LEA: Published at the Workshop on Information Security Application (WISA) in 2013 [128], the Lightweight Encryption Algorithm (LEA) is the Republic of Korea encryption standard KS X 3246 [129] approved by the Korean Cryptographic Module

Validation Program (KCMVP) since 2016 [130]. In addition, it has been standardized lightweight block cipher standard in ISO/IEC-29192-2 [131]. LEA block cipher standard is composed of 4 branches of 32-bit (128-bit block) specialized for software implementations on 32 and 64-bit platforms. LEA can be configured with 128, 192 or 256-bit encryption keys varying the number of rounds of operation to 24, 28 or 32 rounds respectively. For LEA-128, D. Hong *et al.* performed a differential attack over 12 rounds reducing complexities to 2^{84} in time, 2^{100} in data and 2^{76} in memory [128]. For LEA-192, L. Song *et al.* decreased complexities over 14 rounds to $2^{124.02}$ in time and data and 2^{22} in memory [132]. For LEA-256, over 11 rounds, A. D. Dwivedi *et al.* performed a differential attack decreasing the complexities in time, data and memory to 2^{227}, 2^{99} and 2^{22} [133].

Chaskey: Released in 2014, Chaskey is a 128-bit block cipher for 32-bit microcontrollers using a small implementation with a fast decryption throughput [57]. Chaskey is a Message Authentication Code (MAC) algorithm with 128-bit key operating over 8 rounds of encryption. Its security is based on the Even-Mansour scheme performing the XOR-encrypt-XOR operation series on the 128-bit master key. G. Leurent conducted a differential attack on 7 rounds of Chaskey reducing its security to 2^{67} with 2^{48} data [134].

SIMON & SPECK: Developed by the NSA in 2013, Simon and Speck are two lightweight encryption algorithms designed for different applications: SIMON is hardware oriented, while SPECK is software oriented [135]. These two ciphers are available under different versions depending on their block size ranging from 32 to 128 bits and their key size ranging from 64 to 256 bits. Among the multiple cryptanalyses on SIMON and SPECK variants: M. A Abdelraheem *et al.* performed a linear attack on 23 round out of 32 of SIMON-64 reducing its security to 2^{50} with $2^{30.59}$ data [136]. I. Dinur performed a differential cryptanalysis on 12 rounds out of 22 of SPECK-64 reducing its security to 2^{51} with 2^{19} data and 2^{22} memory [137]. J. Alizadeh *et al.* performed a linear attack on 35 out of 72 rounds of SIMON-256 reducing the complexities to 2^{128} in time, 2^{123} in data and memory [138]. F. Abed *et al.* performed a rectangle attack on 18 out of 34 rounds of SPECK-256 reducing the complexity in time, data and memory to $2^{182.7}$, $2^{125.9}$ and $2^{117.9}$ [139].

Table 7 and Table 8 present the cryptanalyses of the most effective attacks on Feistel Network ARX block ciphers running on reduced-round.

Table 7: ARX FN Ciphers Cryptanalysis (part1)

Round	Time (2^x)	Data (2^x)	Memory (2^x)	Cryptanalysis / Attack
SIMON: 32 bits Block & 64 bits Key				
20	31	31	-	Differential [140]
22	31.57	31.57	-	Multi. Linear [141]
23	50	30.59	-	Linear [136]
24	63.57	31.57	-	Multi. Linear [141]
SPECK: 32 bits Block & 64 bits Key				
10	29.2	29	16	Differential [139]
11	46.7	30.1	37.1	Rectangle [139]
12	51	19	22	Differential [137]
13	57	25	22	
14	63	31	22	
SIMON: 48 bits Block & 72 bits Key				
20	52	46	20	Differential [142]
21	61.9	48	43	Zero-Correlation [143]
23	62.10	47.78	-	Linear [136]
SPECK: 48 bits Block & 72 bits Key				
12	43	43	-	Differential [142]
14	65	41	22	Differential [137]
SIMON: 48 bits Block & 96 bits Key				
15	48	43	43	Linear [138]
21	50	45	-	Differential [140]
22	71	45	-	
24	83.10	47.78	-	Linear [136]
SPECK: 48 bits Block & 96 bits Key				
12	43	43	-	Differential [142]
15	89	41	22	Differential [137]
SIMON: 64 bits Block & 96 bits Key				
26	63.9	63	31	Differential [139]
27	88.53	62.53	-	Linear Hull [144]
SPECK: 64 bits Block & 96 bits Key				
15	61.1	61	32	Differential [139]
16	73	64	-	Differential [142]
18	93	61	22	Differential [137]
SIMON: 96 bits Block & 96 bits Key				
35	93.3	93.2	37.8	Differential [139]
SPECK: 96 bits Block & 96 bits Key				
16	85	85	22	Differential [137]
Chaskey 128 bits Block & 128 bits Key				
6	28.6	25	-	Differential [134]
7	67	48	-	
HIGHT: 128 bits Block & 128 bits Key				
10	20	64	small	Mixed-Integer Linear Programming [145]
11	21	64	small	
22	102.35	62	64	Meet-in-the-Middle [127]
26	114.35	61.6	87.6	Impossible Differential [146]
27	120.78	62.79	43	Zero-Correlation [147]
28	125.99	60	-	Impossible Differential [148]
32 (full)	125.93	48	-	Biclique [149]
LEA: 128 bits Block & 128 bits Key				
12	84	100	76	Differential [128]
14	124.02	124.02	22	Differential [132]

Table 8: ARX FN Ciphers Cryptanalysis (part2)

Round	Time (2^x)	Data (2^x)	Memory (2^x)	Cryptanalysis / Attack
SIMON: 64 bits Block & 128 bits Key				
19	66	61	61	Linear [138]
26	94	63	31	Differential [139]
28	119.53	62.53	-	Linear Hull [150]
29	123.53	62.53	-	Linear Hull [144]
SPECK: 64 bits Block & 128 bits Key				
15	61.1	61	32	Differential [139]
16	73	64	-	Differential [142]
19	125	61	22	Differential [137]
XTEA: 64 bits Block & 128 bits Key				
14	85	62.5	-	Impossible Differential [125]
23	105.6	63	103	Impossible Differential [146]
25	110.05	116	-	Truncated Differential [151]
27	115.15	20.5	-	
37	125	63	-	Related Key [152]
SIMON: 128 bits Block & 128 bits Key				
46	125.7	125.6	40.6	Differential [139]
SPECK: 128 bits Block & 128 bits Key				
17	113	113	22	Differential [137]
SIMON: 96 bits Block & 144 bits Key				
28	100	95	95	Linear [138]
35	101.1	93.2	37.8	Differential [139]
36	135.2	94.2	-	Linear Hull [144]
SPECK: 96 bits Block & 144 bits Key				
17	96	92	92	Linear [153]
LEA: 128 bits Block & 192 bits Key				
10	99	99	22	Differential [133]
14	124.02	124.02	22	Differential [132]
SIMON: 128 bits Block & 192 bits Key				
46	142	125.6	40.6	Differential [139]
48	187.6	126.6	-	Linear Hull [144]
SPECK: 128 bits Block & 192 bits Key				
18	128	124	124	Linear [153]
LEA: 128 bits Block & 256 bits Key				
11	227	99	22	Differential [133]
14	250.19	128	142.35	Zero-Correlation Linear [154]
15	252.02	124.02	22	Differential [132]
SIMON: 128 bits Block & 256 bits Key				
35	128	123	123	Linear [138]
41	231	128	-	Matsui's Algorithm 2 [150]
43	232	123	127	Linear [155]
49	232.6	126.6	-	Linear Hull [150]
50	242.6	126.6	-	Linear Hull [144]
SPECK: 128 bits Block & 256 bits Key				
16	111.1	116	64	Differential [139]
18	182.7	125.9	117.9	Rectangle [139]
19	192	124	124	Linear [153]

4.2.4 Stream Ciphers

4.2.4.1 ARX

Salsa20 & Chacha: Unveiled at the eSTREAM conference in 2005, Salsa20 is a ARX based stream cipher using a 512-bit block, a 64-bit nonce and counter, and a 128- or 256-bit key that operates 20 encryption rounds [156]. Chacha is the performance and security optimization of Salsa20 released in 2008 and standardized by the the IETF on RFC 7902 combined with Poly1305 authenticator and integrated within TLS1.2 [157, 158]. K. C. Deepthi *et al.* performed a differential linear attack on 7 rounds of Salsa20 and 6 rounds of Chacha 128-bit key reducing respectively their security to 2^{104} and 2^{101} with 2^{17} and 2^{28} data [159]. While Z. Shao *et al.* reduced the security over 12 rounds of Salsa20 to 2^{224} [160] and A.R. Choudhuri *et al.* reduced the security over 7 rounds of Chacha 256-bit key to 2^{233} (with 2^{96} given) [161].

4.2.4.2 Bivium-like

KATAN: Presented at CHES in 2009, KATAN is a Bivium-like architecture based (simplified variant of the Trivium cipher) stream cipher hardware-oriented algorithm specialized for its very low power consumption capacity and recommended for RFID encryption communications [162]. Katan use an 80-bit encryption key and a 32, 48 or 64-bit block size encrypting data over 254 rounds. T. Fuhr *et al.* performed a meet-in-the-middle attack reducing the security of KATAN-32 121 rounds, KATAN-48 110 rounds and KATAN-64 102 rounds to $2^{77.5}$ in time, 2^{5} in memory and 4 bits in data storage [163].

The Table 9 details the up-to-date cryptanalyses on reduced-round STREAM ciphers.

Table 9: Stream Ciphers Cryptanalysis Summary

Round	**Time** (2^x)	**Data** (2^x)	**Memory** (2^x)	**Cryptanalysis / Attack**
KATAN: 32 bits Block & 80 bits Key				
79	9.79	4.32	-	Algebraic [164]
110	77	7.11	75.1	Meet-in-the-Middle [165]
121	77.5	2	5	Meet-in-the-Middle [163]
201	78.1	1.59	78.1	Meet-in-the-Middle [166]
206	79	1.59	78.1	
KATAN: 48 bits Block & 80 bits Key				
64	14.4	2.32	-	Algebraic [164]
70	34	34	-	Conditional Differential [167]
110	77.5	2	5	Meet-in-the-Middle [163]
146	78.1	1	77	Meet-in-the-Middle [166]
148	79	1	77	
KATAN: 64 bits Block & 80 bits Key				
60	13.4	2.32	-	Algebraic [164]
68	35	35	-	Conditional Differential [167]
102	77.5	2	5	Meet-in-the-Middle [163]
126	78.1	1	77	Meet-in-the-Middle [166]
129	79	1	77	
CHACHA: 512 bits Block & 128 bits Key				
6	101	28	-	Differential-Linear [159]
SALSA20: 512 bits Block & 128 bits Key				
7	104	17	-	Differential-Linear [159]
CHACHA: 512 bits Block & 256 bits Key				
5	16	16	-	Differential-Linear [161]
6	116	116	-	
7	233	96	-	
SALSA20: 512 bits Block & 256 bits Key				
5	8	8	-	Differential-Linear [161]
6	32	32	-	
7	137	61	-	
12	224	-	-	Related-Cipher [160]

4.3 Benchmark & Results

4.3.1 Experiment

In order to evaluate the energy consumption generated by reduced-round cryptography, we selected the lightweight ciphers studied in Section 4.2 and examined their energy consumption during the encryption process. This experience, in the form of a benchmark test 24 lightweight cipher with a different round-reduction on a constrained C1 level IoT platform Texas Instrument Launchpad MSP430FR2355. The launchpad has an ultra-low-power 24 MHz 16-bit MSP430FR2355 MCU with 32 KB of FRAM designed for its ease of use, it's very low power consumption and without hardware acceleration functionality (data encryption).

The power consumption benchmark was realized using the FELICS Framework which contains the implementation of lightweight cryptographic primitives on embedded devices and using the real-time measurement tool Energy Trace present in the Code Composer Studio software.

For our benchmark we used the Scenario 1-II from FELICS frameworks: Encryption using CBC mode without key scheduling. We measured the energy consumed to encrypt 4KB of data per second on the constrained IoT device for each encryption algorithm by varying the number of encryption rounds.

4.3.2 Implementation

FELICS framework is an open-source project that provides ANSI C language implementation with inline assembly instructions for many lightweight ciphers on 8-bit AVR, 16-bit MSP and 32-bit ARM MCU.

In order to integrate reduced-round cryptography, we have modified the FELICS project to count and vary the number of rounds of each encryption algorithm from maximum to minimum (security limit) in order to analyse the energy consumed to encrypt 32 messages of 128 bytes per second (i.e. 4 KB/s).

For compatibility reasons and equity between each encryption algorithm we have used the most up-to-date version that does not contain hardware acceleration and assembly code optimizations. The Table 10 lists, for each cipher, the version used to performed our benchmark compared to the last version proposed by FELICS.

4.3.3 Results

4.3.3.1 Encryption Power Consumption

First, we measured the power consumption by running the encryption benchmarks of 4KB of data per second for 5 minutes for each cipher in full round. Then, we repeated these measurements by reducing each cipher round by one and we repeated this procedure up to the minimum round of every cipher determined by the cryptanalysis presented in Section 4.2.

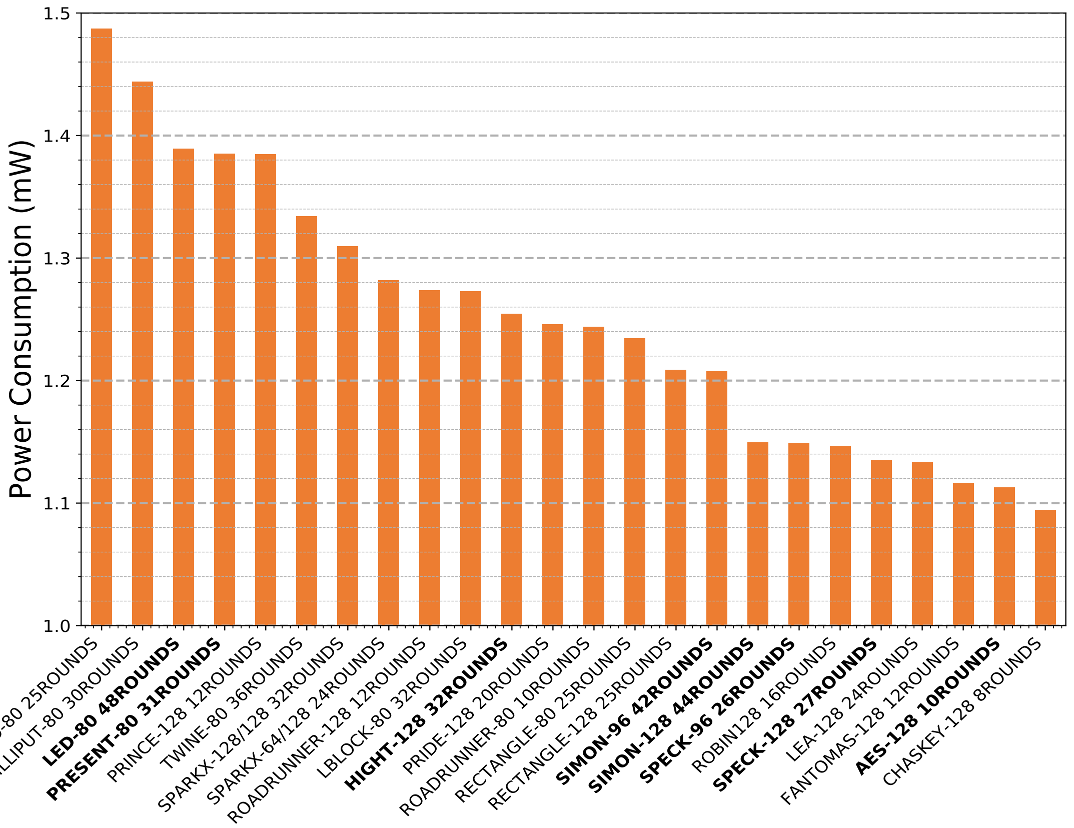
Table 10: FELICS' Ciphers version used

Cipher	Version Used	Last Version
AES-128	6	8
CHASKEY-128	2	6
FANTOMAS-128	2	2
HIGHT-128	20	21
LEA-128	13	14
PRIDE-128	8	8
PRINCE-128	24	24
RECTANGLE-128	5	9
ROAD_RUNNER-128	6	6
ROBIN-128	2	2
SIMON-64_128	3	6
SPARKX-128_128	2	24
SPARKX-64_128	10	35
SPECK-64_128	5	6
Simon-64_96	6	9
Speck-64_96	9	10
LBlock-80	24	24
LED-80	4	4
Lilliput-80	6	7
Piccolo-80	1	1
Present-80	5	6
Rectangle-80	5	9
RoadRunneR-80	2	2
Twine-80	4	4

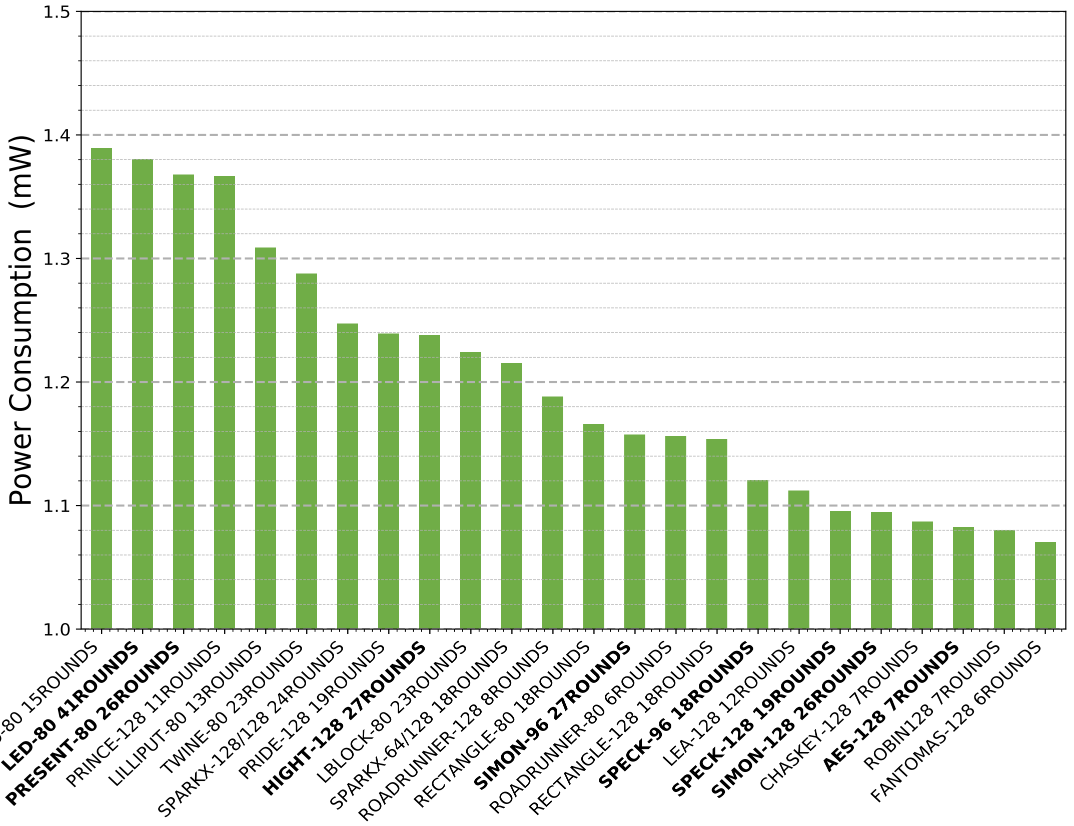
The average power consumption obtained by our benchmark for each lightweight cipher with reduced-round are shown in Figure 3. The power consumption of each lightweight cipher is ranked in descending order at full round in the orange bar chart and at minimum round in the green bar chart. The results show that on average the energy consumption decreased by 0.05 mW between full and minimum round. More precisely, on average the energy consumption decreased by 0.008 mW per reduced-round. Among the results obtained, RoadRunneR 80-bit key reduced to 6 out of 10 rounds and Fantomas 128-bit key reduced to 6 out of 12 rounds respectively reduced the overall power consumption of the IoT device from 1.2438 mW to 1.1560 mW and from 1.1166 mW to 1.0705 mW.

