Abhath Journal of Basic and Applied Sciences., 1(2) (2022) 18-26

Abhath Journal of Basic and Applied Sciences

Website:https://ojs.abhath-ye.com/index.php/OJSRJBAS/about

doi:

Research paper, Short communication, Review, Technical paper

Securing the Internet of Things: A review of Lightweight and low-power Cryptography Techniques

Abdulrazzaq H.A. Al-ahdal[1*], Mouad M.H. Ali[1], Abdullah Alahdal[2], Arafat S.M. Qaed[1], Galal A. Al-Rummana[1]

[1]*Assistant Professor,Computer Science, Faculty of Computer Science & Engineering, Hodeidah University, Yemen*

[2]*CSE Department,University College of Engineering, Osmania University,India*

***Corresponding author E-mail:** alahdal201211@gmail.com*

Received: 07 November 2022, Accepted: 15 December 2022, Published: 30 December 2022

Abstract

The current information related to security and IoT by conducting a theoretical and methodological study. It provides a detailed explanation on what is the meaning of IoT and on the various applications in IoT. It also presents the major issue of the security of IoT for applications. Moreover, the chapter discusses the metrics that define lightweight algorithms in hardware and software, and it presents the reasons for choosing blocks in lightweight algorithms and studying the architecture of those blocks. It also refers to the blocks/algorithms, both the traditional ones and the lightweight ones.

***Keywords**: Symmetric Cryptography, Lightweight Cryptography, Block Cipher, IoT.*

1. Introduction

In the recent decade, the world has been more networked with many variousdevices, such as sensors, smart networking, RFID and IoT, to achieve numerous activities[1]. Currently, IoT is a current technology of smart objects. Laptop, Telephone, Car,Refrigerator are tangible devices known as smart things. IoT is a network of devices ofsmart items which are identified, accessible and controlled by a network of devices. Theyare also able to compute and make decisions. IoT is a worldwide network with dynamiccapabilities that employ standard communication systems. It can operate on actual andvirtual items. It has intelligent platforms for the usage which are readily incorporated into communication technologies [2].In IoT systems, the objects when connected to each other, have restrictions contrasted to Personal Computers. These limits might be displayed in power consumption,execution time and memory cost. These constraints are ineffective on personal Computers[3].On the other hand, security is the process of safeguarding data when they are communicated and shared among objects. Therefore, the data security service is represented in authorization, confidentiality, authentication, and integrity. The security saves data during dissemination in IoT so that it cannot be corrupted or manipulated. Therefore, the security service can be given by cryptography [4]. Cryptography is an essential technique (algorithms) in security that may be characterized as turning original text(plain-text) into mystery-text (cipher-text) and reversing from mysterious-text (cipher-text) into original-text(plain-text). These functions are termed encoding and decoding [5]. Different algorithms such as AES, and DES are traditional cryptographic algorithms.AES is one of the most important algorithms in cryptography and can be implemented on different platforms [6]. It has a block size of 128 bits and a key length of 128,192, 256 bits. DES contains a 56-bit key for 64-bit block encryption. The key in DES is smaller than AES [7], so the security of DES is lower than AES[8]. Traditional encryption algorithms provide high security, but they consume high processing and large consumption of memory as well as power. On this basis, traditional encryption is not suitable for small size devices with limited resources (those devices that work in the Internet of Things). The increase in processors, power consumption and memory cost in traditional encryption lead to the emergence of a new encryption called lightweight cryptography (LWC). This encryption reduces the cost of memory usage and also reduces power to be suitable for devices with limited resources (those devices that work in the Internet of Things).

LWC is the new trend in encryption for resource-limited devices. This is due to the use of simple mathematical operations, lower memory cost, and lower power consumption. In another way, it can make a trade-off between security, performance, and cost, see Figure 1.1. The purpose of LWC is to minimize the total costs of traditional encryption implementation by focusing on numerous factors such as code size, memory cost, execution time and energy usage.

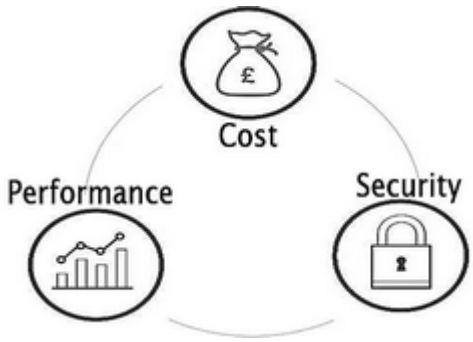

Figure 1.1: Cost, Performance and Security.

1.1 IoT Overview

The Internet of Things (IoT) is a network of interconnected items, each with its own unique identifier, that may gather and share data via the Internet with or without human participation [9]-[13]. All these applications are interconnected to exchange data through the Internet (network of Internet of things). The amount of data will be very large between these devices.

Internet of Things (Smart Home, IoT Wearable's, Smart Grid, Connected Cars, smart healthcare, Smart Cities, Health Care, Smart Farming, and industry 4.0) applications have become very important in our daily life. For this reason, they have become the focus of researchers. See Figure 1.2 and Table 1.1

Figure 1.2: IoT Applications.

In the year 2021, the number of connected devices will be about 20 billion devices in the Internet of Things. As we mentioned earlier, the data that will be generated between them is very large to communicate with each other. In this scenario, we note the great development of Internet of Things technology, which enables it to manage and control these devices. We will also note the flexibility and ease in the way of managing the data collection provided by the Internet of Things from applications.

Table1.1: Applications of Internet of Things

1.	Smart Home	It refers to the capacity to use internet-connected equipment to control household equipment. Applications like (setting alarms, home security controls, complex heating, and lighting)
2.	wearable's	Watches and lockets are examples of IoT wearables that collect information about the user such as health and fitness.
3.	IoT Smart Grid	The Smart Grid is part of an Internet of Things architecture that may be used to remotely monitor and manage anything from lights to traffic signals, traffic congestion, parking spots, road alerts, and early detection of things like power outages caused by earthquakes and harsh weather.
4.	Connected Cars	This includes sharing information about road conditions and incidents with other cars so that preventative steps may be taken.
5.	Smart Cities	It is also considered one of the applications of great importance. Because it solves issues related to traffic accidents, roads and parking lots in cities.
6.	Health Care	This includes gathering information about a person's health.
7.	Smart Farming	Its applications use sensors to monitor the agricultural field and automate the irrigation system (humidity,light,soil moisture, temperature, crop health, and so on). Farmers can keep an eye on their fields from anywhere.
8.	IIoT or industry 4.0	Industry 4.0, also known as IIoT or smart manufacturing, combines physical manufacturing and processes with smart digital technologies, machine learning, and big data to create a more holistic and linked environment for manufacturing and supply chain management firms.

