BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **24**, No 1
Sofia • 2024 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2024-0002

A Survey on Lightweight Cryptographic Algorithms in IoT

P. S. Suryateja, K. Venkata Rao

Dept. of CS&SE, AUCOE, Andhra University, Visakhapatnam, AP, India
E-mails: suryateja.pericherla@gmail.com professor_venkat@yahoo.com

Abstract**:** *The Internet of Things* (*IoT*) *will soon penetrate every aspect of human life. Several threats and vulnerabilities are present due to the different devices and protocols used in an IoT system. Conventional cryptographic primitives or algorithms cannot run efficiently and are unsuitable for resource-constrained devices in IoT. Hence, a recently developed area of cryptography, known as lightweight cryptography, has been introduced, and over the years, numerous lightweight algorithms have been suggested. This paper gives a comprehensive overview of the lightweight cryptography field and considers various popular lightweight cryptographic algorithms proposed and evaluated over the past years for analysis. Different taxonomies of the algorithms and other associated concepts were also provided, which helps new researchers gain a quick overview of the field. Finally, a set of 11 selected ultra-lightweight algorithms are analyzed based on the software implementations, and their evaluation is carried out using different metrics.*

Keywords**:** *Lightweight cryptography, Lightweight cryptography algorithms, IoT security, Cryptography in IoT, Lightweight cryptography in IoT, Ultra-lightweight algorithms.*

1. Introduction

IoT supports ubiquitous computing, which contains devices embedded with microprocessors, microcontrollers, and other components to provide enhanced services. The devices part of the IoT system can be divided into two categories. One is the high-end devices, which contain devices like PCs, laptops, tablets, etc., and the second is the low-end devices, which contain sensors, actuators, RFID tags, etc [1]. These devices often need more resources like processing power, RAM, ROM, battery life, etc. So, there is a need to choose the communication protocols and other underlying technologies carefully so that the performance of these devices does not hinder the user experience [2]. The data communicated by these resource-constrained devices in IoT must be protected to prevent unauthorized access or damage to the IoT environment [3, 4]. So, it is inherent that various security mechanisms need to be considered to find the best possible solution to secure those devices [5].

Applying conventional cryptographic solutions to the resource-deprived devices in IoT for securing the data is no longer an option as the system's performance will be degraded, and the lifetime of the devices will become less [1]. To overcome this, researchers and institutes governing the standardization of various protocols and algorithms introduced LightWeight Cryptography (LWC), which supports security for the data and devices in IoT. This paper provides a comprehensive overview of various popular lightweight cryptographic primitives or algorithms developed and studied over the years. Different taxonomies are provided to aid the new researchers in gaining a quick overview of the field. The contributions of this study are:

- A comprehensive survey of the existing lightweight cryptographic algorithms.
- Various taxonomies to aid new researchers in getting an overview of the LWC algorithms and associated concepts.
- A classification of existing LWC algorithms into ultra-lightweight, low-cost, and lightweight algorithms, including the latest algorithms.
- An analysis of software-oriented ultra-lightweight algorithms.
- Security analysis of PRIDE, ITUbee, and IDEA ciphers.

2. Background

This section introduces the fundamentals related to lightweight cryptography, like the metrics used to evaluate the performance of LWC algorithms, features, security requirements to be satisfied, benchmarking tools, etc.

The resource-constrained devices in IoT are limited concerning different resources like processing power, RAM, ROM, battery, storage, and connectivity. Based on the capabilities of the devices, the implementations of LWC algorithms can be divided into three categories: ultra-lightweight, low-cost, and lightweight [2]. The ultra-lightweight implementations are for the most constrained devices like the standard 8051 microcontroller and the ATtiny45. The low-cost devices are slightly better than the ultra-lightweight devices and perform better, like the ATmega128. Lightweight devices include the other devices that were reported in LWC. Hardware-based implementations of LWC algorithms can be grouped based on the complexity and chip area. Similarly, software implementations of LWC algorithms can be grouped based on the RAM and ROM requirements. The summary of hardware and software-based implementations categorization is given in Table 1.

Table 1. Categorization of hardware-based and software-based implementations

Implementation	Hardware-based implementation		Software-based implementation
Category	Logic Gates Count	ROM	RAM
Ultra-Lightweight	up to 1000	4 KB	256 Bytes
Low-cost	up to 2000	4 KB	8 KB
Lightweight	up to 3000	32 KB	8 KB

Designing and developing a new LWC algorithm requires a focus on three features: security, cost, and performance [6]. To evaluate these features for LWC algorithms, we need metrics. The cost can be measured using the physical area in

terms of Gate Equivalent (GE), memory, and battery power. Performance can be measured using latency and throughput. Finally, security can be measured in terms of security level and various attack models [1]. Different metrics that can be used for LWC algorithms are security level, hardware technology, throughput, latency, power and energy consumption, RAM/ROM, efficiency, and figure of merit [1-3, 7, 8]. The descriptions and corresponding formulae for evaluating the LWC algorithms are given in Table 2.

Lightweight block ciphers can be developed by using smaller block sizes, smaller key sizes, simpler or lesser number of rounds, and simpler key schedules than those used in conventional cryptographic algorithms [9]. Developing new lightweight hash functions can focus on reducing the output size and message sizes.

