Analysis of Lightweight Cryptographic Algorithms for Internet of Things

Renie Mathews[a], Deepa V Jose[b]

[a] Department of Computer Science, CHRIST (Deemed to be University), Bangalore 560029, India
[b] Department of Computer Science, CHRIST (Deemed to be University), Bangalore 560029, India
* *Renie Mathews.* Tel.: 09495258903.
E-mail address: renie.mathews@res.christuniversity.in

ABSTRACT

The exponential advancements in the electronics field resulted in the development of small sensors with the capability to collect information inreal-time, without any human intervention. This is revolutionizing human life and also leads to many security challenges that cannot be ignored. The information collected through the massive internet of things, is analyzed to make intelligent decisions for the users. But the data accumulated can also include sensitive data that needs to be protected from detrimental attacks to maintain the privacy of the user and the integrity of the data. Thus encryption/decryption becomes a critical aspect in the Internet of Things applications; to avoid compromising the data from any intentional/unintentional attacks on the data. As the devices are highly resource-constrained, lightweight cryptographic algorithms that use less power, less memory, and have lesser computation are preferred to the traditional cryptographic methods. This paper analyzes the various symmetric ciphers in hardware architecture in terms of parameters like gate area, throughput, efficiency, memory, and complexity of the algorithm.

Keywords—Cryptography, Lightweight Algorithms, Internet of Things, Sensors, Stream Ciphers, Block Ciphers

I. INTRODUCTION

Universal computing, also called pervasive computing as envisioned by Mark Weiser (Patel & Patel, 2016) [26] has already come a long way in the ever-growing field of computer technology. Devices with computational capabilities are embedded in a variety of objects and are spread throughout the real-world environment [16]. These devices which are connected over the Internet have the capability to interact or communicate among themselves. In the present world relying heavily on these pervasive devices which perform useful tasks making our day-to-day life effortless has become the new normal.

The concept of the Internet of Things (IoT) which was conceived by Kevin Ashton (Kramp et al., 2013) [20] has extended the realm of Ubiquitous computing to a new plane altogether. It can be defined as the interactions between the digital world and the physical environment which has devices like sensors, microprocessors, etc (Bhuvaneswari & Porkodi, 2014) [5]. The devices connected to the web are increasing considerably in the current scenario. The quantity of information available is unimaginable as there is a rich profusion of such devices or things that are connected to the web. Composing context-based (Sethi & Sarangi, 2017) [30] analysis of the multitude of data with little or no human intervention makes the IoT devices like smart devices. IoT is not just a technology but an amalgamation of many technologies enabling us to connect anything, anytime, anywhere for anything (Ali. et al., 2015) [1].

1.1 Importance of IoT

The development of small sensors, the inevitable component of an IoT network, having the capability to collect information in real-time is a milestone in the massive adoption of this technology. The applications of IoT can be seen in a wide variety of aspects from smartphones to smart homes, industry to education, agriculture to satellite communication, and so on. Remote controlling the home front like controlling the temperature of the house, refrigerator, etc to the desired level to the burglar alarms systems is a common reality. The agriculture sector is made smart by using the data analysis reports on soil, water, and weather from the data collected from the rain gauges, temperature sensors, etc, thereby improving the yields through better and healthy crops.

Gadgets, sometimes implants, can be used as health indicators to track and control the patients based on the information received, helping doctors to take preventive measures. This can also be life-saving on many occasions. Various IoT applications are also actively used in patients in assisted living to record their vitals and to ensure a better well being. This in turn relieves the tension of caretakers and improves the quality of life. The usage of IoT in surveillance applications is also quite visible in all areas. Real-time data gathering helps e-commerce vendors to maintain the logistics for the inventory and automatically create orders when the goods in the inventory diminish. The analysis of the data which is abundant can be used for getting intelligent insights into any business and to predict future needs by studying the trends; thus increasing productivity and making viable business decisions. All the above-mentioned applications were just in our imagination a decade before and now have become a reality. Hence this technology enables machine to machine and machine to human communications possible within no time, also capable of handling enormous big data easily to provide useful insights, and have become an inexorable companion of modern lifestyle.