1.2 IoT Security

In the Internet of Things systems, the shift is from desktop computers to small-sized computing devices with limited resources. The interconnection of these small devices via the Internet and across many networks, the exchange of large data, leads to an unprecedented challenge for those users by securing that data [14], [15]. In addition, because IoT devices interact directly with the actual world to gather sensitive data or regulate physical environment variables, they are easily accessible and vulnerable to several security assaults[16], making them an appealing target for attackers [17]. With demands of confidentiality, data integrity, authentication & authorization, availability, privacy & regulation requirements, and frequent system upgrades, all of these factors make cyber-security a serious concern in IoT devices [18]. Cryptography might be one of the most effective techniques in this circumstance to ensure the secrecy, integrity, authentication, and authorization of data transiting across IoT devices [15]. Cryptography might potentially be used to safeguard data that is stored or sent via a network. Traditional PC-based cryptography solutions, on the other hand, are not suited for most IoT devices, such as sensors and RFID tags, owing to resource constraints. This problem can be solved with a lighter version of these methods.

1.4 Measurement of Evaluation LWC

Despite extensive study in programming and equipment used for hardware and software implementation, the results demonstrate that the association between them is difficult to establish due to differences in the execution platform [18].

1.4.1 Hardware Implementation

Given that the hardware is the first, the major measurements are memory utilization, code size, and energy usage. Any lightweight cryptography can only be measured by a particular type of circuit. However, various techniques produce varied simulation results. As a result, there is no way to compare how different algorithms are implemented in hardware. In terms of memory use, lightweight primitives should use smaller blocks with smaller keys. However, rather than using read/write memory, "burning" keys onto devices use read-only structures. It allows the key schedule to use just the master keys for basic tasks. Energy economy is at the heart of hardware implementation, and latency, or the time it takes to perform a certain operation, is one of the factors used to build lightweight block ciphers.

1.4.2 Software Implementation

Lightweight cryptography (LWC) has integrated crucial metrics such as RAM use, code size, and throughput (execution time) in bytes per cycle. The FELICS (Fair Evaluation of Lightweight Cryptographic Systems) framework compares the performance of several lightweight algorithms across different implementations using a variety of measures. The FELICS test for three microcontrollers: 8-bit AVR, 16-bit MSP, and 32-bit ARM-based for all cipher. But we use a framework that supports the AVR ATmega128 (8-bit AVR). Table 1.2 explains all the characteristics of the target devices [19]. FELICS is extremely extensible, allowing it to be used to benchmark new lightweight primitives, extract new metrics, collect performance data for additional target devices, and assess implemented algorithms in new use scenarios [20].

Table 1.2: Key Characteristics of the Three Microcontrollers Used by FELICS[20].

Characteristic	AVR	MSP	ARM
Model	ATmega128	MSP430F1611	Cortex-M3
CPU	8-bit RISC	16-bit RISC	32-bit RISC
Frequency (MHz)	16	8	84
Registers	32	16	21
Architecture	Harvard	von Neumann	Harvard
Flash (KB)	128	48	512
SRAM (KB)	4	10	96
EEPROM (KB)	4	–	–
Supply voltage (V)	4.5 – 5.5	1.8 – 3.6	1.6 – 3.6

1.4.2.1 Metrics

Here we will explain the lightweight metrics that are extracted from the FELICS software simulation for IoT applications. In this version, the power consumption parameter is not extracted because it is closely related to the basic parameters. For each operation needed by the respective module separately, as well as for all operations together, detailed and precise data are provided. Embedded software engineers may utilize the entire findings to determine which cipher operations should be implemented for a given device and application. Because of their usefulness, cycle-accurate and free software simulators of the target embedded devices are favored over-development boards. While a software simulator is simple to download and install, a development board requires purchase and configuration.

1.4.2.2 Code Size

The code size is expressed in bytes and indicates how much space an operation takes up in the target device's non-volatile memory (for example, flash memory).The frameworks employ the GNU size tool to extract the code size for each target device, which lists the section sizes as well as the overall size in bytes for a specific binary file.

1.4.2.3 RAM

The RAM usage is divided into two categories: stack and data. After interruptions and subroutine calls, the stack consumption shows how much RAM was utilized to hold local variables and the return address. The static RAM need is determined by the size of the constants stored in the target device's RAM. It contains information relevant to each case, such as the data to encrypt, the master key, round keys, and initialization vectors. Because the framework doesn't allow for dynamically allocated variables, the heap isn't utilized at all.

1.4.2.4 Execution Time

The number of CPU clock cycles spent executing a particular operation is measured by the execution time. Either cycle-accurate software simulators of the target microcontrollers or development boards are used to derive the metric. The execution time is calculated by dividing the number of cycles in the system timer at the end of the measured operation by the number of cycles in the system timer at the start of the measured operation.

1.5 Why Symmetric Block Cipher?

The sharing of information necessitates secure and efficient encryption/decryption, which includes both symmetric and asymmetric encryptions. Asymmetric ciphers have great security properties, but they are more costly owing to additional computational processes [21]. In general, there are two types of block ciphers: symmetric and asymmetric. Both encoding and decoding take place on a fixed-size block (64 bits or more) at a time in a block cipher, but stream cipher processes the entire message byte by byte (8 bits at a time). Furthermore, stream cipher solely uses the confusion attribute (to use substitution between the ciphertext and the key to make the relationship as complicated as feasible). Block cipher, on the other hand, employs both confusion and dispersion (to use permutation to diffuse the statistical structure of plaintext throughout the bulk of ciphertext [22]).With a block cipher, reversing encrypted text is difficult. However, in a stream cipher, the encryption is done via XOR, which can be readily reversed to plain text. Hash function, on the other hand, is a mathematical process that converts data of any size into a bit string of a predetermined length hash (short). Inverting a one-way function is impossible. For these reasons, a block cipher is favored over a stream cipher in IoT devices, and this study focuses on the block cipher. One of the following structures is used in symmetric lightweight block cipher:

- Substitution-Permutation Network (SPN)
- Feistel Network (FN)
- General Feistel Network (GFN)
- Add-Rotate-XOR (ARX)
- NonLinear-Feedback Shift Register (NLFSR)
- Hybrid.

SPN and FN are the most common structures because of their versatility in terms of implementing the structure based on application needs [23]. Because a round function is applied to just one-half of the state, Feistel structures could well be built with low average power hardware [24].In contrast to SPN structures, Feistel structures use non-linearity in just one-half of the state in each round, hence preserving safety margins generally necessitates a larger round function. When a decision is made between fewer SPN function rounds and more Feistel function rounds with the same level of security and equivalent energy costs, the SPN function may be the better option [24].

1.5.1 Feistel Network

The construction of a block cipher in a Feistel cipher uses asymmetric structure in cryptography. Feistel cipher proposed a strong cipher that alternates substitutions (S-boxes) and permutations (P-boxes). Encryption and decryption operations in the Feistel structure are similar in some cases, requiring a reversal of the key schedule. Therefore, the block cipher requires the implementation of half the size of the code and the circuit. The operation is illustrated in the Feistel cipher in Figure 1.3. The round function is F, and the sub key is K0,…, Kn for the rounds 0, 1, …, n respectively. The operation encryption, divided the plaintext into equal (L0, R0):

For each round i = 0, 1, …, n compute

$L_{i+1} = R_i$

$R_{i+1} = L_i \oplus F(R_i, K_i)$

Then the ciphertext is (R_{n+1}, L_{n+1})

The operation Decryption: a ciphertext (R_{n+1}, L_{n+1})

Computing for i = n, n − 1, …, 0

$R_i = L_{i+1}$

$L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$.

Then (L0, R0) is the plaintext again.