4.3.3.2 Battery Savings

To evaluate the energy benefits of this solution, we estimated the battery gains obtained for each reduced-round lightweight cipher. First, we calculated the global gains for each cipher between full and minimum round. Second, we measured the MSP430FR2355 MCU in idle between each analysis to determine the default power



(a)



(b)

Figure 3: Power consumption comparison of lightweight ciphers in (a) full and (b) minimum rounds.

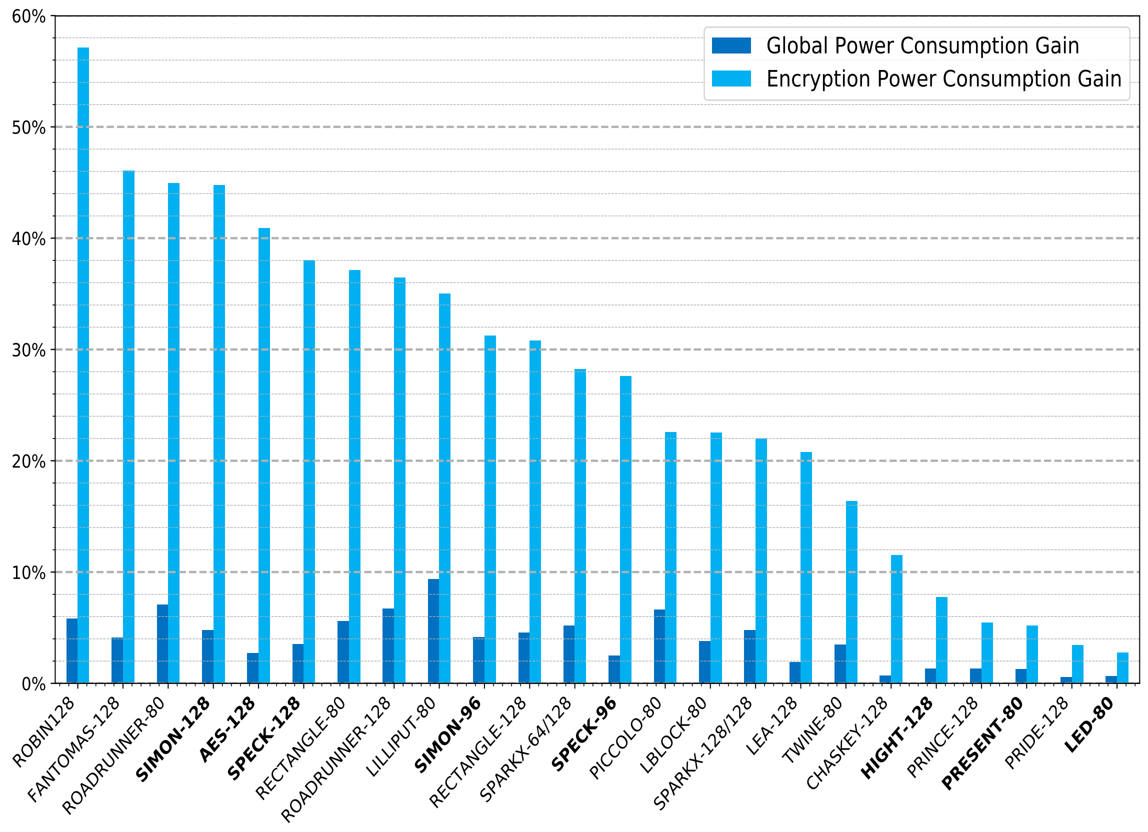


Figure 4: Power gain comparison between lightweight ciphers full and minimum round for 4KB/s data encryption.

consumption and calculated the encryption only gains for each cipher between full and minimum round.

The power consumption gains obtained by reducing each encryption algorithm to their minimum round are shown in Figure 4. The light blue bars show the encryption only gain, while the dark blue bars show the global gain. On average 26.6% encryption only gain, of which 3.9% global gains are obtained by using reduced-round lightweight ciphers. Up to 9.38% of battery can be saved using Lilliput-80 reduced to 13 rounds and up to 57.1% of encryption energy can be saved using Robin-128 reduced to 7 rounds.

Observation: As shown in Table 11, we measured the MSP430FR2355 device power consumption before each new cipher evaluation to ensure accurate encryption only gain estimations. We observe that the IoT device consumption in IDLE mode varies, which explains the non-linearity between the global and encryption gains between each cipher. These variations can be caused by the equipment overheating during the benchmark or by external phenomena such as vibrations, manipulations or low electromagnetic fields.

4.4 Discussion

The power consumption analysis and the battery savings obtained proved the effectiveness of reduced-round cryptography in providing a secure and energy-saving encryption solution. Indeed, the power consumption of each reduced-round lightweight cipher has been lowered, resulting in significant battery savings. The Table 11 shows the Benchmark results on the MSP430: It describes the power consumption and the current in IDLE, full and minimum round as well as the gains obtained for each lightweight cipher. However, future cryptanalyses may weaken the security of the encryption algorithms, forcing the minimum round value to be revised according to the recommended security limit. Dynamic and secure mechanisms must be implemented to configure and update the round number to ensure an appropriate level of security.

Table 11: Reduced-Round Benchmark Results

	IDLE		**Full Round**			**Min Round**			**Battery Savings**	
Cipher	**Power/mW**	**Current/mA**	**Value**	**Power/mW**	**Current/mA**	**Value**	**Power/mW**	**Current/mA**	**Global**	**Encryption**
AES-128	1.0391	0.3165	10	1.1127	0.3389	7	1.0826	0.3297	2.71%	40.9%
HIGHT-128	1.0377	0.3172	32	1.2547	0.3826	27	1.2379	0.3568	1.34%	7.7%
RECTANGLE-128	1.0302	0.3137	25	1.2088	0.341	18	1.1538	0.3325	4.55%	30.8%
LEA-128	1.0305	0.3138	24	1.1336	0.3452	12	1.1122	0.3387	1.89%	20.8%
CHASKEY-128	1.0302	0.3137	8	1.0943	0.3332	7	1.0869	0.331	0.68%	11.5%
FANTOMAS-128	1.0165	0.3128	12	1.1166	0.3401	6	1.0705	0.3266	4.13%	46.1%
SPECK-128	1.0302	0.3137	27	1.1354	0.3298	19	1.0954	0.3237	3.52%	38.0%
ROBIN128	1.0302	0.3137	16	1.1468	0.3492	7	1.0802	0.329	5.81%	57.1%
SIMON-128	1.0269	0.3128	44	1.1497	0.3314	26	1.0947	0.3231	4.78%	44.8%
PRIDE-128	1.0429	0.3176	20	1.2459	0.3485	19	1.2389	0,3474	0.56%	3.4%
PRINCE-128	1.0476	0.319	12	1.3848	0.3704	11	1.3664	0.3676	1.33%	5.5%
RoadRunneR-128	1.0385	0.3162	12	1.2739	0.3521	8	1.1881	0.3389	6.74%	36.4%
SPARKX-64/128	1.046	0.3185	24	1.2818	0.3544	18	1.2152	0.3443	5.20%	28.2%
SPARKX-128/128	1.0253	0.3122	32	1.3097	0.3555	24	1.2471	0.346	4.78%	22.0%
SIMON-96	1.0466	0.3187	42	1.2078	0.3433	27	1.1574	0.3356	4.17%	31.3%
SPECK-96	1.0455	0.3184	26	1.1491	0.3342	18	1.1205	0.3298	2.49%	27.6%
Present-80	1.0473	0.3189	31	1.3853	0.3704	26	1.3677	0.3677	1.27%	5.2%
Piccolo-80	1.0517	0.3203	25	1.4875	0.3866	15	1.3891	0.3716	6.62%	22.6%
Twine-80	1.0497	0.3197	36	1.3343	0.3629	23	1.2877	0.3559	3.49%	16.4%
LBlock-80	1.0571	0.3219	32	1.2729	0.3548	23	1.2243	0.3475	3.82%	22.5%
LED-80	1.056	0.3216	48	1.3894	0.3724	41	1.3802	0.371	0.66%	2.8%
Lilliput-80	1.0573	0.322	30	1.4439	0.3809	13	1.3085	0.3603	9.38%	35.0%
Rectangle-80	1.0495	0.3196	25	1.2347	0.3478	18	1.1659	0.3373	5.57%	37.1%
RoadRunneR-80	1.0484	0.3193	10	1.2438	0.349	6	1.156	0.3356	7.06%	44.9%

Chapter 5

Dynamic Reduced-Round TLS extension

This section present the dynamic reduced-round TLS extension based on the reduced-round cryptography defined in the Chapter 4. First, we detail our reduced-round extensions design within TLS protocol. Second, we describe the dynamic reduced-round mechanism embedded inside our TLS extensions. Third, we outline the implementation of our solution inside WolfSSL embedded TLS library. Finally, we test our solution on a wireless connected IoT platform and analyse the power consumption gains.

5.1 Reduced-Round TLS extension

In this section, we propose a new TLS extension called reduced-round that automatically adapts the power consumption of an IoT device according to its current battery level and an operator-defined policy. We detail the design and the implementation of the reduced-round extension for both *handshake* and *application data record* subprotocols of TLS.

The reduced-round extension is based on TLS 1.3 and follows the IETF recommendations. In addition, this extension is also inspired by already existing and approved TLS version 1.2 and 1.3 extensions. As a result, the reduced-round TLS 1.3 protocol satisfy the same security criteria (data confidentiality and integrity, user authentication and session forward secrecy) and provide security against known attacks such as replay and downgrade attack. In addition, reduced-round extension follows the same threat model as TLS 1.3, listed in the RFC8446 [5]. An IoT device (as a client) can trust the operator (as a server) using digital certificates and trusted third-party certificate authorities (CA) respecting the chain of trust.

5.1.1 The Handshake Protocol

In the TLS protocol, the record handshake phase is a procedure designed to authenticate peer devices and to negotiate the security parameters of their connection. This procedure, generally composed of 6 client/server exchanges, aims to:

- Negotiate the version of the TLS protocol;
- Negotiate the encryption and signature algorithm;
- Authenticate the server to the client;
- Authenticate the client to the server (optional);
- Setting up the master key.

Since TLSv1.2, as defined in RFC 6066 [168], clients can request custom TLS functionalities from the server using the extension field. These extensions, if accepted by the server, are specified in both *ClientHello* and *ServerHello* messages of the TLS Handshake subprotocol. In addition, as defined in the RFC 8446 [5], the arrival of TLS 1.3 further improved the speed and security of the handshake phase by reducing the number of exchanged messages to 4 (without the 0-RTT Resumption mode) and by encrypting the messages right after the *ServerHelloDone* message. For the other extensions, the dynamic number of rounds is integrated within the TLS handshake (version 1.2 and 1.3), in compliance with the rules enforced by the RFCs.

5.1.1.1 Inside *ClientHello*

First message of the Handshake, the *ClientHello* message is initiated from the client to the server (i.e. from the IoT device to the operator). This message contains the latest TLS version supported, the list of supported cipher suites by the IoT device and the extension type value and associated data. Standardized by the Internet Assigned

Numbers Authority (IANA), each TLS extension is identified by a value between 0 and 65535 ($2 bytes$) [169].

For the sake of simplicity, we propose the use of the first (arbitrary) unreserved slot 57 to implement our extension. This 3-byte long payload contains: ($bytes = 0-1$) the extension identifier 57 ($0x0039$) and ($byte = 2$) the battery level of the IoT device between 0 ($0x00$) and 101 ($0x65$). If the battery level is set to 101, then the IoT device is not aware of its battery level.

5.1.1.2 Inside *ServerHello*

Following the *ClientHello* message, the *ServerHello* determines the TLS version, the cipher suite that will be used based on the client-provided information and, if the handshake is not aborted, it confirms the use of proposed extension(s).

In this context, the reduced-round extension payload response contains ($byte = 0$) the minimum round value allowed, or the fixed number of rounds if the client is not aware of its battery level, ($byte = 1$) the operator-defined policy that determines the dynamic number of round used based on the battery level. This policy is set to $0x0$ ($single\_fixed$) if the battery level is not known, $0x1$ if the battery level is known and all intermediate rounds can be used (*proportional*) or $0x2$ if specific round numbers are specified ($multiple\_fixed$). For the latter case, additional bytes must be used to specify the list of fixed possible rounds values to reduce the number of future *UpdateRound* messages. In this case, one additional byte ($byte = 2$) is set to define the number n_{rounds} of fixed round values. The remaining n_{rounds} bytes specify all fixed rounds values.

After transmission of the *ClientHello* and *ServerHello* messages, peers are synchronized. Knowing the battery level and policy, both sides can separately compute the same first number of round used to encrypt and decrypt the first client message. Figure 5 describes the TLSv1.3 protocol establishment with the reduced-round extension.

5.1.2 The Application Data Protocol

The TLS *application_data* subprotocol is a record message type responsible for transporting data securely based on pre-negotiated security parameters and encryption keys. *application_data* 's objectives are:

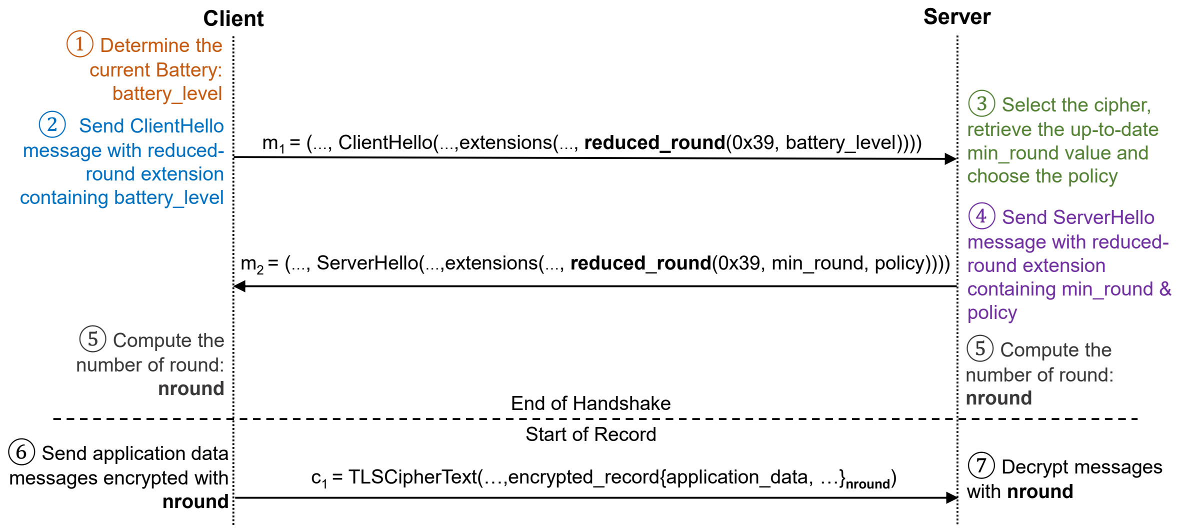

Client
Server
1 Determine the current Battery: battery_level
2 Send ClientHello message with reduced-round extension containing battery_level
m1 = (..., ClientHello(...,extensions(..., reduced_round(0x39, battery_level))))
3 Select the cipher, retrieve the up-to-date min_round value and choose the policy
4 Send ServerHello message with reduced-round extension containing min_round & policy
m2 = (..., ServerHello(...,extensions(..., reduced_round(0x39, min_round, policy))))
5 Compute the number of round: nround
5 Compute the number of round: nround
End of Handshake
Start of Record
6 Send application data messages encrypted with nround
c1 = TLSCipherText(...,encrypted_record{application_data, ...}nround)
7 Decrypt messages with nround

Figure 5: Reduced-round Handshake protocol based on TLS 1.3

- Encrypt/Decrypt outgoing/incoming messages;
- Apply/Check MAC to outgoing/incoming messages;
- Divide output messages into blocks of fixed size and to reassemble them upon arrival;
- Compress/Decompress the output/input blocks (optional).

During this phase, the reduced-round extension dynamically adapt the number of encryption rounds according to the operator-defined policy and the current battery level of the IoT device. Similar to the heartbeat extension [170], our extension evolve on top of the Record layer. To be both functional and secure, we defines 4 new record layer messages: $UpdateRound$, $UpdateMinRound$, $UpdatePolicy$ and $UpdateResponse$. These compact messages contain a message kind field, a response flag (except for $UpdateResponse$) of 1 byte each as well as a payload of variable size, depending on the message kind. The response flag is optionally set and may be used to mimic TLS Acknowledgment when using DTLS (TLS over UDP). This flag is set to $0x0$ is no confirmation is expected and $0x1$ for a response request. The Listing 1 shows the list of implemented message kinds.

Listing 1: List of Policies

```
enum {
    update_round (1),
    update_min_round (2),
    update_policy (3),
    update_response (4),
    (255)
} ReducedRoundMessageType;
```

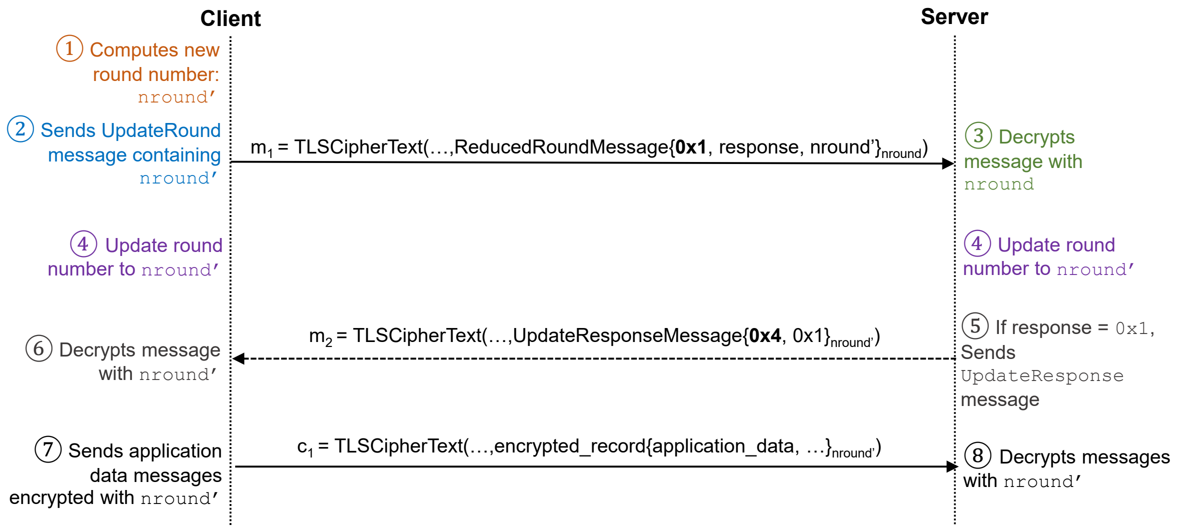
5.1.2.1 Dynamic Round Update

As the battery level decreases (battery-powered IoT device) or increases (energy harvesting systems) over time, it is essential to maintain synchronization between the client and the server by notifying the server of a change in the number of rounds. To securely perform this round modification, the Dynamic Round Update feature is initiated by the IoT device. When the round value changes according to the policy, it sends an encrypted *UpdateRound* message to the operator containing the new round value.

