Journal of King Saud University – Computer and Information Sciences 35 (2023) 101550

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Systematic literature review: Trend analysis on the design of lightweight block cipher

Abdul Alif Zakaria [a,c,*], A.H. Azni [a,b,*], Farida Ridzuan [a,b], Nur Hafiza Zakaria [a], Maslina Daud [c]

[a] Faculty of Science and Technology, Universiti Sains Islam Malaysia, Nilai 71800, Negeri Sembilan, Malaysia
[b] CyberSecurity and System Research Unit, Faculty of Science and Technology, Universiti Sains Islam Malaysia, Nilai 71800, Negeri Sembilan, Malaysia
[c] CyberSecurity Malaysia, Cyberjaya 63000, Selangor, Malaysia

ARTICLE INFO

Article history:
Received 1 February 2023
Revised 28 March 2023
Accepted 2 April 2023
Available online 15 April 2023

Keywords:
Confusion
Cryptanalysis
Diffusion
Evolution of algorithm
Lightweight cryptography
Permutation
Secure design components
Substitution

ABSTRACT

Lightweight block ciphers have become a standard for security protections on IoT devices. Advanced technology is required to secure the data, thus encryption is the method that can provide information security. From previous studies, comparisons of lightweight algorithms in various literature focus on their performance and implementation. However, a lack of analysis has been done on the relationship between the algorithm components and their security strength. This information is crucial for developers in designing secure algorithms. In this paper, a comprehensive systematic literature review on 101 existing lightweight algorithms is presented. This review focuses on the security aspect of lightweight algorithms that cover the identification of secure design components based on substitution and permutation. Security analysis and the evolution of lightweight algorithms are also presented. This research includes the results and discussions to observe the selections of substitution and permutation functions to analyse their impact on the security strength. Recommendations from the developer's insight on methods and considerations for designing an algorithm are also presented. Findings from the research indicate that various techniques can be used to develop a secure algorithm. Most importantly, an algorithm must be provided with confusion and diffusion properties in the design to ensure sufficient security.

Contents

* Corresponding authors at: Faculty of Science and Technology, Universiti Sains Islam Malaysia, Nilai 71800, Negeri Sembilan, Malaysia.
E-mail addresses: alif@cybersecurity.my (A.A. Zakaria), ahazni@usim.edu.my (A.H. Azni), farida@usim.edu.my (F. Ridzuan), mzhafiza@usim.edu.my (N.H. Zakaria), maslina@cybersecurity.my (M. Daud).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2023.04.003
1319-1578/

1. Introduction

The internet of things (IoT) is an in-demand technology adopted in various applications that include sensors, actuators, and monitors that are connected to the Internet. With the continual advancement in technologies, there is an increased demand for IoT applications. It is important to understand the impact of IoT has caused more devices to be connected to the Internet, resulting in huge data exchanges (Saraiva et al., 2019a). Therefore, encryption is the method that can ensure the security of data exchanges in IoT systems.

Due to the lightweight nature of the IoT devices such as the sensors, RFID tags, and smartcards, conventional block cipher schemes such as AES which is relatively heavy in the sense of operations is not suitable for implementation (Nayancy and Chakraborty, 2020). In comparison with the conventional algorithm, a lightweight block cipher is typically simpler than the conventional block cipher with smaller block and key sizes that consist of the following properties. Firstly, applications for constrained devices are unlikely to require the encryption of large amounts of data. Secondly, attackers are lack of data and computing ability, which means lightweight ciphers only need to achieve moderate security (Pei et al., 2018a). Lastly, lightweight ciphers are usually implemented in hardware environments and small parts of them are implemented on software platforms (Mohd et al., 2015a). Hence, a lightweight block cipher is suitable for resource-constrained devices with limited components.

Lightweight block ciphers have since become a default standard when considering security protections on IoT devices (Chen et al., 2020a). The concept of a lightweight algorithm has attracted tremendous attention from academia, industry and, government. Development of lightweight algorithms has begun as early as the 1990's with the introduction of IDEA (Lai and Massey, 1991), Blowfish (Schneier, 1993), and TEA (Wheeler and Needham, 1994). Until now, hundreds of lightweight algorithms were proposed including the recent LBC-IoT (Ramadan et al., 2021), improved SM4 (Chen et al., 2021), and LAO-3D (Zakaria et al., 2022). Designs of lightweight block ciphers have been influenced by existing algorithms such as AES and PRESENT despite their age of development (Girija et al., 2020a). Components from both mentioned algorithms are still relevant and can be applied to construct new lightweight block ciphers.

Based on the research conducted on literature review papers of lightweight block ciphers, the authors only focused on the performance and implementation of the lightweight ciphers (Sevin and Mohammed, 2021a; Dhanda et al., 2020a; Singh et al., 2019a). However, the relationship between the design components and the security strength of the block ciphers has not been analysed in detail thus leaving a research gap to be filled in this paper. Algorithm design components have a significant impact on the strength of lightweight block ciphers. This information is important for the developers in getting the idea of designing secure lightweight block ciphers which have not been discussed in previous literature.

A lightweight block cipher should offer confusion and diffusion characteristics in its design components to provide sufficient security to the algorithm (Alghafis et al., 2021a). Confusion obscures the relationship between the plaintext and ciphertext with substitution, while diffusion spreads the plaintext statistics through the ciphertext using permutation. Substitution and permutation components of the lightweight block ciphers will be the main discussion in this paper. A total of 101 lightweight algorithms are presented to observe the selections of substitution and permutation functions in these block ciphers to analyse their impact on the security strength.

In this research, an in-depth study has been conducted on lightweight block ciphers. The contributions obtained from the research are listed as follows:

- Provide the details of existing algorithms to identify secure design components of lightweight block ciphers.
- Provide the substitution components of existing light-weight block ciphers to analyze their confusion property.
- Provide the permutation components of existing light-weight block ciphers to analyze their diffusion property.
- Provide the security analysis conducted on existing lightweight block ciphers that includes cryptanalysis to analyze their security strength.
- Provide the evolution of lightweight block ciphers that includes encryption algorithm and key schedule algorithm to highlight the evolvement of their implemented cryptographic components.

The organization of the paper is structured as follows. Section 2 defines the methodology used for the systematic review process in the research. Section 3 presents the results from the analysis of existing lightweight block ciphers and their discussions. Section 4 provides the recommendations from the research works. Lastly, Section 5 concludes the research work produced in this paper.

2. Methodology

Literature resources in this article were collected from multiple digital databases being actively used by researchers that include SpringerLink (Springer), IEEE Xplore (IEEE), ScienceDirect (Elsevier), Association for Computing Machinery (ACM), International

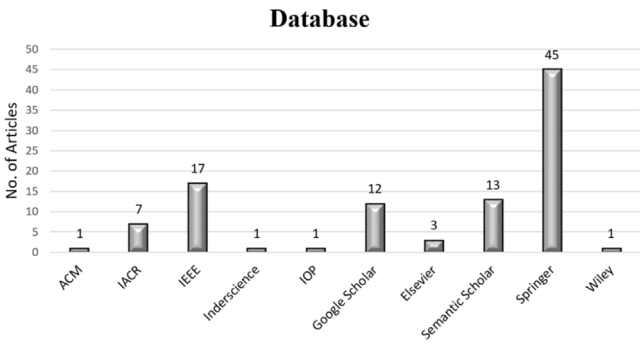

Fig. 1. Resources database.

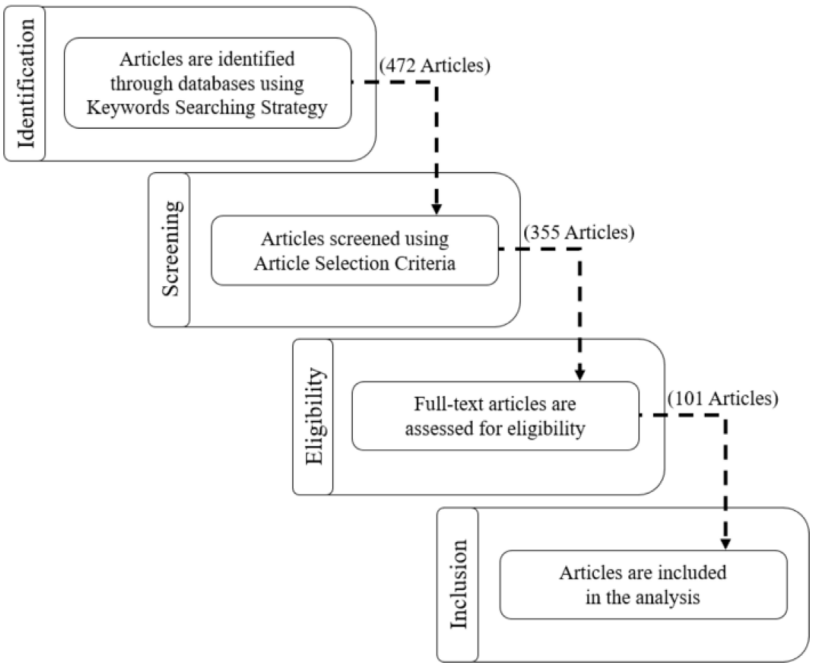

Fig. 2. Systematic literature review analysis flow diagram.

Table 1
Keywords searching strategy.

Keywords
"lightweight" AND "block cipher*" OR "block cipher*"

Association for Cryptologic Research (IACR), Semantic Scholar, Wiley Online Library (Wiley), IOP Publishing (IOP), Inderscience Publishers (Inderscience), and Google Scholar as shown in Fig. 1.

Initially, 472 articles that specifically proposed block ciphers were retrieved from the databases. The articles were reduced to 101 block cipher articles that fall under the category of a lightweight algorithm. Since this paper focuses on an in-depth analysis of lightweight block ciphers, research on algorithms between 1990 and 2022 was conducted to observe the overall insight of cryptographic design components from the early era of lightweight algorithms to the present. The research shows that many old algorithms are still being referred to in recent lightweight block ciphers. It demonstrates the importance of these old designs despite their age of development, thus contributing to ongoing research in the field of cryptography.

The technique used in the systematic literature review adopts the Preferred Reporting Items for Systematic Reviews and Meta-Analysis (PRISMA) (Moher et al., 2009). The review methodology for this work involves four steps that consist of identification, screening, eligibility, and inclusion as depicted in Fig. 2.

The basis of the identification process makes use of the keywords searching strategy as shown in Table 1 to retrieve articles from the database. Keywords searching strategy guides researchers in searching the main information used to describe the research topic. Without the right keywords, it might be difficult to find appropriate articles needed for the literature. Using the keywords searching strategy, 472 articles were retrieved from multiple databases.

Next, the screening process identifies the suitability of literature resources to be included in the survey based on the article selection criteria listed in Table 2. With hundreds of articles found from the database, it is crucial to select significant resources that are useful to the researchers. Type of article, language, the domain of article, cryptographic primitive, block size, key size, and type of encryption are important criteria in selecting the articles. The criteria specify the research requirements so that researchers will not deviate from their research scope. These criteria will guide researchers in structuring the output of the research from the input obtained from the articles. During the screening process, 355 articles were filtered from the resources.

The eligibility process retrieves the full-text articles that have been identified in the previous step. From the 355 identified resources, 101 articles were successfully assessed and downloaded that are used in the next stage of the systematic literature review methodology.

Lastly, the inclusion process gathers all of the assessed articles to be included in the analysis of lightweight block cipher. The sys-

Table 2
Article inclusion and exclusion criteria.

Criteria	Inclusion	Exclusion
Type of Article	Journal and conference proceeding.	Other sources (e.g., PowerPoint slides, thesis, and patent).
Language	English	Non-English (e.g., Chinese, Russian, and Japanese languages).
Domain	Cryptographic algorithm	Other than cryptographic algorithm (e.g., Cryptographic review, attack, and implementation).
Cryptographic Primitive	Block cipher	Other than block cipher (e.g., stream cipher, asymmetric cryptographic, hash function, and random bit/number generator).
Block Size	64-bit and lower.	Larger than 64-bit.
Key Size	128-bit and lower.	Larger than 128-bit.
Encryption Type	Text encryption	Other than text encryption (e.g., image, audio, and video).

Table 3
Security Evaluation criteria of lightweight block cipher.

Criteria	Requirement		
	NESSIE (2000)	NIST (2015)	MySEAL (2017)
Key size	At least 128-bit	At least 112-bit	At least 80-bit
Block size	64-bit	64-bit	At least 64-bit
Analysis	Generic attacks Side-channel attacks	Fault attacks Side-channel attacks Side-channel attacks	Linear cryptanalysis Differential cryptanalysis NIST statistical tests

tematic literature review methodology would help researchers to conduct their research by finding accurate resources. Without using this methodology, researchers may have to read and review thousands of articles with no clear direction.

3. Result and discussion

A total of 101 articles on lightweight block ciphers are included in this research. The extensive analysis of the lightweight algorithms indicates the thoroughness of the conducted research. Results from the analysis show the relationship between cryptographic algorithm design components and the security strength of the block ciphers which have not been technically described in detail before. Therefore, findings from the research are crucial, especially for algorithm developers.

3.1. Lightweight block cipher

Over the past years, several lightweight cryptography projects have been conducted to develop guidelines, recommendations, and standards for lightweight algorithms. The design, security, and performance criteria of lightweight algorithms were provided in each project as a baseline for the algorithm selection criteria. Between 2000 and 2003, the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) is a European project established in search of cryptographic primitives to be recommended to the relevant standardization bodies within Europe (Preneel, 2002). In 2015, the National Institute of Standards and Technology (NIST) has initiated the lightweight cryptography project to explore the needs of the lightweight algorithm in constrained environments which is expected to be completed in 2022 (Turan et al., 2021). Between 2016 and 2020, the National Trusted Cryptographic Algorithm List (MySEAL) project is designed to build a portfolio for the national trusted cryptographic algorithms to be implemented by the Malaysian government and citizens (CyberSecurity Malaysia, 2021). From the three projects, the security evaluation criteria for lightweight block cipher were identified as shown in Table 3 as a reference for developers in designing their cryptographic algorithm.

For the first contribution of the paper, existing lightweight algorithms were analysed to identify secure design components of lightweight block cipher. The properties of the selected algorithms are determined based on the article selection criteria from Table 2 that maps to the security evaluation criteria of lightweight block cipher in Table 3. Component details of lightweight block ciphers that include the block size, key size, structure, number of rounds, type of S-box, cryptographic functions, and their reference algorithm are presented in Table 4–7. Besides the presented algorithm components, an in-depth analysis on the substitution and permutation components implemented by the block ciphers is conducted and their security analysis is discussed in the following sections.

From the reviewed articles, confusion and diffusion are the properties introduced by Claude Shannon that must be associated with lightweight algorithms (Shannon, 1949a). Confusion indicates how changing one bit of the key affects the ciphertexts. A common element for achieving confusion is the substitution function. Meanwhile, diffusion indicates how changing one bit of the plaintext affects the ciphertext for the same key. A simple diffusion element is the bit permutation function. Both confusion and diffusion cannot provide security by themselves alone. The idea is to implement confusion and diffusion elements to develop a secure cryptographic algorithm. This research categorized each of the analysed secure design components of lightweight block cipher from Table 4–7 into confusion and diffusion categories as shown in Fig. 3.

Among the identified secure design components, the most basic mathematical operation used in almost every algorithm is the XOR component that performs bitwise exclusive-or function between a fixed bit input and produces the same length bit output (Adams, 1997a). There are various implementations of the XOR function to increase the diffusion property of a lightweight algorithm. The most implemented XOR function is add round key that operate XOR function between the cipher and the round key. In mixing layer function, the method executes XOR operation on each cipher state bit in sequence (Izadi et al., 2009). Similarly, the mix XOR layer operates a linear transformation that applies XOR operation among cipher states (Li et al., 2018). Counter XOR function is an XOR operation on the round counter and cipher state (Yeoh et al., 2020). L-box computation is a diffusion table that is arranged using XOR and XNOR gate operations (Biswas et al., 2020a). For constant XOR or sometimes called add constants process, the cipher state is XOR with the round constant where the values are obtained either from the table, computation, or round number (Jha et al., 2020).

Multiplication function is another mathematical operation implemented in block ciphers. In modulo multiplication, two numbers are multiplied and the modulus function is operated to obtain the result (Lai and Massey, 1991). Mix nibble performs similarly to the mix column function in AES but operates in the form of nibble (Gong et al., 2011). For the mix column, the multiplication between the diffusion matrix and the cipher state array is performed over $GF(2^4)$ (Li et al., 2018). Besides that, the mix row is a multiplication operation of a given matrix with the cipher state in the finite field $GF(2^4)$ (Liu et al., 2019a).

Matrix operation is also being implemented in cryptographic algorithms to achieve diffusion. Matrix multiplication is constructed based on the parallel application of a simple involution binary matrix multiplication (Standaert et al., 2004). In Maximum Distance Separable (MDS), the matrix applies a quasi-involutive Feistel-MDS transformation on each plane of the cipher cube with a given MDS code (Berger et al., 2015). Meanwhile, for matrix transformation, the matrix is transformed into arrays of cipher state bits (Usman et al., 2017).

Diffusion property in lightweight block ciphers can also be achieved using the transformation technique. Pseudo–Hadamard Transform is an unorthodox linear transformation component that allows the cipher rapid diffusion of small changes in the plaintext or the key over the resulting ciphertext (Massey, 1993). The column-to-Row Transposition is a simple function that transforms a cipher state column to a row representation (Lim and Korkishko, 2005). For Feistel Transformation, the component makes use of XOR and rotation functions in a Feistel-like manner (Zhang et al., 2015). In the reverse order transformation component, the final round output is arranged in reverse order to obtain the ciphertext (Chen et al., 2021).

Permutation is one of the most implemented diffusion components in cryptographic algorithms by developers. An application of this component is called keyed permutation that

Table 4
Lightweight block cipher components.