Table 2. Metrics for evaluating LWC algorithms

Metric	Description
Security level	The security level is the logarithmic measure of the fastest known computational attack and is measured in bits. In most cases, the security level is the length of the key
Hardware technology	It is measured using the Gate Equivalent (GE) metric or logic gates count, which is calculated by dividing the layout area of implementation in μm^2 by the area of a NAND2 gate
Throughput	It is measured in Kb per 1 s, the data processed by the encryption or decryption at a specific processor frequency. In general, the frequency considered for hardware implementation is 100 KHz, and for software implementation is 4 MHz
Latency	It is the number of clock cycles required for processing a single block by the encryption or decryption operation
Power and energy consumption	The power is measured in μW. Power is measured based on the hardware technology used and the GE for hardware implementations. The average power of 0.004 μW for 8-bit and 0.00135 μW for 16-bit microcontrollers is considered for software implementations. Energy consumption per bit for both software and hardware implementations is measured using the formula given below: Energy = (Latency × Power) / (block size) (in μJ)
RAM/ROM	The amount of memory in RAM for storing the intermediate state is considered in bytes. The amount of memory in ROM for storing the actual code of the implementation is considered in bytes
Efficiency	It indicates the trade-off between the implementation size and performance. For hardware implementations, efficiency is calculated using the formula given below: Hardware Efficiency = Throughput / Complexity (in kg GE) For software implementations efficiency is calculated using the formula given below: Software Efficiency = Throughput / Code Size (in KB)
Figure of merit	Figure of Merit (FOM) represents the system's performance and includes the power consumption. It is independent of the process involved. FOM can be calculated using the formula given below. FOM = T/A^2, where T is the throughput and A is the implementation area which is measured in GE

Any LWC algorithm should satisfy four conditions: confidentiality, integrity, authentication, and non-repudiation for accepting it as a perfectly secure algorithm [5, 7, 10, 11]. Symmetric LWC algorithms can only guarantee the first three conditions, whereas asymmetric LWC algorithms can also guarantee non-

repudiation. Some major vulnerabilities in IoT are insufficient authentication and authorization, insecure network services, absence of encryption and integrity checking, insecure software and firmware, and lack of proper physical security [3] [12]. Insecure system updates, privacy, and regulatory standards are significant IoT challenges [1, 13]. Hence, there is a need to develop lightweight solutions to address these vulnerabilities. Significant challenges for deploying various devices in IoT are coordination among multiple networks supporting multiple protocols by integrating them into an IP-based network and guaranteeing the security, privacy, and trust of various entities [9].

There are different benchmarking tools for evaluating the performance of an LWC implementation. The most widely used benchmarking tools are XBX, ATHENA, BLOC, and FELICS [7]. After designing and implementing an LWC algorithm, it must be studied or evaluated against different types of attacks. This process of analyzing a cipher is known as cryptanalysis. The types of attacks can be roughly classified into two categories: generic and non-generic attacks [7]. Some of the attacks under these two categories are presented in Fig. 1. Finally, different institutes, research groups, and vendors are continuously putting their efforts into developing, evaluating, and standardizing the LWC algorithms. Some of the most notable of these bodies are CryptoLux, ECRYPT NET, NSA, NIST LWC-Forum, ISO/IEC, Cryptrec, Ecrypt, IETF, CryptoLUX, Google, Sony, and Intel [1] [7]. The following section introduces a taxonomy of the existing LWC algorithms.

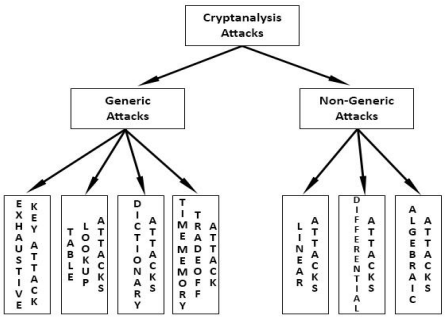

Fig. 1. Types of cryptanalysis attacks

3. Classification of LWC algorithms

This section presents different classifications of the existing LWC algorithms. First, a general classification of LWC algorithms is given and then followed up by two new classifications that are not present in the existing literature.

The LWC algorithms can be categorized into two groups based on the number of keys being used. Algorithms using the same key (secret key or private key) for encryption and decryption are called symmetric algorithms. Algorithms using different keys (private and public keys), one for encryption and the other for decryption, are called asymmetric algorithms. Symmetric algorithms provide confidentiality, integrity, and authentication security services. In contrast, asymmetric algorithms provide confidentiality and integrity by using the receiver's public key and authentication (in the form of a digital signature) and non-repudiation

by using the sender's private key. The downside of symmetric algorithms is the key distribution, and for asymmetric algorithms, it is the usage of large keys (number of bits) for encryption and decryption [1, 10]. The classification of LWC algorithms can be visualized, as shown in Fig. 2.

The symmetric algorithms can be further divided into block and stream ciphers based on how the plaintext is processed. A block cipher processes the plaintext as fixed-size blocks and generates the ciphertext. On the other hand, a stream cipher takes the plaintext as bits and performs operations on the bits to generate ciphertext. The two fundamental operations performed by a block cipher are confusion and diffusion. The confusion operation removes the dependencies between the key and the generated ciphertext to prevent statistical analysis attacks. The diffusion operation removes the dependencies between the plaintext and ciphertext to prevent statistical analysis and guessing attacks. A block cipher uses both confusion and diffusion operations. Whereas a stream cipher only uses confusion operations. So, the security of a block cipher is more than a stream cipher. Another category of symmetric algorithms is a hash function, which takes a variable length text and generates a fixed size output, also called hash or digest. A hash is a one-way function; the generated hash cannot be used to get the original plaintext. A hash is generally used for enforcing integrity.