UpdateRound is a fixed 3-byte long message containing:

- $uint8$ ReducedRoundMessageType $< 0x1 >$;
- $uint8$ responseFlag $< 0x0||0x1 >$;
- $uint8$ round_number $< min\_round..max\_round >$.

This message is encrypted using the last round value known by the server. This value is updated on both sides after the message is correctly transmitted to the server. Optionally, if the response flag is equal to $0x1$, the server sends an *UpdateResponse* message encrypted with the new round value to notify the IoT device of the round change. The detailed step-by-step diagram of the Dynamic Round Update functionality is shown in Figure 6. This process, in addition to being end-to-end encrypted, ensures synchronization of the round number between client and server without requiring the IoT equipment to communicate its battery level.



Client
Server
① Computes new round number: nround′
② Sends UpdateRound message containing nround′
m1 = TLSCipherText(…,ReducedRoundMessage{0x1, response, nround′}nround)
③ Decrypts message with nround
④ Update round number to nround′
④ Update round number to nround′
m2 = TLSCipherText(…,UpdateResponseMessage{0x4, 0x1}nround′)
⑤ If response = 0x1, Sends UpdateResponse message
⑥ Decrypts message with nround′
⑦ Sends application data messages encrypted with nround′
c1 = TLSCipherText(…,encrypted_record{application_data, …}nround′)
⑧ Decrypts messages with nround′

Figure 6: Reduced-round Dynamic Round Update on TLS 1.3.

5.1.2.2 Round Configuration Update

The number of possible encryption rounds is intrinsic to the cipher suite in use: the minimum round value is defined by an operator-stored up-to-date cryptanalysis (see Section 4.2) and finally the operator-set policy. However, there are two scenarios where the minimum round value may change during the lifetime of the TLS connection between the IoT device and the operator: (1) if the operator needs to reduce the security to ensure the communications with a very constrained IoT device, or conversely, (2) if a new cryptanalysis forces to increase the security of an encryption algorithm. To perform this update securely, the operator uses the Round Configuration Update feature by sending an $UpdateMinRound$ message to the IoT device, containing the new minimum round value.

$UpdateMinRound$ is a fixed 3-byte long message with:

- $uint8$ ReducedRoundMessageType $< 0x2 >$;
- $uint8$ responseFlag $< 0x1 >$;
- $uint8$ round_number $< 0x1..max\_round >$.

This $UpdateMinRound$ message must be acknowledged by the IoT device and is encrypted with the current round value. After acknowledgement, both sides update their minimum round property. In return, the IoT device calculates the updated number of rounds to be used considering the new minimum $round\_number$. It then sends an *UpdateRound* message (with an optionally set response flag) with the new calculated round value, encrypted with the current round number. If the value is different from the current one, peers re-synchronize. Figure 7 shows the step-by-step diagram of the Round Configuration Update functionality. This procedure ensures that the minimum is synchronized between client and server via an encrypted channel.

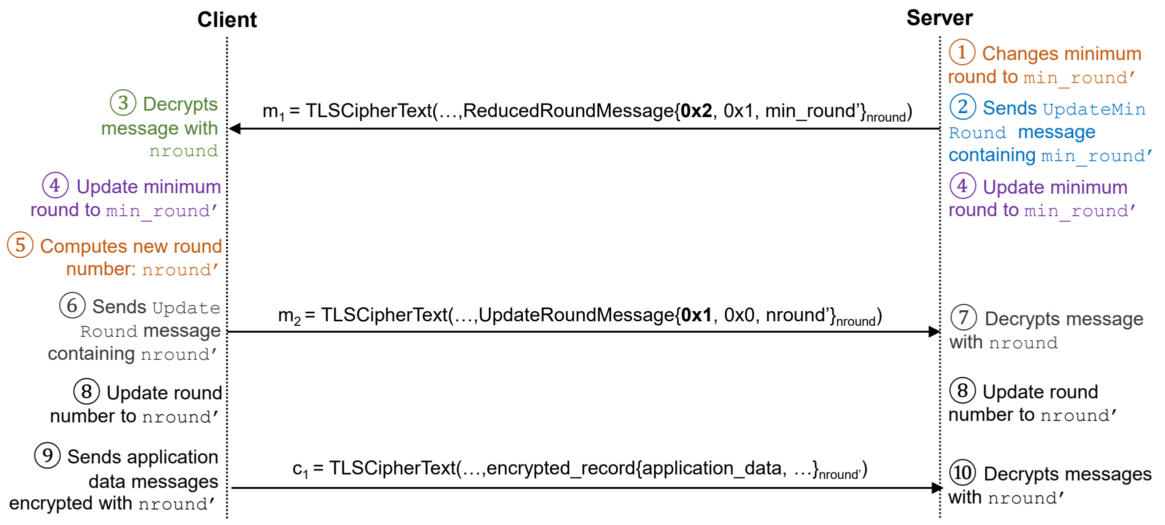

Client
Server
① Changes minimum round to min_round′
② Sends UpdateMin Round message containing min_round′
m_1 = TLSCipherText(…,ReducedRoundMessage{**0x2**, 0x1, min_round'}$_{nround}$)
③ Decrypts message with nround
④ Update minimum round to min_round′
④ Update minimum round to min_round′
⑤ Computes new round number: nround′
⑥ Sends Update Round message containing nround′
m_2 = TLSCipherText(…,UpdateRoundMessage{**0x1**, 0x0, nround'}$_{nround}$)
⑦ Decrypts message with nround
⑧ Update round number to nround′
⑧ Update round number to nround′
⑨ Sends application data messages encrypted with nround′
c_1 = TLSCipherText(…,encrypted_record{application_data, …}$_{nround'}$)
⑩ Decrypts messages with nround′

Figure 7: Reduced-round Round Configuration Update on TLS 1.3.

5.1.2.3 Policy Update

Determined by the operator during the handshake phase, the policy is the main seed governing the appropriate dynamic round number mechanism. The operator may need to change the current policy and, depending on the selected policy, update or communicate the list of permitted round numbers. The policy update is supported by an *UpdatePolicy* message, with a variable size (3 bytes minimum) based on the updated policy. The first 3 bytes contain:

- $uint8$ ReducedRoundMessageType $< 0x3 >$;
- $uint8$ responseFlag $< 0x1 >$;
- $uint8$ policy_value $< 0x0..0xFF >$.

If the new policy is the Single Fixed policy ($0x0$), the payload contains an extra byte that specifies the single round allowed by the operator. If the new policy is the

Multiple Fixed policy ($0x2$), the payload contains the values of the $n_{allowed}$ rounds (1 byte per round value) preceded by a 1-byte size value equal to $n_{allowed}$.

Since the IoT device may not be aware of a user-configured specific policy (which can be defined by the user beyond the three modes proposed in this work), the *UpdatePolicy* procedure can face a "unknown policy" response from the IoT device. In this case, the IoT device must reject the update request ensuring maintenance, repair and operating supply (MRO).

1. known Policy

When the operator wants to change the policy, it sends an *UpdatePolicy* message to the IoT device encrypted with the current round value. The response flag set to $0x1$ implies the IoT device must acknowledge the received policy by sending a response message to the operator. If the IoT device knows the requested policy, it computes the number of rounds to be used based on this new policy. In response, the IoT device sends the new round value through an *UpdateRound* message encrypted with the current number of rounds (response flag set to $0x0$). Finally, both sides update their policy and their stored round value. Figure 8 shows the step-by-step diagram of the Policy Update functionality for an known policy.

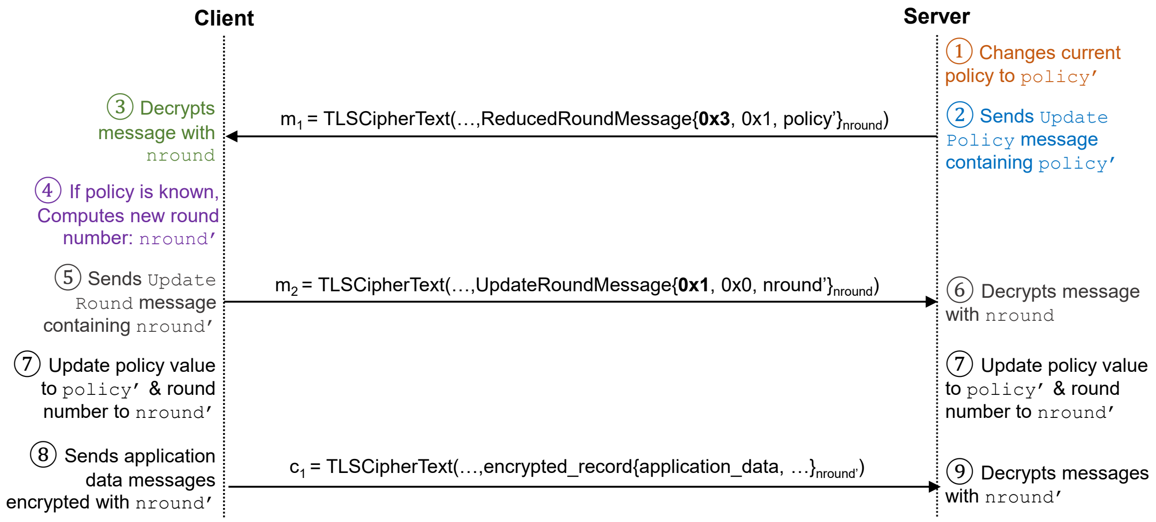

Client
Server
① Changes current policy to policy′
② Sends Update Policy message containing policy′
m1 = TLSCipherText(…,ReducedRoundMessage{0x3, 0x1, policy′}nround)
③ Decrypts message with nround
④ If policy is known, Computes new round number: nround′
⑤ Sends Update Round message containing nround′
m2 = TLSCipherText(…,UpdateRoundMessage{0x1, 0x0, nround′}nround)
⑥ Decrypts message with nround
⑦ Update policy value to policy′ & round number to nround′
⑦ Update policy value to policy′ & round number to nround′
⑧ Sends application data messages encrypted with nround′
c1 = TLSCipherText(…,encrypted_record{application_data, …}nround′)
⑨ Decrypts messages with nround′

Figure 8: Reduced-round Known Policy Update on TLS 1.3.

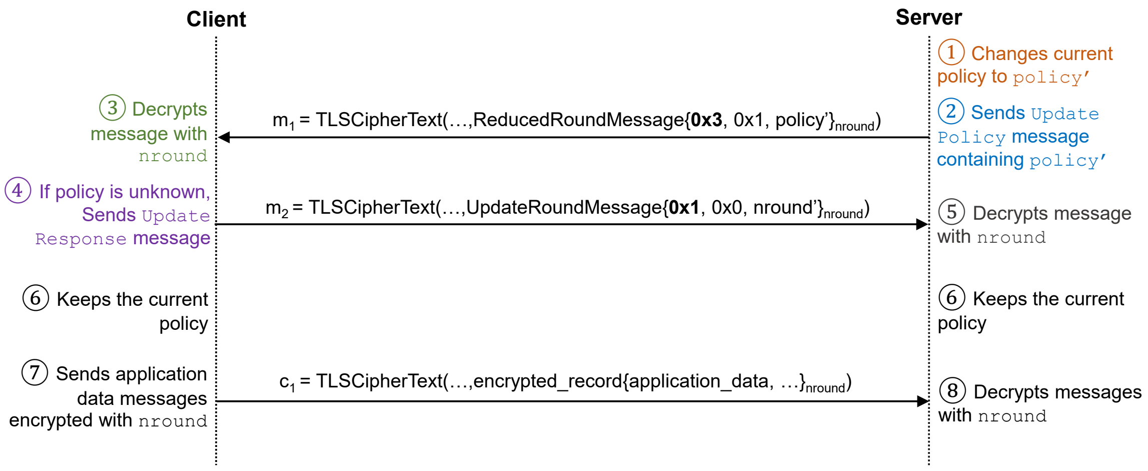

Client
Server
1 Changes current policy to policy′
2 Sends Update Policy message containing policy′
3 Decrypts message with nround
m1 = TLSCipherText(…,ReducedRoundMessage{0x3, 0x1, policy'}nround)
4 If policy is unknown, Sends Update Response message
m2 = TLSCipherText(…,UpdateRoundMessage{0x1, 0x0, nround'}nround)
5 Decrypts message with nround
6 Keeps the current policy
6 Keeps the current policy
7 Sends application data messages encrypted with nround
c1 = TLSCipherText(…,encrypted_record{application_data, …}nround)
8 Decrypts messages with nround

Figure 9: Reduced-round Unknown Policy Update on TLS 1.3.

2. Unknown Policy

Once the *UpdatePolicy* is decrypted by the IoT device, it may be possible that the local TLS implementation does not support the suggested new policy, then, the IoT device sends an error message to the operator using an *UpdateResponse* message (encrypted with the current round number). This message is treated as an error message by the operator that will keep the last IoT-accepted policy and silently discard the last policy change. Figure 9 presents the step-by-step diagram of the Policy Update functionality for an unknown policy.

5.1.2.4 UpdateResponse Message

UpdateResponse message is used in the Dynamic Round Update procedures to acknowledge the round modification on the operator side and in the Policy Update to notify the operator that the requested policy is unknown. The message is fixed to 2 bytes containing:

- $uint8$ ReducedRoundMessageType $< 0x4 >$;
- $uint8$ payload $< 0x0..0x4 >$.

The payload byte contains the previously received $ReducedRoundMessageType$ value to confirm that the message has been received correctly.

5.2 Dynamic Round Number Mechanism

In this subsection we describe the Dynamic round number mechanisms according to the established policy.

To set the encryption round number in real-time, the IoT device must use a dynamic round selector mechanism which is determined according to the policy chosen by the operator. During the handshake phase, after receiving the message from the server, the client is aware of the chosen policy and can thereby start using the appropriate mechanism to initialize the number of rounds and encrypt its first message. Listing 2 shows the policy list: policy id stored in a 4-bit structure and *policy_value* is stored within the 4 remaining bits.

Listing 2: List of Policies

```
enum {
    single_fixed(0),
    proportional(1),
    multiple_fixed(2),
    (16)
} Policies;
```

5.2.1 Single Fixed Policy ($0x0$)

This policy sets the round selection mechanism to a single value determined by the operator and passed to the client within the hello message server. The purpose of this policy is to let the operator set the number of encryption rounds to a specific value between the full and minimum round. This policy is compatible with both battery level-aware and non-battery level-aware IoT devices. The *policy_value* part is not used here.

5.2.2 Proportional Policy ($0x1$)

This policy proportionally determines the number of round based on the current battery level. Each round, from full to minimum round, shares a specific range of battery level determined by the *policy_value* field. Since this policy depends on the battery level, it is compatible with battery level-aware IoT devices only.

We define n_r the number of secure possible rounds of a given encryption algorithm such that:

$$\begin{aligned} & n_r = round_{full} - round_{min} + 1 \\ & where \quad round_{min} \geq 1 \end{aligned} \tag{1}$$

The dynamic round number mechanism under the Proportional policy is defined by the functions F and G such that:

$$\begin{aligned} & \forall B \in [0, 100], \quad F(B) = \left\lceil \frac{n_r B}{100} \right\rceil \\ & and, \quad G(F(B)) = F(B) + round_{min} - 1 \end{aligned} \tag{2}$$

5.2.3 Multiple Fixed Policy ($0x2$)

This policy is equivalent to the Proportional policy except that instead of using all rounds between full and minimum value, it uses a list of rounds fixed by the operator and transmitted to the client in the server hello message. The size of the list must be between 2 and $round_{full} - round_{min}$ included. Since the number of round varies depending on the battery level, this policy is available for battery level-aware IoT devices only. We define n'_r the number of secure allowed rounds of a given encryption algorithm according to the operator round list l_r such that:

$$\begin{aligned} & \forall n \in \mathbb{N}, \quad l_r(1) < l_r(2) < ... < l_r(n) \\ & where \quad n'_r = Card(l_r) \quad \& \quad 1 < n'_r < n_r \end{aligned} \tag{3}$$

The dynamic round number mechanism under the Multiple Fixed policy is defined by the functions F' and G' such that:

$$\begin{aligned} & \forall B \in [0, 100], \quad F'(B) = \left\lceil \frac{n'_r B}{100} \right\rceil \\ & and, \quad G'(F'(B)) = l_r(F'(B)) \end{aligned} \tag{4}$$

5.3 PoC Implementation

We have integrated our reduced-rounds extension into the TLS protocol via the WolfSSL project. This custom build was embedded on the Texas Instrument Tiva C TM4C1294 Connected Launchpad as part of our first PoC. WolfSSL is a lightweight, open-source SSL/TLS library implemented in ANSI C programming language and focusing on constrained devices. WolfSSL can be embedded on limited IoT devices as it has a small code size requirement, uses only a small resource of memory and supports a variety of environments, including real-time operating systems (RTOS) environments. The Tiva C Series microcontroller is an IoT device using a 120 MHz 32-bit ARM Cortex-M4 CPU, 1 MB of flash storage and 256 KB of SRAM suitable for industrial applications.

To build our PoC, we first added our reduced-round extension to the handshake and *application_data* protocols of TLS (version 1.2 and 1.3) and implemented the following features: the *UpdateRound*, *UpdateMinRound*, *UpdatePolicy* and *UpdateResponse* messages with the *single_fixed*, *proportional*, and *multiple_fixed* policies. In the second step, we embedded our modified version of WolfSSL on the connected launchpad coupled with the CC3100 wireless network processor module to evaluate our solution with the 802.11 Wi-Fi network.

5.3.1 WolfSSL Framework Overview

To implement our dynamic reduced-round extension within the WolfSSL Framework we needed to work with a server instance and a client instance which exchange data through TLS packets over the network. WolfSSL library's major files for our implementation are the following:

- *Client.c* and *Server.c*: Containing the main functions to run a Client and a Server instance. We add our extension as a configurable terminal option with arguments to define the type of IoT device (mains or battery-powered) on the client side and the policy (single fixed, proportional or multiple fixed) used on the server side.

- *tls.c* and *tls13.c*: Containing the TLS structure according to messages types, the TLS extensions (initialisation, configuration, management) and the thread-/Semaphore management. We add our Reduced-round extension functions to initialize and to manage it for each message type.

- *ssl.c*: Make the link between *Client.c/Server.c* and *tls.c*. We add the functions to pass our arguments entered in our client and server instances to the functions managing the operation of our extension.
- *internal.c*: Manage the data sending, receiving, storage, encryption and decryption. We make some modification to pass some data from *tls.c/tls13.c* to operate our extension
- *aes.c*: Containing the Encryption and Decryption functions for AES cipher. We add the round number variable and modify the AES Encryption/Decryption function to follow reduced-round functionalities.