No.	Algorithm	Block Size (Bits)	Key Size (Bits)	Structure	Rounds	S-box Bits (Input & Output)	Function		Based on Existing Algorithm	Reference
							Encryption Algorithm	Key Schedule Algorithm		
1	IDEA	64	128	ARX	8.5	None	1. XOR 2. Modulo Addition 3. Modulo Multiplication	1. Key Partition 2. Rotation	None	(Lai and Massey, 1991)
2	Blowfish	64	32–448	FN	16	In:8 Out:32	1. Substitution 2. Modulo Addition 3. XOR	1. Permutation 2. Substitution 3. XOR	None	(Schneier, 1993)
3	SAFER	64	64	SPN	Variable	None	1. XOR 2. Byte Addition 3. Permutation< 4. Pseudo–Hadamard Transform	1. Rotation 2. Modulo Addition	None	(Massey, 1993)
4	RC5	32/64/ 128	0 to 2040	ARX	0–255	None	1. Modulo Addition 2. Rotation 3. XOR	1. Word Conversion 2. Initialize Array 3. Key Mixing	None	(Rivest, 1994)
5	TEA	64	128	FN	64	None	1. XOR 2. Modulo Addition	1. Modulo Addition	None	(Wheeler and Needham, 1994)
6	CAST	64/128	128/160/192/224/256/varies	FN	12/16/48	In:32 Out:32	1. Substitution 2. Rotation 3. Modulo Addition 4. Modulo Subtraction 5.XOR	1. Modulo Addition 2. Modulo Subtraction 3. XOR 4. Shift	None	(Adams, 1997a)
7	MISTY	64	128	FN	8	None	1. FO 2. FI 3. FL	1. FI	None	(Matsui, 1997)
8	SKIPJACK	64	80	FN	32	In:8 Out:8	1. Permutation 2. XOR 3. Round counter	1. Rotation	None	(NSA, 1998a)
9	KHAZAD	64	128	SPN	8	In:8 Out:8	1. Non-linear Layer 2. Linear Diffusion Layer 3. Add Round Key	1. Constant XOR	None	(Barreto and Rijmen, 2000)
10	ICEBERG	64	128	SPN	16	In:4 Out:4 & In:8 Out:8	1. Substitution 2. Permutation 3. Add Round Key 4. Matrix Multiplication	1. Key Expansion 2. Shift 3. Substitution 4. Permutation	None	(Standaert et al., 2004)
11	mCrypton	64	64/96/ 128	SPN	12	In:4 Out:4	1. Substitution 2. Permutation 3. Column-to-Row Transposition 4. Add Round Key	1. Substitution 2. Rotation	None	(Lim and Korkishko, 2005)
12	HIGHT	64	128	ARX	32	None	1. Initial Transformation 2. Round Function 3. Final Transformation	1. Whitening Key Generation 2. Subkey Generation	None	(Hong et al., 2006)
13	SEA	48/96/ 144	48/96/ 144	FN	Variable	In:3 Out:3	1. XOR 2. Substitution 3. Word Rotation 4. Rotation 5. Modulo Addition	1. XOR 2. Substitution 3. Word Rotation 4. Rotation 5. Modulo Addition	None	(Standaert et al., 2006)
14	DESL	64	56	FN	16	In:6 Out:4	1. Add Round Key 2. Substitution 3. Permutation	1. Permutation 2. Rotation	DES (Component)	(Leander et al., 2007)

(*continued on next page*)

Table 4 (*continued*)

No.	Algorithm	Block Size (Bits)	Key Size (Bits)	Structure	Rounds	S-box Bits (Input & Output)	Function		Based on Existing Algorithm	Reference
							Encryption Algorithm	Key Schedule Algorithm		
15	PRESENT	64	80/128	SPN	31	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation	1. Rotation 2. Substitution 3. Counter XOR	None	(Bogdanov et al., 2007)
16	PUFFIN	64	128	SPN	32	In:4 Out:4	1. Substitution 2. Add Round Key 3. Permutation	1. Permutation 2. Bit Inversion	ICEBERG (S-box)	(Cheng et al., 2008)
17	KATAN	32/48/ 64	80	NLFSR	254	None	1. LFSR	1. LFSR	Trivium (Component)	(De Cannière et al., 2009)
18	KTANTAN	32/48/ 64	80	NLFSR	254	None	1. LFSR	1. LFSR	Trivium (Component)	(De Cannière et al., 2009)
19	MIBS	64	64/80	FN	32	In:4 Out:4	1. Add Round Key 2. Substitution 3. Mixing 4. Permutation	1. Rotation 2. Substitution 3. Counter XOR	mCrypton (S-box) & PRESENT (Key Schedule)	(Izadi et al., 2009)
20	PUFFIN2	64	80	SPN	34	In:4 Out:4	1. Substitution 2. Add Round Key 3. Permutation	1. Substitution 2. Permutation 3. Position Selection	ICEBERG (S-box)	(Wang and Heys, 2009)
21	Hummingbird	16	256 + 80 Internal State	Hybrid	4	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation	None	None	(Engels et al., 2010)
22	Improved GOST	64	256	FN	32	In:4 Out:4	1. Modulo Addition 2. Substitution 3. Rotation	None	PRESENT (S-box)	(Poschmann et al., 2010)
23	PRINTcipher	48/96	80/160	SPN	48/96	In:3 Out:3	1. Add Round Key 2. Linear Diffusion 3. Round Counter 4. Keyed Permutation 5. Substitution	1. Keyed Permutation	None	(Knudsen et al., 2010)
24	EPCBC	48/96	96	SPN	32	In:4 Out:4	1. Substitution 2. Permutation	1. F-function 2. Add Round Key	PRESENT (Component & S-box)	(Yap et al., 2011)
25	Hummingbird-2	16	128 + 128 Internal State + 64 IV	Hybrid	4	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation	None	None	(Engels et al., 2011)
26	KLEIN	64	64/80/ 96	SPN	12/16/ 20	In:4 Out:4	1. Sub Nibble 2. Rotate Nibble 3. Mix Nibble	1. Shift 2. Feistel Transformation 3. XOR	AES (Component)	(Gong et al., 2011)
27	LBlock	64	80	FN	32	In:4 Out:4	1. XOR 2. Substitution 3. Rotation 4. Word Permutation	1. Rotation 2. Substitution 3. Constant XOR	None	(Wu and Zhang, 2011)

Table 5
Lightweight Block Cipher Components (Continued).

No.	Algorithm	Block Size (Bits)	Key Size (Bits)	Structure	Rounds	S-box Bits (Input & Output)	Function		Based on Existing Algorithm	Reference
							Encryption Algorithm	Key Schedule Algorithm		
28	LED	64	64/128	SPN	32/48	In:4 Out:4	1. Add Round Key 2. Constant XOR 3. Sub Cell 4. Shift Row 5. Mix Column	None	AES (Component) & PRESENT (S-box)	(Guo et al., 2011)
29	Piccolo	64	80/128	GFN	25/31	In:4 Out:4	1. Whitening Key XOR 2. F-function 3. Permutation	1. Constant Values	None	(Shibutani et al., 2011)
30	TWINE	64	80/128	GFN	32/36	In:4 Out:4	1. Substitution 2. Add Round Key 3. Block Shuffle	1. Counter XOR	None	(Suzaki et al., 2011)
31	PRINCE	64	128	SPN	12	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation 4. Constant XOR	None	None	(Borghoff et al., 2012)
32	Improved IDEA	64	128	ARX	6.5	None	1. XOR 2. Modulo Addition	1. XOR 2. Rotation 3. Modulo Addition	IDEA (Component) & MESH (Key Schedule)	(Lerman et al., 2013)
33	Improved LBlock	64	80	FN	32	In:4 Out:4	1. XOR 2. Substitution 3. Rotation 4. Permutation	1. Rotation 2. Substitution 3. Constant XOR	LBlock (Component, S-box & Key Schedule)	(Al-Dabbagh and Al-Shaikhli, 2013)
34	BEST-1	64	128	ARX	12	None	1. Initial Transformation 2. Round Function 3. Final Transformation	1. Transformation Key Generation 2. Subkey Generation	None	(John, 2014a)
35	ESF	64	80	FN	31	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation3. Counter XOR	1. Shift 2. Substitution	LBlock (Component); PRESENT (Component & Key Schedule) & Serpent (S-box)	(Liu et al., 2014a)
36	Halka	64	80	NLFSR	24	In:8 Out:8	1. Add Round Key 2. Substitution 3. Permutation	1. Rotation 2. Substitution 3. Counter XOR	PRESENT (Key Schedule) & AES (S-box)	(Das, 2014a)
37	HISEC	64	80	GFN	15	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation 4. Rotation 5. XOR	1. Substitution 2. Rotation	PRESENT (Component)	(Al-Dabbagh et al., 2014)
38	Kasumi	64	128	FN	8	In:7 Out:7 & In:9 Out:9	1. Function FL 2. Function FO 3. Function FI	1. KL 2. KO 3. KI	None	(ETSI, 2014a)
39	Khudra	64	80	FN	18	In:4 Out:4	1. F-function 2. Feistel Permutation	1. Round Constant 2. XOR	PRESENT (S-box)	(Kolay and Mukhopadhyay, 2014)
40	OLBCA	64	80	GFN	22	In:4 Out:4	1. F-function 2. Rotation 3. XOR 4. Permutation	1. Substitution 2. Rotation	None	(Al-Dabbagh and Al-Shaikhli, 2014)
41	PRIDE	64	128	SPN	20	In:4 Out:4	1. Add Round Key	None	None	(Albrecht et al.,

(*continued on next page*)

Table 5 (*continued*)

No.	Algorithm	Block Size (Bits)	Key Size (Bits)	Structure	Rounds	S-box Bits (Input & Output)	Function		Based on Existing Algorithm	Reference
							Encryption Algorithm	Key Schedule Algorithm		
							2. Substitution 3. Permutation			2014)
42	CUBE	64/125/ 216	128/250/ 432	SPN	15	In:4 Out:4	1. Add Round Key 2. Substitution 3. MDS Matrix 4. Permutation	1. XOR 2. Matrix Multiplication	NOEKEON (S-box)	(Berger et al., 2015)
43	LILLIPUT	64	80	FN	30	In:4 Out:4	1. Add Round Key 2. Sub Cell 3. Mix Row 4. Mix Column	1. Key Mixing 2. Permutation 3. Subkey Generation	None	(Berger et al., 2015a)
44	Midori	64/128	128	SPN	16/20	In:4 Out:4	1. Sub Cell 2. Shuffle Cell 3. Mix Column 4. Add Round Key	None	None	(Banik et al., 2015)
45	RECTANGLE	64	80/128	SPN	25	In:4 Out:4	1. Add Round Key 2. Sub Column 3. Shift Row	1. Sub Column 2. Feistel Transformation 3. Constant XOR	None	(Zhang et al., 2015)
46	RoadRunneR	64	80/128	FN	10/12	In:4 Out:4	1. Substitution 2. Diffusion Layer 3. Add Round Key	1. Whitening Key Generation 2. Subkey Generation	None	(Baysal and Şahin, 2015)
47	Simeck	32/48/ 64	64/96/ 128	FN	32/36/ 44	None	1. XOR 2. Bitwise AND 3. Rotation	1. Constant XOR 2. LFSR	SIMON (Component) & SPECK (Key Schedule)	(Yang et al., 2015)
48	SIMON	32/48/64/ 96/128	64/72/96/128/ 144/192/256	FN	32/36/42/44/52/ 54/ 68/69/72	None	1. XOR 2. Bitwise AND 3. Rotation	1. Constant XOR 2. LFSR	None	(Beaulieu et al., 2015)
49	SPECK	32/48/64/ 96/128	64/72/96/128/ 144/192/256	ARX	22/23/26/27/28/ 29/32/33/34	None	1. XOR 2. Modulo Addition 3. Rotation	1. XOR 2. LFSR	Threefish (Component)	(Beaulieu et al., 2015)
50	Sriram	64	96/128	SPN	27	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation	1. Rotation 2. Substitution 3. Constant XOR	PRESENT (Key Schedule)	(Ramudu and Shanthi, 2015a)
51	SR-LED	64	128	SPN	12	In:4 Out:4	1. Add Round Key 2. Constant XOR 3. Bit Slice 4. Sub Cell 5. Shift Row 6. Mix Column	1. XOR 2. LFSR	LED (Component); SPECK (Key Schedule) & RECTANGLE (S-box)	(Patil et al., 2015a)
52	VH	64	64/80/96/112/ 128	SPN	10/11/ 12/13/ 14	In:8 Out:8	1. Substitution 2. Permutation 3. Add Round Key	1. XOR 2. Pseudorandom Transformation	None	(Dai et al., 2015)
53	ANU	64	80/128	FN	25	In:4 Out:4	1. Rotation 2. Substitution 3. XOR 4. Permutation	1. Rotation 2. Substitution 3. Constant XOR	PRESENT (Key Schedule)	(Bansod et al., 2016)

Table 6
Lightweight block cipher components (Continued).

No.	Algorithm	Block Size (Bits)	Key Size (Bits)	Structure	Rounds	S-box Bits (Input & Output)	Function		Based on Existing Algorithm	Reference
							Encryption Algorithm	Key Schedule Algorithm		
54	MANTIS	64	128 + 64 Tweak	SPN	14	In:4 Out:4	1. Mix Column 2. Permute Cells 3. Add Round Tweakey 4. Constant XOR 5. Sub Cell	1. Rotation 2. XOR	PRINCE (Component) & Midori (S-box)	(Beierle et al., 2016)
55	PICO	64	128	SPN	32	In:4 Out:4	1. Add Round Key 2. Sub Column 3. Bit Shuffle	1. XOR 2. Rotation	SPECK (Key Schedule)	(Bansod et al., 2016a)
56	QTL	64	64/128	FN	16/20	In:4 Out:4	1. Constant XOR 2. Add Round Key 3. Substitution 4. Permutation	None	PRESENT & mCrypton (S-box)	(Li et al., 2016a)
57	SKINNY	64/128	64/128/192/ 256/384	SPN	32/36/ 40/48/ 56	In:4 Out:4	1. Sub Cell 2. Constant XOR 3. Add Round Tweakey 4. Shift Row 5. Mix Column	1. Tweakey Framework	AES (Component)	(Beierle et al., 2016)
58	SPARX	64/128	128/256	ARX	24/32/40	None	1. Modulo Addition 2. XOR 3. Rotation	1. Modulo Addition 2. Permutation	SPECKEY & NOEKEON (Component)	(Dinu et al., 2016)
59	VAYU	64	80/128	FN	31	In:4 Out:4	1. Substitution 2. Rotation 3. XOR 4. Permutation	1. Rotation 2. Substitution 3. Constant XOR	PRESENT (Key Schedule)	(Bansod, 2016)
60	XSX	64	64	SPN	4	None	1. Add Round Key 2. Bit Switch	1. Random Number Generation 2. XOR	None	(Santos et al., 2016)
61	BORON	64	80/128	SPN	25	In:4 Out:4	1. Add Round Key 2. Substitution 3. Block Shuffle 4. Permutation 5. XOR	1. Rotation 2. Substitution 3. Counter XOR	PRESENT (Key Schedule)	(Bansod et al., 2017a)
62	CHAM	64/128	128/256	ARX	80/96	None	1. Modulo Addition 2. XOR 3. Rotation	1. Rotation 2. XOR	None	(Koo et al., 2017)
63	DLBCA	32	80	FN	15	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation 4. Rotation	1. Substitution 2. Rotation	HISEC (S-box)	(Al-Dabbagh, 2017a)
64	GIFT	64/128	128	SPN	28/40	In:4 Out:4	1. Sub Cell 2. Permutation 3. Add Round Key	1. Rotation 2. Constant XOR	PRESENT (Component)	(Banik et al., 2017)

(*continued on next page*)

Table 6 (*continued*)

No.	Algorithm	Block Size (Bits)	Key Size (Bits)	Structure	Rounds	S-box Bits (Input & Output)	Function		Based on Existing Algorithm	Reference
							Encryption Algorithm	Key Schedule Algorithm		
65	LiCi	64	128	FN	31	In:4 Out:4	1. Substitution 2. Add Round Key 3. Rotation	1. Shift 2. Substitution 3. Counter XOR	PRESENT (Key Schedule)	(Patil et al., 2017)
66	SIT	64	64	Hybrid	5	In:4 Out:4	1. Add Round Key 2. Swapping 3. Substitution	1. Key Partition 2. F-function 3. Matrix Transformation 4. Key Arrangement 5. XOR	KHAZAD (Key Schedule)	(Usman et al., 2017)
67	ANU–II	64	80/128	FN	25	In:4 Out:4	1. Substitution 2. Shift 3. Add Round Key	1. Rotation 2. Substitution 3. Constant XOR	ANU (Component) & PRESENT (Key Schedule)	(Dahiphale et al., 2018)
68	GRANULE	64	80/128	FN	32	In:4 Out:4	1. Permutation 2. Substitution 3. Add Round Key	1. Shift 2. Substitution 3. Counter XOR	PRESENT (Key Schedule)	(Bansod et al., 2018a)
69	Improved DLBCA	32	80	FN	15	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation 4. Rotation	1. Substitution 2. Rotation	DLBCA (Component) & HISEC (S-box)	(Al-Dabbagh et al., 2018a)
70	Improved RoadRunneR	64	80/128	FN	10/12	In:4 Out:4	1. Merged Substitution 2. Diffusion Layer 3. Add Round Key	1. Whitening Key Generation 2. Subkey Generation	RoadRunneR (Component)	(Liu et al., 2018)
71	JAC_Jo	32	64	FN	32	None	1. Rotation 2. Bitwise AND 3. XOR	1. Constant XOR 2. LFSR	Simeck (Component & Key Schedule)	(Shantha and Arockiam, 2018a)
72	MANTRA	64	80/128	FN	32	In:4 Out:4	1. Substitution 2. Add Round Key 3. Rotation	1. Shift 2. Substitution 3. Counter XOR	PRESENT (Key Schedule)	(Bansod et al., 2018a)
73	NUX	64	80/128	FN	31	In:4 Out:4	1. Add Round Key 2. F-function 3. Permutation	1. Rotation 2. Substitution 3. Counter XOR	PRESENT (Key Schedule)	(Bansod et al., 2018a)
74	NVLC	64	80/128	SPN	20	In:4 Out:4	1. Add Round Key 2. Mix Column 3. Substitution 4. Shift	1. Shift 2. Substitution 3. Counter XOR	PRESENT (Key Schedule)	(Abd et al., 2018)
75	RARE	64	80	SPN	13	In:4 Out:4	1. Substitution 2. Permutation	1. Chaotic Function 2. Rotation 3. XOR	None	(Omrani et al., 2018)
76	SAT_Jo	64	80	SPN	31	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation	1. Fibonacci Sequence 2. Modulo Addition	None	(Shantha and Arockiam, 2018b)

Table 6 (*continued*)

No.	Algorithm	Block Size (Bits)	Key Size (Bits)	Structure	Rounds	S-box Bits (Input & Output)	Function		Based on Existing Algorithm	Reference
							Encryption Algorithm	Key Schedule Algorithm		
77	SFN	64	96	Hybrid	32	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation 4. Mix XOR	1. Add Round Key 2. Substitution 3. Mix Column 4. Mix Row/Mix XOR	Midori & PRINCE (S-box)	(Li et al., 2018)
78	BRIGHT	64/128	80/96/ 128/ 192/ 256	FN	32/33/34/35/ 36/37	None	1. Add Round Key 2. ARX Operation 3. Permutation	1. XOR 2. LFSR	SPECK (Key Schedule)	(Sehrawat and Gill, 2019a)
79	CRAFT	64	128 + 64 Tweak	SPN	31	In:4 Out:4	1. Substitution 2. Mix Column 3. Permute Nibble 4. Constant XOR 5. Add Round Tweakey	None	Midori (S-box); Piccolo & Midori (Key Schedule)	(Beierle et al., 2019)

applied permutation in a key-dependent manner (Knudsen et al., 2010). Meanwhile, the word permutation is implemented on a 4-bit cipher state (Wu and Zhang, 2011). The permutation layer is the default Feistel permutation that permutes the cipher state according to the Feistel network (FN) structure (Kolay and Mukhopadhyay, 2014). Other than that, the bit shuffle which is similar to bit permutation, shifted the cipher state bits to the new position using permutation table (Bansod et al., 2016a). For permute cells and shuffle cells, the cipher state cells are permuted using the permutation table (Beierle et al., 2016). Swapping operation is applied to diminish the data originality by altering the order of the cipher state bits (Usman et al., 2017). In block shuffle layer, it takes 16-bit input and gives 16-bit shuffled output using a permutation table (Bansod et al., 2017a). Apart from that, permute nibble function is applied on the nibble positions of the cipher state (Beierle et al., 2019).