The round function F in Feistel structure must not be invertible compared to SP-Structure. This is an advantage of Feistel structure [25], [26].

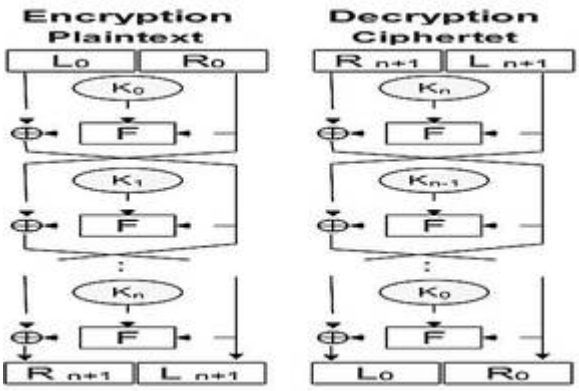

Figure 1.3: Encryption and Decryption Feistel Structure.

1.4.1 Generalized Feistel Network

The generalized Feistel Network (GFN) is a more advanced version of the Feistel cipher. In GFN, the input block is split into two or more sub-blocks, with each sub-block receiving a (classical) Feistel transformation, followed by a cyclic shift proportional to the number of sub-blocks [27].

1.4.2 SP-network structure

Block cipher in SP-network, uses a chain of associated mathematical operations in cryptography. The operations in SP-network are several rounds or “layers" of substitution boxes (S-boxes) and permutation boxes (P-boxes). The S-boxes and P-boxes perform efficiently in hardware, such as XOR and bitwise that transfer blocks of input bits into output bits. The key is used in each round. S-boxes depend on the key. The output of an S-box substitute is an input for another S-box to ensure decryption. In particular, the

length of input and output should be the same, as illustrated in Figure 1.4. A P-box is a mix for all output of each S-box of one round, and make it inputs to S-boxes of the next round. In each round, it uses XOR between input/output from using S-boxes and P-boxes and key. In cryptographic strength, it uses S-box and P-box togetherto satisfy confusion and diffusion properties [28], [29].

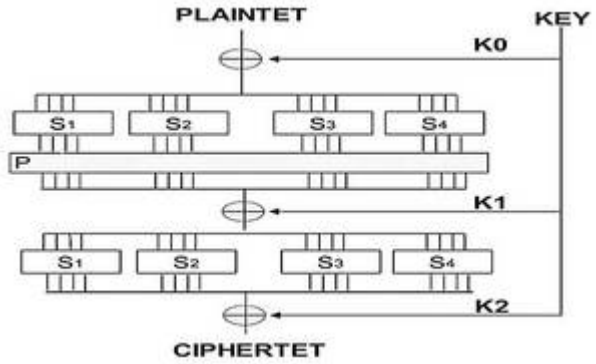

Figure 1.4: SP-Network 2 Rounds Encrypting Plaintext Block 16 Bit.

1.4.3 Nonlinear Feedback Shift Register (NLFSR)

The NLFSR is a kind of nonlinear feedback shift register that may be used in both stream and block ciphers. It makes use of stream cipher building components whose present state is a nonlinear feedback function of their prior state [30].

1.4.4 Hybrid Architecture

Hybrid ciphers combine any two or more kinds (FN, SPN, GFN, NLFSR, ARX) to increase certain characteristics (for example, Code Size (reduces) Memory (reduces), energy (reduces), and so on) according to application needs, or even blend block and stream ciphers.

1.5 Lightweight Block cipher: definition

The general purpose of the block cipher is to provide a semi-random flipping that builds complex protocols. There are several definitions for the lightweight block cipher. In [31], it is an appropriate implementation for low-resource devices, and it treats challenges such as: reduce overhead (silicon area or memory used), low-energy consumption, and sufficient security. Some researchers adopted certain characteristics to define lightweight ciphers [32].To design a lightweight cipher, such requirements must be met as follows:

- **Smaller block sizes**: To keep a memory, block size in lightweight block ciphers should be (64 bits or 80 bits) not as block size used in a conventional AES (128 bits).
- **Smaller key sizes**: For efficiency, key sizes in lightweight block ciphers should be (less than 96 bits)
- **Simpler rounds**: To save area, the S-boxes operations in lightweight block use 4-bit S-boxes. Whereas conventional blocks use 8-bit S-boxes.
- **Simpler key schedules**: To save power consumption, latency and memory, key schedules simple should be used in most of the lightweight block ciphers.

Minimal implementations: Uses of resources: encryption and decryption are not required for all applications. Some applications only support encryption or decryption operations. The block cipher function uses fewer resources.

1.6 Common Lightweight Block Ciphers

AES (Advanced Encryption Standard): It is one of the most important algorithms in cryptography designed by Daemen and Rijmen [33]. It has a block size of 128 bits and key length (128,192, 256) bits. It is built on SP network design philosophy, which combines both substitution and permutation, and is faster in both hardware and software. The algorithm possesses excellent properties, flexible in execution, and also offers high security. But it becomes an obstacle to the hardware limited resources. Although certain implementations of Rijndael have a greater block size and more columns in the state, AES operates on a 4x4 column-major order matrix of bytes called the state. The majority of AES computations are performed in a finite field. It works with a number of rounds to convert the plaintext into ciphertext. Each round contains the following functions :(Shift Rows, Mix Columns, and Add Round Key). It also uses the following functions (Inverse Mix Columns, Add Round Key, Inverse Substitute Bytes, and Inverse Shift Rows) to reverse the encryption process (converting the ciphertext to the original text) depending on the key.

PRESENT: In 2007, Bogdanov, et al [34]proposed an algorithm for a lightweight block cipher. It is used in two situations: the desired low-power consumptions and high chip efficiency. It has particularly high performance in compact hardware implementation. The block cipher works in 31 rounds, depends on SP (Substitution and Permutation) networks for implementation and supports two keys size 80 and 128 bits. The one round consists of an XOR operation, a bitwise permutation, and a nonlinear substitution, which consists of 4-bit input and 4-bit output (4×4) S-boxes, illustrated in Figure 1.5. The design of PRESENT does not use the algebraic unit in the diffusion layer. As the result, it provides hardware efficiency in implementation.

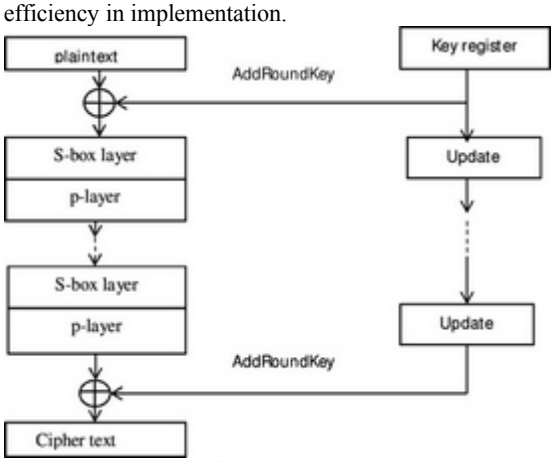

Figure 1.5: Encryption/Decryption in PRESENT.