In addition, we activated DEBUG_WOLFSSL and WOLFSSL_DEBUG_TLS options to enable WolfSSL debugging support and monitor every TLS packet.

5.3.2 TLS Protocol Implementation

5.3.2.1 Handshake

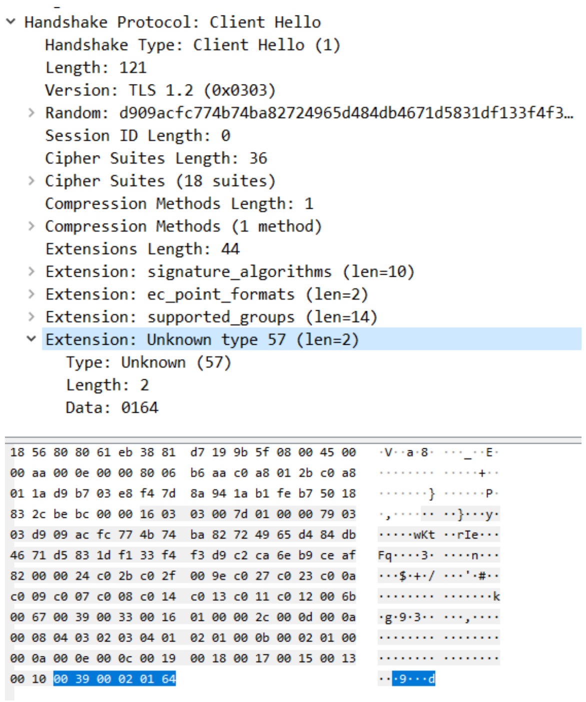
1. ***ClientHello*** **Sending**

In the *ClientHello* message the reduced-round extension data is one byte containing the Battery Level of the Client. Battery level value ranges from 0 to 101:

- 101 ($< 0x65 >$) define a Battery-less IoT Client;
- 0 to 100 ($< 0x00 \ldots 0x64 >$) define the Battery level of the IoT Client.

The Figure 7 shows an example of the *ClientHello* message sent from the Client to the Server with the reduced-round extension: $0x003900020164$

- $< 0x0039 > = 57$ is the IANA number chosen for our reduced-round extension;
- $< 0x0002 >$ is the total length of the extension;
- $< 0x01 >$ is the data length of the extension;



```
⌄ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 121
    Version: TLS 1.2 (0x0303)
  › Random: d909acfc774b74ba82724965d484db4671d5831df133f4f3…
    Session ID Length: 0
    Cipher Suites Length: 36
  › Cipher Suites (18 suites)
    Compression Methods Length: 1
  › Compression Methods (1 method)
    Extensions Length: 44
  › Extension: signature_algorithms (len=10)
  › Extension: ec_point_formats (len=2)
  › Extension: supported_groups (len=14)
  ⌄ Extension: Unknown type 57 (len=2)
      Type: Unknown (57)
      Length: 2
      Data: 0164

18 56 80 80 61 eb 38 81  d7 19 9b 5f 08 00 45 00   ·V··a·8· ···_··E·
00 aa 00 0e 00 00 80 06  b6 aa c0 a8 01 2b c0 a8   ········ ·····+··
01 1a d9 b7 03 e8 f4 7d  8a 94 1a b1 fe b7 50 18   ·······} ······P·
83 2c be bc 00 00 16 03  03 00 7d 01 00 00 79 03   ·,······ ··}···y·
03 d9 09 ac fc 77 4b 74  ba 82 72 49 65 d4 84 db   ·····wKt ··rIe···
46 71 d5 83 1d f1 33 f4  f3 d9 c2 ca 6e b9 ce af   Fq····3· ····n···
82 00 00 24 c0 2b c0 2f  00 9e c0 27 c0 23 c0 0a   ···$·+·/ ···'·#··
c0 09 c0 07 c0 08 c0 14  c0 13 c0 11 c0 12 00 6b   ········ ·······k
00 67 00 39 00 33 00 16  01 00 00 2c 00 0d 00 0a   ·g·9·3·· ···,····
00 08 04 03 02 03 04 01  02 01 00 0b 00 02 01 00   ········ ········
00 0a 00 0e 00 0c 00 19  00 18 00 17 00 15 00 13   ········ ········
00 10 00 39 00 02 01 64                            ···9···d
```

Figure 7: Wireshark Capture of the *ClientHello* message

- $< 0x64 >= 100$ is the Battery Level (battery-powered device).

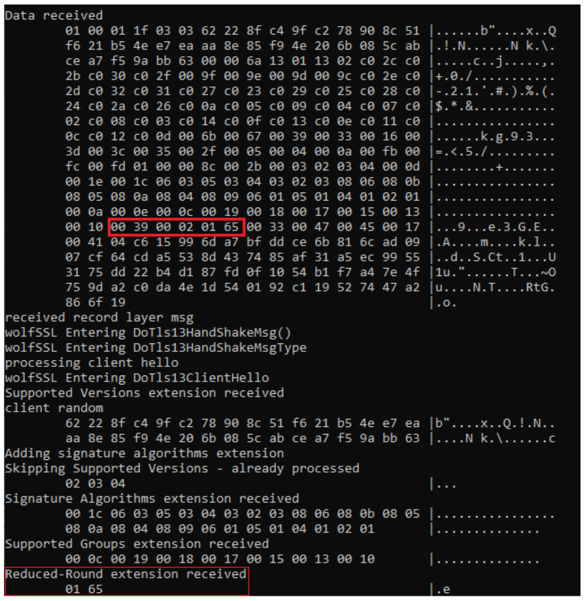
2. *ClientHello* Reception

When the server receives the *ClientHello* message it parse the data in order to find all the extension inside the message. If the server recognize the reduced-round extension, it store the battery level value received and initialize the extension inside the *ServerHello* message. The Figure 8 shows an example of the received *ClientHello* Data inside the Server Instance and the parsing of the extensions.

3. *ServerHello* Sending

In the *ServerHello* message the reduced-round extension data is minimum two bytes containing the input Policy and the minimum round value (depending on the chosen cipher). The policy value is contained between 1 and 3:

- $< 0x01 >$ for the Single Fixed policy;
- $< 0x02 >$ for the Proportional policy;



```
Data received
        01 00 01 1f 03 03 62 22 8f c4 9f c2 78 90 8c 51 |......b"....x..Q
        f6 21 b5 4e e7 ea aa 8e 85 f9 4e 20 6b 08 5c ab |.!.N......N k.\.
        ce a7 f5 9a bb 63 00 00 6a 13 01 13 02 c0 2c c0 |.....c..j.....,.
        2b c0 30 c0 2f 00 9f 00 9e 00 9d 00 9c c0 2e c0 |+.0./...........
        2d c0 32 c0 31 c0 27 c0 23 c0 29 c0 25 c0 28 c0 |-.2.1.'.#.).%.(.
        24 c0 2a c0 26 c0 0a c0 05 c0 09 c0 04 c0 07 c0 |$.*.&...........
        02 c0 08 c0 03 c0 14 c0 0f c0 13 c0 0e c0 11 c0 |................
        0c c0 12 c0 0d 00 6b 00 67 00 39 00 33 00 16 00 |......k.g.9.3...
        3d 00 3c 00 35 00 2f 00 05 00 04 00 0a 00 fb 00 |=.<.5./.........
        fc 00 fd 01 00 00 8c 00 2b 00 03 02 03 04 00 0d |........+.......
        00 1e 00 1c 06 03 05 03 04 03 02 03 08 06 08 0b |................
        08 05 08 0a 08 04 08 09 06 01 05 01 04 01 02 01 |................
        00 0a 00 0e 00 0c 00 19 00 18 00 17 00 15 00 13 |................
        00 10 00 39 00 02 01 65 00 33 00 47 00 45 00 17 |...9...e.3.G.E..
        00 41 04 c6 15 99 6d a7 bf dd ce 6b 81 6c ad 09 |.A....m....k.l..
        07 cf 64 cd a5 53 8d 43 74 85 af 31 a5 ec 99 55 |..d..S.Ct..1...U
        31 75 dd 22 b4 d1 87 fd 0f 10 54 b1 f7 a4 7e 4f |1u.".......T...~O
        75 9d a2 c0 da 4e 1d 54 01 92 c1 19 52 74 47 a2 |u....N.T....RtG.
        86 6f 19                                        |.o.
received record layer msg
wolfSSL Entering DoTls13HandShakeMsg()
wolfSSL Entering DoTls13HandShakeMsgType
processing client hello
wolfSSL Entering DoTls13ClientHello
Supported Versions extension received
client random
        62 22 8f c4 9f c2 78 90 8c 51 f6 21 b5 4e e7 ea |b"....x..Q.!.N..
        aa 8e 85 f9 4e 20 6b 08 5c ab ce a7 f5 9a bb 63 |....N k.\......c
Adding signature algorithms extension
Skipping Supported Versions - already processed
        02 03 04                                        |...
Signature Algorithms extension received
        00 1c 06 03 05 03 04 03 02 03 08 06 08 0b 08 05 |................
        08 0a 08 04 08 09 06 01 05 01 04 01 02 01       |..............
Supported Groups extension received
        00 0c 00 19 00 18 00 17 00 15 00 13 00 10       |..............
Reduced-Round extension received
        01 65                                           |.e
```

Figure 8: Server Instance *ClientHello* message reception and extension parsing

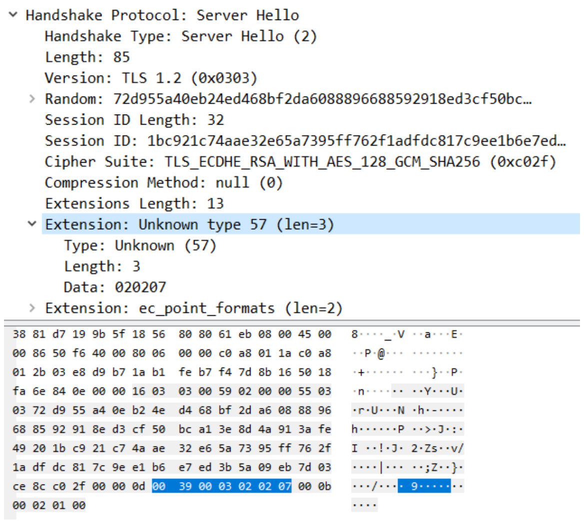
- $< 0x03 >$ for the Multiple Fixed policy.

If the operator chose the Single Fixed policy, the reduced-round extension data length is 3 bytes. First byte contains the Single Fixed policy value, second byte contains the minimum round value and third byte contains the allowed round number.

If the operator chose the Multiple Fixed policy, the reduced-round extension data length depends on the number of reduced-round allowed. Besides the Multiple fixed policy value and the minimum round value, it contains the round list length on one byte and every allowed round value on one byte each.

The Figure 9 shows an example of the *ServerHello* message sent from the Server to the Client with the reduced-round extension: $0x00390003020207$

- $< 0x0039 > = 57$ is the IANA number chosen for our reduced-round extension;
- $< 0x0003 >$ is the total length of the extension;
- $< 0x02 >$ is the data length of the extension;
- $< 0x02 >$ is the Proportional policy;
- $< 0x07 >$ is the minimum round for AES-128.



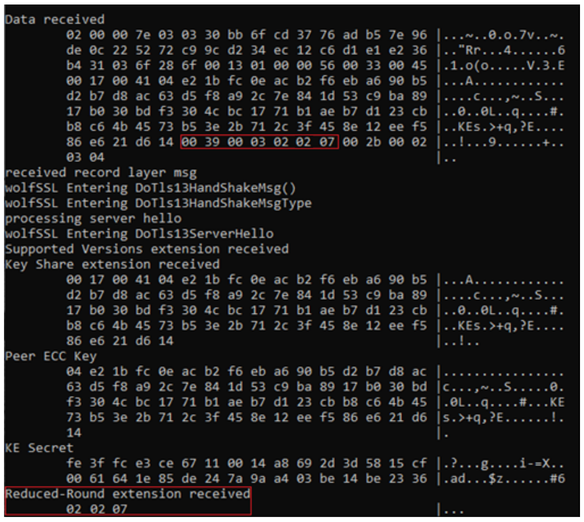
```
v Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 85
    Version: TLS 1.2 (0x0303)
  > Random: 72d955a40eb24ed468bf2da6088896688592918ed3cf50bc…
    Session ID Length: 32
    Session ID: 1bc921c74aae32e65a7395ff762f1adfdc817c9ee1b6e7ed…
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
    Compression Method: null (0)
    Extensions Length: 13
  v Extension: Unknown type 57 (len=3)
      Type: Unknown (57)
      Length: 3
      Data: 020207
  > Extension: ec_point_formats (len=2)

38 81 d7 19 9b 5f 18 56  80 80 61 eb 08 00 45 00   8····_·V ··a···E·
00 86 50 f6 40 00 80 06  00 00 c0 a8 01 1a c0 a8   ··P·@··· ········
01 2b 03 e8 d9 b7 1a b1  fe b7 f4 7d 8b 16 50 18   ·+······ ···}··P·
fa 6e 84 0e 00 00 16 03  03 00 59 02 00 00 55 03   ·n······ ··Y···U·
03 72 d9 55 a4 0e b2 4e  d4 68 bf 2d a6 08 88 96   ·r·U···N ·h·-····
68 85 92 91 8e d3 cf 50  bc a1 3e 8d 4a 91 3a fe   h······P ··>·J·:·
49 20 1b c9 21 c7 4a ae  32 e6 5a 73 95 ff 76 2f   I ··!·J· 2·Zs··v/
1a df dc 81 7c 9e e1 b6  e7 ed 3b 5a 09 eb 7d 03   ····|··· ··;Z··}·
ce 8c c0 2f 00 00 0d 00  39 00 03 02 02 07 00 0b   ···/···· 9·······
00 02 01 00                                        ····
```

Figure 9: Wireshark Capture of the *ServerHello* message

4. *ServerHello* Reception

When the client receives the *ServerHello* message it parses the data in order to find all the extension inside the message. If the client recognizes the reduced-round extension, it stores policy, minimum round value and (optionally) the allowed encryption rounds. The Figure 10 shows an example of the received *ServerHello* Data inside the Client Instance and the parsing of the extensions.



```
Data received
        02 00 00 7e 03 03 30 bb 6f cd 37 76 ad b5 7e 96 |...~..0.o.7v..~.
        de 0c 22 52 72 c9 9c d2 34 ec 12 c6 d1 e1 e2 36 |.."Rr...4......6
        b4 31 03 6f 28 6f 00 13 01 00 00 56 00 33 00 45 |.1.o(o.....V.3.E
        00 17 00 41 04 e2 1b fc 0e ac b2 f6 eb a6 90 b5 |...A............
        d2 b7 d8 ac 63 d5 f8 a9 2c 7e 84 1d 53 c9 ba 89 |....c...,~..S...
        17 b0 30 bd f3 30 4c bc 17 71 b1 ae b7 d1 23 cb |..0..0L..q....#.
        b8 c6 4b 45 73 b5 3e 2b 71 2c 3f 45 8e 12 ee f5 |..KEs.>+q,?E....
        86 e6 21 d6 14 00 39 00 03 02 02 07 00 2b 00 02 |..!...9......+..
        03 04                                           |..
received record layer msg
wolfSSL Entering DoTls13HandShakeMsg()
wolfSSL Entering DoTls13HandShakeMsgType
processing server hello
wolfSSL Entering DoTls13ServerHello
Supported Versions extension received
Key Share extension received
        00 17 00 41 04 e2 1b fc 0e ac b2 f6 eb a6 90 b5 |...A............
        d2 b7 d8 ac 63 d5 f8 a9 2c 7e 84 1d 53 c9 ba 89 |....c...,~..S...
        17 b0 30 bd f3 30 4c bc 17 71 b1 ae b7 d1 23 cb |..0..0L..q....#.
        b8 c6 4b 45 73 b5 3e 2b 71 2c 3f 45 8e 12 ee f5 |..KEs.>+q,?E....
        86 e6 21 d6 14                                  |..!..
Peer ECC Key
        04 e2 1b fc 0e ac b2 f6 eb a6 90 b5 d2 b7 d8 ac |................
        63 d5 f8 a9 2c 7e 84 1d 53 c9 ba 89 17 b0 30 bd |c...,~..S.....0.
        f3 30 4c bc 17 71 b1 ae b7 d1 23 cb b8 c6 4b 45 |.0L..q....#...KE
        73 b5 3e 2b 71 2c 3f 45 8e 12 ee f5 86 e6 21 d6 |s.>+q,?E......!.
        14                                              |.
KE Secret
        fe 3f fc e3 ce 67 11 00 14 a8 69 2d 3d 58 15 cf |.?...g....i-=X..
        00 61 64 1e 85 de 24 7a 9a a4 03 be 14 be 23 36 |.ad...$z......#6
Reduced-Round extension received
        02 02 07                                        |...
```

Figure 10: Client Instance *ServerHello* message reception and extension parsing

5.3.2.2 Application Data

1. Round Update

Described in the Section 5.1.2.1 the Round Update functionality goal is to change the encryption round number that secures the messages sent between the IoT device (Client) and the Operator (Server). Round update functionality and *UpdateRound* message only used for battery IoT device with Proportional and Multiple Fixed Policies and must respond to the following challenges:

- Client does not need to send its battery level after the handshake phase;
- Client and Server must use the numbers of round computed (on the client side) according to the IoT device battery level and/or the policy used;
- The number of rounds must be synchronized between Client and Server;
- Client must not ask for a round number value outside the set [min_round, max_round] for the Encryption Algorithm used.

The *UpdateRound* message Format is detailed on the Table 12. The message must be 3 bytes length and contains the message type, a response flag and the new round value (between minimum and maximum round for the Encryption algorithm used).

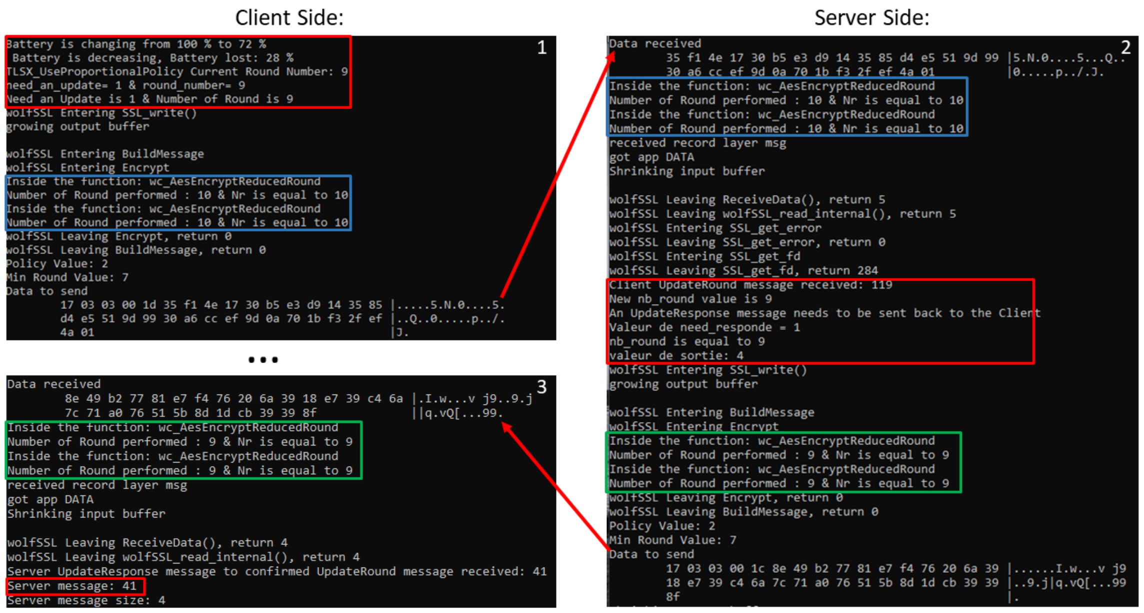
Table 12: *UpdateRound* Message Format

Byte	**0x0**	**0x1**	**0x2**
Value Type	$message\_type$	$response\_flag$	$round\_number$
Value Range	$0x1$	$0x0$ or $0x1$	min_round to max_round
Descritpion	*UpdateRound* message ID	$0x0$ no response asked $0x1$ need a response	New round value

In the *tls.c* file, we add the *SendUpdateRoundMessage* function to verify the number of round to use according to the current battery level before every Client message sending. Depending on the chosen policy, the function determines if the current round number needs to be updated. If a Round Update is needed, it will automatically alert the Client to send an *UpdateRound* message containing the new round number and to follow the Round Update scheme.