Another widely adopted diffusion component in light-weight block cipher is the rotation function. The rotation function is implemented by a given number of bits positioned in a specific direction either to the left or to right (Adams, 1997a). An example of rotation function is called linear-feedback shift register (LFSR) which is a shift register whose input bit is a linear function of its previous state (De Cannière et al., 2009). In rotate nibble or word rotation, the function applies 4-bit rotation to the left or right in a specific position (Gong et al., 2011). For shift row layer, rows of the cipher state cell array are rotated according to the specific position and direction (Beierle et al., 2016). Using shift function, the cipher state bits is shifted by a specific number of bit position either to the left or to right (Abd et al., 2018). Meanwhile, the row round function adjusts the rows of the cipher state arrays (Jawad and Hoomod, 2019). Apart from that, in 3D rotation function, the cipher state is rotated in a specific clockwise direction (Zakaria et al., 2020a).

Besides XOR, other mathematical operations such as byte addition, modulo addition, modulo subtraction, and bitwise AND have been applied in encryption algorithms to achieve the confusion property. For the byte addition function, a byte-by-byte addition operation between two inputs is processed (Massey, 1993). Modulo addition is a process of adding two numbers, then modulo the result by an integer, and taking the remainder as the final answer (Wheeler and Needham, 1994). Bitwise AND operation compares one data bit to the corresponding bit, where the result is set to 0 if any of the two bits are equal to 0, and set to 1 if both bits are equal to 1 (Beaulieu et al., 2015). In modulo subtraction, the cipher state is manipulated by subtracting the plaintext with the round key in binary operation and taking the modulus output (Aboshosha et al., 2019).

Most algorithms adopted substitution components to provide confusion property in the design. The substitution process applies 4 bits input cipher state to the S-box simultaneously and generates 4 bits output (Bansod, 2016). Meanwhile, sub nibble function implements 4-bit nibbles substitution using S-box (Gong et al., 2011). For sub cell function, S-box is applied in parallel to every cipher state cell (Banik et al., 2015). Similarly, sub column is a parallel application of S-boxes in the same cipher state column (Zhang et al., 2015). In merged substitution, eight 4-bit S-boxes are merged and shared the same quadratic permutation, thus reducing the resource overhead (Liu et al., 2018). Another substitution method namely balanced block mixer applies one-to-one value mapping on all inputs of the mixer to change the output values (Jha et al., 2020).

On top of the individual diffusion and confusion functions, there exist components that are able to provide both diffusion and confusion properties at once, which are built based on a combination of multiple cryptographic functions. In the round function, multiple components are used at a time that includes XOR and rotation

Table 7
Lightweight Block cipher components (Continued).

No.	Algorithm	Block Size (Bits)	Key Size (Bits)	Structure	Rounds	S-box Bits (Input & Output)	Function		Based on Existing Algorithm	Reference
							Encryption Algorithm	Key Schedule Algorithm		
80	DoT	64	80/128	SPN	31	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation 4. Rotation	1. Rotation 2. Substitution 3. Counter XOR	PRESENT (Key Schedule)	(Patil et al., 2019)
81	FeW	64	80/128	FN	32	In:4 Out:4	1. F-function 2. Weight Function	1. Rotation 2. Substitution 3. Counter XOR	CLEFIA (Component); PRESENT (Key Schedule) & Hummingbird-2 (S-box)	(Kumar et al., 2019a)
82	FlexAE	64 + Tweak	128 + Tweak	FN	Variable	In:8 Out:8	1. Add Round Key 2. Block Shuffle 3. Substitution	1. Permutation	AES (S-box)	(Marsola do Nascimento and Moreira Xexéo, 2019a)
83	HERMES	64	128	SPN	30	In:4 Out:4	1. Add Round Key 2. Substitution 3. Rotation 4. XOR 5. Permutation	1. Substitution 2. Rotation 3. Modulo Addition 4. Modulo Subtraction	None	(Măluţan et al., 2019)
84	Hybrid PRESENT & Salsa20	64	128	SPN	20	In:4 Out:4	1. Padding 2. Add Round Key 3. Permutation 4. Keystream XOR	1. Chaos Key Generation 2. Quarter Round Function 3. Row Round Function	PRESENT (Component) & Salsa20 (Key Schedule)	(Jawad and Hoomod, 2019)
85	LCA	64	256	FN	16	In:4 Out:4	1. XOR 2. Modulo Addition 3. Modulo Subtraction 4. Rotation	1. Key Partition 2. Rotation	None	(Aboshosha et al., 2019)
86	Loong	64	64/80/ 128	SPN	16/20/ 32	In:4 Out:4	1. Add Round Key 2. Sub Cell 3. Mix Row 4. Mix Column	1. Constant XOR 2. Modulo Addition	None	(Liu et al., 2019a)
87	NUCLEAR	64	128	FN	25	In:4 Out:4	1. Substitution 2. XOR 3. Rotation	1. XOR 2. Rotation	None	(Salunke et al., 2019)
88	ACT	64	80	SPN	31	In:4 Out:4	1. Add Round Key 2. Sub Nibble 3. Permutation	1. LFSR 2. XOR 3. Shift	None	(Jithendra and Kassim, 2020a)
89	ILEA	64	128	SPN	12	In:4 Out:4	1. Balanced Block Mixer 2. Add Round Key 3. Substitution 4. Permutation 5. Constant XOR	None	PRINCE (S-box)	(Jha et al., 2020)
90	Improved Simeck	32/64	64/128	FN	32/44	None	1. XOR 2. Bitwise AND 3. Rotation	1. Constant XOR 2. LFSR	Simeck (Component)	(Encarnacion et al., 2020)
91	LRBC	16	16	Hybrid	24	In:4 Out:4	1. Key XOR/XNOR 2. Substitution 3. Permutation 4. L-box Computation	1. Key Combination	None	(Biswas et al., 2020a)
92	LWE	64	64	Hybrid	3	In:8 Out:8	1. Key XNOR 2. Substitution 3. XOR	1. Substitution 2. XOR 3. Permutation	None	(Toprak et al., 2020a)

Table 7 (*continued*)

No.	Algorithm	Block Size (Bits)	Key Size (Bits)	Structure	Rounds	S-box Bits (Input & Output)	Function		Based on Existing Algorithm	Reference
							Encryption Algorithm	Key Schedule Algorithm		
93	PriPresent	64/80	80/128	SPN	31	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation	1. Random Pentatope Number Generation	None	(Girija et al., 2020a)
94	TED	64	128	FN	26	In:4 Out:4	1. Substitution 2. Rotation 3. Round Constant 4. Add Round Key 5. Permutation	1. Rotation 2. Substitution 3. Counter XOR	PRESENT (Key Schedule)	(Thorat et al., 2020a)
95	T-TWINE	64 + Tweak	80/128	GFN	36	In:4 Out:4	1. Tweak Schedule 2. Substitution 3. Add Round Key	1. Counter XOR	TWINE & SKINNY (Component)	(Sakamoto et al., 2020a)
96	UBRIGHT	32	80	GFN	22	None	1. Add Round Key 2. ARX Operation 3. Permutation	None	BRIGHT (Component); Chaskey & RoadRunneR (Key Schedule)	(Sehrawat and Gill, 2020a)
97	3D RECTANGLE	64	128	SPN	25	In:4 Out:4	1. Add Round Key 2. 3D Rotation 3. Sub Column 4. Shift Row	1. Sub Column 2. Feistel Transformation 3. Constant XOR	RECTANGLE (Component & S-box)	(Zakaria et al., 2020a)
98	μ2	64	80	FN	15	In:4 Out:4	1. Add Round Key 2. Substitution 3. Permutation	1. Rotation 2. Substitution 3. Counter XOR	PRESENT (S-box)	(Yeoh et al., 2020)
99	Improved SM4	64	64	FN	32	In:4 Out:4	1. Rotation 2. XOR 3. Add Round Key 4. Substitution 5. Reverse Order Transformation	1. XOR 2. Rotation	SM4 (Component)	(Chen et al., 2021)
100	LBC-IoT	32	80	FN	32	In:4 Out:4	1. Substitution 2. Permutation 3. Add Round Key	1. Substitution 2. Permutation 3. Rotation	None	(Ramadan et al., 2021)
101	LAO-3D	64	128	SPN	20	In:4 Out:4	1. Add Round Key 2. Sub Column 3. 3D Rotation	1. Nonce XOR 2. Sub Column 3. Row Transformation	3D RECTANGLE	(Zakaria et al., 2022)

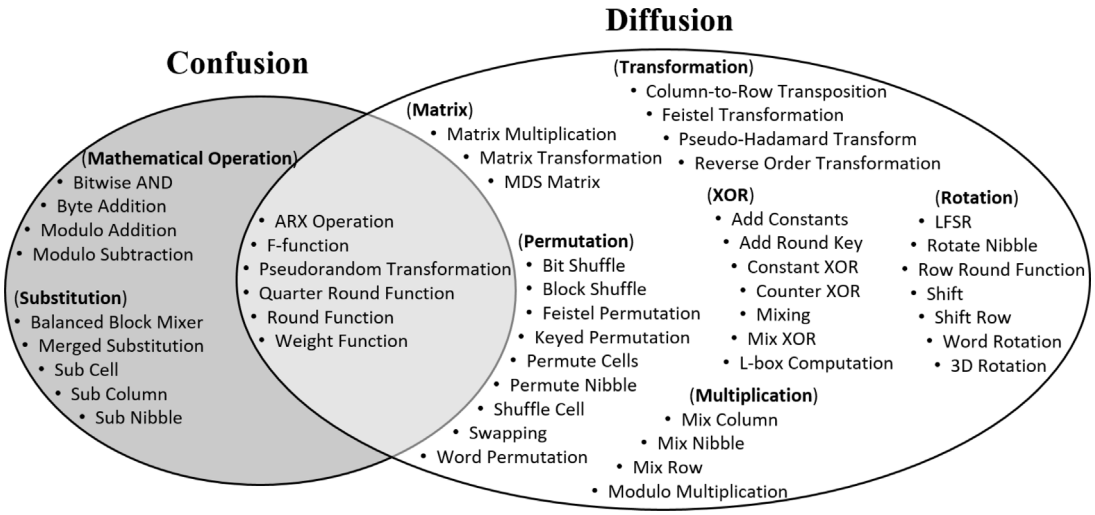

Fig. 3. Secure design components of lightweight block cipher.

(Hong et al., 2006). An F-function which comprises of substitution and diffusion layers is implemented in Feistel network ciphers (Kolay and Mukhopadhyay, 2014). Other than that, the pseudorandom transformation method is used to construct transformation tables to achieve confusion and diffusion properties (Dai et al., 2015). ARX (add–rotate–XOR) operation relies on modular addition to provide non-linearity while wordwise rotations and XOR provide diffusion property (Sehrawat and Gill, 2019a). Meanwhile, the quarter round function implements XOR, AND, and rotation operations (Jawad and Hoomod, 2019). Besides the above functions, weight function considers rotation and XOR operations as the linear functions and S-box as the non-linear function (Kumar et al., 2019a).

Two most common components used to construct a secure lightweight algorithm are the substitution and permutation (Banik et al., 2017). These two components contribute to the addition of confusion and diffusion properties in the algorithm. The substitution component replaces a cipher string by mapping a value input to another output using a transformation table such as an S-box. Meanwhile, the permutation component rearranges a cipher string by reordering the positions of each value input to produce an output using a permutation table or other rotation techniques. Analyses of the substitution and permutation components are discussed in Sections 3.2 and Section 3.3.

3.2. Substitution component

S-box or substitution box is an important component in the design of block ciphers to obscure the relationship between the key and ciphertext, thus ensuring confusion property (Omrani et al., 2018). An S-box takes an input bit m, and transforms it into an output bit n, where n is not necessarily equal to m. These transformations are repeated a few times depending on the number of elements in the S-box. The primary usage of the S-box is to provide a nonlinear element in the block cipher, which increases the complexity of the algorithm to make it more secure (Patil et al., 2019). The introduction of the nonlinearity element from the S-box would effectively resist different types of cryptanalytic attacks. There are three types of substitution components in lightweight block ciphers that include the single S-box, multiple S-boxes, and multiple S-box sizes. These S-boxes implementations are analysed to observe their impact on each lightweight algorithm which highlights the second contribution of the paper.4-bit S-box has been the most implemented substitution component in lightweight block ciphers. This type of S-box is compact and efficient even if the number of encryption rounds has to be increased to maintain security margins. 4-bit S-box reduces the resources consumed by the hardware circuit and meets the requirements of lightweight cipher. This type of S-box has optimum cycle count and adequate security (Izadi et al., 2009). The cost for the compact S-box is considerably low due to its small area footprint, thus making it low energy consumption (Poschmann et al., 2010). 4-bit S-box not only makes the cipher design robust but also introduces a great avalanche effect (Bansod et al., 2016a). The robustness of the 4-bit S-box fulfils the requirement to be resistant against cryptanalytic attacks (Liu et al., 2018).

Out of the 101 lightweight block ciphers, 20 algorithms do not implement S-box in their ciphers. KATAN is an example of a block cipher with no S-box implementation to reduce the burden on the hardware requirements. Non-S-box-based lightweight algorithms are tuned for optimal performance in hardware and software (Koo et al., 2017). However, to provide confusion property without using S-box, other substitution functions such as MA-box (Multiplication and Addition) and AX-box (Addition and XOR) were applied in lightweight algorithms (Lerman et al., 2013).

The smallest S-box size found in this research is the 3-bit S-box that contains eight elements compared to 16 elements in the 4-bit S-box. An example of a 3-bit S-box algorithm is the SEA block cipher that is constructed using a simple algebraic structure. Although the size is small, the 3-bit S-box is optimal with respect to linear and differential properties (Knudsen et al., 2010).

One of the uncommon types of S-box identified in Table 9 is the DESL algorithm that contains 6-bit input and produces 4-bit output. This type of S-box is inspired by the DES block cipher. However, the DESL applied only one S-box instead of eight S-boxes in DES to reduce the gate complexity, thus performing better than the DES algorithm.

AES-type 8-bit S-box is argued to be unsuitable for lightweight block cipher because of its large area and power consumption compared to the compact 4-bit S-box (Biswas et al., 2020a). However, there are algorithms that used 8-bit S-box such as Halka, KHAZAD, and VH in their cipher design. Other than these, Kasumi applied 7-bit and 9-bit S-boxes in its encryption algorithm. The reason for using larger S-box is because it is more resistant to differential and linear cryptanalysis (Schneier, 1993).

Multiple S-boxes have been applied simultaneously in many algorithms. As an example, two S-boxes implemented in SFN have improved the security of the cipher. For four S-boxes used in

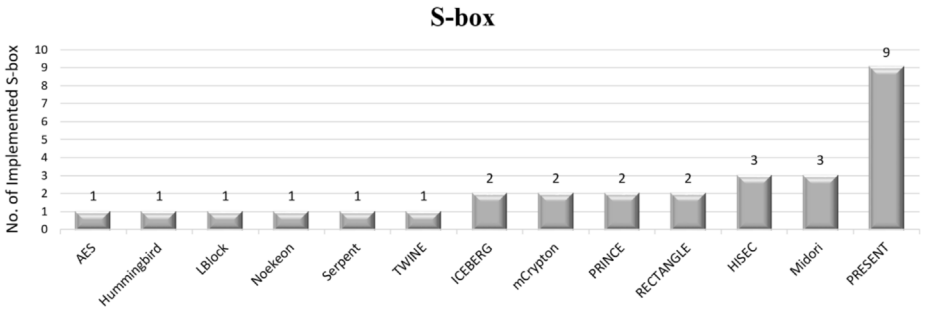

Fig. 4. Implemented existing S-boxes.

mCrypton, it takes only 128 bytes of storage that is considered small enough to be used in limited computing environments. In addition, the implementation of four different S-boxes is primarily to avoid symmetries when different bytes of the input are equal. The multiple S-boxes used in block ciphers are carefully chosen to fulfil S-box properties such as no fixed point, completed, non-linearity, differential probability, and optimum algebraic order (Wu and Zhang, 2011). Multiple S-boxes along with the diffusion layer in block cipher can make cryptanalytic attacks not practical to be executed (Usman et al., 2017).

Another important point to highlight is the implementation of existing S-boxes in lightweight algorithms. The most reused S-boxes in block ciphers are PRESENT, Midori, HISEC, RECTANGLE, PRINCE, and ICEBERG as shown in Fig. 4. Despite the age of PRESENT, its S-box is still being applied in many algorithms because it is strong with regard to linear and differential properties and

Table 8
Substitution function.