Hummingbird: Engels et al. designed Hummingbird [35],and Hummingbird-2 [29], [36]. It's an ultra-lightweight cryptographic for encryption and authentication such as small hardware devices like RFID tags. The block size is 16 bits. It consists of 4 rounds and the last round only includes the key mixing and the S-box substitution steps. Like any other hybrid (structure), one round consists of three stages: a key mixing step, a substitution layer, and a permutation layer. This block size is suitable for constrained devices because it deals with small messages. Security is weak in hummingbird and hummingbird-2 because of the smaller size of the block.

PRINT-Cipher: Knudsen et al. [37] proposed the PRINT-cipher block code for IC printing as one of the low-constrained devices. It is built on an SP network design structure. There are two operations for cipher state: collecting a round key by using bitwise XOR and shuffling by fixed linear diffusion layer. Each S-box contains a 3-bit entry which is permuted in a key-dependent permutation layer; lastly, the cipher state is mixed using a layer of b3 nonlinear S-box substitutions, illustrated in Figure 1.6 for one round of PRINT- Cipher.

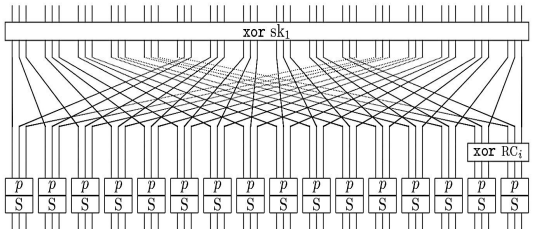

Figure 1.6:PRINT Cipher (One Round).

KLEIN: According to Gong et al. [38], it is one of the newest Lightweight block ciphers. The block size is a fixed 64-bit and key length 64, 80 or 90-bits. It follows the Substitution-Permutation Network (SPN) architecture. The KLEIN cipher has a 4-bit S-box and operations to make mixes from the AES and from PRESENT as well. In the block cipher, the key length and block size refer to trade-offs between performance and security, considering security measured by block size and performance measured by key length. Figure 1.7 shows the structure of KLEIN.

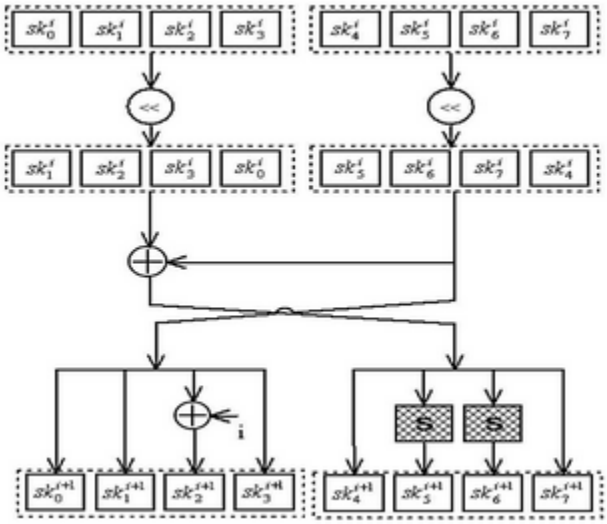

Figure 1.7: Structure of KLEIN.

LED: It means (Lightweight Encryption Device), proposed by Guo et al. [39], in 2011. It is one of the newest lightweight ciphers. It provides security against all state-of-the-art attacks. The cipher state uses a 4-bits matrix to represent arranged concepts, with all nibble showing an element from GF (24) with a polynomial as 〖 x〗 ^4 + x + 1. The S-box in the LED is the same as the PRESENT. Also, the LED is the same as lightweight has function PHOTON and the architecture is based on the Substitution-Permutation Network (SPN).

PRINCE: Borghoff et al. [40]state that the importance of PRINCE lightweight block cipher lies in taking low latency as the main priority. The internal construction of the block is based on the SPN architecture. The cipher with a slightly different key can provide decryption by reusing the encryption process. The block size 64 bits uses a 4-bit S-box.

HIGHT: Itmeans (high security and lightweight) proposed by Hong et al.[41]. The block cipher is an ARX based on a generalized Feistel Structure (GFS). The block cipher uses the operations XOR and bitwise rotations for input, and XOR or addition modulo 28 for output. The block cipher consists of 64-bit block size and key size 128-bit. Three functions to encrypt the entered text together with the key in HIGHT cipher (first transformation, round function and final transformation) see Figure 1.8.The Whitening-keys and sub-keys are generated by the Key Schedule procedure. For the initial and final transformations, there are eight Whitening keys (WK7—WK0). For round functions, 128 subkeys (SK127—SK0) are utilized in total, with four subkeys each round. The processes of coding and decoding in HIGHT are similar.

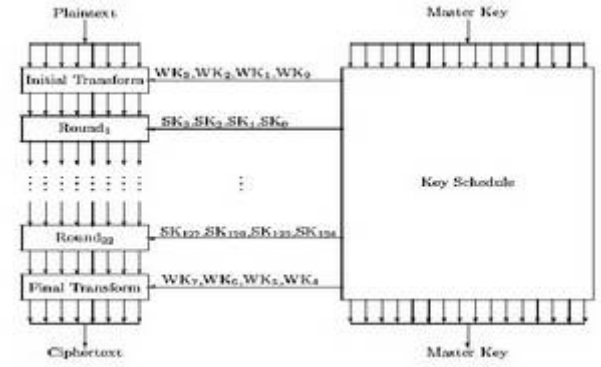

Figure 1.8: Structure of HIGHT for Encryption/Decryption.

DESL & DESXL: DESL and DESXL (DES Lightweight) are proposed by Poschmann et al. [42]. The primary idea of DESL and SESXL is to reduce gate complexity to limit the size of hardware. DESL and DESXL use only one single box rather than 8-boxes from the original DES. DESL and DESXL by a single S-box are strong against the attack and solve the weakness of DES. The security strength and reducing wiring costs are done by remain the original first permutation and its inverse. The block size 64-bits and small key size 56-bits achieve limited protection. Thus, applications that possess short-term security DESL are desired.

CLEFIA: Itis a lightweight block cipher that provides efficiency in implementations of hardware and software, proposed by Shirai et al. [43]. The CLEFIA block cipher has four branches from a generalized Feistel structure (GFS). CLEFIA has two round F-functions, every function uses 32-bits per round. Each branch consists of 32 bits to making block size 128 bits. The F-function process inputdividesthe 32-bits into two parts: (S) a substitution step leads to a simple S-box substitution and (D) a publishing step leads to a linear combination of substituted. CLEFIA uses this scenario to guarantee immunity against pioneer attacks.

KATAN and KTANTAN: They are two block ciphers proposed by De Cannièreet al.[36], [44] KATAN and KTANTAN for new family lightweight based on bivium stream cipher designed proposed at [45]. This design has a structure called nonlinear feedback shift register (NLFSR) in Feistel structure. The block size of cipher is 32-, 48- and 64 bits and key length 80 bits. Utilization of the physical footprint is at the heart of these two designs, at the cost of some speed. KATAN is less compact than KTANTAN. KTANTAN does not change the key because it is used in devices that have a fixed key. The key schedule is the only difference between KATAN and KTANTAN.KATAN and KATANTAN use shift registers.Therefore, KATAN and KATANTAN are appropriate for RFID devices [44].