When the Server receives the *UpdateRound* message, both Client and Server will modify their current round with the new round value and the *UpdateRoundNumber* function. The return value is different if the Client is asking for an *UpdateResponse*. If the Client is asking for a response (response flag = $0x1$), the Server sends an *UpdateResponse* with the value encrypted with the new round number. The function *UpdateResponse* is 2 bytes containing the message type ($0x4$) and the message type it responds to ($0x1$).

Figure 11 show an example of the round update functionality within the reduced-round extension. Both Client (on the left) and Server (on the right) are shown, the battery is decreasing from 100% to 72% with proportional policy.



```
Battery is changing from 100 % to 72 %
 Battery is decreasing, Battery lost: 28 %
TLSX_UseProportionalPolicy Current Round Number: 9
need_an_update= 1 & round_number= 9
Need an Update is 1 & Number of Round is 9
wolfSSL Entering SSL_write()
growing output buffer

wolfSSL Entering BuildMessage
wolfSSL Entering Encrypt
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 10 & Nr is equal to 10
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 10 & Nr is equal to 10
wolfSSL Leaving Encrypt, return 0
wolfSSL Leaving BuildMessage, return 0
Policy Value: 2
Min Round Value: 7
Data to send
        17 03 03 00 1d 35 f1 4e 17 30 b5 e3 d9 14 35 85 |.....5.N.0....5.
        d4 e5 51 9d 99 30 a6 cc ef 9d 0a 70 1b f3 2f ef |..Q..0.....p../.
        4a 01                                           |J.
```

```
Data received
        35 f1 4e 17 30 b5 e3 d9 14 35 85 d4 e5 51 9d 99 |5.N.0....5...Q..
        30 a6 cc ef 9d 0a 70 1b f3 2f ef 4a 01          |0.....p../.J.
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 10 & Nr is equal to 10
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 10 & Nr is equal to 10
received record layer msg
got app DATA
Shrinking input buffer

wolfSSL Leaving ReceiveData(), return 5
wolfSSL Leaving wolfSSL_read_internal(), return 5
wolfSSL Entering SSL_get_error
wolfSSL Leaving SSL_get_error, return 0
wolfSSL Entering SSL_get_fd
wolfSSL Leaving SSL_get_fd, return 284
Client UpdateRound message received: 119
New nb_round value is 9
An UpdateResponse message needs to be sent back to the Client
Valeur de need_responde = 1
nb_round is equal to 9
valeur de sortie: 4
wolfSSL Entering SSL_write()
growing output buffer

wolfSSL Entering BuildMessage
wolfSSL Entering Encrypt
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
wolfSSL Leaving Encrypt, return 0
wolfSSL Leaving BuildMessage, return 0
Policy Value: 2
Min Round Value: 7
Data to send
        17 03 03 00 1c 8e 49 b2 77 81 e7 f4 76 20 6a 39 |......I.w...v j9
        18 e7 39 c4 6a 7c 71 a0 76 51 5b 8d 1d cb 39 39 |..9.j|q.vQ[...99
        8f                                              |.
```

```
Data received
        8e 49 b2 77 81 e7 f4 76 20 6a 39 18 e7 39 c4 6a |.I.w...v j9..9.j
        7c 71 a0 76 51 5b 8d 1d cb 39 39 8f             ||q.vQ[...99.
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
received record layer msg
got app DATA
Shrinking input buffer

wolfSSL Leaving ReceiveData(), return 4
wolfSSL Leaving wolfSSL_read_internal(), return 4
Server UpdateResponse message to confirmed UpdateRound message received: 41
Server message: 41
Server message size: 4
```

Figure 11: Update Round functionality Example

2. Minimum Round Update

Detailed in the Section 5.1.2.2 the round configuration goal is to change the minimum round number that can be used to secure the messages sent between the IoT device (Client) and the Operator (Server). Round update functionality and *UpdateRound* message only affect battery IoT device using Proportional and Multiple Fixed Policies. However, this functionality can be used for battery IoT device using Single Fixed policy in case of a policy modification (using *UpdatePolicy* message). This functionality must respond to the following challenges:

- Min Round update must be initiated by the Operator to ensure security;
- Client and Server must use the same number of round at the end of the min round update mechanism;
- Operator must not ask for a min round number value outside the set [1 , *max_round*] for the Encryption Algorithm used.

The *UpdateMinRound* message Format is described on the Table 13. The message must be 3 bytes length and contains the message type, a response flag and the new minimum round value (between one and maximum round for the Encryption algorithm used).

Table 13: *UpdateMinRound* Message Format

Byte	**0x0**	**0x1**	**0x2**
Value Type	*message_type*	*response_flag*	*round_number*
Value Range	$0x2$	$0x1$	1 to *max_round*
Description	*UpdateMinRound* message ID	Need a response	New min. round value

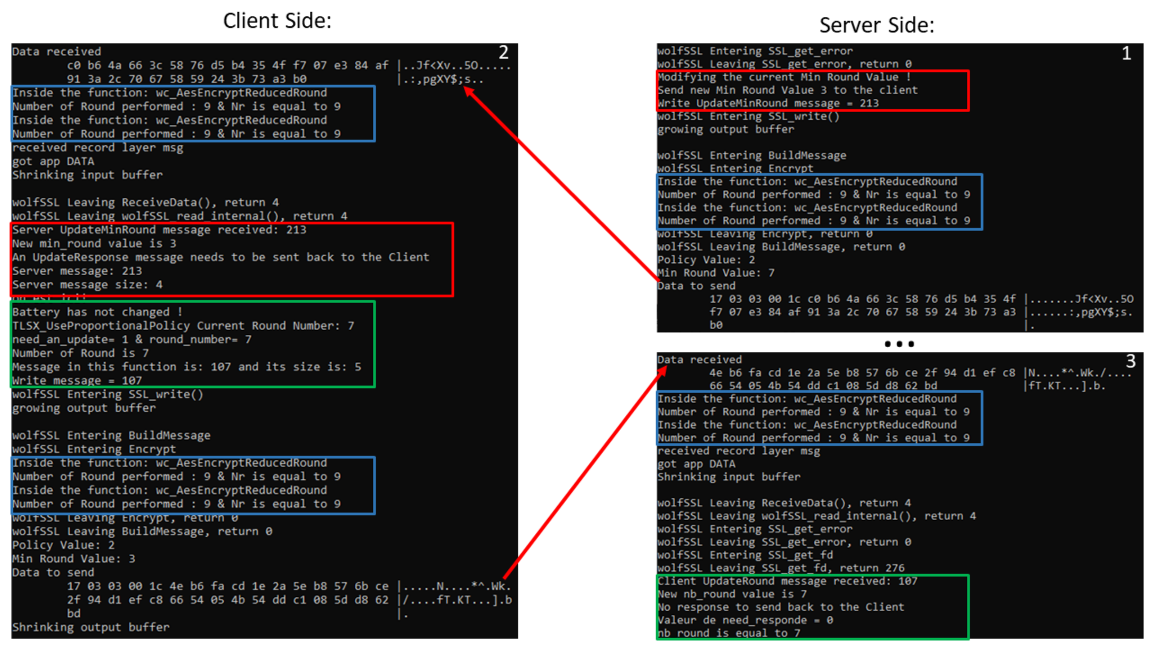
In the *tls.c* file, we add the *SendUpdateMinRoundMessage* function on the server side to verify and sets the minimum round number. *MinRoundNumber* must be different than the previous value already stored, greater than 0 and lower than the *max_round* value of the encryption algorithm. If a Min Round Update is needed, it will automatically alert the Client to send an *UpdateMinRound* message and to follow the Min Round Update scheme describe in the previous slide.

When the client receives the *UpdateMinRound* message, both Client and Server will modify their *min_round* number with the new one with the *UpdateRoundNumber* function. The response flag is always $0x1$ because the Server always needs a response after an *UpdateMinRound* message. Inside ClientRead, When the Client parses the *UpdateMinRound* message, the Client stores this new *min_round* value and computes the current round value in order to send back a *UpdateRound* message containing the data: $0x0100XX$

- $< 0x01 >$ for the message type: *UpdateRound*;
- $< 0x00 >$ for the response flag: No response is requested;

- $< 0xXX >$ the new round value to use: $XX = round\_nb$.

Figure 12 show an example of the round configuration functionality within the reduced-round extension. Both Client (on the left) and Server (on the right) are shown, the battery is from 58% with proportional policy and the minimum round is modified from 7 to 3 rounds.



```
Data received
        c0 b6 4a 66 3c 58 76 d5 b4 35 4f f7 07 e3 84 af |..Jf<Xv..5O.....
        91 3a 2c 70 67 58 59 24 3b 73 a3 b0             |.:,pgXY$;s..
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
received record layer msg
got app DATA
Shrinking input buffer

wolfSSL Leaving ReceiveData(), return 4
wolfSSL Leaving wolfSSL_read_internal(), return 4
Server UpdateMinRound message received: 213
New min_round value is 3
An UpdateResponse message needs to be sent back to the Client
Server message: 213
Server message size: 4
[illegible]
Battery has not changed !
TLSX_UseProportionalPolicy Current Round Number: 7
need_an_update= 1 & round_number= 7
Number of Round is 7
Message in this function is: 107 and its size is: 5
Write message = 107
wolfSSL Entering SSL_write()
growing output buffer

wolfSSL Entering BuildMessage
wolfSSL Entering Encrypt
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
wolfSSL Leaving Encrypt, return 0
wolfSSL Leaving BuildMessage, return 0
Policy Value: 2
Min Round Value: 3
Data to send
        17 03 03 00 1c 4e b6 fa cd 1e 2a 5e b8 57 6b ce |.....N....*^.Wk.
        2f 94 d1 ef c8 66 54 05 4b 54 dd c1 08 5d d8 62 |/....fT.KT...].b
        bd                                              |.
Shrinking output buffer
```

```
wolfSSL Entering SSL_get_error
wolfSSL Leaving SSL_get_error, return 0
Modifying the current Min Round Value !
Send new Min Round Value 3 to the client
Write UpdateMinRound message = 213
wolfSSL Entering SSL_write()
growing output buffer

wolfSSL Entering BuildMessage
wolfSSL Entering Encrypt
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
wolfSSL Leaving Encrypt, return 0
wolfSSL Leaving BuildMessage, return 0
Policy Value: 2
Min Round Value: 7
Data to send
        17 03 03 00 1c c0 b6 4a 66 3c 58 76 d5 b4 35 4f |.......Jf<Xv..5O
        f7 07 e3 84 af 91 3a 2c 70 67 58 59 24 3b 73 a3 |......:,pgXY$;s.
        b0                                              |.
```

• • •

```
Data received
        4e b6 fa cd 1e 2a 5e b8 57 6b ce 2f 94 d1 ef c8 |N....*^.Wk./....
        66 54 05 4b 54 dd c1 08 5d d8 62 bd             |fT.KT...].b.
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 9 & Nr is equal to 9
received record layer msg
got app DATA
Shrinking input buffer

wolfSSL Leaving ReceiveData(), return 4
wolfSSL Leaving wolfSSL_read_internal(), return 4
wolfSSL Entering SSL_get_error
wolfSSL Leaving SSL_get_error, return 0
wolfSSL Entering SSL_get_fd
wolfSSL Leaving SSL_get_fd, return 276
Client UpdateRound message received: 107
New nb_round value is 7
No response to send back to the Client
Valeur de need_responde = 0
nb_round is equal to 7
```

Figure 12: Update Minimum Round functionality Example

3. Policy Update

Introduced in the Section 5.1.2.3 the policy update functionality goal is to change the policy used to decide the number of round to secure the messages sent between the IoT device (Client) and the Operator (Server). The policy update functionality must respond to the following challenges:

- Policy update must be initiated by the Operator to ensure security;
- Client and Server must use the same updated number of round from the new policy at the end of the policy update mechanism;
- The *UpdatePolicy* message format must be adapted for the asked policy;

- Police update functionality must ensure dynamic input value for Single and Multiple Fixed policies.

The *UpdatePolicy* message format is detailed on the Table 14. The message must be minimum 3 Bytes (Proportional policy) length to maximum $4 + [list\_roundlength]$ Bytes (Multiple Fixed policy) and contains the message type, a response flag, the new policy (between one and three). If the Simple Fixed Policy is specified, the *UpdateMinRound* message must contain the round value selected by the operator. If the Multiple Fixed policy is specified, the *UpdateMinRound* message must contains the round list length and each round number selected by the operator.

Table 14: *UpdatePolicy* Message Format

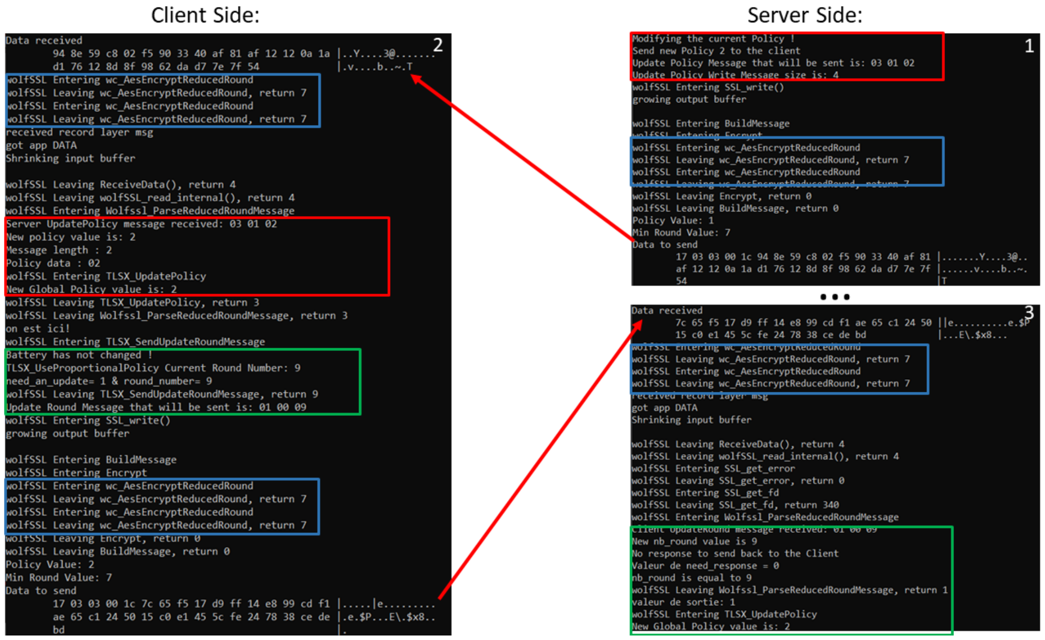
Byte	**0x0**	**0x1**	**0x2**	**0x3**	**0x4**		**0xn**
Value Type	*message_type*	*response_flag*	*policy_value*	*round_number* (Single Fixed Policy) — **or** — *round_list_length* (Multiple Fixed Policy)	$round\_number_1$ (Multiple Fixed Policy)		$round\_number_n$ (Multiple Fixed Policy)
Value Range	$0x3$	$0x1$	$0x1$ to $0x3$	*min_round* to *max_round* — **or** — $0x1$ to $0x255$	*min_round* to *max_round*	...	*min_round* to *max_round*
Descri-ption	*UpdatePolicy* message ID	Need a response	New policy value	Single Fixed round value — **or** — Multiple Fixed round list length	Multiple Fixed first round value		Multiple Fixed last round value

Inside the *tls.c* file, we add *SendUpdatePolicyMessage* function called by the server to verify if the new policy is different from the current one in order to prepare the *UpdatePolicy* message.