No.	Algorithm	Rounds	S-box			Max DDT	Max LAT	Maximum Significant Attack Round	
			Input & Output (Bits)	Table	Design			Differential	Linear
1	ACT	31	In:4 Out:4	(4,B,D,8,1,6,2,F,A,5,E,3,9,C,7,0)	Own	4	4	15	15
2	ANU	25	In:4 Out:4	(2,9,7,E,1,C,A,0,4,3,8,D,F,6,5,B)	Own	4	4	15	12
3	ANU–II	25	In:4 Out:4	(E,4,B,1,7,9,C,A,D,2,0,F,8,5,3,6)	Own	4	4	12	12
4	BORON	25	In:4 Out:4	(E,4,B,1,7,9,C,A,D,2,0,F,8,5,3,6)	Own	4	4	15	15
5	CUBE	15	In:4 Out:4	(7,A,2,C,4,8,F,0,5,9,1,E,3,D,B,6)	NOEKEON	4	4	9	10
6	DoT	31	In:4 Out:4	(3,F,E,1,0,A,5,8,C,4,B,2,9,7,6,D)	Own	4	4	30	24
7	FeW	32	In:4 Out:4	(2,E,F,5,C,1,9,A,B,4,6,8,0,7,3,D)	Hummingbird	4	4	11	11
8	GIFT	28	In:4 Out:4	(1,A,4,C,6,F,3,9,2,D,B,7,5,0,8,E)	Own	4	4	14	13
9	GRANULE	32	In:4 Out:4	(E,7,8,4,1,9,2,F,5,A,B,0,6,C,D,3)	Own	4	4	14	14
10	Improved RoadRunneR	10	In:4 Out:4	(0,8,6,D,5,F,7,C,4,E,2,3,9,1,B,A)	Own	4	4	N/A	N/A
11	Improved SM4	32	In:4 Out:4	(0,C,9,D,3,5,B,1,6,E,A,8,7,4,2,F)	Own	4	4	N/A	N/A
12	KLEIN	12	In:4 Out:4	(7,4,A,9,1,F,B,0,C,3,2,6,8,E,D,5)	Own	4	4	9	9
13	LBC-IoT	32	In:4 Out:4	(0,8,6,D,5,F,7,C,4,E,2,3,9,1,B,A)	Own	4	4	N/A	N/A
14	LCA	16	In:4 Out:4	(8,7,3,C,D,B,4,1,6,A,9,F,0,5,E,2)	Own	4	4	N/A	N/A
15	LiCi	31	In:4 Out:4	(3,F,E,1,0,A,5,8,C,4,B,2,9,7,6,D)	Own	4	4	12	12
16	LILLIPUT	30	In:4 Out:4	(4,8,7,1,9,3,2,E,0,B,6,F,A,5,D,C)	Own	4	4	17	17
17	Loong	16	In:4 Out:4	(C,A,D,3,E,B,F,7,9,8,1,5,0,2,4,6)	Own	4	4	4	4
18	LRBC	24	In:4 Out:4	(Biswas et al., 2020a)	Own	N/A	N/A	N/A	N/A
19	MANTRA	32	In:4 Out:4	(2,5,D,A,F,3,4,9,B,0,6,C,8,E,1,7)	Own	4	4	16	16
20	MIBS	32	In:4 Out:4	(4,F,3,8,D,A,C,0,B,5,7,E,2,6,1,9)	mCrypton	4	4	20	20
21	NUCLEAR	25	In:4 Out:4	(E,4,B,1,7,9,C,A,D,2,0,F,8,5,3,6)	Own	4	4	20	16
22	NUX	31	In:4 Out:4	(E,7,8,4,1,9,2,F,5,A,B,0,6,C,D,3)	Own	4	4	20	15
23	NVLC	20	In:4 Out:4	(1,0,5,3,E,2,F,7,D,A,9,B,C,8,4,6)	Own	4	4	N/A	N/A
24	Piccolo	25	In:4 Out:4	(E,4,B,2,3,8,0,9,1,A,7,F,6,C,5,D)	Own	4	4	7	7
25	PICO	32	In:4 Out:4	(1,2,4,D,6,F,B,8,A,5,E,3,9,C,7,0)	Own	4	4	24	24
26	PRIDE	20	In:4 Out:4	(0,4,8,F,1,5,E,9,2,7,A,C,B,D,6,3)	Own	4	4	N/A	N/A
27	RARE	13	In:4 Out:4	(D,E,0,8,B,1,9,F,6,3,7,A,2,5,C,4)	Own	4	4	N/A	N/A
28	RoadRunneR	10	In:4 Out:4	(0,8,6,D,5,F,7,C,4,E,2,3,9,1,B,A)	Own	4	4	5	5
29	SAT_Jo	31	In:4 Out:4	(1,0,E,9,B,D,6,7,2,F,5,C,4,A,8,3)	Own	4	4	N/A	N/A
30	SKINNY	32	In:4 Out:4	(C,6,9,0,1,A,2,B,3,8,5,D,4,E,7,F)	Own	4	4	N/A	N/A
31	Sriram	27	In:4 Out:4	(6,3,5,8,1,E,2,B,F,C,9,7,A,0,4,D)	Own	4	4	N/A	N/A
32	TED	26	In:4 Out:4	(9,4,F,A,E,1,0,6,C,7,3,8,2,B,5,D)	Own	4	4	12	12
33	VAYU	31	In:4 Out:4	(6,3,A,5,C,8,1,B,0,D,9,E,F,2,7,4)	Own	4	4	8	8
34	PRINCE	12	In:4 Out:4	(B,F,3,2,A,C,9,1,6,7,8,0,E,5,D,4)	Own	4	4	N/A	N/A
35	ILEA	12	In:4 Out:4	(B,F,3,2,A,C,9,1,6,7,8,0,E,5,D,4)	PRINCE	4	4	N/A	N/A
36	PUFFIN	32	In:4 Out:4	(D,7,3,2,9,A,C,1,F,4,5,E,6,0,B,8)	ICEBERG	4	4	N/A	N/A
37	PUFFIN2	34	In:4 Out:4	(D,7,3,2,9,A,C,1,F,4,5,E,6,0,B,8)	ICEBERG	4	4	N/A	N/A
38	TWINE	32	In:4 Out:4	(C,0,F,A,2,B,9,5,8,3,D,7,1,E,6,4)	Own	4	4	15	15

(*continued on next page*)

Table 8 (*continued*)

No.	Algorithm	Rounds	S-box			Max DDT	Max LAT	Maximum Significant Attack Round	
			Input & Output (Bits)	Table	Design			Differential	Linear
39	T-TWINE	36	In:4 Out:4	(C,0,F,A,2,B,9,5,8,3,D,7,1,E,6,4)	TWINE	4	4	19	19
40	Midori	16	In:4 Out:4	(C,A,D,3,E,B,F,7,8,9,1,5,0,2,4,6)	Own	4	4	7	7
41	CRAFT	31	In:4 Out:4	(C,A,D,3,E,B,F,7,8,9,1,5,0,2,4,6)	Midori	4	4	18	22
42	MANTIS	14	In:4 Out:4	(C,A,D,3,E,B,F,7,8,9,1,5,0,2,4,6)	Midori	4	4	N/A	N/A
43	RECTANGLE	25	In:4 Out:4	(6,5,C,A,1,E,7,9,B,0,3,D,8,F,4,2)	Own	4	4	15	14
44	3D RECTANGLE	25	In:4 Out:4	(6,5,C,A,1,E,7,9,B,0,3,D,8,F,4,2)	RECTANGLE	4	4	N/A	N/A
45	SR-LED	12	In:4 Out:4	(6,5,C,A,1,E,7,9,B,0,3,D,8,F,4,2)	RECTANGLE	4	4	N/A	N/A
46	HISEC	15	In:4 Out:4	(F,C,2,7,9,0,5,A,1,B,E,8,6,D,3,4)	Own	4	4	N/A	N/A
47	DLBCA	15	In:4 Out:4	(F,C,2,7,9,0,5,A,1,B,E,8,6,D,3,4)	HISEC	4	4	N/A	N/A
48	Improved DLBCA	15	In:4 Out:4	(F,C,2,7,9,0,5,A,1,B,E,8,6,D,3,4)	HISEC	4	4	N/A	N/A
49	OLBCA	22	In:4 Out:4	(F,C,2,7,9,0,5,A,1,B,E,8,6,D,3,4)	HISEC	4	4	8	8
50	PRESENT	31	In:4 Out:4	(C,5,6,B,9,0,A,D,3,E,F,8,4,7,1,2)	Own	4	4	20	16
51	μ2	15	In:4 Out:4	(C,5,6,B,9,0,A,D,3,E,F,8,4,7,1,2)	PRESENT	4	4	6	6
52	EPCBC	32	In:4 Out:4	(C,5,6,B,9,0,A,D,3,E,F,8,4,7,1,2)	PRESENT	4	4	N/A	N/A
53	Hybrid PRESENT & Salsa20	20	In:4 Out:4	(C,5,6,B,9,0,A,D,3,E,F,8,4,7,1,2)	PRESENT	4	4	N/A	N/A
54	Improved GOST	32	In:4 Out:4	(C,5,6,B,9,0,A,D,3,E,F,8,4,7,1,2)	PRESENT	4	4	N/A	N/A
55	Khudra	18	In:4 Out:4	(C,5,6,B,9,0,A,D,3,E,F,8,4,7,1,2)	PRESENT	4	4	6	6
56	LED	32	In:4 Out:4	(C,5,6,B,9,0,A,D,3,E,F,8,4,7,1,2)	PRESENT	4	4	6	6
57	PriPresent	31	In:4 Out:4	(C,5,6,B,9,0,A,D,3,E,F,8,4,7,1,2)	PRESENT	4	4	N/A	N/A
58	LAO-3D	20	In:4 Out:4	(C,5,6,B,9,0,A,D,3,E,F,8,4,7,1,2)	PRESENT	4	4	6	7
59	PRINTcipher	48	In:3 Out:3	(0,1,3,6,7,4,5,2)	Own	N/A	N/A	24	24
60	SEA	Variable	In:3 Out:3	(0,5,6,7,4,3,1,2)	Own	N/A	N/A	N/A	N/A
61	Halka	24	In:8 Out:8	(Das, 2014a)	Own	N/A	N/A	N/A	N/A
62	KHAZAD	8	In:8 Out:8	(Barreto and Rijmen, 2000)	Own	N/A	N/A	N/A	N/A
63	LWE	3	In:8 Out:8	(Toprak et al., 2020a)	Own	N/A	N/A	N/A	N/A
64	SKIPJACK	32	In:8 Out:8	(NSA, 1998a)	Own	N/A	N/A	N/A	N/A
65	VH	10	In:8 Out:8	(Dai et al., 2015)	Own	N/A	N/A	4	4
66	FlexAE	Variable	In:8 Out:8	(Marsola do Nascimento and Moreira Xexéo, 2019a)	AES	N/A	N/A	9	10
67	DESL	16	In:6 Out:4	(E,5,7,2,B,8,1,F,0,A,9,4,6,D,C,3) (5,0,8,F,E,3,2,C,B,7,6,9,D,4,1,A) (4,9,2,E,8,7,D,0,A,C,F,1,5,B,3,6) (9,6,F,5,3,8,4,B,7,1,C,2,0,E,A,D)	Own	N/A	N/A	N/A	N/A
68	Blowfish	16	In:8 Out:32 (4 S-boxes)	(Schneier, 1993)	Own	N/A	N/A	N/A	N/A

has a low area footprint (Poschmann et al., 2010). Meanwhile, ICEBERG S-box implements separate logic gates to generate output bits in which the cost of the S-boxes does not dominate the hardware resources (Cheng et al., 2008). In RECTANGLE S-box, the clustering of linear and differential trails is prevented thus increasing the security of the algorithm (Patil et al., 2015a). Other than that, HISEC S-box provides confusion property which gives non-linearity elements to the cipher (Al-Dabbagh, 2017a). Midori S-box has been selected based on its low energy consumption which is needed for resource-constrained devices (Beierle et al., 2019). On the other hand, lightweight algorithm can be optimized with respect to latency using PRINCE S-box (Jha et al., 2020).

An S-box has a huge influence on differential and linear cryptanalysis results (Cui and Jin, 2017a). From the S-box, the Differential Distribution Table (DDT) and Linear Approximation Table (LAT) are generated to be applied in differential and linear cryptanalysis respectively. DDT and LAT tables can determine the number of active S-boxes, thus analysing the algorithm strength against cryptanalytic attacks. For a 4-bit S-box, the threshold of the maximum DDT and LAT values for an optimum S-box is equal to four as obtained by most of the analysed algorithms in Table 8 and Table 9. Further explanations of the differential and linear cryptanalysis are discussed in Section 3.4.

As a summary of this section, a comparison of substitution functions features is presented in Table 10. Four types of S-boxes that include small S-box (4-bit and lower size S-box), large S-box (bigger than 4-bit size S-box), different input and output sizes S-box, and multiple S-boxes are compared to observe the characteristics of each implementation in block ciphers. The results from the comparison show that the substitution features are dependent on the size of the S-boxes that influence the advantages or disadvantages of the substitution functions. Since lightweight block cipher is supposedly designed to be simpler than the conventional algorithm, substitution function such as 4-bit S-box is preferred by developers given its adequate security for resource-constrained devices. The findings of the substitution function features are used in the identification of the secure cryptographic components.

3.3. Permutation component

Permutation or linear layer is a core component in Substitution Permutation Network (SPN) lightweight block cipher (Albrecht et al., 2014). The component can influence the security and efficiency of the algorithm. In general, there are two types of permutation components which are the permutation and rotation functions. For the third contribution of the paper, Table 11 to Table 12 listed the implementation of permutation components in lightweight algorithms that includes the data type and permutation method. In addition, the table listed the type of rotation, rotation direction, and rotation parameter. This information provides a broad overview of the various strategies that can be used in identifying the secure cryptographic components.

Table 9
Substitution function (Continued).

No.	Algorithm	Rounds	S-box			Max DDT	Max LAT	Maximum Significant Attack Round	
			Input & Output (Bits)	Table	Design			Differential	Linear
69	Kasumi	8	In:7 Out:7 & In:9 Out:9	(ETSI, 2014a)	Own	N/A	N/A	N/A	N/A
70	QTL	16/20	In:4 Out:4 (2 S-boxes)	(C,5,6,B,9,0,A,D,3,E,F,8,4,7,1,2) (4,F,3,8,D,A,C,0,B,5,7,E,2,6,1,9)	PRESENT & mCrypton	4/4	4	6	6
71	SFN	32	In:4 Out:4 (2 S-boxes)	(C,A,D,3,E,B,F,7,8,9,1,5,0,2,4,6) (B,F,3,2,A,C,9,1,6,7,8,0,E,5,D,4)	Midori & PRINCE	4	4	16	16
72	SIT	5	In:4 Out:4 (2 S-boxes)	(3,F,E,0,5,4,B,C,D,A,9,6,7,8,2,1) (9,E,5,6,A,2,3,C,F,0,4,D,7,B,1,8)	Own	4	4	N/A	N/A
73	ICEBERG	16	In:4 Out:4 (2 S-boxes) & In:8 Out:8	(D,7,3,2,9,A,C,1,F,4,5,E,6,0,B,8) (4,A,F,C,0,D,9,B,E,6,1,7,3,5,8,2) & (Standaert et al., 2004)	Own	N/A	N/A	N/A	N/A
74	Hummingbird	4	In:4 Out:4 (4 S-boxes)	(8,6,5,F,1,C,A,9,E,B,2,4,7,0,D,3) (0,7,E,1,5,B,8,2,3,A,D,6,F,C,4,9) (2,E,F,5,C,1,9,A,B,4,6,8,0,7,3,D) (0,7,3,4,C,1,A,F,D,E,6,B,2,8,9,5)	Own	N/A	N/A	N/A	N/A
75	Hummingbird-2	4	In:4 Out:4 (4 S-boxes)	(7,C,E,9,2,1,5,F,B,6,D,0,4,8,A,3) (4,A,1,6,8,F,7,C,3,0,E,D,5,9,B,2) (2,F,C,1,5,6,A,D,E,8,3,4,0,B,9,7) (F,4,5,8,9,7,2,1,A,3,0,E,6,C,D,B)	Own	N/A	N/A	N/A	N/A
76	mCrypton	12	In:4 Out:4 (4 S-boxes)	(4,F,3,8,D,A,C,0,B,5,7,E,2,6,1,9) (1,C,7,A,6,D,5,3,F,B,2,0,8,4,9,E) (7,E,C,2,0,9,D,A,3,F,5,8,6,4,B,1) (B,0,A,7,D,6,4,2,C,E,3,9,1,5,F,8)	Own	4	4	8	8
77	ESF	31	In:4 Out:4 (8 S-boxes)	(Liu et al., 2014a)	Serpent	N/A	N/A	N/A	N/A
78	CAST	12/16/48	In:32 Out:32 (8 S-boxes)	(Adams, 1997a)	Own	N/A	N/A	N/A	N/A
79	LBlock	32	In:4 Out:4 (10 S-boxes)	(Wu and Zhang, 2011)	Own	4	4	15	15
80	Improved LBlock	32	In:4 Out:4 (10 S-boxes)	(Wu and Zhang, 2011)	LBlock	4	4	13	13
81	HERMES	30	In:4 Out:4 (16 S-boxes)	(Măluţan et al., 2019)	Own	N/A	N/A	N/A	N/A

3.3.1. Permutation function

Permutation function is one of the permutation components in a block cipher design. The function consists of three methods that include the Feistel permutation, formulation permutation, and permutation table. Feistel permutation implemented in lightweight algorithms is divided into block and nibble types. Blowfish Feistel network discards the bitwise permutation to introduce simple operations that are efficient on microprocessors (Schneier, 1993). The block permutation layer in Feistel design of MIBS used simple wiring that does not require extra gate (Izadi et al., 2009). Block permutation layer increased the active S-boxes of VAYU algorithm (Bansod, 2016). In LBC-IoT, two blocks swap of the Feistel structure is used to enhance diffusion property (Ramadan et al., 2021). On the other hand, the nibble permutation of Feistel structure in Khudra reduced the number of S-box usage (Kolay and Mukhopadhyay, 2014). Although the S-box usage is reduced, the block cipher does not require an additional diffusion layer to secure the algorithm.

Formulation-based permutation is another option besides the conventional methods. Instead of using predetermined permutation parameters, this method executes on-the-fly mathematical computation to obtain the parameters. The bitwise formulation permutation provides fast diffusion without hardware implementation cost in ESF algorithm (Liu et al., 2014a). Bit formulation permutation is designed to maximize the number of active S-boxes of μ2 block cipher (Yeoh et al., 2020). Apart from the bit formulation method, LRBC block cipher implemented round transposition operation using nibble formulation permutation (Biswas et al., 2020a). Meanwhile, ILEA applied byte permutation to provide additional security to the algorithms (Jha et al., 2020).

Permutation table is a common function in block ciphers. The table can be structured into different forms of permutation such as bit, nibble, byte, or block to suit the algorithm. ICEBERG permutation table is designed to disturb the bit alignment of the algorithm to provide resistance against cryptanalytic attacks (Standaert et al., 2004). Bit permutation table is implemented in PRESENT given the simplicity of the method which is being adopted in many algorithms (Bogdanov et al., 2007). It can be seen that the bit permutation table of PUFFIN is involution and satisfies the property that no two outputs of an S-box are connected to the same S-box in the next encryption round (Cheng et al., 2008). Bit permutation table of Halka algorithm proved that no trails can be constructed thus preventing structural attacks (Das, 2014a). The regular bit permutation table helps VAYU to improve the robustness of the algorithm round function (Bansod, 2016). The bit permutation is easy to be implemented in hardware and software environments of SFN algorithm (Li et al., 2018). Minimal memory requirement of DoT block cipher is achieved using bit permutation table with high diffusion mechanism (Patil et al., 2019). Cipher bits that are permuted using permutation table give complete diffusion with just three rounds of ACT algorithm

Table 10
Features of substitution functions.