1.2 Limitations of IoT Devices

Due to the variety of applications that IoT facilitates, exponential growth in the usage of IoT devices is prevalent now. It is expected that every user should be aware of the limitations of it to eliminate the chances of becoming victim of the inherent hidden vulnerabilities. There are some limitations to the IoT devices which have to be addressed as every device a person now has, is becoming a smart device. The number of devices in the physical environment will keep rising as more IoT applications are developed for the users. The IoT environment should be able to accommodate all the basic functionalities even with the constant growth of the Internet without affecting the performance of the system (Miorandi et al., 2012) [24]. Many of the IoT devices are expected to be small in size owing to the requirement to be embedded in different devices or in sensitive areas where human intervention is risky or even impossible. This obviously creates restrictions in the various resources like memory and processing capabilities which foils the usage of traditional routing and security mechanisms used in communication networks [5]. In turn, this issue demanded the development of new methods of communication protocols, security mechanisms, etc specifically for IoT. Besides all, the lack of a common standardized architecture makes communication cumbersome in IoT networks as various vendors follow different protocols and architectures.

IoT environment is largely a heterogeneous environment owing to the collection of different ‘things’, possibly of different nature in all aspects. The heterogeneity is visible in software, hardware, and network platforms. It is expected that there should be methods that these “things” would be able to collaborate their functionalities seamlessly. The heterogeneity of the devices is closely associated with the network, syntactic, and semantic interoperability challenges (Hossain et al., 2015) [17]. Protocols and standards should be developed for smooth communication between the different networks to hide the intricacies of different networks at the interface itself.

Similarly, there would be issues due to the syntactic differences of the data structure formats used by the communicating devices, which will also have to be addressed. Efforts should be taken to combat the semantic incompatibility as the devices may use different languages and the meaning of the data should not be lost in the process. The data accumulated by the IoT devices fall in the category of 'Big Data'. The large volume of data will be stored in the centralized cloud or servers. Algorithms have to be developed to draw useful insights from these data ensuring data security. Sometimes the data collected may be incomplete or lost in transit. In such situations, it is needed to extrapolate the lost data. Sophisticated algorithms have to be developed to extract context-based information for applications.

II. SECURITY CHALLENGES IN IoT

Security challenges in IoT can be looked into different perspectives of its layered structure say; hardware and software levels, based on networking and application levels, etc. Any IoT application results in gathering enormous quantities of data. In the majority of these applications, the data collected seems to be quite private and sensitive like medical data, personal data, etc which need to be protected from outside theft. If proper preventive measures and security mechanisms are not taken for the same, it can result in data leakage/inside/outside data theft through various active and passive attacks which may not be noticed in the initial stages. Several risks, threats, attacks, and vulnerabilities are associated with IoT. A major challenge to be addressed is to provide awareness for IoT users to adopt relevant security mechanisms in their IoT network. Since vendor-specific devices are used commonly, regular patch-ups and middleware updates may not happen; and even in some cases, the vendor itself may not exist after a few years. Many of the devices are plug and play which can be easily infused or taken away from the network. In such scenarios, attackers can easily infiltrate and gain control over the network and perform the intended malicious activities resulting in the slowing down of the resources and the network or complete loss of data. A common attack that is faced by IoT networks is the various versions of the Denial of Service. The Mirai Botnet which occurred in October 2016 is one such kind. Several such attacks happen every moment despite having security mechanisms and intrusion detection systems. Novel attacks and threats get generated and the only solution to withstand this is to be literate about the issue and adopt possible countermeasures.