LBlock: Wu and Zhang [46] in 2010 proposed for lightweight block cipher that uses Feistel Structure. The block LBlock has a 64-bit block size and a key length of 80-bit.The LBlock has three Functions: encryption algorithm, decryption algorithm, and key scheduling. Every round in LBlock has two round functions: confusion and diffusion (permutation of eight 4-bit words).Therefore, Feistel structures are suitable to minimize the number and size of the S-box. The one round of LBlock divided data into two parts: the first half goes through round function, and the second half applies simple rotation operation, therefore, the more round iteration leads to security margins.

TWINE: According to Suzaki et al. [47], the block cipher has a 64-bit block size and key length 80- and 128 bits. The structure that uses TWINE is a generalized Feistel structure (GFS) with 16 branches. The Feistel function, called eight times per round, consists of simply XORing a sub-key and the application of a 4-bit S-box. In The TWINE, a round function has a nonlinear layer using 4-bit S-boxes and a diffusion layer, which permutes the 16 blocks. The diffusion layer provides diffusion better because it does not use the circular shift in diffusion layer designs.

SIMON: It is one of the families of block cipher lightweight as proposed by Beaulieu et al. [48] in 2013. The structure SIMON

is a Feistel structure. It providesa target in hardware implementations to optimal performance. The block has sizes of variable lengths (32, 48, 64, 96,128) for keys of variable lengths (64, 72, 96, 128, 144, 192, 256). The encryption and decryption block operates in rounds (32 and 72). The nonlinear function in SIMON uses a bitwise XOR operation and rotates (every round), making it suitable for hardware. Figure 1.9shows the work of the block of one round.

SPECK: It is one of the families of block cipher lightweight as proposed by Beaulieu et al. [48] in 2013. The structure SPECK is ARX. It provides a target in software implementations to optimal performance. The block operates with the same block size, key size and number of rounds as the block of Simon. The nonlinear function in SPECK uses the modular addition operation, XOR and rotate (every round), making it suitable for software. Figure 1.10 shows the work of the block for one round.

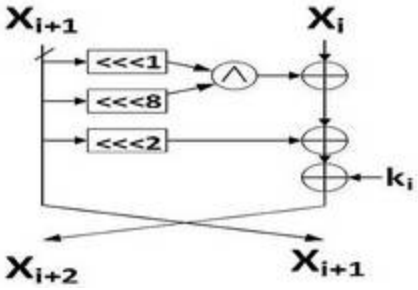

Figure 1.9: Block of SIMON for One Round.

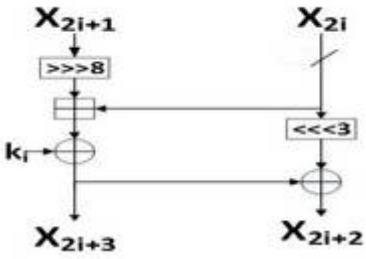

Figure 1.10: Block of SPECK for One Round.

Piccolo: According to Shibutani et al. [49],it is a new block cipher of lightweight. It is a generalized Feistel network (GFN). The Piccolo has block size 64-bits and 80- and 128-bits keys length. Piccolo achieves low power consumptions and high security, based on the design that is based on (half-word based round permutation) and (permutation for key expanding). Therefore, it is suitable for sensor nodes and FRID devices.

LEA: According to Hong et al. [50], it means Lightweight block Encryption Algorithm. It was designed by the Electronics and Telecommunication Research Institute of Korea (ETRIK) for software-oriented. It can be implemented on different platforms. The software encryption in LEA is fast on most common processors. The operations in LEA are simple Rotation, XOR and Addition, and they do not use operation S-Box. It provides implementation with high efficiency in software/hardware.

QTL: According to Lang Li et al.[51], the structure used in QTL is a generalized Feistel structure. The QTL has a block size of 64-bits with 46-bit or 128-bit keys length. The one round can process half the block message in Feistel structure, but QLT can change all messages. The QTL uses diffusion of the (SPNs) structure, which achieve security in Feistel-type structures. It reduces the power consumption and area and does not use the key schedule in designing. It achieves high security and cost implementation in hardware.

Blowfish: It is a symmetric encryption technique, which means it encrypts and decrypts communications using the same secret key. It is a hybrid architecture. It separates the data (plaintext) into 64-bit blocks of predetermined size. It can handle a wide range of variable size lengths, from 32 bits to 448 bits. In Figure 1.11 we notice P boxes are employed in the F-function to implement each round of the blowfish block [52]. P boxes are 32 bits in size and there is a total of 18 of them.

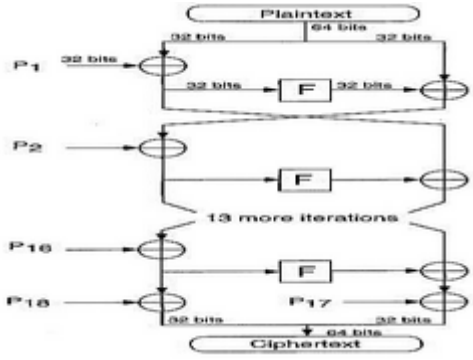

Figure 1.11: Block of Blowfish.

BORON: Bansod et al. [53] suggested an LWC technique based on SPN. It supports 128/80 key bits along with a 64-bit plaintext block. It is based on a total of 25 rounds, where the 4-bit to 4-bit S-boxes are used, followed by round shift, permutations, and XOR operations. It is also immune to attacks, differential and linear.

LiCi: Patil et al. [54] suggested an LW block cipher algorithm. It considers 31 consecutive rounds paired with 4 × 4 LW S-boxes. The cipher LiCi is a balanced network of Feistel structure, and its architecture supports 128 bits key for 64-bits plaintext.

1.7 Performance Analysis

FELICS tool (described in the previous section) can extract measurements (code size, memory, execution time (for encryption/decryption)). These are lightweight algorithm measurements (capable of running on IoT devices with high efficiency). A measurement (code size, memory, and execution time (for encryption/decryption)) whose value is lower is considered better.Table 1.3 and Figure 1.12, explain the lightweight algorithms as well as the values extracted by the FELICS tool for measurements (code size, memory, and execution time (encoding/decoding)). On an LW device, the code size of ciphers has an impact on performance. Most LW devices, such as microcontroller chips, embedded devices, RFID tags, and so on, have enough memory (for example, flash memory) to store cryptographic algorithms. From Figure 1.1٣, the algorithms PRINCE, LBLOCK, and AES have the highest values in code size, while the algorithms RoadRunneR, PICCOL and TWINE have the lowest values in code size, respectively.