Feature	Small S-box	Large S-box	Different Input and Output Sizes S-box	Multiple S-boxes
Area Footprint	↓	↑	↓ (Small S-box) or ↑ (Large S-box)	↓ (Small S-box) or ↑ (Large S-box)
Avalanche Effect	↑	↑	↑	↑
Compact	✓	*x*	✓ (Small S-box) or *x* (Large S-box)	✓ (Small S-box) or *x* (Large S-box)
Complexity	↓	↑	↓ (Small S-box) or ↑ (Large S-box)	↓ (Small S-box) or ↑ (Large S-box)
Confusion Property	✓	✓	✓	✓
Cryptanalytic Attacks Resistant	✓	✓	✓	✓
Cycle Count	↓	↑	↓ (Small S-box) or ↑ (Large S-box)	↓ (Small S-box) or ↑ (Large S-box)
Diminish Data Originality	✓	✓	✓	✓
Gate Count	↓	↑	↓ (Small S-box) or ↑ (Large S-box)	↓ (Small S-box) or ↑ (Large S-box)
Implementation Cost	↓	↑	↓ (Small S-box) or ↑ (Large S-box)	↓ (Small S-box) or ↑ (Large S-box)
Involution	✓	✓	✓	✓
Latency	↓	↑	↓ (Small S-box) or ↑ (Large S-box)	↓ (Small S-box) or ↑ (Large S-box)
Lightweight	✓	*x*	✓ (Small S-box) or *x* (Large S-box)	*x*
Power Consumption	↓	↑	↓ (Small S-box) or ↑ (Large S-box)	↓ (Small S-box) or ↑ (Large S-box)
Robustness	✓	✓	✓	✓
Security	✓	✓	✓	✓
Speed	↑	↓	↑ (Small S-box) or ↓ (Large S-box)	↑ (Small S-box) or ↓ (Large S-box)
Storage Requirement	↓	↑	↓ (Small S-box) or ↑ (Large S-box)	↓ (Small S-box) or ↑ (Large S-box)

"✓" (Yes) and "*x*" (No) represent the availability of each feature, while "↑" (High) and "↓" (Low) indicate the valuation.

(Jithendra and Kassim, 2020a). Besides the bit permutation, a nibble-based permutation table is also being used in block ciphers such as TWINE that achieved hardware efficiency (Suzaki et al., 2011). Midori is another algorithm that applied the nibble-type permutation table to improve diffusion speed and increase the number of active S-boxes in each encryption round (Banik et al., 2015). Other implementations of the permutation table are SKIPJACK which used byte permutation and LILLIPUT that adopted block permutation to maximize the number of active S-boxes of the algorithms (Berger et al., 2015a).

Table 11 to Table 12 show a type of permutation categorized as the unspecified method. Unlike the Feistel, table, and formulation methods, algorithms implemented this type of permutation do not specify the technique used to generate their permutation parameters. The unspecified permutation can be divided into the bit, block, byte, and nibble implementations. Bit permutation provides full dependency after a minimal number of encryption rounds as implemented in PRINTcipher (Knudsen et al., 2010). The bit permutation of an improved LBlock produced a higher number of active S-box because the method spreads the cipher bits into four S-boxes, thus making it secure against cryptographic attacks (Wu and Zhang, 2011). Besides the bit permutation method, block permutation is applied in block ciphers such as SIT to diminish the data originality by altering the order of bits block in the algorithm (Usman et al., 2017). To improve the diffusion rate of UBRIGHT, a block permutation is applied in the last step of every encryption round (Sehrawat and Gill, 2020a). Similar to the block permutation method, byte permutation provides a systematic way to ensure SAFER archives the necessary diffusion (Massey, 1993). In addition, to enhance diffusion property, Piccolo adopted the byte-based permutation between every encryption round (Shibutani et al., 2011). Meanwhile, another method to be highlighted is the nibble permutation that applies a 4-bit permutation operation. Nibble permutation is considered cheap in hardware and software environments as implemented in mCrypton (Lim and Korkishko, 2005). Loong reported that the algorithm does not require resources in hardware implementation using the nibble permutation method (Liu et al., 2019a). PRINCE that adopted nibble permutation guaranteed at least 16 S-boxes are activated in four consecutive encryption rounds (Borghoff et al., 2012). Lastly, CRAFT attains full diffusion after seven rounds with nibble permutation and is resistant against differential and linear attacks (Beierle et al., 2019).

A comparison of permutation function features is presented in Table 13. Three types of permutation functions that include Feistel permutation, formulation permutation, and table permutation are compared to observe the characteristics of each implementation in lightweight block ciphers.The results from the comparison show that all permutation functions offer many benefits for lightweight block cipher adoption due to their excellent characteristics, especially for software and hardware implementations. Although the strength of each permutation function is not similar, every function has its advantages in increasing the security of lightweight algorithms. This statement is supported by the implementation of permutation functions in the majority of the analysed algorithms as listed in Table 11 to Table 12, showing the importance of permutation functions in algorithm design.

3.3.2. Rotation function

Rotation function is a simple permutation operation adopted in many lightweight block ciphers. In general, there are three types of rotation directions including one-way, two-way, and 3-dimensional rotations. On top of the different rotation directions, implementation of the function can be applied in multiple forms of input such as bit, byte, and nibble rotations. Throughout the conducted analysis, there are two factors that influence the output of the function. The first factor is the rotation direction and the other is the rotation parameter. Although the factors have been identified, the best combinations cannot be concluded because the strength of the cipher output is subject to the overall structure of the dedicated algorithm. However, the advantages of each type of rotation are discussed in this section for a better understanding of the function.

One-way bit rotation to the left direction is the most applied rotation function among lightweight block ciphers as shown in Table 11 to Table 12. Improved LBlock modified the bit rotation method by 20-bit to produce better diffusion than 8-bit used in the original LBlock cipher, thus increasing the active S-boxes (Al-Dabbagh and Al-Shaikhli, 2013). The P-layer of RECTANGLE adopted bit rotation that makes each column dependent on some other columns, which aimed to provide high diffusion (Zhang et al., 2015). Linear function on each byte of RoadRunneR provides diffusion using the bit rotation method (Baysal and Şahin, 2015). The bit rotation method is efficient for hardware implementations as adopted in Simeck (Yang et al., 2015). JAC_Jo used bit rotation in

Table 11
Permutation and rotation functions.

No.	Algorithm	Rounds	Permutation		Rotation			Maximum Significant Attack			
								Differential		Linear	
			Data Type	Method	Data Type	Direction	Parameter	Rounds	Active S-boxes	Rounds	Active S-boxes
1	IDEA	8.5	Block	Unspecified		None				N/A	
2	Blowfish	16	Block	Feistel		None				N/A	
3	SAFER	Variable	Byte	Unspecified		None				N/A	
4	RC5	0–255		None	Bit	Left	Formulation			N/A	
5	TEA	64		None		None				N/A	
6	CAST	12	Block	Feistel	Bit	Left	Varies			N/A	
7	MISTY	8	Block	Feistel		None				N/A	
8	SKIPJACK	32	Byte	Table		None				N/A	
9	KHAZAD	8		None		None				N/A	
10	ICEBERG	16	Bit	Table		None				N/A	
11	mCrypton	12	Bit Nibble	Unspecified		None		8	32	8	32
12	HIGHT	32	Byte	Unspecified	Bit	Left	1, 2, 3, 4, 6 & 7			N/A	
13	SEA	Variable		None	Bit	Left	1			N/A	
						Right	1				
					Byte	Left	1				
14	DESL	16	Bit Block	Unspecified Feistel		None				N/A	
15	PRESENT	31	Bit	Table		None		20	40	16	20
16	PUFFIN	32	Bit	Table		None				N/A	
17	KATAN	254		None	Bit	Right	1			N/A	
						Left	1				
18	KTANTAN	254		None	Bit	Right	1			N/A	
						Left	1				
19	MIBS	32	Nibble Block	Table Feistel		None		20	30	20	35
20	PUFFIN2	34	Bit	Table		None				N/A	
21	Hummingbird	4		None	Bit	Left	6 & 10			N/A	
22	Improved GOST	32	Block	Feistel	Bit	Left	11			N/A	
23	PRINTcipher	48	Bit	Formulation		None		24	24	24	24
24	EPCBC	32	Bit	Formulation		None				N/A	
25	Hummingbird-2	4		None	Bit	Left	1, 3 & 8			N/A	
						Right	1				
26	KLEIN	12	Nibble	Unspecified		None		9	33	9	33
27	LBlock	32	Nibble Block	Unspecified Feistel	Bit	Left	8	15	32	15	32
28	LED	32		None	Nibble	Left	1, 2 & 3	6	37	6	37
29	Piccolo	25	Byte Block	Unspecified		None		7	7	7	7
30	TWINE	32	Nibble	Table		None		15	32	15	32
31	PRINCE	12	Nibble	Unspecified		None				N/A	
32	Improved IDEA	6.5	Block	Unspecified		None				N/A	
33	Improved LBlock	32	Bit Block	Unspecified Feistel	Bit	Left	20	13	32	13	32
34	BEST-1	12	Byte	Unspecified	Bit	Left	1, 3, 4 & 6			N/A	
35	ESF	31	Bit Block	Formulation Feistel	Bit	Left	7			N/A	
36	Halka	24	Bit	Table		None				N/A	
37	HISEC	15	Bit Block	Unspecified Feistel	Bit	Left	20			N/A	
38	Kasumi	8	Block	Feistel		None				N/A	
39	Khudra	18	Nibble Block	Feistel		None		6	6	6	6
40	OLBCA	22	Block	Unspecified	Bit	Left	12	8	14	8	14
41	PRIDE	20	Bit	Unspecified		None				N/A	
42	CUBE	15		None	Bit	3D	Varies	9	33	10	38
43	LILLIPUT	30	Block	Table Feistel		None		17	36	17	34
44	Midori	16	Nibble	Table		None		7	35	7	35
45	RECTANGLE	25		None	Bit	Left	1, 12 & 13	15	32	14	32
46	RoadRunneR	10	Block	Feistel	Bit	Left	1, 12 & 13	5	36	5	36
47	Simeck	32/36/44		None	Bit	Left	1 & 5			N/A	
48	SIMON	32/36/(42 or 44)		None	Bit	Left	1, 2 & 8			N/A	
49	SPECK	22/(22 or 23)/(26 or 27)		None	Bit	Left	2 & 3			N/A	
						Right	7 & 8				
50	Sriram	27	Bit	Table		None				N/A	
51	SR-LED	12		None	Nibble	Left	1, 2 & 3			N/A	
52	VH	10	Bit	Table		None		4	21	4	24
53	ANU	25	Bit Block	Table Feistel	Bit	Left Right	3 8	15	40	12	36
54	MANTIS	14	Bit	Table		None				N/A	
55	PICO	32	Bit	Table		None		24	48	24	45

Table 12
Permutation and rotation functions (Continued).

No.	Algorithm	Rounds	Permutation		Rotation			Maximum Significant Attack			
								Differential		Linear	
			Data Type	Method	Data Type	Direction	Parameter	Rounds	Active S-boxes	Rounds	Active S-boxes
56	QTL	16	Bit Block	Table Feistel		None		6	42	6	42
57	SKINNY	32	None		Nibble	Right	1, 2 & 3			N/A	
58	SPARX	24	Block	Unspecified	Bit	Left Right	8 8			N/A	
59	VAYU	31	Bit Block	Table Feistel	Bit	Left Right	3 & 7 3 & 7	8	54	8	36
60	XSX	4	Bit	Formulation		None				N/A	
61	BORON	25	Nibble	Unspecified	Bit	Left	1, 4, 7 & 9	15	40	15	40
62	CHAM	80	Block	Unspecified	Bit	Left	1 & 8			N/A	
63	DLBCA	15	Bit Block	Unspecified Unspecified	Bit	Left	12			N/A	
64	GIFT	28	Bit	Table		None		14	28	13	26
65	LiCi	31	Block	Feistel	Bit	Left Right	3 7	12	39	12	36
66	SIT	5	Block	Feistel		None				N/A	
67	ANU–II	25	Block	Feistel	Bit	Left Right	10 3	12	39	12	42
68	GRANULE	32	Nibble Block	Unspecified Feistel	Bit	Left Right	2 7	14	46	14	46
69	Improved DLBCA	15	Bit Block	Unspecified Feistel	Bit	Left	12			N/A	
70	Improved RoadRunneR	10/12	Bit Block	Table Feistel		None				N/A	
71	JAC_Jo	32	Block	Feistel	Bit	Left	1 & 5			N/A	
72	MANTRA	32	Block	Feistel	Bit	Left Right	3 8	16	32	16	32
73	NUX	31	Bit Block	Table Feistel	Bit	Left Right	8 3	20	36	15	39
74	NVLC	20	None		Bit Byte	Left Left	Formulation 2			N/A	
75	RARE	13	Bit	Unspecified		None				N/A	
76	SAT_Jo	31	Bit	Table		None				N/A	
77	SFN	32	Bit	Table		None		16	29	16	29
78	BRIGHT	32	Block	Feistel	Bit	Left	2 & 6			N/A	
79	CRAFT	31	Nibble	Unspecified		None		18	32	22	32
80	DoT	31	Byte	Unspecified	Bit	Left Right	25 31	30	35	24	40
81	FeW	32	Byte Block	Unspecified Feistel	Bit	Left	1, 5, 9 & 12	11	34	11	34
82	FlexAE	Variable	Nibble Block	Unspecified Feistel		None		9	26		N/A
83	HERMES	30	Bit	Table	Bit	Left Right	Formulation			N/A	
84	Hybrid PRESENT & Salsa20	20	Bit	Table	Bit	Left	7, 9, 13 & 18			N/A	
85	LCA	16	Block	Feistel	Bit	Left	13			N/A	
86	Loong	16	Nibble	Unspecified		None		4	32	4	32
87	NUCLEAR	25	Block	Feistel	Bit	Left Right	3 6	20	35	16	32
88	ACT	31	Bit	Table		None		15	30	15	32
89	ILEA	12	Byte	Formulation		None				N/A	
90	Improved Simeck	32/44	Block	Feistel	Bit	Left Right	3 5			N/A	
91	LRBC	24	Nibble	Unspecified Formulation		None				N/A	
92	LWE	3	Nibble Byte Block	Unspecified		None				N/A	
93	PriPresent	31	Bit	Table		None				N/A	
94	TED	26	Bit Block	Formulation Feistel	Bit	Left Right	11 7	12	36	12	33
95	T-TWINE	36	Nibble	Table		None		19	32	19	32
96	UBRIGHT	22	Block	Unspecified	Bit	Left	2 & 6			N/A	
97	3D RECTANGLE	25	None		Bit	3D	Varies			N/A	
98	μ2	15	Bit Block	Formulation Feistel		None		6	36	6	36
99	Improved SM4	32	None		Bit	Left	1, 4, 6 & 9			N/A	
100	LBC-IoT	32	Bit Block	Table Feistel	Bit	Left	7			N/A	
101	LAO-3D	25	None		Bit	3D	Varies	6	25	7	13

Table 13
Features of Permutation functions.

Feature	Feistel Permutation	Formulation Permutation	Permutation Table
Avalanche Effect	↑	↑	↑
Complexity	↓	↑	↓
Cryptanalytic Attacks Resistant	✓	✓	✓
Diffusion Property	✓	✓	✓
Diminish Data Originality	✓	✓	✓
Efficiency	↑	↑	↑
Gate Count	↓	↓	↓
Implementation Cost	↓	↑	↓
Increase Active S-boxes	✓	✓	✓
Involution	✓	✓	✓
Lightweight	✓	✓	✓
Randomization	↓	↑	↑
Robustness	✓	✓	✓
Security	✓	✓	✓
Simplicity	↑	↓	↑
Speed	↑	↓	↑
Storage Requirement	↓	↓	↑

"✓" (Yes) represents the availability of each feature, while "↑" (High) and "↓" (Low) indicate the valuation.

Table 14
Features of Rotation functions.

Feature	One-way Direction (Left or Right Rotation)	Two-way Direction (Left and Right Rotation)	Multiple Directions (3D Rotation)
Avalanche Effect	↑	↑	↑
Complexity	↓	↓	↓
Cryptanalytic Attacks Resistant	✓	✓	✓
Diffusion Property	✓	✓	✓
Diminish Data Originality	✓	✓	✓
Efficiency	↑	↑	↑
Gate Count	↓	↓	↓
Implementation Cost	↓	↓	↓
Increase Active S-boxes	✓	✓	✓
Involution	✓	✓	✓
Lightweight	✓	✓	✓
Randomization	↓	↓	↑
Robustness	✓	✓	✓
Security	✓	✓	✓
Simplicity	↑	↑	↑
Speed	↑	↑	↑
Storage Requirement	↓	↓	↑

"✓" (Yes) represents the availability of each feature, while "↑" (High) and "↓" (Low) indicate the valuation.

its encryption function and key schedule to harden the differential and linear parameters of the cipher (Shantha and Arockiam, 2018a). Lastly, NVLC produced full diffusion to prevent shortcut attack accumulation that depends on the bit rotation method (Abd et al., 2018).

Two-way bit rotation directions that consist of the left and right rotations are often being implemented in light-weight algorithms. The bit rotations method is applied in SEA algorithm to provide predictable low-cost diffusion (Standaert et al., 2006). Hummingbird-2 adopted left and right bit rotations to increase the resistance of the algorithm against related-key attacks (Engels et al., 2011). By proper usage of the bit rotations method, ANU–II algorithm is able to generate a maximum number of active S-boxes in minimum number of rounds (Dahiphale et al., 2018). The bit rotations method helped NUX in reducing the number of GEs of the algorithm (Bansod et al., 2018a). Improved Simeck modified the bit rotations method to enhance its avalanche effect (Encarnacion et al., 2020).

Another type of permutation method in block ciphers is the cube permutation or known as 3-dimensional rotation that has been adopted in CUBE, 3D RECTANGLE, and LAO-3D algorithms to add better permutation and diffusion layer in the ciphers. The 3D rotation permutes the cipher bits in multiple directions, which is better than the conventional rotation method that applies one-way and two-way rotation directions (Zakaria et al., 2020a).

Besides implementing conventional predetermined rotation parameters, there are algorithms that used on the fly mathematical computations to obtain the parameters. As an example, the strength of RC5 relied on the data-dependent bit rotation method (Rivest, 1994). An advantage of the data-dependent bit rotation method used in CAST is the immunity against linear and differential attacks (Adams, 1997a).

Byte and nibble rotations are the alternative methods apart from the bit rotation functions mentioned above. The two algorithms that implemented the byte rotation function are NVLC and SEA. On the other hand, nine block ciphers adopted nibble rotation in their algorithm design. Nibble rotation implemented in SKINNY maximizes the bounds on the number of active S-boxes (Beierle et al., 2016). Using nibble rotation on LED, any rotation property between the algorithm columns is removed after the execution of two encryption rounds which is unlikely to lead to any type of attack (Guo et al., 2011).

A comparison of rotation function features is presented in Table 14. Three types of rotation functions are highlighted including one-way direction (left or right rotation), two-way directions (left and right rotation), and multiple directions (3D rotation) to observe the characteristics of each implementation in lightweight block ciphers.

The results from the comparison show that the one-way rotation is the most implemented function in lightweight block ciphers. However, the research shows that the two-way rotation function also shares similar features as the one-way rotation function. Apart from that, the only difference between the three functions is the 3D rotation that requires a higher storage requirement. Despite the disadvantage of the function, the 3D rotation managed to provide better randomization of the block cipher output compared to the conventional rotation methods, thus increasing the security of the lightweight algorithm. The adoption of the 3D rotation has increased the confusion and diffusion characteristics of existing block ciphers.