The fundamental security requirements for any information should also be guaranteed for IoT systems for effective throughput and performance. Thus methods are needed to protect the privacy of the sensitive data and also preserve the integrity of the data. Data confidentiality has to be ensured by allowing only authorized entities to access and modify the data. In IoT both users and objects access information. Hence access control and object authentication mechanisms should deny unauthorized access to information (Buchanan et al., 2017). The information is encrypted to maintain the confidentiality of the data avoiding passive attacks on the messages sent between the devices. Data Integrity is essentially the reliability of data. The massive amounts of data moving between nodes in the IoT network should be protected from passive and active attacks by adversaries. This can be ensured by encryption algorithms where only the intended user can decrypt the information. Efficient key management schemes must be made available when exchanging keys between the users which do not reveal the key. Client privacy is a great concern as a lot of personal information is collected in the process of analyzing the data. Personal information should be accessed only by authorized users. Data authentication is the process of preventing access to information by unauthorized users. Security in IoT should be dealt with utmost importance; otherwise, it can tamper with individuals/organization's resources, data, and routine.

2.1 Requirement of Lightweight Security Algorithms

The IoT devices are highly resource-constrained in terms of size, memory and energy backup. Because of this, the traditionally sound and highly secure cryptographic algorithms cannot be used in IoTs as these complex algorithms will consume far more memory while implementing the algorithm [2]. The power consumption of the computationally intensive algorithms is high for the devices running on battery. The above factors resolve this issue and provide security which was essential for the development of lightweight cryptographic algorithms with the main feature of less computational overhead [8]. So lightweight cryptographic algorithms are chosen to provide security in the resource limited devices like IoT devices. These algorithms use less memory, power and implementation size with high throughput (Wheeler & Needham, 1995). But the algorithms are less secure than the conventional cryptographic methods when traded with the cost of implementing the ciphers in the tiny devices like the RFID tags, sensors.

III. LITERATURE REVIEW

The lightweight cryptographic algorithms can be either symmetric or asymmetric [18]. In symmetric algorithms, the plain text is encrypted and decrypted using a public key [10]. These are either stream ciphers or block ciphers [11]. Asymmetric ciphers have two keys such as public key and private key. The symmetric ciphers are the commonly used ciphers for lightweight solutions as they are easier to implement [12].

Tiny Encryption Algorithm (TEA) with 64 rounds and 128-bit key size was proposed in 1994 (Wheeler & Needham, 1995) [35]. To overcome the weakness of the related key attacks and it is a bad hash function the authors of TEA, Wheeler, and Needham again modified it and proposed XTEA (extended TEA) in 1997 with a complex key scheduling method [19] [28]. But the Gate Equivalent (GE) exceeded the desired gate area which was to be between 1000 and 3000. The Advanced Encryption Standard (AES) proposed the highly secure round key technique in 1998, which makes it difficult to break the encrypted message. Though AES cannot be categorized as LWC, it is considered as the basis for most of the lightweight methods as it was standardized by NIST in the year 2001 (Rothke, 2007) [27]. The mCrypton block cipher (Lim & Korkishko, 2005) [22] similar to AES was designed for low-cost IoT devices like the RFID tags and sensors in 2005 has 3 key lengths possible – 64, 96, and 128 but was found to be prone to relate key rectangle attacks. Unlike the SPN and Feistel networks used in the previous ciphers mentioned, HIGHT (Scott et al., 2006) [29] is an ARX (Addition Rotation XOR) cipher whose design principle does not include S- boxes and hence is easy to implement on the hardware but the cipher had some inherent security issues. Bogdanov et al. designed the famous PRESENT in the year 2007 cipher which was an ultra-lightweight block cipher with the smallest hardware footprint with only 1000 gates (Bogdanov et al., 2007) [6]. The major security concern is that the keystream is not random enough, letting the attacker to easily decipher theciphertext. GRAIN and TRIVIUM are the finalists in the eStream project for the lightweight stream cipher category having less gate count. GRAIN (Hell et al., 2007) [15] has a keystream of 80 bits and uses less memory and power consumption. TRIVIUM (De Cannière, 2006) [11] has better hardware implementation and is inspired from block ciphers. DESL and DESXL ciphers were introduced by George Leander et al, which are lightweight adaptations of the traditional DES cipher (Leander et al., 2007) [21]. ICEBERG was an involutive cipher which could change keys with every clock cycle, but the number of gates required was 5800 (Cheng & Heys, 2008) [9]. The major difference is that these versions of DES use only one S-box instead of the 8 S-boxes used in the original, which is repeated for achieving the functionality. The 64 bit block cipher with 80/128 bit keys, TWINE proposed by Suzaki et al. is based on the generalized Feistel Structure (GFS) and can be implemented both in software and hardware (Suzaki et al., 2011) [34].