When the Client receives the *UpdatePolicy* message, Client will modify its policy value and update is round value according to the new policy with the function

UpdatePolicy. The response flag is always 0x1 because the Server always needs a response after an *UpdatePolicy* message. Inside *ClientRead*, When the Client parses the *UpdatePolicy* message, the Client stores this new policy value and computes the current round value in order to send back a *UpdateRound* message: $0x0100XX$

- $< 0x01 >$ for the message type: *UpdateRound*;
- $< 0x00 >$ for the response flag: No response is requested;
- $< 0xXX >$ the new round value to use: $XX = round\_number$.



```
Data received
        94 8e 59 c8 02 f5 90 33 40 af 81 af 12 12 0a 1a |..Y....3@.......
        d1 76 12 8d 8f 98 62 da d7 7e 7f 54             |.v....b..~.T
wolfSSL Entering wc_AesEncryptReducedRound
wolfSSL Leaving wc_AesEncryptReducedRound, return 7
wolfSSL Entering wc_AesEncryptReducedRound
wolfSSL Leaving wc_AesEncryptReducedRound, return 7
received record layer msg
got app DATA
Shrinking input buffer

wolfSSL Leaving ReceiveData(), return 4
wolfSSL Leaving wolfSSL_read_internal(), return 4
wolfSSL Entering Wolfssl_ParseReducedRoundMessage
Server UpdatePolicy message received: 03 01 02
New policy value is: 2
Message length : 2
Policy data : 02
wolfSSL Entering TLSX_UpdatePolicy
New Global Policy value is: 2
wolfSSL Leaving TLSX_UpdatePolicy, return 3
wolfSSL Leaving Wolfssl_ParseReducedRoundMessage, return 3
on est ici!
wolfSSL Entering TLSX_SendUpdateRoundMessage
Battery has not changed !
TLSX_UseProportionalPolicy Current Round Number: 9
need_an_update= 1 & round_number= 9
wolfSSL Leaving TLSX_SendUpdateRoundMessage, return 9
Update Round Message that will be sent is: 01 00 09
wolfSSL Entering SSL_write()
growing output buffer

wolfSSL Entering BuildMessage
wolfSSL Entering Encrypt
wolfSSL Entering wc_AesEncryptReducedRound
wolfSSL Leaving wc_AesEncryptReducedRound, return 7
wolfSSL Entering wc_AesEncryptReducedRound
wolfSSL Leaving wc_AesEncryptReducedRound, return 7
wolfSSL Leaving Encrypt, return 0
wolfSSL Leaving BuildMessage, return 0
Policy Value: 2
Min Round Value: 7
Data to send
        17 03 03 00 1c 7c 65 f5 17 d9 ff 14 e8 99 cd f1 |.....|e.........
        ae 65 c1 24 50 15 c0 e1 45 5c fe 24 78 38 ce de |.e.$P...E\.$x8..
        bd                                              |.
```

```
Modifying the current Policy !
Send new Policy 2 to the client
Update Policy Message that will be sent is: 03 01 02
Update Policy Write Message size is: 4
wolfSSL Entering SSL_write()
growing output buffer

wolfSSL Entering BuildMessage
wolfSSL Entering Encrypt
wolfSSL Entering wc_AesEncryptReducedRound
wolfSSL Leaving wc_AesEncryptReducedRound, return 7
wolfSSL Entering wc_AesEncryptReducedRound
wolfSSL Leaving wc_AesEncryptReducedRound, return 7
wolfSSL Leaving Encrypt, return 0
wolfSSL Leaving BuildMessage, return 0
Policy Value: 1
Min Round Value: 7
Data to send
        17 03 03 00 1c 94 8e 59 c8 02 f5 90 33 40 af 81 |.......Y....3@..
        af 12 12 0a 1a d1 76 12 8d 8f 98 62 da d7 7e 7f |......v....b..~.
        54                                              |T
```

• • •

```
Data received
        7c 65 f5 17 d9 ff 14 e8 99 cd f1 ae 65 c1 24 50 ||e..........e.$P
        15 c0 e1 45 5c fe 24 78 38 ce de bd             |...E\.$x8...
wolfSSL Entering wc_AesEncryptReducedRound
wolfSSL Leaving wc_AesEncryptReducedRound, return 7
wolfSSL Entering wc_AesEncryptReducedRound
wolfSSL Leaving wc_AesEncryptReducedRound, return 7
received record layer msg
got app DATA
Shrinking input buffer

wolfSSL Leaving ReceiveData(), return 4
wolfSSL Leaving wolfSSL_read_internal(), return 4
wolfSSL Entering SSL_get_error
wolfSSL Leaving SSL_get_error, return 0
wolfSSL Entering SSL_get_fd
wolfSSL Leaving SSL_get_fd, return 340
wolfSSL Entering Wolfssl_ParseReducedRoundMessage
Client UpdateRound message received: 01 00 09
New nb_round value is 9
No response to send back to the Client
Valeur de need_response = 0
nb_round is equal to 9
wolfSSL Leaving Wolfssl_ParseReducedRoundMessage, return 1
valeur de sortie: 1
wolfSSL Entering TLSX_UpdatePolicy
New Global Policy value is: 2
```

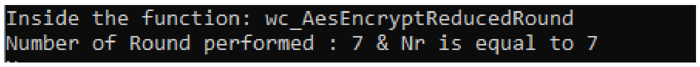
Figure 13: Update Policy functionality Example

Figure 13 shows an example of the policy update functionality within the reduced-round extension. Both Client (on the left) and Server (on the right) are shown, the battery is 72% and the policy is updated from from Single Fixed (with 7 round) to proportional .

5.3.3 Dynamic Reduced-Round Mechanism

5.3.3.1 AES Reduced-Round Encryption

Inside *aes.c* file, we modified AES encryption function used for both encryption and decryption. We create a variable that takes the current round value (defined by the operator policy and the IoT device battery level), and a counter to count each round to verify AES encryption stop at the current round number. The Figure 14 shows the debugging output inside the client and the server instances to verify the number of round performed by AES encryption function.



```
Inside the function: wc_AesEncryptReducedRound
Number of Round performed : 7 & Nr is equal to 7
```

Figure 14: AES Reduced-Round Encryption debugging example

5.3.3.2 Operator Policies

1. Single Fixed Policy

The single fixed policy sets a single round value allowed by the operator. The round value ranges from the minimum round allowed by the operator and the full round for the given cipher. The round value allowed is chosen by the operator and then used to define the fixed round number by the IoT device. Inside the *tls.c* file, we add the *SingleFixedInput* function called by the client if the operator configured the single fixed policy. This function verifies and store the operator-chosen encryption round within the client instance.

2. Proportional Policy

The proportional policy dynamically sets the encryption round value to use according the current battery level of the IoT device. The round values range from the minimum round allowed by the operator and the full round for the given cipher. The policy must ensure that each round must share a battery level equal range. Within the *tls.c* file, the *UseProportionalPolicy* function is called by the client if the operator configured the proportional policy in order to compute the number of round to use according to the current battery level.

The Figure 15 shows the *UseProportionalPolicy* function that compute the number of round according to different client battery level values in client instances.

3. Multiple Fixed Policy

The multiple fixed policy sets the encryption round value to use according to a list allowed round define by the operator and the current battery level of the IoT

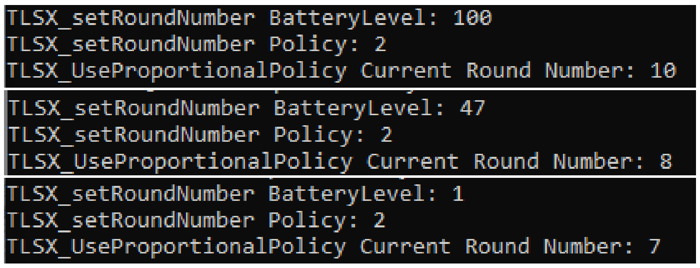

```
TLSX_setRoundNumber BatteryLevel: 100
TLSX_setRoundNumber Policy: 2
TLSX_UseProportionalPolicy Current Round Number: 10
TLSX_setRoundNumber BatteryLevel: 47
TLSX_setRoundNumber Policy: 2
TLSX_UseProportionalPolicy Current Round Number: 8
TLSX_setRoundNumber BatteryLevel: 1
TLSX_setRoundNumber Policy: 2
TLSX_UseProportionalPolicy Current Round Number: 7
```

Figure 15: Proportional Policy round computation example

device . First, this policy is used by the operator to sets the list of allowed round values. Second, using this policy and the round list, the IoT device select the current number of round according to the current battery level. The multiple fixed policy must ensure the following challenges:

- The round list length must be greater than 1 (Single Fixed policy) and less than *max_round–min_round* (Proportional policy);
- Regardless of the number of rounds allowed, each round must share a battery level equal range;
- Order and sort the round list sent by the operator to ensure the proper functioning of reduced-round extension with this policy.

Inside the *tls.c* file, we add the $UseMultipleFixedPolicy$ function called by the IoT device if the operator specifies to use the Multiple Fixed Policy. This function returns the round number that is needed to encrypt the message between the Client and the Server.

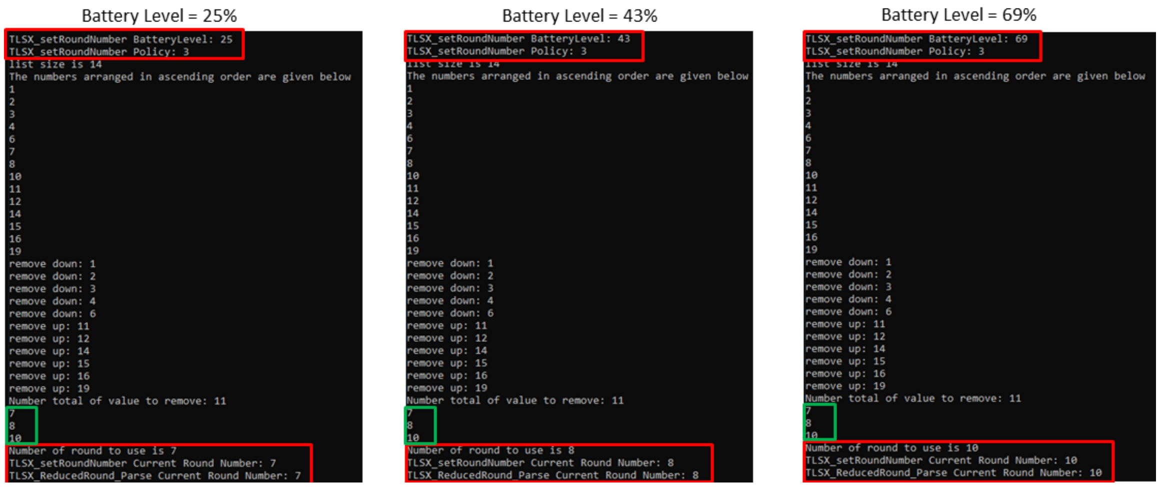
To avoid malfunctions due to user input errors, this function makes same optimization on the operator list:

- Sort the round list in ascending to ensure that the minimum number of rounds is used for low battery values;
- Remove all the round values greater than the maximum round possible to avoid unnecessary over-consumption of the battery;
- Remove all the round values lower than the minimum number of rounds fixed to ensure a sufficient security.

When the Operator initiate the server instance with Multiple Fixed Policy a message is prompt to ask the Operator to enter is round value list according to 2 possibilities:

1. Enter the values one by one: list length first and round values individually;
2. Enter the round value list: $round\_nb_1, \ldots, round\_nb_n$.

The Figure 16 show client instance of round value selection for the Multiple Fixed policy for the same round list $L = \{2, 4, 1, 19, 6, 7, 11, 16, 10, 8, 14, 12, 15, 3\}$ for three different Battery level: 25% on the left, 43% in the middle and 69% on the right.



```
TLSX_setRoundNumber BatteryLevel: 25
TLSX_setRoundNumber Policy: 3
list size is 14
The numbers arranged in ascending order are given below
1
2
3
4
6
7
8
10
11
12
14
15
16
19
remove down: 1
remove down: 2
remove down: 3
remove down: 4
remove down: 6
remove up: 11
remove up: 12
remove up: 14
remove up: 15
remove up: 16
remove up: 19
Number total of value to remove: 11
7
8
10
Number of round to use is 7
TLSX_setRoundNumber Current Round Number: 7
TLSX_ReducedRound_Parse Current Round Number: 7
```

```
TLSX_setRoundNumber BatteryLevel: 43
TLSX_setRoundNumber Policy: 3
list size is 14
The numbers arranged in ascending order are given below
1
2
3
4
6
7
8
10
11
12
14
15
16
19
remove down: 1
remove down: 2
remove down: 3
remove down: 4
remove down: 6
remove up: 11
remove up: 12
remove up: 14
remove up: 15
remove up: 16
remove up: 19
Number total of value to remove: 11
7
8
10
Number of round to use is 8
TLSX_setRoundNumber Current Round Number: 8
TLSX_ReducedRound_Parse Current Round Number: 8
```

```
TLSX_setRoundNumber BatteryLevel: 69
TLSX_setRoundNumber Policy: 3
list size is 14
The numbers arranged in ascending order are given below
1
2
3
4
6
7
8
10
11
12
14
15
16
19
remove down: 1
remove down: 2
remove down: 3
remove down: 4
remove down: 6
remove up: 11
remove up: 12
remove up: 14
remove up: 15
remove up: 16
remove up: 19
Number total of value to remove: 11
7
8
10
Number of round to use is 10
TLSX_setRoundNumber Current Round Number: 10
TLSX_ReducedRound_Parse Current Round Number: 10
```

Figure 16: Update Policy functionality Example

4. Policy Redirection

In case of battery-less IoT device, only the single fixed policy is allowed (not depending of the battery level). If the operator initializes a proportional or multiple fixed policy for a battery-less IoT device ($0x65$ battery level value inside the $ClientHello$ message), the server instance redirects the operator to a single fixed policy. Then, the server asks the operator to set the round value, the default one is full round value (10 for AES-128). In addition, server instance informs the operator of the policy redirection.

5.4 Scenarios & Results

5.4.1 Experiments

Finally, to validate our solution, we evaluated the behaviour of our TLS extension according to different scenarios using TLS1.3 and 128-bit AES encryption, between the IoT device (client) and a server (workstation).

- *Scenario 1*: Single Fixed policy to 10 rounds (equivalent to TLS communication without reduced-round extension) representing the maximum security level;
- *Scenario 2*: Single Fixed policy to 7 rounds representing the minimum level of security;
- *Scenario 3*: Proportional policy varying the security level from 7 to 10 encryption rounds based on the battery level;
- *Scenario 4*: Multiple Fixed Policy to 8 and 9 rounds including a policy update to Single Fixed policy to 7 rounds (battery level at 40%) and a minimum round update from 7 to 8 rounds (battery level at 20%).

In all scenarios, we measured the energy consumption of the Connected Launchpad (client) to communicate with a server (operator) using reduced-round extension over TLS with the Power-Z KM001 USB meter.

5.4.2 Results

5.4.2.1 Power Consumption within TLS

First, we evaluated the power consumption of scenarios sending 286 B of data per second in wireless communication through the TLS1.3 protocol and our reduced-round extensions. The battery level was simulated to demonstrate the IoT device ability to dynamically change its security level according to the policy chosen by the operator. In addition, the minimum round and policy update procedures initiated by the operator are functional on the IoT device. Figure 17 shows the power consumption of each scenario running on the Tiva C Launchpad according to the battery level and the number of AES-128 rounds to encrypt the data. As Scenario 1 and Scenario 2

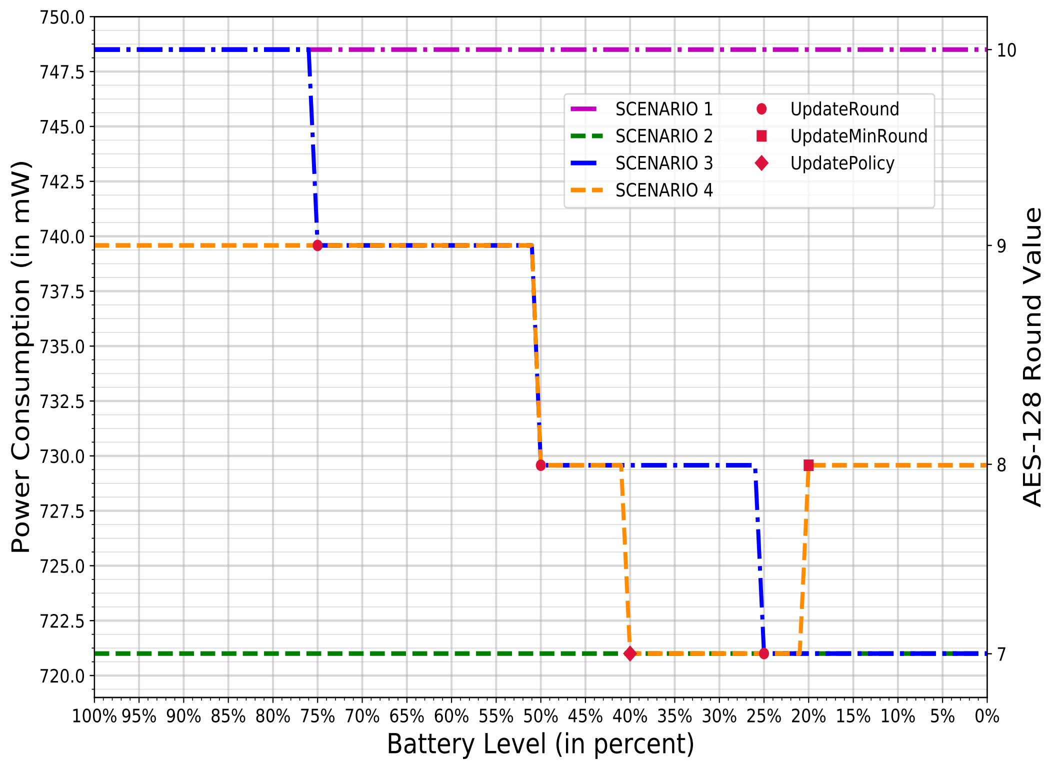


Figure 17: Power consumption of Reduced-Round TLS Extensions Scenarios with AES-128 on TM4C1294 Connected Launchpad.

are under the Single Fixed policy, they do not change encryption rounds despite the battery depletion. Scenario 1 consumes 748.5 mW while Scenario 2 consumes 721 mW, indicating a 27.5 mW reduction in power consumption from 10 to 7 rounds of AES-128 encryption. Scenario 3 shows a decrease in power consumption as a function of the decrease in battery level, affecting the number of AES encryption rounds used (using $UpdateRound$ messages). Scenario 4 shows the variation of the AES round value and thus of the energy consumption depending on the battery level and the $UpdatePolicy$ (at Single Fixed 7 rounds) and $UpdateMinRound$ (at 8 rounds) messages sent by the operator. The average power consumption of Scenarios 3 and 4 is 734.7 mW and 732.9 mW, respectively.

5.4.2.2 Battery Savings

To quantify the benefits of our solution, we computed and analyzed the energy gains realized by the scenarios compared to a TLS communication with full AES-128 (Scenario 1). Figure 18 shows the global power gains (light blue chart) and the protocol

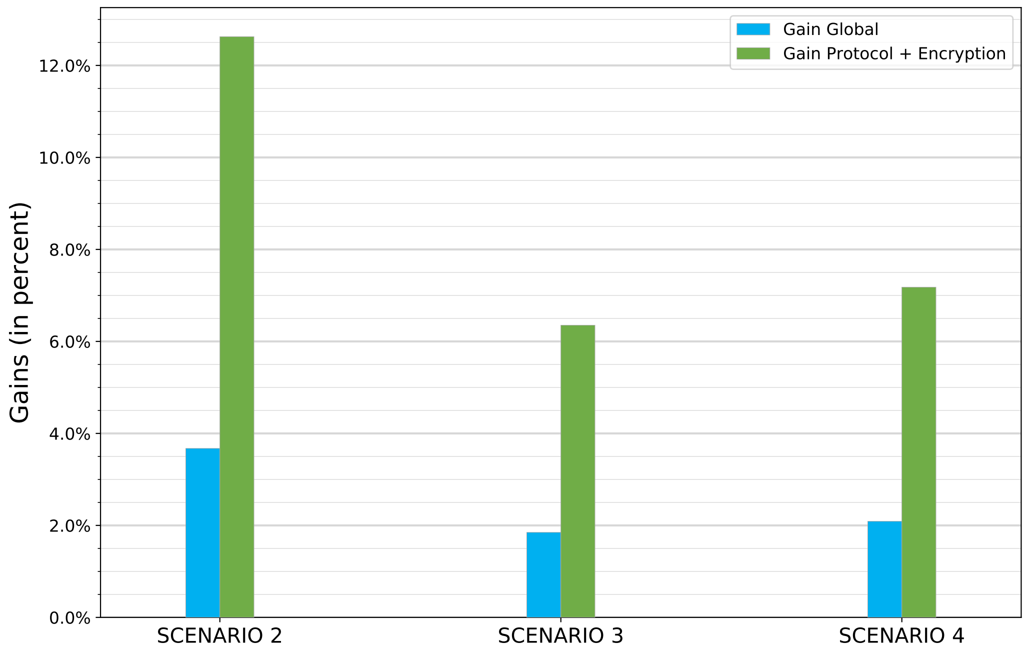


Figure 18: Power gain between Reduced-Round TLS Extensions Scenarios with AES-128 compared to full AES-128.

and encryption power gains (green chart) for each scenario. Scenarios 3 and 4 result in battery savings of 1.85% and 2.09% respectively, with protocol and encryption power gains of 6.35% and 7.18%. Up to 3.67% battery savings and 12.63% protocol and encryption power savings using AES-128 with 7 encryption rounds (Scenario 2). As an example, for a 10,000 mAh battery capacity, Scenarios 2, 3, and 4 gain 9,172, 4,534, and 5,135 seconds after one full charge, respectively.