3.4. Security analysis

In this section, comparisons of security analysis of existing lightweight block ciphers are presented that highlights the fourth contribution of the paper. These comparisons are made based on the cryptanalysis which is the most common method in assessing the security strength of cryptographic algorithms. The security analysis is influenced by the substitution and permutation components implemented in each algorithm as described in earlier sections.

Cryptanalysis is an effective technique used to evaluate the strength of algorithms against cryptographic attacks or in other words, is to break the cryptosystem. The evaluation technique helps cryptographic developers to design more secure block ciphers by finding their weaknesses to improve and strengthen the algorithms. This paper focuses on the two most common attacks, which are differential cryptanalysis and linear cryptanalysis. Hence, it is necessary to analyse both cryptanalysis techniques' results to compare the security strength of existing block ciphers.

Differential cryptanalysis was proposed by Eli Biham and Adi Shamir to attack Data Encryption Standard (DES) (Biham and Shamir, 1991a). It is one of the most significant attacks for block ciphers. Differential cryptanalysis is also known as the chosen-plaintext attack that exploits the high probability of plaintext pair differences and the corresponding ciphertexts differences (Cheng et al., 2008). The technique determines the complexity of the attack whereby an upper bound can be estimated by observing the active S-boxes and the highest differential probability obtained from the Differential Distribution Table. To apply differential cryptanalysis on an n-bit block cipher, a predictable difference has to be propagated in all except for a few encryption rounds with a probability greater than 2^{1-n}.

On the other hand, linear cryptanalysis or recognized as the known-plaintext attack was used as a theoretical attack on the DES (Matsui, 1993). Linear cryptanalysis is a powerful technique to assess the resistance of a block cipher against attacks. This technique exploits the high probability occurrences of linear expression that contains plaintext, ciphertext, and secret key (Bansod, 2016). To execute a linear attack, the attacker requires the knowledge of the plaintext subset and its corresponding ciphertext. Using the Linear Approximation Table, the attacker is capable of tracking the linear trails and number of active S-boxes. If the amplitude of linear propagation is larger than $2^{-n/2}$, an n-bit block cipher can be attacked using linear cryptanalysis.

The security of a block cipher against differential and linear cryptanalysis is determined by the upper bound probability of the best differential characteristic and the linear trail of the algorithm (Sajadieh et al., 2017a). These two bounds are described by the maximum differential and linear probability of the S-boxes and the lower bound of the active S-boxes. An S-box is active in differential cryptanalysis if its input difference is nonzero. Meanwhile, for linear cryptanalysis, an S-box is active if its output mask is nonzero. A block cipher is considered more secure if the number of active S-boxes is high.

There are a few factors that may contribute to a higher number of active S-boxes from a block cipher. For instance, the robust design of PICO able to generate a large number of active S-boxes in lesser rounds that can defeat linear and differential attacks (Bansod et al., 2016a). Bit permutation of ANU helps in increasing the active S-boxes of the block cipher (Bansod et al., 2016). By proper usage of shift operators in LiCi, the algorithm can generate a maximum number of active S-boxes (Patil et al., 2017). Block shuffle and the circular shift implemented in DoT have increased the number of active S-box (Patil et al., 2019). Strong permutation layer of TED able to activate a high number of S-boxes (Thorat et al., 2020a). Other factors that influence the number of active S-boxes that have been mentioned in previous sections are the implementation of the bit rotation method, bit formulation permutation, block permutation layer, nibble permutation, block permutation table, and nibble permutation table. Fig. 5 and Fig. 6 present the comparison of existing lightweight algorithms according to the number of active S-boxes against the number of cipher rounds for differential and linear cryptanalysis.

The selection of substitution and permutation components in a block cipher give an impact on the cryptanalysis results. Maximum encryption rounds that can be attacked would give the developer a confidence level in determining the number of rounds required by the algorithm to prevent successful cryptanalysis attacks. The maximum attack rounds of differential and linear cryptanalysis obtained by existing algorithms are displayed in Fig. 7 and Fig. 8. On average, the number of full encryption rounds set for each block ciphers is double the size of the maximum significant attack resulting from the cryptanalysis. This is sufficient to avoid successful attacks on the full rounds of the cipher that may lead to security problems and secret key exposure.

From the analysis involving existing lightweight block ciphers, multiple methods were adopted by developers to design secure algorithms in an effort to defeat differential and linear cryptanalysis. Bitwise permutation in PRINTcipher gives full cipher dependency after minimal encryption rounds (Knudsen et al., 2010). Improved LBlock managed to activate 32 S-boxes in 13 rounds using the modified bit rotation method, thus enhancing the security of its original algorithm (Al-Dabbagh and Al-Shaikhli, 2013). RECTANGLE established excellent security using a bit-slice technique with a number of rounds that can be attacked (Zhang et al., 2015). GIFT also attains full diffusion in three rounds with its S-box and the bit permutation to improve differential and linear bounds (Banik et al., 2017). The maximum number of active S-boxes in minimum encryption rounds achieved in NUX can prevent successful differential and linear attacks (Bansod et al., 2018a). Furthermore, data permutation of ACT combined with S-box properties gives complete diffusion in three rounds offers excellent security (Jithendra and Kassim, 2020a).

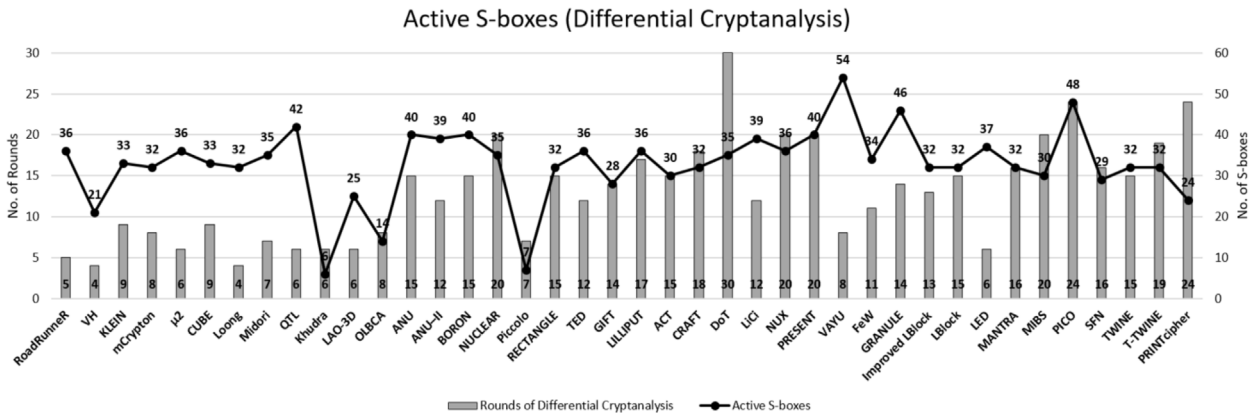

Fig. 5. Active S-boxes of differential cryptanalysis.

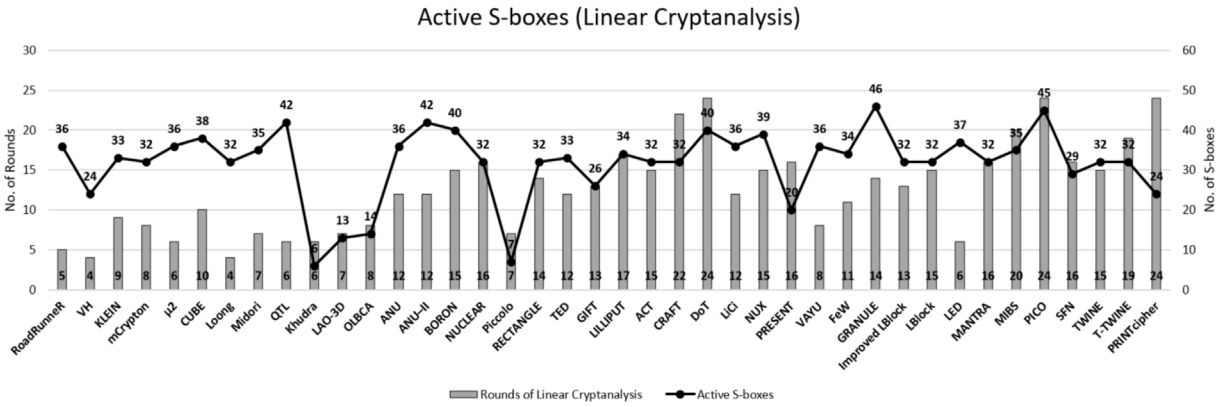

Fig. 6. Active S-boxes of linear cryptanalysis.

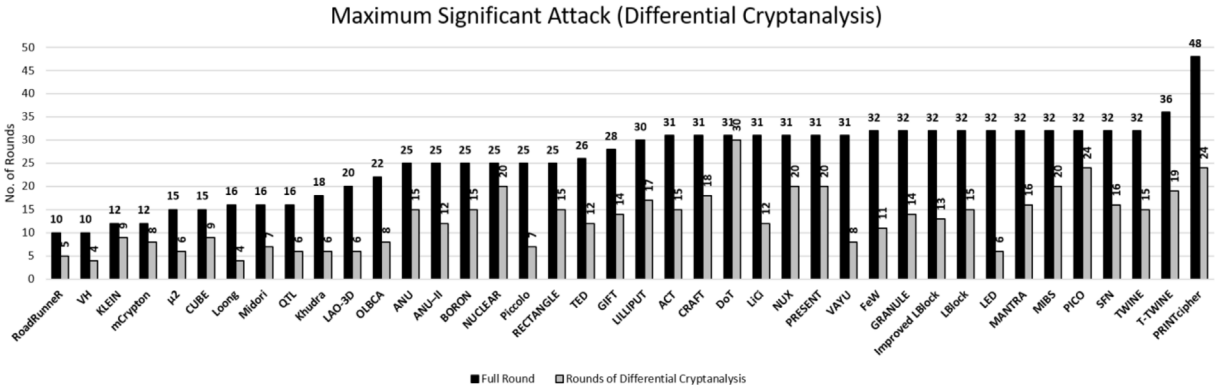

Fig. 7. Maximum attack rounds of differential cryptanalysis.

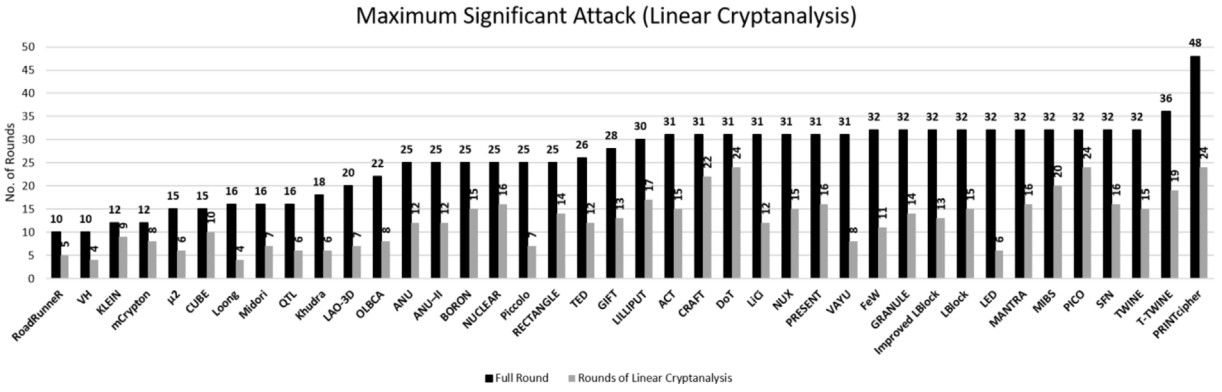

Fig. 8. Maximum attack rounds of linear cryptanalysis.

3.5. Evolution of lightweight block ciphers

There are various characteristics to be assessed in designing lightweight block ciphers which include power consumption, memory requirement, complexity, robustness, throughput, and strength against cryptanalytic attacks. Hundreds of new algorithms have been developed as early as the 1990's until today. New techniques were introduced to improve the previous lightweight algorithms. However, there are still many existing algorithms that are being referred to in the development of new algorithms. For the fifth contribution of the paper, this section highlights the justification behind the decision of new algorithms to adopt existing algorithm components in their design. Fig. 9 displays the evolution of cryptographic algorithms from the year 1990 to 2022. In the figure, the grey coloured oval shape indicates the referred block cipher while the white coloured rectangular shape represents the referring algorithm.

3.5.1. Encryption algorithm

The major decision for choosing PRESENT as the base algorithm in many block ciphers is due to its ultra-lightweight characteristics and implementation performance (Jawad and Hoomod, 2019). PRESENT is examined as an efficient algorithm for resource-constrained devices. Bitwise permutation method of PRESENT provides fast diffusion without hardware implementation cost (Liu et al., 2014a). The algorithm focused on hardware efficiency and

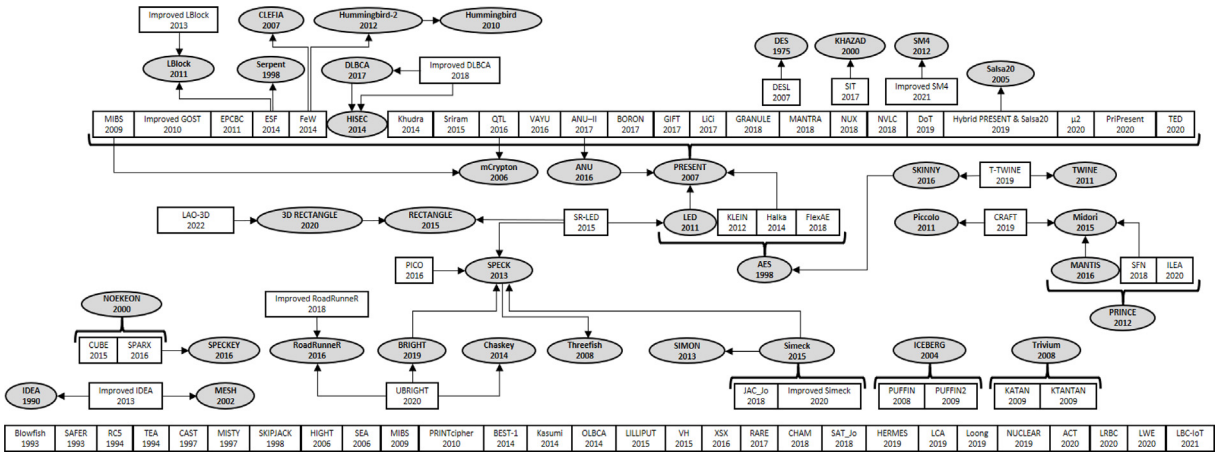

Fig. 9. Evolution of cryptographic algorithms.

achieved simplicity of linear layer that can be implemented with a minimum number of processing elements. PRESENT is also resistant against both differential and linear cryptanalysis (Yap et al., 2011). Among algorithms that adopted PRESENT components in their block cipher design are µ2, EPCBC, ESF, GIFT, HISEC, Hybrid PRESENT & Salsa20, Improved GOST, Khudra, LED, and QTL.

Another algorithm being referred to by lightweight developers is the AES block cipher. AES-like block cipher structure appears to be the ideal solution for a secure cryptographic design. LED used AES-like design principles which allows the algorithm to derive simple bounds on the number of active S-boxes during encryption (Guo et al., 2011). For better software performance, the design of KLEIN adopted AES byte-oriented structure (Gong et al., 2011). Later, SKINNY applied the ShiftRows function from AES but rotated the cell array to the right instead of left for better permutation (Beierle et al., 2016).

LBlock that employed variant Feistel structure with SPN round function is adopted by ESF to make the algorithm lighter without compromising security (Liu et al., 2014a). LBlock can also be implemented efficiently in both hardware and software platforms. Apart from that, an improved version of LBlock used bit permutation instead of word permutation to improve the algorithm diffusion property (Al-Dabbagh and Al-Shaikhli, 2013).

SIMON is an optimized block cipher for hardware, while SPECK is optimized for software implementation. Later, Simeck inherits SIMON and SPECK block ciphers, which is a more compact and efficient that provides the same level of security and properties. In 2018, Simeck is adopted by JAC_Jo with better performance in terms of power consumption, latency, and throughput without sacrificing security (Shantha and Arockiam, 2018a). Simeck is built on ARX structure and operates fast on software with different file sizes demonstrating swift encryption time that is designed for IoT environment (Encarnacion et al., 2020).

NOEKEON employs a sequence of self-inverse transformations that can be implemented easily in hardware or software (Daemen et al., 2000). The algorithm is designed according to a variant of the wide-trail strategy. In 2015, CUBE implemented NOEKEON S-box in the design (Berger et al., 2015). Later, the L-function of SPARX is borrowed from NOEKEON using 16 and 32-bit rotations because it is cheap on processors (Dinu et al., 2016).

RECTANGLE enables lightweight and fast implementations using the bit-slice method. The block cipher is developed based on a simple model design suitable for low-cost hardware and efficient deployment in software (Zhang et al., 2015). RECTANGLE has efficient encryption speed performance among the existing lightweight algorithms. In 2020, an enhancement of the RECTANGLE was proposed using a 3-dimensional permutation known as 3D RECTANGLE to improve its diffusion property (Zakaria et al., 2020a).

LED is an algorithm with a robust design that can provide high security. Its cryptanalysis resistance and design are competent enough to be secure against any possible types of attacks (Guo et al., 2011). The algorithm has low GEs and memory requirements that are suitable for applications in limited-resource devices such as RFID tags. An improvement of the block cipher was introduced in SR-LED that modified the original S-box and key schedule algorithm (Patil et al., 2015a).

RoadRunneR is one of the most software-efficient light-weight block ciphers. The security strength of the algorithm is provable against linear and differential attacks. In 2018, the improved RoadRunneR was developed by introducing a merged substitution function (Liu et al., 2018). Two years later, UBRIGHT adopted the key schedule algorithm of RoadRunneR in the block cipher (Sehrawat and Gill, 2020a).

There is a long list of existing algorithms being adopted by new designs such as DESL that inherited DES due to its efficiency in hardware with its bit permutation and small S-boxes (Leander et al., 2007). Complete diffusion in a single round achieved in IDEA is enhanced in the improved IDEA algorithm to provide resistance against cryptanalysis attacks (Lerman et al., 2013). Improved DLBCA used the substitution layer of the original DLBCA that produces confusion property and gives nonlinearity to the cipher (Al-Dabbagh et al., 2018a). ANU-II improved the efficiency of the original ANU in terms of memory size, latency, throughput, and power (Dahiphale et al., 2018). FeW implemented CLEFIA that has been a standard in the International Organisation of Standardisation (ISO) and International Electrotechnical Commission (IEC) (Kumar et al., 2019a).