Table 1.3: Results for Common Cipher Implementations on AVR Architecture in FELICS Tools

Cipher	Device	Block Size (bit)	Key Size (bit)	Code Size (byte)	RAM (byte)	Encrypted-Key Schedule	Encryption (cycles)	Decryption (cycles)
AES	AVR	128	128	23464	720	2424	5225	5242
RC5	AVR	64	128	20444	360	30744	5244	5239
PRINCE	AVR	64	128	23838	176	675	7044	7047
HIGHT	AVR	64	128	13716	288	1615	3459	3543
LBLOCK	AVR	64	80	23718	306	4824	4772	4799
PICCOL	AVR	64	80	1534	126	1563	12630	12709
LILLIPUT	AVR	64	80	3908	276	12778	10934	11424
TWINE	AVR	64	80	2204	214	5047	10303	10183
RoadRunneR	AVR	64	80	1426	142	967	3658	3682
LED	AVR	64	80	4108	358	369	66950	71061

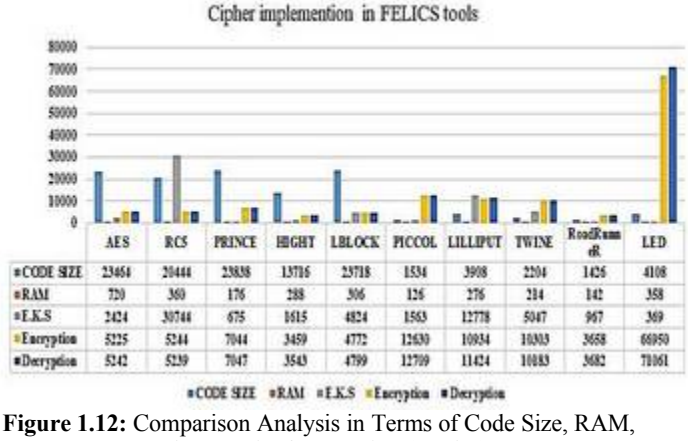

Figure 1.12: Comparison Analysis in Terms of Code Size, RAM, andEncryption/Decryption (Cycles).

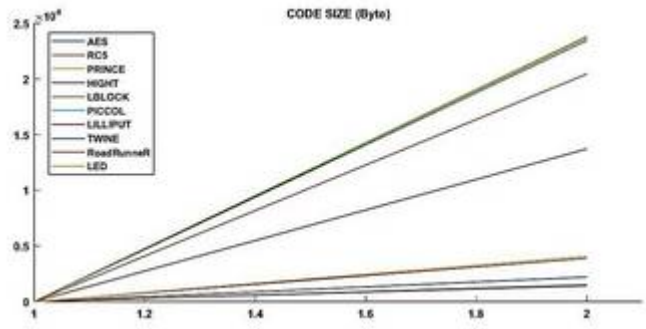

Figure 1.1٣: Different Ciphers are Compared in Terms of Code Size.

The RAM for LW devices is restricted, the utilized RAM size of the ciphers is critical for speed. As a result, the most efficient ciphers are those that require the least amount of RAM when running like PICCOL, RoadRunneR, and PRINCE, as shown in Figure 1.14. Another case is between the two algorithms (HIGHT and LED). The HIGHT requires a higher code size in encryption and decryption and requires less memory expenditure compared to the LED algorithm (which needs the opposite). Due to the increased number of rounds in these ciphers, usually, rounds (3 to 20) must be in the algorithm for the encryption to be lightweight while retaining the strength of the algorithm. As a result, LED is superior to HIGHT since it uses less memory for devices of IoT. The security strength of a cipher is dependent on the complexity of calculations, which is a problem. However, there is a balance to be struck between security and complexity.

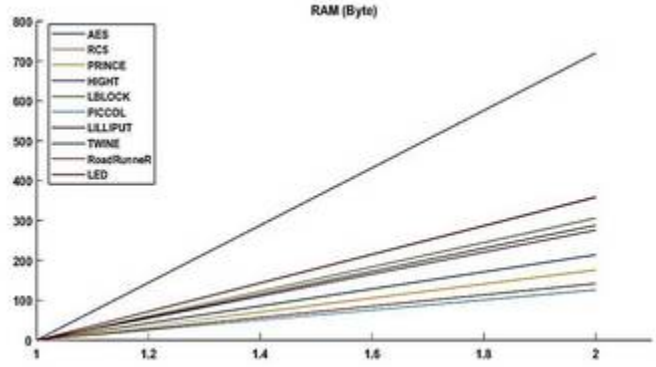

Figure 1.14: Different Ciphers are Compared in Terms of RAM.

In Figures 1.15, 1.16 the execution time (encoding / decoding cycles) should require the lowest values, because it is related to the process of power consumption (the higher the execution time, the higher the power consumption). Because the gadgets with limited resources(IoT devices) are battery-operated, they are low-powered. As a result, power consumption should be kept to a minimum, yet cipher security should be adequate. In other words, there is a trade-off between security and lightweight. The security of a cipher is entirely dependent on the keys used to encode and decode each message block. Therefore, one of the most important operations in encryption is the process of key generation (it must be complex) because it is used in all cycles of encryption and decryption. However, its complexity (arithmetic) does not affect the performance of devices when encrypting or decrypting the block data. We note that the HIGHT, RoadRunneR and LBlock are fewer in the number of execution cycles (less energy consumption). But HIGHT has a higher execution of master scheduling cycles than the rest of the blocks (RoadRunneR and LBlock) and this is due to the complexity of the key generation process in the HIGHT block.

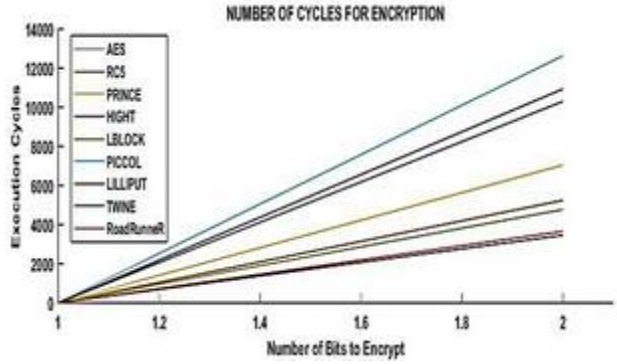

Figure 1.15: Execution Time (Cycle) for Multiple Ciphers in the Varying Block of Encryption.

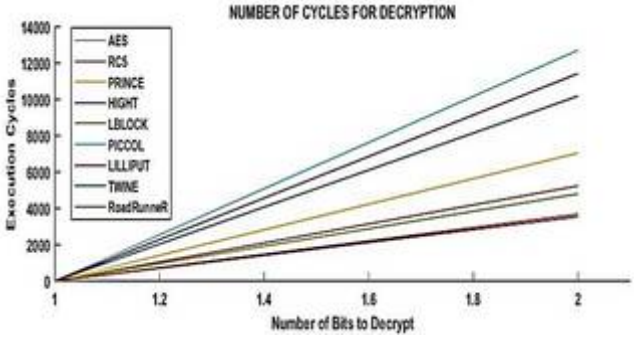

Figure 1.16: Execution Time (Cycle) for Multiple Ciphers in the Varying Block of Decryption.

The power use of the algorithm is measured by the following equation [55] (1.1):

$$E=I * VCC* Ţ*N \qquad (1.1)$$

Here, VCC is the supply voltage of the system. I is the average current in amperes in which T is consumed in seconds.

Ţ is the clock period. N is the number of the clock cycle. So, the clock period is Ţ=1/f Sec/Cycle. Atmel Atmega128 (AVR) typically takes voltage in the range of 2.7~5.5, current 20 mA on average, and also runs at 16 MHz. Figure 2.17 displays the comparison of power usage among current ciphers with the suggested cipher

Figure 1.17: Power Consumption of Ciphers.