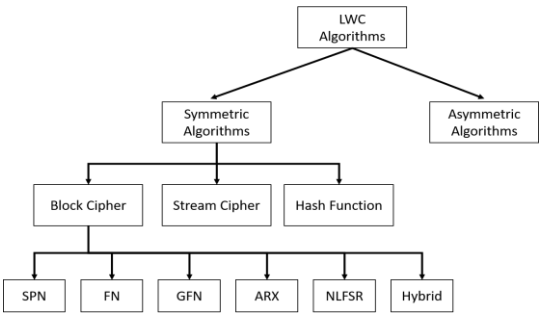

Fig. 2. Classification of LWC algorithms

A block cipher internally uses a specific structure for achieving security. In general, the different structures that can be used are Substitution-Permutation Network (SPN), Feistel Network (FN), General Feistel Network (GFN), Add-Rotate-XOR (ARX), Non-Linear Feedback Shift Register (NLFSR), and Hybrid structure [1]. An **SPN** processes the data by performing different operations using S-boxes (Substitution Boxes) and permutation tables. An **FN** divides the text into equal halves, and on one half, it applies diffusion. Also, at the beginning of each round, it swaps the two halves. A **GFN** divides the plaintext into sub-blocks and performs operations on the sub-blocks. It also applies a cyclic shift proportional to the number of created sub-blocks. **ARX** structure uses addition, rotation, and XOR operations for encryption and decryption. **NLFSR** uses the basic building blocks of a stream cipher. A block's current state is based on its previous state, which is a non-linear feedback value. Finally, **a hybrid** structure uses a combination of the structures mentioned above based on the application's requirements.

Various lightweight block ciphers that were developed and analyzed over the years are listed in Table 3, Table 4, and Table 5. Stream ciphers use structures like

Linear Feedback Shift Register (LFSR), Counter bits (CT), SHift Registers (SHR), Pseudo Random Number Generators (PRNGs), and Generalized NLFSR (GNLFSR). Various lightweight stream ciphers that were developed and analyzed over the years are listed in Table 3, Table 4, and Table 5. Various hash generation algorithms which were developed over the years are listed in these tables. All the lightweight block ciphers, stream ciphers, and hash algorithms can be categorized into different generations [2] based on the year in which the algorithm was published. These categories and associated algorithms are presented in Table 3, Table 4, and Table 5. The format of the algorithm names consists of the algorithm name, followed by the structure used by that algorithm and finally ending with the year in which that algorithm was developed.

Table 3. Early light-weight generation algorithms (before 2005)

Type of algorithm	Algorithm names
Block cipher	DES (FN 1977), DESX (FN 1984), KeeLoq (NLFSR 1985), GOST (FN 1989), IDEA (ARX 1991), TEA (FN 1994), XTEA (FN 1997), 3DES (FN 1998), AES (SPN 1998), XXTEA (FN 1998), Seed (FN 1999), Camellia (Hybrid 2000), KHAZAD (SPN 2000), NOEKEON (SPN 2000), ICEBERG (SPN 2004)
Stream cipher	A5/1 (LFSR 1987), Rabbit (CT 2003)
Hash Algorithm	-

Table 4. First light-weight generation algorithms (2005-2012)

Type of algorithm	Algorithm names
Block cipher	mCrypton (SPN 2005), HIGHT (Hybrid 2006), SEA (FN 2006), CLEFIA (GFN 2007), DESL (FN 2007), DESXL (FN 2007), KASUMI (FN 2007), PRESENT (SPN 2007), PUFFIN (SPN 2008), HB (Hybrid 2009), KATAN (NLFSR 2009), KTANTAN (NLFSR 2009), MIBS (FN 2009), PUFFIN-2 (SPN 2009), TWIS (GFN 2009), PRINTcipher (SPN 2010), EPCBC (SPN 2011), HB2 (Hybrid 2011), Klein (SPN 2011), LBlock (Hybrid 2011), LED (SPN 2011), Piccolo (GFN 2011), TWINE (GFN 2011), PICARO (SPN 2012), PRINCE (SPN 2012), SCREAM (SPN 2012)
Stream cipher	Salsa 20/r (ARX 2005), Grain (Hybrid 2005), Trivium (SHR 2005), Grain128a (Hybrid 2006), CHACHA (ARX 2008), MICKEY (Hybrid 2008), Sosemanuk (Hybrid 2008), Encoro80 (PRNG 2008), Encoro128 (PRNG 2009), SNOW-3G (Hybrid 2010), A2U2 (Hybrid 2011), Quavium (SHR 2012)
Hash Algorithm	KECCAK (2010), Lesamanta (2010), Quark (2010), PHOTON (2011), SPONGENT (2011), GLUON (2012)

Table 5. Second light-weight generation algorithms (2013 – present)