SIMON and SPECK algorithms designed by NSA [3], are block ciphers with the Feistel structure and support several key sizes, block sizes, and round numbers(Beaulieu et al., 2015). Both have hardware and software implementations but SIMON performs well when implemented in hardware and SPECK is found to be better in software implementation. A stream cipher analysis and the different design methods are conducted in (Manifavas et al., 2015) [23], which shows the ciphers are easy to implement, as streams of bits are XORed with the keystream.

The simplified key scheduling used is found to be not immune enough to the Man-in-the-middle attack. A cipher that is resistant to almost all attacks known called LEA was developed in the year 2013 by the developers of HIGHT. It is a 128-bit block cipher that has varying key sizes (Singh et al., 2017) [32]. Elliptic Curve Cryptography (ECC) is the most attractive cipher among the public key or asymmetric ciphers and is best for confidentiality and non-repudiation (Bhardwaj et al., 2017) [4]. In (Hatzivasilis et al., 2018) [14] explained the review of the lightweight block ciphers has been done on various parameters. The block cipher works on blocks of data chunks. Still, ECC and different versions of ECC seem to be more embraced.

IV. LIGHTWEIGHT CRYPTOGRAPHIC ALGORITHMS - A COMPARISON

The basic and essential parameters for comparing the lightweight algorithms are Key length or Key size, Block Size, and Number of rounds used by the algorithm.

The key length is usually measured in the number of bits and it determines the security level that can be achieved by the cipher. As the length of the key decreases, the key related attacks increases.AES is a highly secure cipher with little known attacks like Biclique cryptanalysis. Hence the key size of the ciphers should be preferably somewhere near the key length of AES (128 bits). Keeping with that condition most of the lightweight ciphers are designed with a key size of 128 except for ciphers like PRESENT which has a key length of 80 [7]. But as the key size becomes smaller, the information becomes more prone to attacks as the key can be detected with little effort. The key length is considered in conjunction with the block size and the number of rounds. In each round, the plain text is changed using a new key, and hence with more rounds, the ciphertext is more secure.

Some of the most popular ciphers taken in this study are Tiny Encryption Algorithm(TEA), Extended the Tiny Encryption Algorithm (XTEA), Advanced Encryption Standard (AES), ICEBERG, mCrypton, HIGHT, PRESENT, TRIVIUM, and GRAIN. TEA recorded in C (Wheeler & Needham, 1995) could be implemented easily in other languages effectively to encrypt the information. TEA takes 64 block size data and a 128-bit length key and produces independent output. It also uses a magic number, which is of the golden ratio, and is stored in the variable delta which is as represented in equation (1).

$$delta = \left(\sqrt{5} - 1\right)2^{31} \quad \text{(Wheeler \& Needham, 1995)} \qquad (1)$$

A different value of delta is used in the algorithm making each the round of the cipher different from each other making the cipher more secure.
Since the algorithm is very small it can be implemented in hardware also just as it can be implemented in software using the desired language choice. The problem with the algorithm is that there are 3 equivalent keys thus reducing the effective number of keys from 128 to 126 making it vulnerable to attacks. The simple key scheduling made a slide attack probable if it were not for the changing delta values (Shepherd, 2007) [31]. The extended version of TEA, XTEA, was developed to overcome the flaws found during the cryptanalysis of TEA. The 64-bit block is divided into 2 halves and Feistel network routines of 32 rounds are applied to each half. In XTEA instead of the usual XOR operation, addition is done during the encryption process and subtraction is carried out during the decryption process. All operations that are done are modulo 2^{32} in XTEA. A permutation function and a function for generating the subkey are applied to the two halves as given in equation 2.