5.5 Discussion

Through the first implementation of our reduced-round TLS extension PoC, we have been able to implement a dynamic, energy-efficient and secure solution based on reduced-round cryptography. The establishment of secure update mechanisms ensures the confidentiality and integrity of the data communicated and the resilience of our system. The power consumption analysis performed on different deployment scenarios of our extension with AES-128 on TLS1.3 highlighted significant battery savings resulting in a prolonged IoT device lifespan. However, these energy savings can be further enhanced by improving the implementation of our PoC. Among the possible improvements, the lightweight ciphers studied in Section 4.2 could be integrated within TLS cipher suites (in AEAD ciphers format) to further reduce the TLS power consumption. Other improvement options are presented in the Chapter 6.

Chapter 6

Discussions

In this section we will discuss the limits of our project in order to define the improvement strategies in future works. We have developed the dynamic reduced-round mechanism within TLS protocol to provide a secure IoT communication solution generalized to various battery-powered constrained IoT devices. Specifically, we carefully implemented our extension to be compatible with the latest versions of TLS, to integrate all update mechanisms in a secure way and to be easily applicable. However, this project remains a first PoC of the dynamic reduced-round TLS extension, which could be extended to any constrained device, either regarding the protocol used or the architecture. The different improvement axes of this project are the following:

- Integrate lightweight ciphers under TLS protocol: Lightweight primitives analyzed in reduced-round cryptography can be implemented in the WolfSSL library in order to analyze the power consumption of the TLS protocol using these ciphers in full and reduced-round. This improvement can be used to perform a power-consumption benchmark of our TLS extension depending on the lightweight cipher used.

- Diversify battery-powered constrained IoT architecture tests: For our first PoC, the Tiva C TM4C1294 connected launchpad was powered from the mains and the real-time battery level was simulated. In addition, the design of the IoT devices differs widely, they are based on CPUs with different capacity limits. We suggest to test our solution on several battery-powered platforms using different well known IoT-based MCUs such as: 8-bit AVR, 16-bit MSP, 32-bit ARM Cortex M, 32-bit PIC and ESP32 MCUs.

- Reduced-Round DTLS extension: Some IoT application protocols are based on a transport over UDP which differs from TCP protocol structure. Therefore, UDP communications are secured with a different security protocol: DTLS. Adapting our extension on DTLS will enable the compatibility of our reduced-round solution on the two major transport protocols UDP and TCP and ensure secure communications for all IoT application protocols. This implementation can be performed within the WolfSSL library which supports the latest version of DTLS (DTLS1.2).

- MQTT and CoAP over Reduced-Round TLS/DTLS: Since our solution is integrated within TLS protocol, it can be deployed on any application protocol running on TCP. Thus, MQTT protocol can be secured using our extensions over TLS. Moreover, with the future integration of our solution over DTLS, we will be able to deploy our solution on the CoAP application protocol as well. Applying MQTT and CoAP over Reduced-Round TLS and DTLS will further decrease the power consumption of the IoT devices. We suggest to deploy these two solutions in order to analyze and compare the resulting battery savings.

- Reduced-round mechanism for 6LoWPAN infrastructure: Integrate our solution on 6LoWPAN protocol to provide secure communication for constrained 6LoWPAN infrastructures. Combining the 6LoWPAN protocol with our solution over TLS/DTLS will enable secure communication for most constrained devices by minimizing resource and energy requirements.

- Power Consumption Reduced-Round Mechanism: We propose to vary the encryption round number according to the evolution of the real-time consumption of the IoT device. Adding this feature in our solution allows the reduced-round mechanism to be dynamic for all constrained IoT devices instead of focusing only on battery-powered devices. This feature will be added to the current implementation of our TLS reduced-round extension in the WolfSSL library. Operators will be able to define a dynamic mechanism for mains powered IoT devices.

- Perform an in-depth security analysis of our solution using analysis tools, assistance platforms and pentesting investigation in order to establish a complete threat model.

Finally, as a final goal, we plan to submit our extension to the IETF for an expert review and to standardize our extension within the TLS protocol. We wish our project will allow the development of a new (optional) solution for constrained devices communication over TLS which can be applied by academic and industrial research communities.

Chapter 7

Conclusion

While worldwide deployment of connected objects is increasing and new IoT applications are being developed, these devices face two major challenges: environmental impact and communication security. The annual global power and battery consumption of IoT devices is increasing dramatically. In addition, these small devices and especially constrained devices face potential vulnerabilities due to their limited capacity, low-cost design and long lifespan. The development of secure energy-efficient communications guaranteeing data confidentiality, integrity and availability is a live issue in the research community. Thus, the objective of this thesis was to propose a new solution integrating a lightweight and self-monitored mechanism that dynamically balances communication security and power consumption according to the IoT devices current battery level. First, we evaluated the security and power consumption performance of reduced-round cryptography on different lightweight ciphers. Second, we proposed a mechanism to control power consumption by dynamically adjusting the security level of encrypted communications based on reduced-round cryptography. Third, we designed, implemented and evaluated our dynamic reduced-round mechanism integrated in TLS communication protocol. Through the reduced-round cryptography we were able to adapt the level of security according to the application of the device and its constraints. Cryptanalysis overview of lightweight primitives helped us to evaluate the security of each cipher and to determine their minimum round number according to the level of security suggested. Our benchmarks on a constrained IoT device showed the power consumption decrease between full and reduced-round on several lightweight ciphers. According to the results we achieved up to 57.1% energy cost reduction for the encryption process and 9.4% battery savings.

With the dynamic reduced-round TLS extension we have designed a dynamic solution to secure the communication of battery-powered constrained devices through the TLS protocol reducing the impact on power and resource consumption. Secure schemes for round number, minimum round number and policy updates allow us to keep our dynamic reduced-round mechanism operational, configurable and secure. Our first PoC on a real-world IoT device implementation, through the analysis of the scenarios, highlighted the energy consumption reduction compared to the traditional use of the TLS protocol with AES-128 full. We observed up to 12.7% energy reduction in TLS protocol usage including the encryption process resulting in 3.7% global battery savings.

Finally, the research in this thesis can contribute to an innovative solution for the IoT security and green communications domains,relevant for the general public and to be deployed on a large number of IoT devices. The reduced-round TLS extensions still need several improvements and revisions such as: reduced-round DTLS support, MQTT and CoAP communications over TLS/DTLS reduced-round experiments and a complete security analysis through analysis tools and penetration testing (see Chapter 6). However, the new approach introduced in this thesis for energy-efficient IoT communications security mechanisms can be beneficial for future IoT applications.

Bibliography

[1] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[2] M. Fagan, K. N. Megas, K. Scarfone, and M. Smith, "Foundational cybersecurity activities for iot device manufacturers," National Institute of Standards and Technology, Tech. Rep., 2020.

[3] "Iot devices in the enterprise 2020: Shadow iot threat emerges," Zscaler, Inc., Tech. Rep., Feb. 2020. [Online]. Available: https://www.zscaler.com/resources/industry-reports/iot-in-the-enterprise.pdf

[4] "Energy efficiency of the internet of things: Technology and energy assessment report," International Energy Agency (IEA), Tech. Rep., Apr. 2016. [Online]. Available: https://www.iea-4e.org/document/384/energy-efficiency-of-the-internet-of-things-technology-and-energy-assessment-report

[5] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8446.txt

[6] D. Dinu, A. Biryukov, J. Großschädl, D. Khovratovich, Y. Le Corre, and L. Perrin, "Felics–fair evaluation of lightweight cryptographic systems," 2015.

[7] W. Lardier, Q. Varo, and J. Yan, "Dynamic reduced-round cryptography for energy-efficient wireless communication of smart iot devices," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[8] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the internet of things (iot)," *IEEE Internet Initiative*, vol. 1, no. 1, pp. 1–86, 2015.

[9] C. MacGillivray and D. Reinsel, "Worldwide global datasphere iot device and data forecast, 2019–2023," International Data Corporation (IDC), Tech. Rep., May 2019. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=US45066919

[10] A. Deol, K. Figueredo, S.-W. Lin, B. Murphy, D. Seed, and J. Yin, "Advancing the industrial internet of things," Industrial Internet Consortium and oneM2M, Tech. Rep., Dec. 2019. [Online]. Available: https://www.iiconsortium.org/pdf/IIC_oneM2M_Whitepaper_final_2019_12_12.pdf

[11] G. Erboz, "How to define industry 4. 0: The main pillars of industry 4. 0," in *7th International Conference on Management (ICoM 2017), At Nitra, Slovakia*, 2017, pp. 1–2.

[12] N. Bansal, "Microsoft azure iot platform," in *Designing Internet of Things Solutions with Microsoft Azure*. Springer, 2020, pp. 33–48.

[13] "Battery market for iot by type, rechargeability, end-use application, and geography - global forecast to 2025," Markets and Markets, Tech. Rep., May 2020.

[14] C. Bormann, M. Ersue, and A. Keränen, "Terminology for Constrained-Node Networks," RFC 7228, May 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7228.txt

[15] D. T. Elgamal and K. E. Hickman, "The SSL Protocol," Internet Engineering Task Force, Internet-Draft draft-hickman-netscape-ssl-00, Apr. 1995, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-hickman-netscape-ssl-00

[16] T. Polk and S. Turner, "Prohibiting Secure Sockets Layer (SSL) Version 2.0," RFC 6176, Mar. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6176.txt

[17] A. O. Freier, P. Karlton, and P. C. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0," RFC 6101, Aug. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6101.txt

[18] R. Barnes, M. Thomson, A. Pironti, and A. Langley, "Deprecating Secure Sockets Layer Version 3.0," RFC 7568, Jun. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7568.txt

[19] C. Allen and T. Dierks, "The TLS Protocol Version 1.0," RFC 2246, Jan. 1999. [Online]. Available: https://rfc-editor.org/rfc/rfc2246.txt

[20] B. Möller, T. Duong, and K. Kotowicz, "This poodle bites: exploiting the ssl 3.0 fallback," *Security Advisory*, 2014.

[21] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," RFC 4346, Apr. 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4346.txt

[22] K. Moriarty and S. Farrell, "Deprecating TLSv1.0 and TLSv1.1," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-oldversions-deprecate-09, Nov. 2020, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-tls-oldversions-deprecate-09

[23] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5246.txt

[24] C. OpenSSL, "Ssl/tls toolkit," *The document is available in http://www. openssl. org*, 2011.

[25] "Openssl commands 52.43% market share in network security," 2020. [Online]. Available: https://enlyft.com/tech/products/openssl

[26] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," RFC 6347, Jan. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6347.txt

[27] L. bTrade, "Fips 140-2 non-proprietary security policy," 2011.

[28] T. K. Ferrell, U. D. Ferrell, and C. Spitzer, "Rtca do-178c/eurocae ed-12c," *Digital Avionics Handbook*, pp. 16–1, 2017.

[29] R. Bagnara, A. Bagnara, and P. M. Hill, “The misra c coding standard and its role in the development and analysis of safety-and security-critical embedded software,” in *International Static Analysis Symposium.* Springer, 2018, pp. 5–23.

[30] S. Khillari, “Iot security market— growth, trends and forecast report, 2026,” 2020.

[31] T. allen, “Cybersecurity considerations in iot,” Oct 2019. [Online]. Available: https://www.nist.gov/itl/applied-cybersecurity/nist-cybersecurity-iot-program/about/cybersecurity-considerations-iot

[32] M. Bhuptani and S. Moradpour, *RFID field guide: deploying radio frequency identification systems.* Prentice Hall PTR, 2005.

[33] J. Decuir *et al.*, “Bluetooth 4.0: low energy,” *Cambridge, UK: Cambridge Silicon Radio SR plc*, vol. 16, 2010.

[34] Z. Alliance, “The zigbee alliance,” 2020. [Online]. Available: https://zigbeealliance.org/

[35] Z.-w. Alliance, “Z-wave: Safer, smarter homes start with z-wave,” 2020. [Online]. Available: https://www.z-wave.com/

[36] S. Li, L. Da Xu, and S. Zhao, “5g internet of things: A survey,” *Journal of Industrial Information Integration*, vol. 10, pp. 1–9, 2018.

[37] S. Sigfox, “Sigfox technical overview,” 2018.

[38] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, “Lorawan specification,” *LoRa alliance*, 2015.

[39] S. Grant, “3gpp low power wide area technologies,” Global System for Mobile Communications Association (GSMA), Tech. Rep., 2016.

[40] “Ieee standard for low-rate wireless networks,” *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, pp. 1–800, 2020.

[41] S. Chakrabarti, G. Montenegro, R. Droms, and james woodyatt, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) ESC Dispatch Code Points and Guidelines," RFC 8066, Feb. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8066.txt

[42] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7252.txt

[43] "Message queuing telemetry transport (mqtt)," apr 2019. [Online]. Available: https://mqtt.org/

[44] J. Lee, S.-Y. Cho, and Y. Chung, "Coap-http proxy server based on zigbee network," *International Information Institute (Tokyo). Information*, vol. 19, no. 11A, p. 5291, 2016.

[45] Y.-H. Jang, S.-C. Park, and S.-H. Yoon, "Design and implementation of mqtt-based standby power reduction system in z-wave network environment," *Journal of Korea Multimedia Society*, vol. 23, no. 3, pp. 421–429, 2020.

[46] C.-M. Kim, H.-W. Kang, S.-I. Choi, and S.-J. Koh, "Implementation of coap/6lowpan over ble networks for iot services," *Journal of broadcast engineering*, vol. 21, no. 3, pp. 298–306, 2016.

[47] T. Zillner and S. Strobl, "Zigbee exploited: The good, the bad and the ugly," *Black Hat*, 2015. [Online]. Available: https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly.pdf

[48] D. Miessler, "Securing the internet of things: Mapping attack surface areas using the owasp iot top 10," in *RSA Conference*, 2015.

[49] R. Bodenheim, J. Butts, S. Dunlap, and B. Mullins, "Evaluation of the ability of the shodan search engine to identify internet-facing industrial control devices," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 2, pp. 114–123, 2014.

[50] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, “Understanding the mirai botnet,” in *26th {USENIX} security symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.

[51] E. Doerr, “The enemy within: Modern supply chain attacks,” in *BlackHat Europe*, August 2019. [Online]. Available: https://i.blackhat.com/USA-19/Thursday/us-19-Doerr-The-Enemy-Within-Modern-Supply-Chain-Attacks.pdf

[52] J. Kirk, “Pacemaker hack can deliver deadly 830-volt jolt,” *Computerworld*, vol. 17, 2012.

[53] M. S. Turan, K. A. McKay, Ç. Çalık, D. Chang, and L. Bassham, “Status report on the first round of the nist lightweight cryptography standardization process,” *National Institute of Standards and Technology, Gaithersburg, MD, NIST Interagency/Internal Rep.(NISTIR)*, 2019.

[54] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, “The simon and speck lightweight block ciphers,” in *Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp. 1–6.

[55] E. Barker, “Guideline for using cryptographic standards in the federal government: Cryptographic mechanisms,” National Institute of Standards and Technology, Tech. Rep., 2016.

[56] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, “The led block cipher,” in *International workshop on cryptographic hardware and embedded systems.* Springer, 2011, pp. 326–341.

[57] N. Mouha, B. Mennink, A. Van Herrewege, D. Watanabe, B. Preneel, and I. Verbauwhede, “Chaskey: an efficient mac algorithm for 32-bit microcontrollers,” in *International Conference on Selected Areas in Cryptography.* Springer, 2014, pp. 306–323.

[58] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, “Twine: A lightweight, versatile block cipher,” in *ECRYPT Workshop on Lightweight Cryptography*, 2011.

[59] 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.

[60] F. Siddiqui, J. Beley, S. Zeadally, and G. Braught, “Secure and lightweight communication in heterogeneous iot environments,” *Internet of Things*, p. 100093, 2019.

[61] G. Selander, J. P. Mattsson, F. Palombini, and L. Seitz, “Object Security of CoAP (OSCOAP),” Internet Engineering Task Force, Internet-Draft draft-ietf-core-object-security-00, 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-core-object-security-00

[62] R. H. Randhawa, A. Hameed, and A. N. Mian, “Energy efficient cross-layer approach for object security of coap for iot devices,” *Ad Hoc Networks*, vol. 92, p. 101761, 2019.

[63] T. Fischer, H. Linka, M. Rademacher, K. Jonas, and D. Loebenberger, “Analyzing power consumption of tls ciphers on an esp32,” *crypto day matters 30*, 2019.

[64] P. Li, J. Su, and X. Wang, “itls/idtls: Lightweight end-to-end security protocol for iot through minimal latency,” in *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, 2019, pp. 166–168.

[65] K. Tange, D. Howard, T. Shanahan, S. Pepe, X. Fafoutis, and N. Dragoni, “rtls: Lightweight tls session resumption for constrained iot devices,” in *2020 International Conference on Information and Communications Security*, 2020.

[66] B. Rashidi, “Flexible structures of lightweight block ciphers present, simon and led,” *IET Circuits, Devices Systems*, vol. 14, no. 3, pp. 369–380, 2020.

[67] M. Suárez-Albela, T. M. Fernández-Caramés, P. Fraga-Lamas, and L. Castedo, “A practical performance comparison of ecc and rsa for resource-constrained iot devices,” in *2018 Global Internet of Things Summit (GIoTS)*, 2018, pp. 1–6.

[68] X. Yao, Z. Chen, and Y. Tian, “A lightweight attribute-based encryption scheme for the internet of things,” *Future Generation Computer Systems*,

vol. 49, pp. 104 – 112, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X14002039

[69] R. Arshad, S. Zahoor, M. A. Shah, A. Wahid, and H. Yu, "Green iot: An investigation on energy saving practices for 2020 and beyond," *IEEE Access*, vol. 5, pp. 15 667–15 681, 2017.

[70] M. Moreno, B. Úbeda, A. F. Skarmeta, and M. A. Zamora, "How can we tackle energy efficiency in iot basedsmart buildings?" *Sensors*, vol. 14, no. 6, pp. 9582–9614, 2014.

[71] M. A. Zamora-Izquierdo, J. Santa, and A. F. Gómez-Skarmeta, "An integral and networked home automation solution for indoor ambient intelligence," *IEEE Pervasive Computing*, vol. 9, no. 4, pp. 66–77, 2010.

[72] C. Karakus, A. C. Gurbuz, and B. Tavli, "Analysis of energy efficiency of compressive sensing in wireless sensor networks," *IEEE Sensors Journal*, vol. 13, no. 5, pp. 1999–2008, 2013.

[73] C. Perera, D. S. Talagala, C. H. Liu, and J. C. Estrella, "Energy-efficient location and activity-aware on-demand mobile distributed sensing platform for sensing as a service in iot clouds," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 171–181, 2015.

[74] M. Fagan, K. N. Megas, K. Scarfone, and M. Smith, "Nistir 8259a: Iot device cybersecurity capability core baseline," National Institute of Standards and Technology (NIST), Tech. Rep., 2020.

[75] "303 645 v2.1.1, cyber; cyber security for consumer internet of things: Baseline requirements," European Telecommunications Standards Institute (ETSI), Tech. Rep., 2020.

[76] N.-F. Standard, "Announcing the advanced encryption standard (aes)," *Federal Information Processing Standards Publication*, vol. 197, no. 1-51, pp. 3–3, 2001.

[77] L. Hathaway, "National policy on the use of the advanced encryption standard (aes) to protect national security systems and national security information," *National Security Agency (NSA)*, vol. 23, 2003.