Type of algorithm	Algorithm names
Block cipher	ITUbee (FN 2013), LEA (Hybrid 2013), SIMON (FN 2013), SPECK (FN 2013), Zorro (SPN 2013), BEST-1 (ARX 2014), Chaskey (Hybrid 2014), Fantomas (Hybrid 2014), FeW (FN 2014), Halka (NLFSR 2014), HISEC (GFN 2014), I-PRESENT (SPN 2014), LAC (FN 2014), OLBCA (SPN 2014), PRESENT-GRP (Hybrid 2014), PRIDE (SPN 2014), RC5 (FN 2014), RECTANGLE (SPN 2014), Robin (Hybrid 2014), MIDORI (SPN 2015), PICO (SPN 2015), ANU (FN 2016), LAX (Hybrid 2016), MANTIS (SPN 2016), QTL (FN 2016), RoadRunneR (FN 2016), SPARX (Hybrid 2016), DLBCA (FN 2017), GIFT (SPN 2017), LiCi (FN 2017), SIT (Hybrid 2017), SFN (Hybrid 2018), ACE (ARX 2019), ASCON (SPN 2019), COMET (Hybrid 2019), ESTATE (SPN 2019), ForkAE (Hybrid 2019), GIFT-COFB (SPN 2019), Gimli (ARX 2019), ISAP (SPN 2019), LOTUS-LOCUS (Hybrid 2019), mixFeed (SPN 2019), Oribatida (FN 2019), Pyjamask (SPN 2019), SAEAES (SPN 2019), Saturnin (Hybrid 2019), SKINNY (SPN 2019), SPARKLE (Hybrid 2019), SPIX (Hybrid 2019), SpoC (Hybrid 2019), SUNDAE-GIFT (SPN 2019), WAGE (Hybrid 2019)
Stream cipher	WG-8 (Hybrid 2013), Sprout (Hybrid 2015), Fruit-v2 (Hybrid 2016), Plantlet (Hybrid 2016), Espresso (GNLFSR 2017), Lizard (NLFSR 2017)
Hash Algorithm	L-Hash (2013), Hash-One (2016), Neeva (2016)

The aforementioned lightweight block cipher algorithms can also be classified into ultra-lightweight, low-cost, and lightweight categories based on the measures mentioned in Section 2. This classification of existing algorithms can help researchers and manufacturers choose one or more ciphers for developing IoT devices or protocols. For classifying the lightweight block ciphers, results from various performance analysis papers were considered [1, 2, 8, 9, 12, 14-19]. Among the implementations on different architecture types and devices, the maximum values for GE, ROM, and RAM were considered for the classification. The performance of algorithms proposed in recent years that were mentioned in Table 3, Table 4, and Table 5, particularly in the second light-weight generation, was not evaluated on resource-constrained devices. Instead, they were evaluated on robust systems like Raspberry Pi, etc. So, those algorithms were not considered in this classification. There is a need to evaluate the performance of the latest algorithms on resource-constrained IoT devices.

Table 6. Classification of lightweight block ciphers

Category	Metric	Block ciphers
Ultra-lightweight	GE	KTANTAN
	ROM/RAM	PRIDE, LEA, ITUbee, Camellia, KASUMI, TEA, XTEA, NOEKEON, SEA, DESL, mCrypton, IDEA, MIBS
Low-cost	GE	KTANTAN, GOST, KATAN, PUFFIN-2, QTL, SIMON, LBlock, EPCBC, Piccolo, MIBS, SPECK, MIDORI, HISEC, RECTANGLE
	ROM/RAM	PRIDE, GOST, KATAN, PUFFIN-2, QTL, SIMON, LBlock, EPCBC, Piccolo, MIBS, SPECK, MIDORI, HISEC, RECTANGLE, SPECK, Piccolo, PRESENT-GRP, Zorro, TWINE
Lightweight	GE	KTANTAN, GOST, KATAN, PUFFIN-2, QTL, SIMON, LBlock, EPCBC, Piccolo, MIBS, SPECK, MIDORI, HISEC, RECTANGLE, PRESENT, TWINE, PUFFIN, Klein, DESL, I-PRESENT, NOEKEON, HIGHT, DESXL, LED, KASUMI, PRINCE, XTEA, TEA, mCrypton, SEA, LEA, ICEBERG, CLEFIA, Camellia, AES
	ROM/RAM	PRIDE, GOST, KATAN, PUFFIN-2, QTL, SIMON, LBlock, EPCBC, Piccolo, MIBS, SPECK, MIDORI, HISEC, RECTANGLE, SPECK, Piccolo, PRESENT-GRP, Zorro, TWINE, DES, DESX, CLEFIA, Robin, HB2, Klein, Fantomas, PRINTcipher, LED, SIMON, KATAN, LBlock, PRESENT, HIGHT, GOST, PRINCE, KTANTAN, DESXL, AES, HB

The classification of lightweight block ciphers is presented in Table 6. The algorithms in the table are mentioned in the order of metric values from low to high. For example, in the ultra-lightweight category, PRIDE has the lowest requirement for ROM/RAM, and MIBS has the highest requirement.

The following section presents an analysis of the ultra-lightweight algorithms identified in Table 6 based on the metrics given in Section 2.

4. Analysis of ultra-lightweight algorithms

This section analyses various ultra-lightweight algorithms identified in Section 3 concerning the metrics mentioned in Section 2.

Most of the IoT devices that upcoming innovators or micro-organizations create use insecure or no encryption techniques in their communications [12], which can lead to spoofing and traffic analysis attacks [3, 10]. There has yet to be a consensus on which encryption or authentication algorithms will be used for IoT devices or

networks to ensure security. Since 2015, NIST has been searching for suitable LWC algorithms. In 2019, it started a process for standardizing LWC algorithms for resource-constrained devices in IoT. In the second round, NIST announced 32 algorithms as candidates for final consideration [12]. Two features or characteristics of LWC algorithms, namely cost and performance, are achieved using simple round functions, simple keys, small block sizes, and simple key scheduling. The final feature, security, can be achieved using the six internal structures: SPN, FN, GFN, ARX, NLFSR, or hybrid [1]. According to a recent study conducted by NIST, all existing LWC algorithms must satisfy cost, security, and performance features. So, there is a need to analyze existing algorithms further or develop new LWC algorithms [3].