$$Permutatio \quad nfunction = \left(R_n \prec\prec 4 \oplus R_n \succ\succ 5\right) + R_n, \qquad (2)$$

where $\oplus$ is a bitwise XOR operation and $R_n \prec\prec 4$ is the logical left shifting of the right half by 4 bits and $R_n \succ\succ 5$ is logical right shifting by 5 bit (San & At, 2012) . The subkey is generated by the function $sum + k(sum)$ where $k(sum)$ selects one block out of the four 32 bit blocks depending on the bit value of bits 1 and 0 or bits 12 and 11 of sum (Kaps, 2008). The permutation function and the subkey function are then XOR-ed and is applied to the two halves to get the final values (Kaps, 2008).
In AES, The 128 bit block of data is considered as 16 bytes which are arranged in a 4 X 4 matrixes. The number of rounds varies depending on the key sizes as shown in Table 1. Each of the rounds then uses a separate 128-bit key which is obtained from the original key. A reverse method of the above encryption process is done for the decryption process (Muhammad Abdullah, 2017) [25]. These individual rounds consist of various variable which are mentioned below.

a) ByteSubstitution: This gives a 4 X 4 matrix obtained from the lookup table.
b) ShiftRows: In this where each row is shifted to the left and the resultant matrix has the same 16 bytes but rearranged due to shifting to each other.
c) MixColumns: It gives a new matrix of entirely different 16 bytes where a mathematical function is applied to the bytes of each column giving new values in its place.
d) AddRoundKey: This matrix is now considered as 128-bit and is XORed with the 128-bit round key. The process is repeated until the cipher text is obtained in the last round.

The involution block cipher is normally used in hardware implementations and has repetitive rounds which are identical depending on the key. The encryption and the decryption differ only in the key scheduling. Non- Linearity in the cipher implementation is achieved by the substitution boxes and the permutation method. The permutation method applies 8 permutations in parallel where these are bit permutations on 8-bit blocks. This is viewed as 8 identical 8 X 8 S boxes. The key addition is done by a bitwise XOR with the key vector along with matrix multiplications to achieve the linear layer (Standaert et al., 2004)[33]. This is the case of ICEBERG. The mCrypton cipher algorithm uses a 4 X 4 nibble matrix of 4 bits. The substitution is achieved by nibble wise substitutions on this matrix. The function operates on the nibble word on the ith row or column as in equation (3), where (a_0, a_1, a_2, a_3) is the nibble word.

$$\gamma_i(a) = (S_i(a_0), S_{i+1}(a_1), S_{i+2}(a_2), S_{i+3}(a_3)) \tag{3}$$

A bit permutation is carried out column-wise using be masking nibbles. A transposition is done by moving the nibble at the ith row and the jth column to the jth row and the ith column. Then the round key is XORed to the matrix. Each round of the encryption process consists of the substitution, permutation, transposition, and the addition of the key to the 4 X 4 matrix (Lim & Korkishko, 2005).

In HIGHT the plain text is transformed as input on which the first round function is applied by using the key whitening mechanism of 8 bytes. The whitening keys are obtained from the key scheduling process from some of the bytes of the key and the subkeys that are generated using the addition and XOR operations, which is also used in the encryption process. In the other rounds the bits are shifted and XOR-ed and in the final round again the key whitening mechanism is applied (Hong et al., 2006). In the PRESENT algorithm, 31 rounds are used to generate the round key K_1 to K_{31} (Bogdanov et al., 2007) using substitution and permutation operations and the K_{32} is used in the last round to get the ciphertext using XOR with the result from the previous round(B et al., 2017). The cipher is made lightweight by using a single S box which is repeated 16 times. In TRIVIUM the stream cipher generates a long keystream of bits from the master key and it also uses an initialization vector. The incoming stream of data is XOR-ed with the keystream updating the IV(Christophe & Preneel, 2006). This process is done for encryption. It is again created and XOR-ed with the ciphertext for the decryption process. According to GRAIN the key is initialized by setting the NFLSR with the key bit values and the LFSR is set with the value from the 64 bit IV and the rest of the bits are set to 1. The algorithm is clocked where the output bits are fed back and the result is then XORed with that of LFSR and NLFSR (Diedrich et al., 2016). The block size (BS), key size (KS), and the number of rounds® for these popular ciphers are given in Table 1.