The list of algorithms continues with KATAN that adopts a stream cipher structure similar to Trivium using a two-register variant of Bivium (De Cannière et al., 2009). MANTIS adopted the structure of MIDORI to ensure a high number of active S-boxes and implemented the low-latency design of PRINCE (Beierle et al., 2016). The round function of SPECKEY is chosen by SPARX because of its superior implementation properties (Dinu et al., 2016). T-TWINE reused the original TWINE design with an additional tweakey algorithm of SKINNY to build a tweakable block cipher with minimum cost for both design and implementation (Sakamoto et al., 2020a). Lastly, improved SM4 enhanced the original SM4 that is a commercial algorithm developed by the country of China (Chen et al., 2021). Other algorithms that are not discussed in this section can be referred to in Fig. 10.

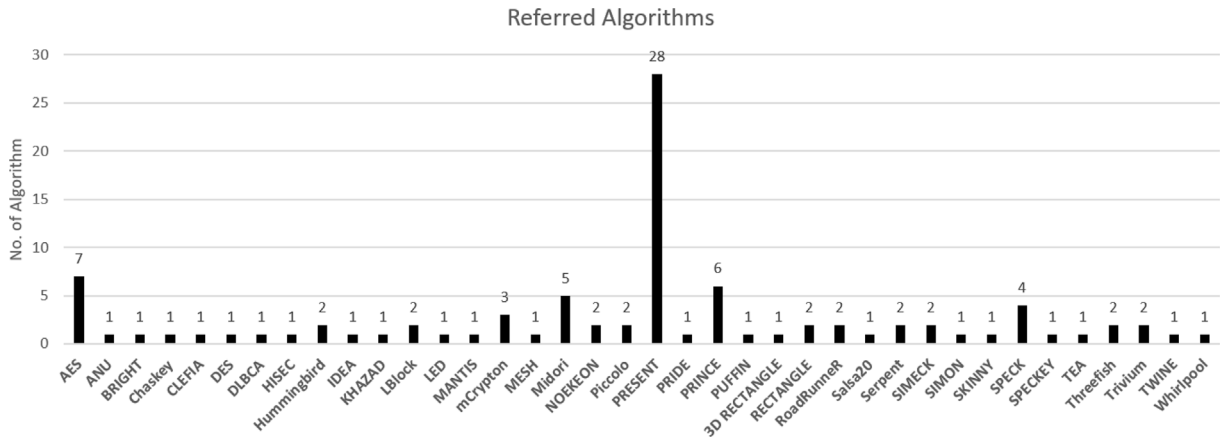

Fig. 10. Referred algorithms.

3.5.2. Key schedule algorithm

PRESENT is the most popular lightweight block cipher and the key schedule algorithm is considered one of the most robust key-scheduling designs (Thorat et al., 2020a). Since PRESENT key schedule uses the round-dependent counter and a nonlinear operation to mix the contents of the key register, the algorithm is secure against related key attacks (Izadi et al., 2009). There is no evidence of a successful cryptanalytic attack can be applied to PRESENT key scheduling algorithm. ANU–II, Sriram, and VAYU key scheduling are motivated by PRESENT design. NVLC is designed with efficient key schedule algorithm of PRESENT to defeat cryptanalytic attacks (Abd et al., 2018). DoT is encouraged by PRESENT key scheduling because it is a strong algorithm (Patil et al., 2019).

SPECK key schedule is the other algorithm being implemented in new designs. National Security Agency (NSA) launched SPECK which is considered as one of the most ultra-lightweight ciphers (Beaulieu et al., 2015). SPECK key scheduling is compact in memory size requirement that makes a big impact on the GEs of the algorithm. SPECK generates unpredicted keys that help in increasing resistance against the slide and meet in the middle attacks (Patil et al., 2015a). The key schedule algorithm has a great avalanche effect as one-bit change can change the complete key structure that makes the round keys random. SPECK is best suited in improving the key scheduling of block ciphers such as SR-LED to resist related key attacks (Patil et al., 2015a). Simeck learns the idea of reusing the round function to update the roundkey registers from SPECK (Yang et al., 2015). The key scheduling of PICO is motivated by SPECK that does not include the nonlinear layer in the design thus making it compact (Bansod et al., 2016a). BRIGHT round keys generation method is inspired by SPECK key schedule algorithm due to its simple internal structure (Sehrawat and Gill, 2019a).

Apart from PRESENT and SPECK, the following key schedule algorithms are adopted in new lightweight block ciphers. The round key bits in the improved IDEA are overlapped and nonlinearly dependent on each other to enhance the simple round key bit permutation mapping in the original IDEA (Lerman et al., 2013). SIT used an F-function in its key schedule algorithm that applied tweaked KHAZAD to ensure complex dependency of the output bits on its input bits (Usman et al., 2017). JAC_Jo adopted Simeck that is designed to have a small area because of the simplified key schedule (Shantha and Arockiam, 2018a). Hybrid PRESENT & Salsa20 implemented the key schedule algorithm of Salsa20 that is considered one of the efficient block ciphers to be used in resource-constrained devices (Jawad and Hoomod, 2019). To ensure the key schedule of CRAFT is small and lightweight, the same round keys are used in an alternating way as in Piccolo and Midori (Beierle et al., 2019). Key scheduling of UBRIGHT adopted Chaskey and RoadRunneR that do not apply any key schedule algorithm where the round keys are generated on the fly (Sehrawat and Gill, 2020a).

Section 3.5 summarizes that the development of light-weight block ciphers is still ongoing even though the research has been in place since 1990. The evolution of lightweight algorithms demonstrates the continuity of the research that has managed to gain interest from researchers around the world. Every year, new ideas and innovations are proposed to increase the strength of lightweight algorithms. However, the design of the old algorithms cannot be neglected as some of their cryptographic components are still valid until now. Therefore, the developer may take some of the ideas from existing designs and propose improvements from the cryptographic components to develop secure lightweight block ciphers.

4. Recommendation

In designing a lightweight algorithm, several factors need to be taken into account. Firstly, understanding the security evaluation criteria adopted by international lightweight cryptography projects as presented in Section 3.1 is required to comprehend the lightweight block cipher requirements. These criteria will guide the developers in preparing a block cipher that is competitive with standard algorithms to facilitate comparisons in terms of performance and security.

Then, the objective of developing a lightweight block cipher should be determined, whether the algorithm is designed to focus on implementation performance, aim for high security, or a combination of performance and security. The objective will help in the development of the block cipher because the design of an algorithm will determine where it is suitable to be applied in the future. Different types of algorithm motivation can lead to different design methodologies and analysis techniques to justify the selection of lightweight block cipher. Therefore, it is important to maintain the design objective throughout the development of the algorithm.

After that, existing algorithms need to be referred to as a benchmark for the developers in designing their lightweight block cipher. Each algorithm has its characteristics such as structures, components, and strengths as explained in Section 3.1 and Section 3.5. These characteristics influence an algorithm in achieving the development objective. From the research, substitution and permutation are the most common components used by developers that contribute to the addition of confusion and diffusion properties in the algorithms. Besides that, the research highlights existing algorithms that are being referred by recent lightweight ciphers. The cryptographic components from the existing block ciphers might be useful in constructing a new algorithm.

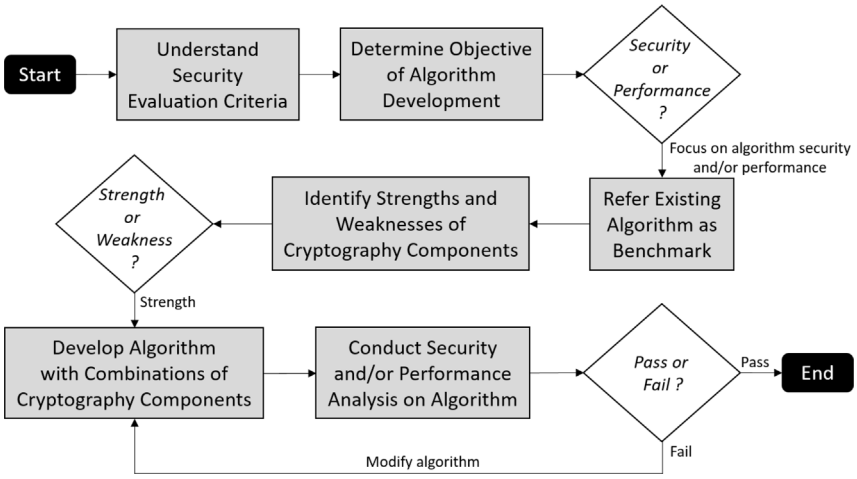

Fig. 11. Lightweight block cipher development process.

Next, the strengths of each cryptographic component implemented in existing algorithms have to be identified as the foundation of the new block cipher. These components have their advantages that will determine the strength of the algorithm as analyzed in Section 3.2 and Section 3.3. A secure algorithm is developed with a combination of several cryptographic components such as substitution and permutation. Therefore, the developers need to find a proper combination of components that can produce a secure algorithm. Findings from the research recommend some implementations of substitution and permutation components in the lightweight algorithm. For the substitution component, 4-bit S-box is suggested over the other types of S-boxes due to its adequate security for resource-constrained devices. Meanwhile, two observations were discovered on the permutation component that consists of permutation and rotation functions. Feistel permutation, formulation permutation, and table permutation functions can be used to increase the security of lightweight block cipher. On the other hand, one-way rotation, two-way rotation, and 3D rotation functions are recommended for algorithm implementation.

Finally, security analysis on the designed algorithm as discussed in Section 3.4 must be conducted to justify the security strength of the new algorithm. Usually, the best algorithm design will only be obtained after several modifications. Each modification must be tested multiple times to validate the strength of the algorithm as shown in Fig. 11. This is the most difficult and time-consuming stage in the development of a cryptographic algorithm but it is one of the most important parts of the research work. The research recommends cryptanalysis attacks as the methods to evaluate the security strength of lightweight block cipher.

5. Conclusion

This paper has provided a comprehensive review of lightweight cryptographic solutions for resource-constrained environments that include 101 selected lightweight block ciphers. From the analysis of the lightweight block ciphers, it is observed that numerous techniques can be implemented to design a secure cryptographic algorithm. The analysed cryptographic components and characteristics of the lightweight block ciphers have provided abundant information from the developers' point of view. Combination of substitution and permutation functions are found to be the solution in designing a secure algorithm as both components are able to provide confusion and diffusion properties that are required in lightweight block ciphers. The development trend shows that most recent lightweight block ciphers implement single 4-bit S-box, apply 128-bit or lower key size, and execute more than 20 encryption rounds. In the future, it is expected that the development trend will continue in order to balance the security and efficiency. Meeting the objective of this paper, the findings from the research would help researchers in terms of understanding the secure design of lightweight block cipher. From this research, it is encouraged that more researchers design lightweight block ciphers that are suitable for resource-constrained devices by emphasizing security aspects to deal with increasingly challenging cyber-attacks.

Funding

This research was funded by the Ministry of Higher Education (MOHE) of Malaysia under the Fundamental Research Grants Scheme (FRGS/1/2019/ICT03/USIM/02/1). The authors also would like to express their gratitude to Universiti Sains Islam Malaysia (USIM) and CyberSecurity Malaysia (CSM) for the support and facilities provided.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Open Access funding provided by the Universiti Sains Islam Malaysia.

References

Abd Al-Rahman, Seddiq Q., Sagheer, Ali Makki, Dawood, Omar A., 2018. NVLC: New variant lightweight cryptography algorithm for internet of things. In: Annual International Conference on Information and Sciences. IEEE, pp. 176–181.

Aboshosha Bassam W., Dessouky Mohamed M., Ramdan Rabie A., El-Sayed, Ayman, 2019. LCA- Lightweight cryptographic algorithm for IoT constraint resources. In: International Conference on Electronic Engineering, pp. 374–380.

Adams, Carlisle M., 1997a. Constructing symmetric ciphers using the CAST design procedure. Des. Codes, Cryptogr 12, 3, 283–316.

Albrecht, Martin R., Driessen, Benedikt, Kavun, Elif Bilge, Leander, Gregor, Paar, Christof, Yalçın, Tolga, 2014. Block ciphers - Focus on the linear layer (feat PRIDE). In: Annual Cryptology Conference. Springer, Berlin, Heidelberg, pp. 57–76.

Al-Dabbagh, Sufyan Salim Mahmood, 2017a. Design 32-bit lightweight block cipher algorithm (DLBCA). Int. J. Comput. Appl 166 (8), 17–20.

Al-Dabbagh, Sufyan Salim Mahmood, Al-Shaikhli, Imad Fakhri Taha, 2013. Improving the security of LBlock lightweight algorithm using bit permutation. In: International Conference on Advanced Computer Science Applications and Technologies. IEEE, pp. 296–299.

Al-Dabbagh, Sufyan Salim Mahmood, Al-Shaikhli, Imad Fakhri Taha, 2014. OLBCA: A new lightweight block cipher algorithm. In: International Conference on Advanced Computer Science Applications and Technologies. IEEE, pp. 15–20.

Al-Dabbagh Sufyan Salim Mahmood , Al Shaikhli Imad Fakhri Taha, Alahmad Mohammad A., 2014. HISEC: A new lightweight block cipher algorithm. In: International Conference on Security of Information and Networks, pp. 151–156.

Al-Dabbagh, Sufyan Salim Mahmood, Sulaiman, Alyaa Ghanim, Shaikhli, Imad Fakhri Taha Al, Al-Enezi, Khalid Abdulkareem, Alenezi, Abdulrahman Yousef, 2018a. Improving the cost factor of DLBCA lightweight block cipher algorithm. Indones. J. Electr. Eng. Comput. Sci 10 (2), 786–791.

Alghafis, Abdullah, Munir, Noor, Khan, Majid, 2021a. An encryption scheme based on chaotic Rabinovich-Fabrikant system and S8 confusion component. Multimed. Tools Appl. 80, 7967–7985.

Banik, Subhadeep, Bogdanov, Andrey, Isobe, Takanori, Shibutani, Kyoji, Hiwatari, Harunaga, Akishita, Toru, Regazzoni, Francesco, 2015. Midori: A block cipher for low energy. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, pp. 411–436.

Banik Subhadeep, Pandey Sumit Kumar, Peyrin Thomas, Sasaki Yu, Sim Siang Meng, Todo Yosuke, 2017. GIFT: A small present towards reaching the limit of lightweight encryption. In: International Conference on Cryptographic Hardware and Embedded Systems, pp. 321–345.

Bansod, Gaurav, 2016. A new ultra lightweight encryption design for security at node level. Int. J. Secur. Appl. 10 (12), 111–128.

Bansod, Gaurav, Patil, Abhijit, Sutar, Swapnil, Pisharoty, Narayan, 2016. ANU: an ultra lightweight cipher design for security in IoT. Secur. Commun. Networks 9 (18), 5238–5251.

Bansod, Gaurav, Pisharoty, Narayan, Patil, Abhijit, 2016a. PICO: An ultra lightweight and low power encryption design for ubiquitous computing. Def. Sci. J. 66 (3), 259–265.

Bansod, Gaurav, Pisharoty, Narayan, Patil, Abhijit, 2017a. BORON: An ultra-lightweight and low power encryption design for pervasive computing. Front. Inf. Technol. Electron. Eng. 18 (3), 317–331.

Bansod, Gaurav, Pisharoty, Narayan, Patil, Abhijit, 2018a. GRANULE: An ultra lightweight cipher design for embedded security. IACR Cryptol. IACR Cryptol. ePrint Arch, 1–12.

Bansod, Gaurav, Pisharoty, Narayan, Patil, Abhijit, 2018a. MANTRA: An ultra lightweight cipher design for ubiquitous computing. Int. J. Ad Hoc Ubiquitous. Comput. 28 (1), 13–26.

Bansod, Gaurav, Sutar, Swapnil, Patil, Abhijit, Pisharoty, Narayan, 2018a. NUX: A lightweight block cipher for security at wireless sensor node level. Int. J. Bioeng. Life Sci. 5 (1), 1–8.

Barreto Paulo S.L.M., Rijmen Vincent, 2000. The Khazad legacy-level block cipher. In: Primitive submitted to NESSIE 97 1–20.

Baysal, Adnan, Şahin, Suhap, 2015. RoadRunneR: A small and fast bitslice block cipher for low cost 8-bit processors. In: Lightweight Cryptography for Security and Privacy. Springer, Cham, pp. 58–76.

Beaulieu, Ray, Treatman-Clark, Stefan, Shors, Douglas, Weeks, Bryan, Smith, Jason, Wingers, Louis, 2015. The SIMON and SPECK lightweight block ciphers. Annual Design Automation Conference. IEEE, 1–6.

Beierle, Christof, Jean, Jeremy, Kolbl, Stefan, Leander, Gregor, Moradi, Amir, Thomas Peyrin, Yu., Sasaki, Pascal Sasdrich, Sim, Siang Meng, 2016. The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Annual International Cryptology Conference. Springer, Berlin, Heidelberg, pp. 123–153.

Beierle, Christof, Leander, Gregor, Moradi, Amir, Rasoolzadeh, Shahram, 2019. CRAFT: Lightweight tweakable block cipher with efficient protection against DFA attacks. IACR Trans. Symmetric Cryptol 1, 5–45.

Berger, Thierry P., Francq, Julien, Minier, Marine, 2015. CUBE cipher: A family of quasi-involutive block ciphers easy to mask. In: International Conference on Codes, Cryptology, and Information Security. Springer, Cham, pp. 89–105.

Berger, Thierry P., Francq, Julien, Minier, Marine, Thomas, Gael, 2015a. Extended generalized feistel networks using matrix representation to propose a new lightweight block cipher: LILLIPUT. IEEE Trans. Comput 65 (7), 2074–2208.

Biham, Eli, Shamir, Adi, 1991a. Differential cryptanalysis of DES-like cryptosystems. J. Cryptol 4 (1), 3–72.

Biswas, A., Majumdar, A., Nath, S., Dutta, A., Baishnab, K.L., 2020a. LRBC: A lightweight block cipher design for resource constrained IoT devices. J. Ambient Intell. Humaniz. Comput., 1–15

Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C., 2007. PRESENT: An ultra-lightweight block cipher. In: International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, pp. 450–466.

Borghoff, Julia, Canteaut, Anne, Guneysu, Tim, Kavun, Elif Bilge, Knezevic, Miroslav, Knudsen, Lars R., Leander, Gregor, Nikov, Ventzislav, Paar, Christof, Rechberger, Christian, Rombouts, Peter, Thomsen, Søren S., Yalcın, Tolga, 2012. PRINCE - A low-latency block cipher for pervasive computing applications. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, pp. 208–225.

Chen, Zhanwen, Chen, Jiageng, Meng, Weizhi, Teh, Je Sen, Li, Pei, Ren, Bingqing, 2020a. Analysis of differential distribution of lightweight block cipher based on parallel processing on GPU. J. Inf. Secur. Appl. 55, 1–10.

Chen, B.W., Xia, X., Liang, Q.M., Zhong, W.D., 2021. Lightweight design of SM4 algorithm and realization of threshold scheme. In: Journal of Physics: Conference Series, pp. 1–14.

Cheng, Huiju, Heys, Howard M., Wang, Cheng, 2008. PUFFIN: A novel compact block cipher targeted to embedded digital systems. In: EUROMICRO Conference on Digital System Design Architectures, Methods and Tools. IEEE, pp. 383–390.