The only ultra-lightweight algorithm based on the GE metric is **KTANTAN**. The hardware implementation of KTANTAN requires 588 logic gates and is suitable for resource-constrained devices like RFID devices. KTANTAN supports a key of size 80 bits and blocks of 32, 48, and 64 bits size. The number of rounds is 254. The key is hardwired, which enables KTANTAN to have a low footprint. Recovery of a full 80-bit length key using a related key attack for KTANTAN was demonstrated, making it insecure. Also, the software implementation of KTANTAN requires more resources than other contemporary software-based ultra-lightweight algorithms. The ultra-lightweight algorithms based on the ROM/RAM metric values from low to high are PRIDE, LEA, ITUbee, Camellia, KASUMI, TEA, XTEA, NOEKEON, SEA, DESL, mCrypton, IDEA, and MIBS. **PRIDE** is the best ultra-lightweight, software-based implementation. It is an SPN-based algorithm that utilizes 20 rounds. It uses keys of 128 bits in length and a block size of 64 bits. It requires only 266 bytes of ROM. PRIDE is superior to many other algorithms regarding latency and energy consumption.

LEA (Lightweight block Encryption Algorithm) is a software-based algorithm developed based on ARX architecture. Encryption performed by LEA is very fast, using 128, 192, and 256-bit keys for 24, 28, and 32 rounds, respectively. The block size is 128 bits. LEA requires only 590 bytes of ROM and 32 bytes of RAM. LEA became insecure when the 128-bit key retrieval was demonstrated. **ITUbee** is a block cipher based on the FN structure. It does not contain a key schedule and uses round constants. The key size and block size is 80 bits. It requires a ROM size of 716 bytes. **Camellia** is a highly standardized software-based implementation. It is famous as it has similar processing functionality to AES. It uses the same block and key sizes as AES. It requires 1262 bytes of ROM and 12 bytes of RAM. **KASUMI** is a software-based implementation using an FN structure to encrypt the plaintext. It is widely used in GSM, GPRS, and UMTS. It uses a key size of 128 bits and a block size of 64 bits. It uses eight rounds for processing. It requires 1264 bytes of ROM and 24 bytes of RAM. TEA (Tiny Encryption Algorithm) is an efficient FN structure algorithm. It requires 1350 bytes of ROM and only 13 bytes of RAM. The key size, block size, and number of rounds in TEA are 128 bits, 64 bits, and 64, respectively. TEA has weaknesses like equivalent key attacks and its unsuitability as a hash.

XTEA (eXtended TEA or Block TEA) was developed to overcome the weaknesses of TEA. It uses a more complex key scheduling algorithm and can work

on arbitrary block sizes. XTEA requires 1400 bytes of ROM and 11 bytes of RAM. However, related-key attacks and other attacks have already been demonstrated on it which makes it insecure. **NOEKEON** is a SPN-based block cipher whose key and block size are 128 bits. It used 16 rounds for processing the data. NOEKEON requires 2780 bytes of ROM and only 34 bytes of RAM. **SEA** (Scalable Encryption Algorithm) is a FN-based algorithm that is used in resource-constrained devices. The main features of SEA are implementation in assembly language, on-the-fly derivation of keys, and efficient encryption and decryption. SEA requires 2800 bytes of ROM and only 24 bytes of RAM. **DESL** is a variant of DES that uses the same key and block sizes and also several rounds. DESL reduces the gate complexity by reducing the 8 S-boxes to only 1 S-box. DESL provides protection against linear, differential, and Davis-Murphy attacks. DESL requires 3100 bytes of ROM.

mCrypton (miniature Crypton) is an SPN-based block cipher that uses a small block size of 64 bits. It uses variable keys of sizes 64, 96, and 128 bits. It uses 13 rounds for processing the data. It is generally used in low-cost RFID tags and sensors. mCrypton uses 3100 bytes of ROM and 28 bytes of RAM. **IDEA** (International Data Encryption Algorithm) is an ARX-based algorithm that uses keys of size 128 bits and a block size of 64 bits. It uses 8.5 rounds for processing the data. It does not use any P-boxes or S-boxes. IDEA uses 3140 bytes of ROM and 23 bytes of RAM. **MIBS** is an FN-based block cipher that uses keys of sizes 64 and 80 bits. The block size is 64 bits, and it uses 32 rounds for processing the data. MIBS utilizes the S-box of mCrypton and key scheduling of PRESENT cipher. MIBS requires 3180 bytes of ROM and 29 bytes of RAM.

The software implementations of the ultra-lightweight algorithms mentioned above are compared based on the performance analysis performed on an 8-bit microcontroller. The result of the comparison is shown in Table 7.