TABLE 1 – COMPARISON BASED ON PARAMETERS (BS, KS, R)

Algorithm	**Block size**	**Key Size**	**Rounds**
TEA	64	128	64
XTEA	64	128	64
AES	128	128 192 256	10 12 16
ICEBERG	64	128	16
mCrypton	64	128	16
HIGHT	64	128	13
PRESENT	64	80	32
TRIVIUM	1	80	-
GRAIN	1	80	-

Other criteria should also be considered by the lightweight ciphers for various devices like smaller hardware implementations which are proportional to the number of gates used (GE), Energy consumed, Power Consumption, Latency, Throughput, Hardware Efficiency, the software implementation measured in terms of the memory used for storing the cipher, and resistance to possible attacks on security.

The hardware implementation of any cipher considers the GE to measure the area and the complexity of the cipher which translates to the cost of implementing the same. To be considered as lightweight the GE should be between 1000 and 3000 the AES cipher though very efficient exceeds the boundary of 3000 GE and hence cannot be considered for resource-constrained devices. PRESENT is the best cipher suited for ultra-low constrained devices with 1000gates. The RFID tags being very small, the circuit area in the chips are very limited. Even though the AES algorithm consumes less power, manufacturers prefer ciphers like PRESENT for devices like RFID tags. These are also suited for applications for agriculture where many sensors are needed to collect the data and the data need not be protected.

PRESENT and AES use the SPN (Substitution Permutation network). The difference between them is in the number of S-Boxes used. The AES used 8X 8 bit S-boxes whereas PRESENT was made compact and less complex by reducing the S-Boxes to 4 X 4 bit S-boxes. Power consumption is another parameter that is closely related to GE or the circuit area. The majority of the IoT devices are battery charged or rely on the server to which it is connected for the power requirements, methods which consume less power are preferred [1]. The power used by AES is 6.12 µW, HIGHT is 5.47 µW, mCrypton is 4.66 µW, DESL is 1.93 µW whereas PRESENT significantly uses less power of 1.80 µW. Thus, the ciphers which use very little power are ideal for devices that are installed for collecting data in places like agricultural farms or remote locations where there is no power.

The number of clock cycles needed for encryption/decryption is referred to as latency. Though the PRESENT algorithm has a higher latency than ICEBERG and mCrypton, the energy consumption of the PRESENT is lower than the other ciphers. The efficiency of the cipher when implemented in hardware increases when the throughput of the encryption/decryption process increases and the complexity of the algorithm which is the computational work needed for the process decreases. This is higher in HIGHT than XTEA, PRESENT. The various parameters taken for comparison are GE,Power used(P),Latency(L), throughput(T) and H/W Efficient(H/W E) and is shown in Table 2.

TABLE 2 – COMPARISON BASED ON PARAMETERS (GE, P, L, T, H/W E)

Algorithm	GE	Power used	Latency	Through put	H/W Efficiency
TEA	3872	7	512	6.25	1.61
XTEA	2636	4.74	705	9. 08	3.44
AES	3400	6.12	1032	12. 4	3.46
ICEBERG	3872	-	-	-	-
mCrypton	2594	4.66	190	33.51	12.91
HIGHT	3038	5.47	34	188	61.67
DESL	1848	-	-	-	-
PRESENT	1000	1.80	561	11.4	11.40
TRIVIUM	2599	-	-	100	-
GRAIN	1294	-	-	100	-

The function of a cryptographic algorithm is to secure the information from hackers. Cryptanalysis is conducted on the ciphers to see how far they can resist any attack and tries to withstand the external interferences. An intentional attacker will try to decode the secret key and find the key from the information by carefully analyzing the variations in the power consumption during the encryption process etc. Here, some of the popular lightweight ciphers are analyzed based on the cryptanalysis.