1.8 Trends of Design for LWC Algorithm

Smart and light technologies are modern trends for the world that use devices such as RFID, sensor and embedded systems. As a consequence, researchers have been developing and proposing a range of cryptographic algorithms to suit these devices.The NIST provides an LWC project that describes the issues and develops a technique for the standardization of lightweight cryptographic algorithms.In this project, numerous metrics were identified to evaluate the lightweight properties, in hardware implementation chip size (area or resource) and/or energy consumption (performance), and in software implementation code size, RAM sizeand/or execution time (This is our field of study).To provide knowledge for selecting or determining cipher of LWC in design directions, the following are required:

- **Block size and key length**: Large block and key size cannot achieve space reduction. Memory is affected by large block size and large key size; therefore, it must block the size and key size which must be small to save memory. For instance,the block size in lightweight block ciphers should be (64 bits or 80 bits) not as the block size that is used in a conventional AES (128 bits). Key sizes in lightweight block ciphers should be (less than 96 bits). The key size acceptable by NIST is less than 112-bit and is equal to or more than 64-bits [56], as PRESENT of 80-bit. On the other hand, various issues of security, if we use a block size less than 32 bits, the birthday attacks will be possible. Blocks of 64-bit and keys size of 80-bit are popular parameters for ciphers of lightweight.

- **Key schedule:**Uses simple key schedules in most of the lightweight block ciphers to save power consumption, latency and memory. As a result, some attacks can be possible to get keys during generating sub-keys such as weak keys and related keys but can prevent these attackers by using key derivation function (KDF) of security, described in [57], [58] and [59]. When the security level is not more important than the implementation cost, for applications, the good selection is a simple key schedule and Feistel structure, SPNs, and a simple key schedule when moderate security is the best choice.
- **Simpler rounds**: To save area, the S-boxes operations in lightweight blocks use 4-bit S-boxes while conventional blocks use 8-bit S-boxes.
- **Minimal implementations**: Uses of resources: encryption and decryption are not required for all applications. Some applications need either encryption or decryption operations. The block cipher functions use fewer resources.
- **A combination of Feistel, SP and more architectural methods** increases the complexity of the encryption.
- **Thealgorithms(designed) flexibility** is decently efficient on a wide range of 8-bit Micro-controllers.

1.9 Conclusion

The current review in terms of IoT security.First, knowledge of the Internet of things and its applications and knowledge of the security required for this study is presented. Second, the general standards required for lightweight algorithms (which work on Internet of Things devices) are discussed. Then, a general study of algorithms in the field of the Internet of Things is provided. And finally, the recommendations that must be followed to design lightweight algorithms that work on Internet of Things devices and apply them in this paper are highlighted.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conficts of interest.

Acknowledgments

This work was conducted during our work at Hodeidah University.

How to Cite : Al-ahdal A. H.A, Ali M.M.H., Alahdal A., Qaed A.S.M. & Al-Rummana G.A. (2022). Securing the Internet of Things: A review of Lightweight and low-power Cryptography Techniques, *Abhath Journal of Basic and Applied Sciences*, 1(2), 18-26.

References

1. **McKay N. M. Larry Bassham KA, , and Turan M. S. (2017).** "NISTIR 8114 Report on Lightweight Cryptography,"
2. **Nandhini P., Vanitha V., and Scholar P. (2017).** A Study of Lightweight Cryptographic Algorithms for IoT, Int. J. Innov. Adv. Comput. Sci. IJIACS ISSN, 6(1), 2347–8616.
3. **A. M. I. Alkuhlani and S. B. Thora. (2018).** "Lightweight Anonymity-Preserving Authentication and Key Agreement Protocol for the Internet of Things Environment",108–125.
4. **K. T. Nguyen, M. Laurent, and N. (2015).** Oualha, "Survey on secure communication protocols for the Internet of Things," Ad Hoc Networks, 32, 17–31. doi: 10.1016/j.adhoc.2015.01.006.
5. **Kahn, D. The Code breakers (1996). 1181,** ISBN 0-684-83130-9. Look for the 1967 rather than the 1996 edition.
6. **M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. (2005).** "AES implementation on a grain of sand," IEE Proc. - Inf. Secur., 152(1), 13. doi: 10.1049/ip-ifs:20055006.
7. **N. F. Standard.(1999).** Data Encryption Standard (DES). Federal Information Processing Standards Publication.
8. **N. F. Standard. (2001)**. Announcing the advanced encryption standard (AES). Federal Information Processing Standards Publication, 197, 1-51.
9. **G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. (2009).** Fitton, "Smart objects as building blocks for the internet of things," IEEE Internet Comput., 14(1), 44–51. doi: 10.1109/MIC.143
10. **N. Pérez Moldón. (2019)**."Security in IoT ecosystems.".
11. **E. Brown, 2018**, "21 open source projects for IoT," Linux.com. Retrieved, 23.
12. **S. Charmonman and P. Mongkhonvanit. (2015**). "Internet of things in e-business," in Proceeding of the 10th International Conference on e-Business King Mongkut's University of Technology Thonburi. 1–9.
13. **"The trouble with the internet of things," Aug (2015)**. [Online]. Available: https://data.london.gov.uk/blog/the-trouble-with-the-internet-of-things/.
14. **K. McKay, L. Bassham, M. S. Turan, and N. Mouha, (2017),** "Report on lightweight cryptography (nistir8114)," National Institute of Standards and Technology (NIST.
15. **B. J. Mohd and T. Hayajneh. (2018)**. "Lightweight Block Ciphers for IoT: Energy Optimization and Survivability Techniques," IEEE Access. 6, 35966–35978. doi: 10.1109/ACCESS.2018.2848586.
16. **S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, (2017),** "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions," J. Ambient Intell. Humaniz. Comput. 1–18. doi: 10.1007/s12652-017-0494-4.
17. **W. Feng, Y. Qin, S. Zhao, and D. Feng. (2018)**. "AAoT: Lightweight attestation and authentication of low-resource things in IoT and CPS," Comput. Networks, 134, 167–182, 2018, doi: 10.1016/j.comnet.01.039.
18. **A. Banafa. (2017).** "Three major challenges facing IoT," IEEE IoT Newsletter.
19. Retrieved.**(2019)** https://www.cryptolux.org/index.php/FELICS.
20. **Dinu, D. D.(2017)**. Efficient and secure implementations of lightweight symmetric cryptographic primitives (Doctoral dissertation, University of Luxembourg, Luxembourg, Luxembourg.
21. **Martin Ågren. (2012).** Doctoral dissertation On Some Symmetric Lightweight Cryptographic Designs.
22. **W. Stallings. (2019).** Cryptography and Network Security: Principles and Practice. [Online].Available: Retrieved .http://uru.ac.in/uruonlinelibrary/Cyber_Security/Cryptography_and_Network_Security.pdf
23. **G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, and C. Manifavas.(2018**). "A review of lightweight block ciphers," J. Cryptogr. Eng., 8(2), 141–184, doi: 10.1007/s13389-017-0160-y.
24. **S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni.(2015).** "Midori: A block cipher for low energy," in Inter- national Conference