Table 7. Software implementation of ultra-lightweight algorithms on an 8-bit microcontroller

No	Algorithm name	ROM (B)	RAM (B)	Latency (Cycles per 1 block)	Energy (mJ per 1 bit)	TP at 4 MHz (Kbps)	S/W Efficiency (Kbps / KB)
1	PRIDE	266	0	1514	6	169	635.33
2	LEA	-	-	-	-	-	-
3	ITUbee	716	0	2937	11.7	122.7	272.5
4	Camellia	1262	12	68260	256	8	6.33
5	KASUMI	1264	24	11939	47.6	21.4	16.93
6	TEA	1140	0	6299	34.3	40.8	35.78
7	XTEA	1246	0	19936	70	33.7	40
8	NOEKEON	364	32	23517	95.9	21.7	59.61
9	SEA	2132	0	41604	173.7	47	98.21
10	DESL	3098	0	8365	34.5	31.3	6.86
11	mCrypton	1076	28	22656	68	15.5	14.4
12	IDEA	836	232	22792	34.3	94.8	159.06
13	MIBS	-	-	-	-	-	-

TP in the table refers to throughput. It is to be noted that performance analysis of LEA and MIBS algorithms was not done using an 8-bit microcontroller. In the case of multiple versions of the same algorithm, the maximum values were considered for different metrics in the table. PRIDE and NOEKEON require the least

bit of ROM. XTEA, Camellia, and KASUMI use approximately the same amount of ROM. SEA and DESL require more ROM when compared to the other algorithms.

Camellia consumes the least amount of RAM, while IDEA consumes the highest amount of RAM. PRIDE and ITUbee have the least amount of latency. NOEKEON, IDEA, and mCrypton have approximately the same amount of latency. The Camellia algorithm incurs the highest latency among all the algorithms. PRIDE and ITUbee are the most energy-conserving algorithms. Whereas Camellia consumes the most energy, followed by the SEA algorithm.

PRIDE algorithm gives the best throughput, followed by the ITUbee algorithm. Camellia gives the worst throughput, followed by the mCrypton algorithm. PRIDE algorithm gives the highest software efficiency, followed by the ITUbee algorithm. Camellia, DESL, mCrypton, and KASUMI offer the least software efficiency. PRIDE and ITUbee offer the best performance when considering all the metrics, whereas Camellia and SEA have the worst. Although PRIDE and ITUbee are the best ultra-lightweight algorithms, attacks have already been demonstrated against them, making them insecure. Next, we discuss the various attacks on PRIDE, ITUbee, and IDEA algorithms. These best-performing software-based ultra-lightweight algorithms were selected based on the data provided in Table 7.

Dai and Chen [20] performed related-key differential attacks to find 8 2-round iterative characteristics in the key scheduling algorithm. Based on the 18-round related-key differentials and other observations in the linear layer, they presented an attack on the full PRIDE cipher. This attack was possible due to a weakness in the key schedule algorithm and due to the linear layer in PRIDE. Zhao et al. [21] identified 16 different 2-round iterative characteristics using differential attack. This was due to the weakness in the S-box and the linear layer of the PRIDE algorithm. Based on these observed differentials, they were able to launch a differential attack on the 18-round PRIDE cipher. No attack was reported by them on the full 20-round PRIDE cipher.

Yang et al. [22] used an automatic search method to find 56 iterative differential characteristics of PRIDE. Based on three of those characteristics, they built a 15-round differential and performed a differential attack on a 19-round PRIDE. Tezcan et al. claim that attacks on PRIDE done by authors previously didn't consider two properties, namely undistributed bits and differential factors. Due to this, even though the report attacks are correct, the time complexity of the attacks is much more than reported by the authors. Lallemand and Rasoolzadeh [23] proposed a single key differential attack on 18-round PRIDE. They claimed that the previous differential attacks were incorrect due to a miscomputation of the known bits. The authors claim that their attack is the first single key differential attack that was successful on the 18-round PRIDE cipher. Hou et al. [24] proposed a new differential fault analysis attack that does not require any information on plaintext and ciphertext. Their approach is a fully automated one that scans the assembly implementation of the PRIDE cipher and recovers the last round key. The advantage of this method is that it can be applied to with any input data.

Soleimany [25] identified several weaknesses in the round-reduced versions of the ITUbee cipher. The author shows the existence of a deterministic related-key

differential distinguisher for an 8-round version of the ITUbee cipher. These results exploit the flaws in the key schedule and round constants used in ITUbee. Fu et al. [26] proposed a novel differential fault attack in combination with traditional differential analysis on the ITUbee cipher. The authors found weaknesses in the cipher structure and the round function of the ITUbee cipher. They simulated their theoretical attack for recovering the encryption key. Kang et al. [27] came up with a new attack based on differential fault analysis and meet-in-the-middle attack. The attacks were based on the observations of the cipher's round function and its structure. The authors claim that they were able to overcome the countermeasures proposed in a previous work.

Meier [28] performed a differential analysis attack on a 2-round IDEA cipher. The author exploited the new arithmetic properties of the basic operations used in the round function of the IDEA cipher. Biham, Dunkelman and Keller [29] proposed an attack on the 6-round IDEA cipher by exploiting its weak key schedule algorithm. Their approach uses a combination of square-like techniques with linear cryptanalysis. Biham, Dunkelman and Keller [30] devised multiple attacks on reduced-round IDEA cipher. Authors performed a known plaintext 5-round attack, a related-key rectangle attack on 7-round IDEA, and a related key attack on 7.5-round IDEA cipher. These attacks were viable due to the linear key schedule of IDEA.

Clavier, Gierlichs and Verbauwhede [31] performed differential fault analysis on the IDEA cipher and were successful in retrieving 93 out of 128 key bits. It requires only 10 faults to be exploited. The attack was possible due to weakness within the structure of the IDEA cipher. Biryukov et al. [32] identified a large weak-key class of IDEA cipher. This was because the non-linear part is based on the multiplication with a chosen master key and due to the linear key schedule. So, there is a need to redesign the key schedule of IDEA. The summary of the security issues and attacks performed on PRIDE, ITUbee, and IDEA ciphers is listed in Table 8.