Cui, Ting, Jin, Chenhui, 2017a. Classification of SPN structures from the viewpoint of structural cryptanalysis. IEEE Access 6, 9733–9739.

CyberSecurity Malaysia, 2021. MySEAL - National Trusted Cryptographic Algorithm List. Retrieved January 29, 2022 from https://myseal.cybersecurity.my/en/index.html.

Daemen, Joan, Peeters, Michaël, Van Assche, Gilles, Rijmen, Vincent, 2000. Nessie proposal: NOEKEON. In: First Open NESSIE Workshop, pp. 213–230.

Dahiphale Vijay, Bansod Gaurav, Patil Jagdish, 2018. ANU-II: A fast and efficient lightweight encryption design for security in IoT. In: International Conference on Big Data, IoT and Data Science, IEEE, pp. 130–137.

Dai, Xuejun, Lu, Yuhua Huang, Chen, Tingting Lu, Fei, Su., 2015. VH: A lightweight block cipher based on dual pseudo-random transformation. In: International Conference on Cloud Computing and Security. Springer, Cham, pp. 3–13.

Das, Sourav, 2014a. A lightweight, software friendly block cipher using ultra-lightweight 8-bit S-box. IACR Cryptol. ePrint Arch, 1–16.

De Cannière, Christophe, Dunkelman, Orr, Knezevic, Miroslav, 2009. KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In: International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, pp. 272–288.

Dhanda, Sumit Singh, Singh, Brahmjit, Jindal, Poonam, 2020a. Lightweight cryptography: A solution to secure IoT. Wirel. Pers. Commun. 112, 1947–1980.

Dinu, Daniel, Perrin, Léo, Udovenko, Aleksei, Velichkov, Vesselin, Großschädl, Johann, Biryukov, Alex, 2016. Design strategies for ARX with provable bounds: Sparx and LAX (Full Version). In: International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, pp. 484–513.

Encarnacion, Philipcris C., Gerardo, Bobby D., Hernandez, Alexander A., 2020. Modified round function of SIMECK 32/64 block cipher. Int. J. Adv. Trends Comput. Sci. Eng. 9, 1, 258–266.

Engels, Daniel, Fan, Xinxin, Gong, Guang, Hu, Honggang, Smith, Eric M., 2010. Hummingbird: Ultra-lightweight cryptography for resource-constrained devices. In: International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, pp. 3–18.

Engels, Daniel, Saarinen, Markku-Juhani O., Schweitzer, Peter, Smith, Eric M., 2011. The Hummingbird-2 lightweight authenticated encryption algorithm. In: International Workshop on Radio Frequency Identification: Security and Privacy Issues. Springer, Berlin, Heidelberg, pp. 19–31.

ETSI, 2014a. Universal mobile telecommunications system (UMTS); LTE; 3G security; specification of the 3GPP confidentiality and integrity algorithms; document 2: Kasumi specification (3GPP TS 35.202 version 12.0.0 Release 12).

Girija, M., Manickam, P., Ramaswami, M., 2020a. PriPresent: An embedded prime lightweight block cipher for smart devices. Peer-to-Peer Netw. Appl. 14, 1–11.

Gong, Zheng, Nikova, Svetla, Law, YeeWei, 2011. KLEIN: A new family of lightweight block ciphers. In: International Workshop on Radio Frequency Identification: Security and Privacy Issues. Springer, Berlin, Heidelberg, pp. 1–18.

Guo, Jian, Peyrin, Thomas, Poschmann, Axel, Robshaw, Matt, 2011. The LED block cipher*. In: International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, pp. 326–341.

Hong, Deukjo, Sung, Jaechul, Hong, Seokhie, Lim, Jongin, Lee, Sangjin, Koo, Bon-Seok, Lee, Changhoon, Chang, Donghoon, Lee, Jesang, Jeong, Kitae, Kim, Hyun, Kim, Jongsung, Chee, Seongtaek, 2006. HIGHT: A new block cipher suitable for low-resource device. In: International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, pp. 46–59.

Izadi, Maryam, Sadeghiyan, Babak, Sadeghian, Seyed Saeed, Khanooki, Hossein Arabnezhad, 2009. MIBS: A new lightweight block cipher. In: International Conference on Cryptology and Network Security. Springer, Berlin, Heidelberg, pp. 334–348.

Jawad Kubba, Zaid M., Hoomod, Haider K., 2019. A hybrid modified lightweight algorithm combined of two cryptography algorithms PRESENT and Salsa20 using chaotic system. In: International Conference of Computer and Applied Sciences. IEEE, pp. 199–203.

Jha, Pallavi, Zorkta, Haythem Yosef, Allawi, Dahham, Al-Nakkar, Maher Riad, 2020. Improved lightweight encryption algorithm (ILEA). In: International Conference for Emerging Technology. IEEE, pp. 1–4.

Jithendra, K.B., Kassim, Shahana T., 2020a. ACT: An ultra-light weight block cipher for internet of things. Int. J. Comput. Digit. Syst. 9 (5), 921–929.

John, Jacob, 2014a. BEST-1: A light weight block cipher. IOSR J. Comput. Eng. 16 (2), 91–95.

Knudsen, Lars, Leander, Gregor, Poschmann, Axel, Robshaw, Matthew J.B., 2010. PRINTcipher: A block cipher for IC-printing. In: International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, pp. 16–32.

Kolay, Souvik, Mukhopadhyay, Debdeep, 2014. Khudra: A new lightweight block cipher for FPGAs. In: International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer, Cham, pp. 126–145.

Koo, Bonwook, Roh, Dongyoung, Kim, Hyeonjin, Jung, Younghoon, Lee, Dong-Geon, Kwon, Daesung, 2017. CHAM: A family of lightweight block ciphers for resource-constrained devices. In: International Conference on Information Security and Cryptology. Springer, Cham, pp. 3–25.

Kumar, Manoj, Pal, Saibal K., Panigrahi, Anupama, 2019a. FeW: A lightweight block cipher. Turkish J. Math. Comput. Sci 11 (2), 58–73.

Lai, Xuejia, Massey, James L., 1991. A proposal for a new block encryption standard. In: Workshop on the Theory and Application of of Cryptographic Techniques. Springer, Berlin, Heidelberg, pp. 389–404.

Leander, Gregor, Paar, Christof, Poschmann, Axel, Schramm, Kai, 2007. New lightweight DES variants. In: International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, pp. 196–210.

Lerman, Liran, Nakahara, Jorge, Veshchikov, Nikita, 2013. Improving block cipher design by rearranging internal operations. In: International Conference on Security and Cryptography. IEEE, pp. 1–12.
Li, Lang, Liu, Botao, Wang, Hui, 2016a. QTL: A new ultra-lightweight block cipher. Microprocess. Microsyst 45, 45–55.
Li, Lang, Liu, Botao, Zhou, Yimeng, Zou, Yi, 2018. SFN: A new lightweight block cipher. Microprocess. Microsyst. 60, 138–150.
Lim Chae Hoon, Korkishko Tymur, 2005. mCrypton - A lightweight block cipher for security of low-cost RFID tags and sensors. In: International Workshop on Information Security Applications. Springer, Berlin, Heidelberg, pp. 243–258.
Liu, Xuan, Zhang, Wen-ying, Liu, Xiang-zhong, Liu, Feng, 2014a. Eight-sided fortress: A lightweight block cipher. J. China Univ. Posts Telecommun. 21 (1), 104–128.
Liu, Juhua, Li, Wei, Bai, Guoqiang, 2018. An improved s-box of lightweight block cipher Roadrunner for hardware optimization. In: China Semiconductor Technology International Conference. IEEE, pp. 1–4.
Liu, Bo-Tao, Li, Lang, Wu, Rui-Xue, Xie, Ming-Ming, Li, Qiu-Ping, 2019a. Loong: A family of involutional lightweight block cipher based on SPN structure. IEEE Access 7, 136023–136035.
Marsola do Nascimento Eduardo, Moreira Xexéo José Antônio. 2019a. FlexAEAD v1.1 -A lightweight AEAD cipher with integrated authentication. J. Inf. Secur. Cryptogr 6(1), 15–24.
Massey, James L., 1993. SAFER K-64: A byte-oriented block-ciphering algorithm. In: International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, pp. 1–17.
Matsui, Mitsuru, 1993. Linear cryptanalysis method for DES cipher. In: Workshop on the Theory and Application of of Cryptographic Techniques. Springer, Berlin, Heidelberg, pp. 386–397.
Matsui, Mitsuru, 1997. New block encryption algorithm MISTY. In: International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, pp. 54–68.
Mohd, Bassam J., Hayajneh, Thaier, Vasilakos, Athanasios V., 2015a. A survey on lightweight block ciphers for low-resource devices. J. Netw. Comput. Appl. 58, 73–93.
Moher David, Liberati Alessandro, Tetzlaff Jennifer, Altman Douglas G., The PRISMA Group. 2009a. Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. PLoS Med 6(7), 1–6.
Măluțan, Sergiu Beniamin, Dragomir, Ioana Roxana, Lazăr, Marilena, Vitan, Dragoș, 2019. HERMES, a proposed lightweight block cipher used for limited resource devices. In: International Conference on Speech Technology and Human-Computer Dialogue. IEEE, pp. 1–6.
Nayancy, Sandip Dutta, Chakraborty, Soubhik, 2020. A survey on implementation of lightweight block ciphers for resource constraints devices. J. Discret Math. Sci. Cryptogr., 1–22
NSA, 1998a. Skipjack and KEA Algorithm Specifications. 1– 23.
Omrani Tasnime, Becheikh Rabei, Mannai Olfa, Rhouma Rhouma, Belghith Safya, 2018. RARE: A robust algorithm for rapid encryption. In: International Conference for Internet Technology and Secured Transactions. IEEE, pp. 23–28.
Patil, Abhijit, Bansod, Gaurav, Pisharoty, Narayan, 2015a. Hybrid lightweight and robust encryption design for security in IoT. Int. J. Secur. Appl. 9 (12), 85–98.
Patil Jagdish, Bansod Gaurav, Kant Kumar Shashi, 2017. LiCi: A new ultra-lightweight block cipher. In: International Conference on Emerging Trends and Innovation in ICT. IEEE, pp. 40–45.
Patil, Jagdish, Bansod, Gaurav, Kant, Kumar Shashi, 2019. DoT: A new ultra-lightweight SP network encryption design for resource-constrained environment. International Conference on Data Engineering and Communication Technology, Advances in Intelligent Systems and Computing, vol. 828. Springer, Singapore, pp. 249–257.
Pei, Chao, Xiao, Yang, Liang, Wei, Han, Xiaojia, 2018a. Trade-off of security and performance of lightweight block ciphers in industrial wireless sensor networks. EURASIP J. Wirel. Commun. Netw. 117, 1–18.
Poschmann, Axel, Ling, San, Wang, Huaxiong, 2010. 256 bit standardized crypto for 650 GE - GOST revisited. In: International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, pp. 219–233.
Preneel, Bart, 2002. New European schemes for signature, integrity and encryption (NESSIE): A status report. In: International Workshop on Public Key Cryptography. Springer, Berlin, Heidelberg, pp. 297–309.
Ramadan, Rabie A., Aboshosha, Bassam W., Yadav, Kusum, Alseadoon, Ibrahim M., Kashout, Munawar J., Elhoseny, Mohamed, 2021. LBC-IoT: Lightweight block cipher for IoT constraint devices. Comput. Mater. Contin. 67 (3), 3563–3579.
Ramudu, Sri, Shanthi, G., 2015a. Implementation of an ultra-lightweight block cipher. Int. J. Mag. Eng. Technol. Manag. Res. 2, 2, 233–243.
Rivest, Ronald L., 1994. The RC5 encryption algorithm. In: International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, pp. 86–96.
Sajadieh, Mahdi, Mirzaei, Arash, Mala, Hamid, Rijmen, Vincent, 2017a. A new counting method to bound the number of active S-boxes in Rijndael and 3D. Des. Codes, Cryptogr. 83, 2, 327–343.
Sakamoto Kosei, Minematsu Kazuhiko, Shibata Nao, Shigeri Maki, Kubo Hiroyasu, Funabiki Yuki, Bogdanov Andrey, Morioka Sumio, Isobe Takanori, 2020a. Tweakable TWINE: Building a tweakable block cipher on generalized Feistel structure. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. 103(12), 1629–1639.
Salunke, Rutuja, Bansod, Gaurav, Naidu, Praveen, 2019. Design and implementation of a lightweight encryption scheme for wireless sensor nodes. In: Advances in Intelligent Systems and Computing. Springer, Cham, pp. 566–581.
Santos, Ricardo Jorge, Vieira, Marco, Bernardino, Jorge, 2016. XSX: Lightweight encryption for data warehousing environments. In: International Conference on Big Data Analytics and Knowledge Discovery. Springer, Cham, pp. 281–295.
Saraiva, Daniel A.F., Leithardt, Valderi Reis Quietinho, de Paula, Diandre, Mendes, André Sales, González, Gabriel Villarrubia, Crocker, Paul, 2019a. PRISEC: Comparison of symmetric key algorithms for IoT devices. Sensors 19 (19), 1–23.
Schneier, Bruce, 1993. Description of a new variable-length key, 64-bit block cipher (Blowfish). In: International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, pp. 191–204.
Sehrawat Deepti, Gill, Nasib Singh. 2019a. BRIGHT: A small and fast lightweight block cipher for 32-bit processor. Int. J. Eng. Adv. Technol. 8(5), 1549–1556.
Sehrawat Deepti, Gill Nasib Singh, 2020a. Ultra BRIGHT: A tiny and fast ultra lightweight block cipher for IoT. Int. J. Sci. Technol. Res. 9(2), 1063–1068.
Sevin, Abdullah, Mohammed, Abdu Ahmed Osman, 2021a. A survey on software implementation of lightweight block ciphers for IoT devices. J. Ambient Intell. Humaniz., 1–15
Shannon, C.E., 1949a. Communication theory of secrecy systems. Bell Syst. Tech. J. 28 (4), 656–715.
Shantha Mary Joshitta R., Arockiam, L., 2018. A novel block cipher for enhancing data security in healthcare internet of things. In: Journal of Physics: Conference Series. pp. 1–11.
Shantha Mary Joshitta R., Arockiam, L., 2018. SAT-Jo: An enhanced lightweight block cipher for the Internet of Things. In: International Conference on Intelligent Computing and Control Systems. IEEE, pp. 1146–1150.
Shibutani, Kyoji, Isobe, Takanori, Hiwatari, Harunaga, Mitsuda, Atsushi, Akishita, Toru, Shirai, Taizo, 2011. Piccolo: An ultra-lightweight blockcipher. In: International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, pp. 342–357.
Singh, Pulkit, Acharya, Bibhudendra, Chaurasiya, Rahul Kumar, 2019a. A comparative survey on lightweight block ciphers for resource constrained applications. Int. J. High Perform. Syst. Archit. 8, 250–270.
Standaert, Francois-Xavier, Piret, Gilles, Rouvroy, Gael, Quisquater, Jean-Jacques, Legat, Jean-Didier, 2004. ICEBERG: An involutional cipher efficient for block encryption in reconfigurable hardware. In: International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, pp. 279–298.
Standaert, Francois-Xavier, Piret, Gilles, Gershenfeld, Neil, Quisquater, Jean-Jacques, 2006. SEA: A scalable encryption algorithm for small embedded applications. In: International Conference on Smart Card Research and Advanced Applications. Springer, Berlin, Heidelberg, pp. 222–236.
Suzaki, Tomoyasu, Minematsu, Kazuhiko, Morioka, Sumio, Kobayashi, Eita, 2011. Twine: A lightweight, versatile block cipher. In: ECRYPT Workshop on Lightweight Cryptography. Springer, Berlin, Heidelberg, pp. 146–169.
Thorat, Chandrama, Inamdar, Vandana, Jadhav, Bhagvat, 2020a. TED: A lightweight block cipher for IoT devices with side-channel attack resistance. Int. J. Inf. Technol. Secur. 12 (2), 83–96.
Toprak, Sezer, Akbulut, Akhan, Aydın, Muhammet Ali, Zaim, Abdül Haim, 2020a. LWE: An energy-efficient lightweight encryption algorithm for medical sensors and IoT devices. Electrica 20 (1), 71–81.
Turan Meltem Sönmez, McKay Kerry A, Çalık Çağdaş, Chang Donghoon, Bassham Larry. 2021. Status report on the first round of the NIST lightweight cryptography standardization process. NIST Internal or Interagency Report (NISTIR) 8369.
Usman, Muhammad, Irfan Ahmed, M., Aslam, Imran, Khan, Shujaat, Shah, Usman Ali, 2017. SIT: A lightweight encryption algorithm for secure. Internet Things Int. J. Adv. Comput. Sci. Appl. 8 (1), 402–411.
Wang Cheng, Heys, Howard M., 2009. An ultra compact block cipher for serialized architecture implementations. In: Canadian Conference on Electrical and Computer Engineering. IEEE, pp. 1085–1090.
Wheeler, David J., Needham, Roger M., 1994. TEA, a tiny encryption algorithm. In: International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, pp. 363–366.
Wu, Wenling, Zhang, Lei, 2011. LBlock: A lightweight block cipher. In: International Conference on Applied Cryptography and Network Security. Springer, Berlin, Heidelberg, pp. 327–344.
Yang, Gangqiang, Zhu, Bo, Suder, Valentin, Aagaard, Mark D., Gong, Guang, 2015. The Simeck family of lightweight block ciphers. In: International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, pp. 307–329.
Yap, Huihui, Khoo, Khoongming, Poschmann, Axel, Henricksen, Matt, 2011. EPCBC - A block cipher suitable for electronic product code encryption. In: International Conference on Cryptology and Network Security. Springer, Berlin, Heidelberg, pp. 76–97.
Yeoh, Wei-Zhu, Teh, Je Sen, Sazali, Mohd Ilyas Sobirin Mohd, 2020. 2: A Lightweight block cipher. In: Computational Science and Technology. Springer, Singapore, pp. 281–290.
Zakaria, Abdul Alif, Azni, A.H., Ridzuan, Farida, Zakaria, Nur Hafiza, Daud, Maslina, 2020a. Extended RECTANGLE algorithm using 3D bit rotation to propose a new lightweight block cipher for IoT. IEEE Access 8, 198646–198658.
Zakaria, Abdul Alif, Azni, A.H., Ridzuan, Farida, Zakaria, Nur Hafiza, Daud, Maslina, 2022. LAO-3D: A symmetric lightweight block cipher based on 3D permutation for mobile encryption application. Symmetry 14 (10), 2042.
Zhang, Wentao, Bao, Zhenzhen, Lin, Dongdai, Rijmen, Vincent, Yang, Bohan, Verbauwhede, Ingrid, 2015. RECTANGLE: A bit-slice lightweight block cipher suitable for multiple platforms. Sci. China Inf. Sci. 58 (12), 1–15.