on the Theory and Application of Cryptology and Information Security. Springer. 411–436.
25. **Menezes, Alfred J.; Oorschot, Paul C. van; Vanstone, Scott A. (2001)**. Handbook of Applied Cryptography (Fifth ed.). 251, ISBN 0849385237.
26. **M. Luby and C. Rackoff. (1988).** "How to Construct Pseudorandom Permutations from Pseudorandom Functions," SIAM J. Comput., 17(2). 373–386,doi.org/10.1137/0217022.
27. **T. Suzaki and K. Minematsu.(2010).** "Improving the generalized Feistel," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 6147 , 19–39, doi: 10.1007/978-3-642-13858-4_2.
28. **Preneel, B., &Rijmen, V.(1998)**. Principles and performance of cryptographic algorithms. Dr. Dobb's Journal: Software Tools for the Professional Programmer. 23(12), 126-130.
29. **N. Ferguson, (2010)**, "The Skein Hash Function Family," Argument, 30(4),79, [Online]. Available: http://www.schneier.com/skein.html.
30. **A. Bogdanov.(2007).** "Cryptanalysis of the KeeLoq block cipher.," IACR Cryptol. ePrint Arch., 2007,55.
31. **J. Daemen. (1995).** "Cipher and Hash Function Design Strategies Based on Linear and Differential Cryptanalysis," Ph. D. Thesis. 267, [Online]. Available: http://jda.noekeon.org/JDA_Thesis_1995.pdf.
32. **P. Barreto and V. Rijmen.(2000).** "The khazad legacy-level block cipher," Primitive submitted to NESSIE, vol. 97.
33. **J. Daemen and V. Rijmen.(2002)**."The Design of Rijndael,AES - The Advanced Encryption Standard" New York,255,[Online].Available: http://portal.acm.org/citation.cfm?id=560131.
34. **A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, and A. Poschmann.(2007)**. "PRESENT : An Ultra-Lightweight Block Cipher,". 450–466.
35. **X. Fan, H. Hu, G. Gong, E. M. Smith, and D. Engels. (2009)**. "Lightweight implementation of hummingbird cryptographic algorithm on 4-bit microcontrollers," Int. Conf. Internet Technol. Secur. Trans. ICITST 2009, no. Icc. 5–7, doi: 10.1109/ICITST.2009.5402515.
36. **D. Engels, M. J. O. Saarinen, P. Schweitzer, and E. M. Smith. (2012).** "The hummingbird-2 lightweight authenticated encryption algorithm," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 7055, 19–31, doi: 10.1007/978-3-642-25286-0_2.
37. **L. Knudsen, G. Leander, A. Poschmann, and M. J. B. Robshaw. (2010).** "PRINTcipher: A block cipher for IC-printing," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 6225, 16–32, doi: 10.1007/978-3-642-15031-9_2.
38. **Z. Gong, S. Nikova, and Y. W. Law.(2012).** "KLEIN: A new family of lightweight block ciphers," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 7055 , 1–18, doi: 10.1007/978-3-642-25286-0_1.
39. **J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw,.(2011).** "The LED block cipher," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 6917, 326–341, doi: 10.1007/978-3-642-23951-9_22.
40. **J. Borghoff et al. (2012)**. "PRINCE – A Low-Latency Block Cipher for pervasive computing applications Extended Abstract," Int. Conf. Theory Appl. Cryptol. Inf. Secur., no. 11061130539. 208–225.
41. **D. Hong et al. (2006)**."HIGHT : A new block cipher suitable for low-resource device," . 46–59.
42. **G. Leander, C. Paar, A. Poschmann, and K. Schramm, (2007),** "New lightweight des variants," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 4593, 196–210, doi: 10.1007/978-3-540-74619-5_13.
43. **T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata. (2007).** "The 128-Bit Blockcipher CLEFIA (Extended Abstract)," . 181–195, doi: 10.1007/978-3-540-74619-5_12.
44. **C. De Cannière, O. Dunkelman, and M. Knežević. (2009).** "KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 5747, 272–288, doi: 10.1007/978-3-642-04138-9_20.
45. **C. De Canni, "Trivium," . (2008). pp. 244–266,**.
46. **W. Wu and L. Zhang. (2011).** "LBlock: A lightweight block cipher," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 6715 , 327–344, doi: 10.1007/978-3-642-21554-4_19.
47. **T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, (2011).** "Twine: A lightweight, versatile block cipher," ECRYPT Work. pn Light. Cryptogr. LC11. 146–169, 2011, [Online]. Available: http://www.nec.co.jp/rd/media/code/research/images/twine_LC11.pdf.
48. **R. Beaulieu and S. Treatman-clark, (2013).** "The Simon and Speck Families of Lightweight Block Ciphers,".
49. **K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, (2011).** "Piccolo: An ultra-lightweight blockcipher," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 6917, 342–357, doi: 10.1007/978-3-642-23951-9_23.
50. **D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu, and D.-G. Lee,(2014).** "LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors,". 3–27.
51. **L. Li, B. Liu, and H. Wang, (2016).** "QTL: A new ultra-lightweight block cipher," Microprocess. Microsyst., vol. 45. 45–55, Aug. 2016, doi: 10.1016/j.micpro.2016.03.011.
52. **M. N. Valmik and P. V. K. Kshirsagar, (2014).** "Blowfish Algorithm," IOSR J. Comput. Eng., vol. 16, no. 2. 80–83, doi: 10.9790/0661-162108083.
53. **G. Bansod, N. Pisharoty, and A. Patil, (2017).** "BORON: an ultra-lightweight and low power encryption design for pervasive computing," Front. Inf. Technol. Electron. Eng., vol. 18, no. 3. 317–331, doi: 10.1631/FITEE.1500415.
54. **J. Patil, G. Bansod, and K. S. Kant, (2017).** "LiCi: A new ultra-lightweight block cipher," in 2017 International Conference on Emerging Trends & Innovation in ICT (ICEI), Feb. 2017. 40–45, doi: 10.1109/ETIICT.2017.7977007.
55. **M. Alizadeh, M. Salleh, M. Zamani, S. Jafar, and K. Sasan,(2015).** "Security and Performance Evaluation of Lightweight Cryptographic Algorithms in RFID," Recent Res. Commun. Comput., no. November 2015. 45–50, 2012, [Online]. Available: http://goo.gl/ej5iEr.
56. **Barker, E., and Roginsky, A.(2015).** Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths, NIST Special Publication (SP) 800-131A Revision 1, National Institute of Standards and Technology, Gaithersburg, Maryland, November 2015, https://doi.org/10.6028/NIST.SP.800-131Ar1.
57. **Chen, L. (2011).** Recommendation for Key Derivation through Extraction-then-Expansion, NIST Special Publication (SP) 800-56C, National Institute of Standards and Technology, Gaithersburg, Maryland, November 2011, https://doi.org/10.6028/NIST.SP.800-56C.
58. **Chen, L. (2009).** Recommendation for Key Derivation Using Pseudorandom Functions (Revised), NIST Special Publication (SP) 800-108, National Institute of Standards and Technology,Gaithersburg,Maryland,October 2009, https://doi.org/10.6028/NIST.SP.800-108.
59. **Dang, Q. (2011)**, Recommendation for Existing Application-Specific Key Derivation Functions, NIST Special Publication (SP) 800-135 Revision 1, National Institute of Standards and Technology, Gaithersburg, Maryland, December 2011, https://doi.org/10.6028/NIST.SP.800-135r1