Table 8. Summary of security issues and attacks on PRIDE, ITUbee, and IDEA ciphers

Cipher	Type	NR	KS	Type of attack	Weakness
PRIDE	SPN	20	128	Related-key differential attack on 20 rounds	Key schedule algorithm and a linear layer of cipher
				Differential attack on 18 rounds	S-box and linear layer of cipher
				Automated differential attack on 19 rounds	S-box and linear layer of cipher
				Single key differential attack on 18 rounds	S-box and linear layer of cipher
				Differential fault analysis attack on 20 rounds	Cipher structure
ITUBEE	FN	20	80	Related-key differential attack on 8 rounds	Key schedule and round constants
				Differential fault attack and differential analysis on 20 rounds	Cipher structure and round function
				Differential fault analysis and meet-in-the-middle attack on 20 rounds	Cipher structure and round function
IDEA	ARX	8.5	128	Differential analysis attack on 2 rounds	Arithmetic properties of the basic operations used in the cipher
				Square-like attack and linear cryptanalysis on 6 rounds	Key schedule algorithm
				Related-key attack on 7.5 rounds	Linear key schedule
				Differential fault analysis attack for deducing 93 bits out of 128 in the key	Cipher structure
				Weak-key classes	Linear key schedule and multiplication operation

In the above table, NR represents several rounds in the cipher and KS represents the key size of the cipher in bits. The study of various attacks on the ciphers reveals that the current trend of attacking the ciphers uses differential fault analysis and hybrid attacks.

5. Conclusion

IoT amalgamates existing and new technologies with broad applications in many areas. The devices used for sensing and communication are often resource-constrained. Hence, there is a need to use existing LWC algorithms or develop new algorithms based on the application to provide security and privacy. This paper provides a brief survey of the existing LWC algorithms and a taxonomy that aids new researchers in gaining insight into them. This paper also categorizes the existing LWC algorithms into ultra-lightweight, low-cost, and lightweight algorithms, which is not seen in previous research. Finally, based on the analysis of LWC algorithm implementations, the best software-oriented algorithms in the ultra-lightweight category were identified as PRIDE, ITUbee, and IDEA. As these algorithms are still vulnerable to different attacks, the future research direction involves designing a new LWC algorithm that outperforms them and resists different attacks.

References

1. Thakor, V. A., M. A. Razzaque, M. R. A. Khandaker. Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities. – IEEE Access: Practical Innovations, Open Solutions, Vol. **9**, 2021, pp. 28177-28193.
2. Hatzivasilis, G., K. Fysarakis, I. Papaefstathiou, C. Manifavas. A Review of Lightweight Block Ciphers. – Journal of Cryptographic Engineering, Vol. **8**, 2018, No 2, pp. 141-184.
3. Dhanda, S. S., B. Singh, P. Jindal. Lightweight Cryptography: A Solution to Secure IoT. – Wireless Personal Communications. 2020, pp. 1947-1980.
4. Alazzam, H., O. AbuAlghanam, Q. M. Al-zoubi, A. Alsmady, E. Alhenawi. A New Network Digital Forensics Approach for Internet of Things Environment Based on Binary Owl Optimizer. – Cybernetics and Information Technologies, Vol. **22**, 2022, No 3, pp. 146-160.
5. Khan, M. N., A. Rao, S. Camtepe. Lightweight Cryptographic Protocols for IoT-Constrained Devices: A Survey. – IEEE Internet of Things Journal, Vol. **8**, 2021, No 6, pp. 4132-4156.
6. Sallam, S., B. D. Beheshti. A Survey on Lightweight Cryptographic Algorithms. – In: Proc. of IEEE Region 10 Conference (TENCON'18), IEEE, 2018.
7. Julian Okello, W., Q. Liu, F. Ali Siddiqui, C. Zhang. A Survey of the Current State of Lightweight Cryptography for the Internet of Things. – In: Proc. of International Conference on Computer, Information and Telecommunication Systems (CITS'17), 2017, pp. 292-296.
8. Nayancy, D. S., S. Chakraborty. A Survey on Implementation of Lightweight Block Ciphers for Resource Constraints Devices. – Journal of Discrete Mathematical Sciences & Cryptography, Vol. **2020**, pp. 1-22.
9. Singh, S., P. K. Sharma, S. Y. Moon, J. H. Park. Advanced Lightweight Encryption Algorithms for IoT Devices: Survey, Challenges and Solutions. – Journal of Ambient Intelligence and Humanized Computing, 2017.