[78] P. Derbez, P.-A. Fouque, and J. Jean, "Improved key recovery attacks on reduced-round aes in the single-key setting," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer, 2013, pp. 371–387.

[79] I. Nikolić, L. Wang, and S. Wu, "Cryptanalysis of round-reduced `LED`," in *International Workshop on Fast Software Encryption.* Springer, 2013, pp. 112–129.

[80] V. Grosso, G. Leurent, F.-X. Standaert, and K. Varıcı, "Ls-designs: Bitslice encryption for efficient masked software implementations," in *International Workshop on Fast Software Encryption.* Springer, 2014, pp. 18–37.

[81] A. D. Dwivedi, S. Dhar, G. Srivastava, and R. Singh, "Cryptanalysis of round-reduced fantomas, robin and iscream," *Cryptography*, vol. 3, no. 1, p. 4, 2019.

[82] G. Leander, B. Minaud, and S. Rønjom, "A generic approach to invariant subspace attacks: Cryptanalysis of robin, iscream and zorro," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer, 2015, pp. 254–283.

[83] M. R. Albrecht, B. Driessen, E. B. Kavun, G. Leander, C. Paar, and T. Yalçın, "Block ciphers–focus on the linear layer (feat. pride)," in *Annual Cryptology Conference.* Springer, 2014, pp. 57–76.

[84] Q. Yang, L. Hu, S. Sun, K. Qiao, L. Song, J. Shan, and X. Ma, "Improved differential analysis of block cipher pride," in *International Conference on Information Security Practice and Experience.* Springer, 2015, pp. 209–219.

[85] D. Dinu, L. Perrin, A. Udovenko, V. Velichkov, J. Großschädl, and A. Biryukov, "Design strategies for arx with provable bounds: Sparx and lax," in *International Conference on the Theory and Application of Cryptology and Information Security.* Springer, 2016, pp. 484–513.

[86] R. Ankele and E. List, "Differential cryptanalysis of round-reduced sparx-64/128," in *International Conference on Applied Cryptography and Network Security.* Springer, 2018, pp. 459–475.

[87] M. Tolba, A. Abdelkhalek, and A. M. Youssef, "Multidimensional zero-correlation linear cryptanalysis of reduced round sparx-128," in *International Conference on Selected Areas in Cryptography.* Springer, 2017, pp. 423–441.

[88] F.-X. Standaert, G. Piret, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat, "Iceberg: An involutional cipher efficient for block encryption in reconfigurable hardware," in *International Workshop on Fast Software Encryption.* Springer, 2004, pp. 279–298.

[89] Y. Sun, M. Wang, S. Jiang, and Q. Sun, "Differential cryptanalysis of reduced-round iceberg," in *International Conference on Cryptology in Africa.* Springer, 2012, pp. 155–171.

[90] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *International workshop on cryptographic hardware and embedded systems.* Springer, 2007, pp. 450–466.

[91] J. Y. Cho, "Linear cryptanalysis of reduced-round present," in *Cryptographers' Track at the RSA Conference.* Springer, 2010, pp. 302–317.

[92] J. Nakahara, P. Sepehrdad, B. Zhang, and M. Wang, "Linear (hull) and algebraic cryptanalysis of the block cipher present," in *International Conference on Cryptology and Network Security.* Springer, 2009, pp. 58–75.

[93] J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger *et al.*, "Prince–a low-latency block cipher for pervasive computing applications," in *International conference on the theory and application of cryptology and information security.* Springer, 2012, pp. 208–225.

[94] A. Canteaut, T. Fuhr, H. Gilbert, M. Naya-Plasencia, and J.-R. Reinhard, "Multiple differential cryptanalysis of round-reduced prince," in *International Workshop on Fast Software Encryption.* Springer, 2014, pp. 591–610.

[95] F. Abed, C. Forler, E. List, S. Lucks, and J. Wenzel, "Biclique cryptanalysis of present, led, and klein," *Cryptology ePrint Archive, Report*, vol. 5912012, 2012.

[96] B. Collard and F.-X. Standaert, “A statistical saturation attack against the block cipher present,” in *Cryptographers’ Track at the RSA Conference.* Springer, 2009, pp. 195–210.

[97] K. Jeong, H. Kang, C. Lee, J. Sung, and S. Hong, “Biclique cryptanalysis of lightweight block ciphers present, piccolo and led.” *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 621, 2012.

[98] H. Mala, M. Dakhilalian, V. Rijmen, and M. Modarres-Hashemi, “Improved impossible differential cryptanalysis of 7-round aes-128,” in *International Conference on Cryptology in India.* Springer, 2010, pp. 282–291.

[99] A. Bogdanov, D. Khovratovich, and C. Rechberger, “Biclique cryptanalysis of the full aes,” in *International conference on the theory and application of cryptology and information security.* Springer, 2011, pp. 344–371.

[100] Y. Wei, “Bit-pattern based integral attack on iceberg,” in *2015 International Conference on Intelligent Networking and Collaborative Systems.* IEEE, 2015, pp. 370–373.

[101] P. Morawiecki, “Practical attacks on the round-reduced prince,” *IET Information Security*, vol. 11, no. 3, pp. 146–151, 2016.

[102] P. Derbez and L. Perrin, “Meet-in-the-middle attacks and structural analysis of round-reduced prince,” *Journal of Cryptology*, pp. 1–32, 2020.

[103] S. Rasoolzadeh and H. Raddum, “Cryptanalysis of prince with minimal data,” in *International Conference on Cryptology in Africa.* Springer, 2016, pp. 109–126.

[104] V. Lallemand and S. Rasoolzadeh, “Differential cryptanalysis of 18-round pride,” in *International Conference on Cryptology in India.* Springer, 2017, pp. 126–146.

[105] W. Wu and L. Zhang, “Lblock: a lightweight block cipher,” in *International Conference on Applied Cryptography and Network Security.* Springer, 2011, pp. 327–344.

[106] M. Xie and Q. Zeng, "Related-key impossible boomerang cryptanalysis on lblock-s," *Journal on Communications*, vol. 38, no. 5, pp. 66–71, 2017.

[107] A. Baysal and S. Şahin, "Roadrunner: A small and fast bitslice block cipher for low cost 8-bit processors," in *Lightweight Cryptography for Security and Privacy*. Springer, 2015, pp. 58–76.

[108] Q. Yang, L. Hu, S. Sun, and L. Song, "Extension of meet-in-the-middle technique for truncated differential and its application to roadrunner," in *International Conference on Network and System Security*. Springer, 2016, pp. 398–411.

[109] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: an ultra-lightweight blockcipher," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011, pp. 342–357.

[110] M. Z. Ahangarkolaei, S. R. H. Najarkolaei, S. Ahmadi, and M. R. Aref, "Zero correlation linear attack on reduced round piccolo-80," in *2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*. IEEE, 2016, pp. 66–71.

[111] M. Minier, "On the security of piccolo lightweight block cipher against related-key impossible differentials," in *International Conference on Cryptology in India*. Springer, 2013, pp. 308–318.

[112] T. P. Berger, J. Francq, M. Minier, and G. Thomas, "Extended generalized feistel networks using matrix representation to propose a new lightweight block cipher: Lilliput," *IEEE Transactions on Computers*, vol. 65, no. 7, pp. 2074–2089, 2015.

[113] C. Boura, M. Naya-Plasencia, and V. Suder, "Scrutinizing and improving impossible differential attacks: applications to clefia, camellia, lblock and simon," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2014, pp. 179–199.

[114] Y. Sasaki and Y. Todo, "Tight bounds of differentially and linearly active s-boxes and division property of lilliput," *IEEE Transactions on Computers*, vol. 67, no. 5, pp. 717–732, 2017.

[115] T. Ashur, O. Dunkelman, and N. Masalha, "Linear cryptanalysis reduced round of piccolo-80," in *International Symposium on Cyber Security Cryptography and Machine Learning.* Springer, 2019, pp. 16–32.

[116] Y. Todo, "Impossible differential attack against 14-round *Piccolo*-80 without relying on full code book," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 99-A, no. 1, pp. 154–157, 2016. [Online]. Available: https://doi.org/10.1587/transfun.E99.A.154

[117] Y. Wang, W. Wu, and X. Yu, "Biclique cryptanalysis of reduced-round piccolo block cipher," in *International Conference on Information Security Practice and Experience.* Springer, 2012, pp. 337–352.

[118] Y. Wang and W. Wu, "Improved multidimensional zero-correlation linear cryptanalysis and applications to lblock and twine," in *Australasian Conference on Information Security and Privacy.* Springer, 2014, pp. 1–16.

[119] M. Çoban, F. Karakoç, and Ö. Boztaş, "Biclique cryptanalysis of twine," in *International Conference on Cryptology and Network Security.* Springer, 2012, pp. 43–55.

[120] T. Isobe and K. Shibutani, "Security analysis of the lightweight block ciphers xtea, led and piccolo," in *Australasian Conference on Information Security and Privacy.* Springer, 2012, pp. 71–86.

[121] S. Ahmadi, Z. Ahmadian, J. Mohajeri, and M. R. Aref, "Low-data complexity biclique cryptanalysis of block ciphers with application to piccolo and hight," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 10, pp. 1641–1652, 2014.

[122] A. Bogdanov, C. Boura, V. Rijmen, M. Wang, L. Wen, and J. Zhao, "Key difference invariant bias in block ciphers," in *International Conference on the Theory and Application of Cryptology and Information Security.* Springer, 2013, pp. 357–376.

[123] S. R. H. Najarkolaei, M. Z. Ahangarkolaei, S. Ahmadi, and M. R. Aref, "Biclique cryptanalysis of twine-128," in *2016 13th International Iranian Society*

of Cryptology Conference on Information Security and Cryptology (ISCISC). IEEE, 2016, pp. 46–51.

[124] R. M. Needham and D. J. Wheeler, "Tea extensions," *Report, Cambridge University*, 1997.

[125] D. Moon, K. Hwang, W. Lee, S. Lee, and J. Lim, "Impossible differential cryptanalysis of reduced round xtea and tea," in *International Workshop on Fast Software Encryption.* Springer, 2002, pp. 49–60.

[126] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong *et al.*, "Hight: A new block cipher suitable for low-resource device," in *International Workshop on Cryptographic Hardware and Embedded Systems.* Springer, 2006, pp. 46–59.

[127] Y. Sasaki and L. Wang, "Meet-in-the-middle technique for integral attacks against feistel ciphers," in *International Conference on Selected Areas in Cryptography.* Springer, 2012, pp. 234–251.

[128] D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu, and D.-G. Lee, "Lea: A 128-bit block cipher for fast encryption on common processors," in *International Workshop on Information Security Applications.* Springer, 2013, pp. 3–27.

[129] "Ks x 3246: Lea-128 national standard of republic of korea," National Radio Research Agency, Tech. Rep., 2016. [Online]. Available: https://rra.go.kr/ko/reference/kcsList_view.do?nb_seq=1923&cpage=4&nb_type=6&searchCon=&searchTxt=&sortOrder=

[130] "Delivering various information about encryption technology and policies based on information protection," Korean Cryptographic Module Validation Program (KCMVP), Tech. Rep., 2016. [Online]. Available: https://seed.kisa.or.kr/kisa/algorithm/EgovLeaInfo.do

[131] "Iso/iec 29192-2: Information security — lightweight cryptography — part 2: Block ciphers," International Organization for Standardization (ISO), Tech. Rep., 2019. [Online]. Available: https://www.iso.org/standard/78477.html

[132] L. Song, Z. Huang, and Q. Yang, "Automatic differential analysis of arx block ciphers with application to speck and lea," in *Australasian Conference on Information Security and Privacy.* Springer, 2016, pp. 379–394.

[133] A. D. Dwivedi and G. Srivastava, "Differential cryptanalysis of round-reduced lea," *IEEE Access*, vol. 6, pp. 79 105–79 113, 2018.

[134] G. Leurent, "Improved differential-linear cryptanalysis of 7-round chaskey with partitioning," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer, 2016, pp. 344–371.

[135] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The simon and speck families of lightweight block ciphers." *IACR Cryptol. ePrint Arch.*, vol. 2013, p. 404, 2013.

[136] M. A. Abdelraheem, J. Alizadeh, H. A. Alkhzaimi, M. R. Aref, N. Bagheri, and P. Gauravaram, "Improved linear cryptanalysis of reduced-round simon-32 and simon-48," in *International Conference on Cryptology in India.* Springer, 2015, pp. 153–179.

[137] I. Dinur, "Improved differential cryptanalysis of round-reduced speck," in *International Conference on Selected Areas in Cryptography.* Springer, 2014, pp. 147–164.

[138] J. Alizadeh, N. Bagheri, P. Gauravaram, A. Kumar, and S. K. Sanadhya, "Linear cryptanalysis of round reduced simon." *IACR Cryptol. ePrint Arch.*, vol. 2013, p. 663, 2013.

[139] F. Abed, E. List, S. Lucks, and J. Wenzel, "Differential cryptanalysis of round-reduced simon and speck," in *International Workshop on Fast Software Encryption.* Springer, 2014, pp. 525–545.

[140] N. Wang, X. Wang, K. Jia, and J. Zhao, "Improved differential attacks on reduced simon versions." *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 448, 2014.

[141] T. Ashur, "Improved linear trails for the block cipher simon." *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 285, 2015.

[142] A. Biryukov, A. Roy, and V. Velichkov, "Differential analysis of block ciphers simon and speck," in *International Workshop on Fast Software Encryption*. Springer, 2014, pp. 546–570.

[143] L. Sun, K. Fu, and M. Wang, "Improved zero-correlation cryptanalysis on simon," in *International Conference on Information Security and Cryptology*. Springer, 2015, pp. 125–143.

[144] M. A. Abdelraheem, J. Alizadeh, H. A. Alkhzaimi, M. R. Aref, N. Bagheri, P. Gauravaram, and M. M. Lauridsen, "Improved linear cryptanalysis of reduced-round simon," *Cryptology ePrint Archive, Report 2014/681*, 2014.

[145] J. Yin, C. Ma, L. Lyu, J. Song, G. Zeng, C. Ma, and F. Wei, "Improved cryptanalysis of an iso standard lightweight block cipher with refined milp modelling," in *International Conference on Information Security and Cryptology*. Springer, 2017, pp. 404–426.

[146] J. Chen, M. Wang, and B. Preneel, "Impossible differential cryptanalysis of the lightweight block ciphers tea, xtea and hight," in *International Conference on Cryptology in Africa*. Springer, 2012, pp. 117–137.

[147] L. Wen, M. Wang, A. Bogdanov, and H. Chen, "Multidimensional zero-correlation attacks on lightweight block cipher hight: improved cryptanalysis of an iso standard," *Information Processing Letters*, vol. 114, no. 6, pp. 322–330, 2014.

[148] J. Lu, "Cryptanalysis of reduced versions of the hight block cipher from ches 2006," in *International Conference on Information Security and Cryptology*. Springer, 2007, pp. 11–26.

[149] J. Song, K. Lee, and H. Lee, "Biclique cryptanalysis on lightweight block cipher: Hight and piccolo," *International Journal of Computer Mathematics*, vol. 90, no. 12, pp. 2564–2580, 2013.

[150] J. Alizadeh, H. AlKhzaimi, M. R. Aref, N. Bagheri, P. Gauravaram, and M. M. Lauridsen, "Improved linear cryptanalysis of round reduced simon." *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 681, 2014.

[151] Y. Ko, S. Hong, W. Lee, S. Lee, and J.-S. Kang, "Related key differential attacks on 27 rounds of xtea and full-round gost," in *International Workshop on Fast Software Encryption.* Springer, 2004, pp. 299–316.

[152] C. Bouillaguet, O. Dunkelman, G. Leurent, and P.-A. Fouque, "Another look at complementation properties," in *International Workshop on Fast Software Encryption.* Springer, 2010, pp. 347–364.

[153] Y. Liu, K. Fu, W. Wang, L. Sun, and M. Wang, "Linear cryptanalysis of reduced-round speck," *Information Processing Letters*, vol. 116, no. 3, pp. 259–266, 2016.

[154] K. Zhang, J. Guan, and B. Hu, "Zero correlation linear cryptanalysis on lea family ciphers," *Journal of Communications*, vol. 11, no. 7, pp. 677–685, 2016.

[155] X.-L. Yu, W.-L. Wu, Z.-Q. Shi, J. Zhang, L. Zhang, and Y.-F. Wang, "Zero-correlation linear cryptanalysis of reduced-round simon," *Journal of Computer Science and Technology*, vol. 30, no. 6, pp. 1358–1369, 2015.

[156] D. J. Bernstein, "Salsa20 specification," *eSTREAM Project algorithm description, http://www. ecrypt. eu. org/stream/salsa20pf. html*, 2005.

[157] ——, "Chacha, a variant of salsa20," in *Workshop Record of SASC*, vol. 8, 2008, pp. 3–5.

[158] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols," RFC 7539, May 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7539.txt

[159] K. K. Deepthi and K. Singh, "Cryptanalysis for reduced round salsa and chacha: revisited," *IET Information Security*, vol. 13, no. 6, pp. 591–602, 2019.

[160] Z.-y. Shao and L. Ding, "Related-cipher attack on salsa20," in *2012 Fourth International Conference on Computational and Information Sciences.* IEEE, 2012, pp. 1182–1185.

[161] A. R. Choudhuri and S. Maitra, "Significantly improved multi-bit differentials for reduced round salsa and chacha," *IACR Transactions on Symmetric Cryptology*, pp. 261–287, 2016.

[162] C. De Canniere, O. Dunkelman, and M. Knežević, "Katan and ktantan—a family of small and efficient hardware-oriented block ciphers," in *International Workshop on Cryptographic Hardware and Embedded Systems.* Springer, 2009, pp. 272–288.

[163] T. Fuhr and B. Minaud, "Match box meet-in-the-middle attack against katan," in *International Workshop on Fast Software Encryption.* Springer, 2014, pp. 61–81.

[164] G. V. Bard, N. T. Courtois, J. Nakahara, P. Sepehrdad, and B. Zhang, "Algebraic, aida/cube and side channel analysis of katan family of block ciphers," in *International Conference on Cryptology in India.* Springer, 2010, pp. 176–196.

[165] T. Isobe and K. Shibutani, "All subkeys recovery attack on block ciphers: Extending meet-in-the-middle approach," in *International Conference on Selected Areas in Cryptography.* Springer, 2012, pp. 202–221.

[166] S. Rasoolzadeh and H. Raddum, "Improved multi-dimensional meet-in-the-middle cryptanalysis of katan," *Tatra Mountains Mathematical Publications*, vol. 67, no. 1, pp. 149–166, 2016.

[167] S. Knellwolf, W. Meier, and M. Naya-Plasencia, "Conditional differential cryptanalysis of nlfsr-based cryptosystems," in *International Conference on the Theory and Application of Cryptology and Information Security.* Springer, 2010, pp. 130–145.

[168] D. E. E. 3rd, "Transport Layer Security (TLS) Extensions: Extension Definitions," RFC 6066, Jan. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6066.txt

[169] J. A. Salowey and S. Turner, "IANA Registry Updates for TLS and DTLS," RFC 8447, Aug. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8447.txt

[170] R. Seggelmann, M. Tuexen, and M. Williams, "Transport layer security (tls) and datagram transport layer security (dtls) heartbeat extension," *Internet Engineering Task Force, RFC*, vol. 6520, 2012.