10. M o u s a v i, S. K., A. G h a f f a r i, S. B e s h a r a t, H. A f s h a r i. Security of Internet of Things Based on Cryptographic Algorithms: A Survey. – Wireless Networks, Vol. **27**, 2021, No 2, pp. 1515-1555.
11. M u s t a f a, G., R. A s h r a f, M. M i r z a, A. J a m i l, A. M u h a m m a d. A Review of Data Security and Cryptographic Techniques in IoT Based Devices. – In: Proc. of 2nd International Conference on Future Networks and Distributed Systems, ACM, New York, NY, USA, 2018.
12. F o t o v v a t, A., G. M. E. R a h m a n, S. S. V e d a e i, K. A. W a h i d. Comparative Performance Analysis of Lightweight Cryptography Algorithms for IoT Sensor Nodes. – IEEE Internet of Things Journal, Vol. **8**, 2021, No 10, pp. 8279-8290.
13. S u d h a, K. S., N. J e y a n t h i. A Review on Privacy Requirements and Application Layer Security in Internet of Things (IoT). – Cybernetics and Information Technologies, Vol. **21**, 2021, No 3, pp. 50-72.
14. B h a r d w a j, I., A. K u m a r, M. B a n s a l. A Review on Lightweight Cryptography Algorithms for Data Security and Authentication in IoTs. – In: Proc. of 4th International Conference on Signal Processing, Computing and Control (ISPCC'17), IEEE, 2017.
15. S u r e n d r a n, S., A. N a s s e f, B. D. B e h e s h t i. A Survey of Cryptographic Algorithms for IoT Devices. – In: Proc. of IEEE Long Island Systems, Applications, and Technology Conference (LISAT'18), IEEE, 2018.
16. G o y a l, T. K., V. S a h u l a, D. K u m a w a t. Energy Efficient Lightweight Cryptography Algorithms for IoT Devices. – IETE Journal of Research, Vol. **2019**, pp. 1-14.
17. D u t t a, I. K., B. G h o s h, M. B a y o u m i. Lightweight Cryptography for Internet of Insecure Things: A Survey. – In: Proc. of 9th IEEE Annual Computing and Communication Workshop and Conference (CCWC'19), IEEE, 2019.
18. G u n a t h i l a k e, N. A., W. J. B u c h a n a n, R. A s i f. Next Generation Lightweight Cryptography for Smart IoT Devices: Implementation, Challenges and Applications. – In: Proc. of 5th IEEE World Forum on Internet of Things (WF-IoT'19), IEEE, 2019.
19. G u n a t h i l a k e, N. A., A. A l-D u b a i, W. J. B u c h a n a. Recent Advances and Trends in Lightweight Cryptography for IoT Security. – In: Proc. of 16th International Conference on Network and Service Management (CNSM'20), IEEE, 2020.
20. D a i, Y., S. C h e n. Cryptanalysis of Full PRIDE Block Cipher. – Science China Information Sciences, Vol. **60**, 2017, No 5.
21. Z h a o, J., X. W a n g, M. W a n g, X. D o n g. Differential Analysis on Block Cipher PRIDE. – In: Cryptology ePrint Archive. Vol. **2014**.
22. Y a n g, Q., L. H u, S. S u n, K. Q i a o, L. S o n g, J. S h a n, X. M a. Improved Differential Analysis of Block Cipher PRIDE. Information Security Practice and Experience. – In: Lecture Notes in Computer Science. Cham, Springer International Publishing, 2015, pp. 209-219.
23. L a l l e m a n d, V., S. R a s o o l z a d e h. Differential Cryptanalysis of 18-Round PRIDE. Lecture Notes in Computer Science. – In: Lecture Notes in Computer Science. Cham, Springer International Publishing, 2017, pp. 126-146.
24. H o u, X., J. B r e i e r, F. Z h a n g, Y. L i u. Fully Automated Differential Fault Analysis on Software Implementations of Block Ciphers. – In: IACR Transactions on Cryptographic Hardware and Embedded Systems. Vol. **2019**. pp. 1-29.
25. S o l e i m a n y, H. Self-Similarity Cryptanalysis of the Block Cipher ITUbee. – IET Information Security, Vol. **9**, 2015, No 3, pp. 179-184.
26. F u, S., G. X u, J. P a n, Z. W a n g, A. W a n g. Differential Fault Attack on ITUbee Block Cipher. – ACM Transactions on Embedded Computing Systems, Vol. **16**, 2017, No 2, pp. 1-10.
27. K a n g, Y., Q. Y u, L. Q i n, G. Z h a n g. Meet-in-the-Middle Differential Fault Analysis on ITUbee Block Cipher. – Symmetry, Vol. **15**, 2023, No 6, p. 1196.
28. M e i e r, W. On the Security of the IDEA Block Cipher. Advances in Cryptology. – In: EUROCRYPT 1993. Berlin, Heidelberg, Springer, 2007, pp. 371-385.
29. B i h a m, E., O. D u n k e l m a n, N. K e l l e r. A New Attack on 6-Round IDEA. Fast Software Encryption. – In: Lecture Notes in Computer Science. Berlin, Heidelberg, Springer, 2007, pp. 211-224.
30. B i h a m, E., O. D u n k e l m a n, N. K e l l e r. New Cryptanalytic Results on IDEA. Advances in Cryptology – ASIACRYPT 2006. – In: Lecture Notes in Computer Science. Berlin, Heidelberg, Springer, 2006, pp. 412-427.

31. C l a v i e r, C., B. G i e r l i c h s, I. V e r b a u w h e d e. Fault Analysis Study of IDEA. Topics in Cryptology – CT-RSA 2008. – In: Lecture Notes in Computer Science. Berlin, Heidelberg, Springer, 2008, pp. 274-287.
32. B i r y u k o v, A., J. N a k a h a r a, B. J. P r e n e e l, J. V a n d e w a l l e. New Weak-Key Classes of IDEA. Information and Communications Security. – In: Lecture Notes in Computer Science. Berlin, Heidelberg, Springer, 2002, pp. 315-326.

Received: 23.10.2023; Second Version: 03.01.2024; Accepted: 15.01.2024 (fast track)