Whether in hardware or software, PRESENT is one of the popular lightweight ciphers. Ciphers like SIMON, SPECK, and SEA are better in terms of small code size and low RAM usage than PRESENT. Also, they belong to ciphers with less than 1000 GE category. The ciphers have a weakness to various attacks and hence PRESENT is preferred to them. The security analysis of the PRESENT cipher has proved that it is vulnerable to some attacks, particularly multiple fault injections (Breier & He, 2016) and the secret key could be recovered using differential fault analysis. HIGHT uses a masking technique to overcome the Differential Power Analysis attack (DPA) (Gong et al., 2016) [13]. The same technique used with LEA did not reduce the processing speed but was found that the speed increased 2.7 times than HIGHT. Although LEA consumes more area than HIGHT, it can be implemented in objects which are not restricted by area, ensuring better security against DPA.

Among the variants of DES (DESL and DESXL) DESL is robust against many attacks. As it uses a single S-Box which is repeated 8 times introducing the non- linearity nature of the S-boxes .DESL is found to be resistant to linear and differential cryptanalysis and the David Murphy attacks (Leander et al., 2007). The Stream Cipher GRAIN provides high security with better hardware implementation and is well suited for RFID tags. The speed of the cipher can be increased at the cost of the hardware. GRAIN is resistant to most attacks like the algebraic attacks and fault attack (Hell et al., 2007). The 80-bit stream cipher TRIVIUM though very simple in hardware implementation is open to differential fault analysis attacks (Hosseinzadeh, 2016).

V. CONCLUSION AND FUTURE SCOPE

Over the years, the world is witnessing accelerated growth in the number of new IoT devices. Human beings rely on these indispensable devices to complement their life. With more information load in the network, security becomes a formidable issue that has to be addressed. Various lightweight cryptographic algorithms have been developed for these highly resource-limited devices. There is no single lightweight cipher that satisfies all the criteria, giving the best performance in hardware or software and also giving the best security possible for the information communicated between the devices. Depending on the requirement of the embedded applications, sometimes performance will be traded for cost if the device is ultra-limited in size. In certain other situations, if the data is very sensitive, then the security will be paramount to the cost or performance. Thus the ciphers are chosen based on the need of the applications. The development of a lightweight cryptographic algorithm suited for IoT applications without any compromise in security aspects is the need of the hour to be focused on. This study highlights the features of various lightweight symmetric ciphers suitable for different IoT applications.

REFERENCES

1. Ali., Z. H., Ali., H. A., & Badaway., M. M. (2015). Internet of Things (IoT): Definitions, Challenges and Recent Research Directions. *International Journal of Computer Applications*, *128*(1), 37–47. https://doi.org/10.5120/ijca2015906430
2. B, C., Kumar.V, K., & Shatharama Rai C. (2017). Comparative Study of cryptographic encryption algorithms. *IOSR Journal of Electronics and Communication Engineering*, *12*(03), 66–71. https://doi.org/10.9790/2834-1203026671
3. Beaulieu, R., Treatman-Clark, S., Shors, D., Weeks, B., Smith, J., & Wingers, L. (2015). The SIMON and SPECK lightweight block cIPhers. *Proceedings - Design Automation Conference*, *2015-July*. https://doi.org/10.1145/2744769.2747946
4. Bhardwaj, I., Kumar, A., & Bansal, M. (2017). A review on lightweight cryptography algorithms for data security and authentication in IoTs. *4th IEEE International Conference on Signal Processing, Computing and Control, ISPCC 2017*, *2017-Janua*(October), 504–509. https://doi.org/10.1109/ISPCC.2017.8269731
5. Bhuvaneswari, V., & Porkodi, R. (2014). The internet of things (IOT) applications and communication enabling technology standards: An overview. *Proceedings - 2014 International Conference on Intelligent Computing Applications, ICICA 2014*, *March*, 324–329. https://doi.org/10.1109/ICICA.2014.73
6. Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., & Poschmann, A. (2007). *PRESENT : An Ultra-Lightweight Block Cipher*. 450–466.
7. Breier, J., & He, W. (2016). Multiple Fault Attack on PRESENT with a Hardware Trojan Implementation in FPGA. *Proceedings - 2015 International Workshop on Secure Internet of Things, SIoT 2015*, 58–64. https://doi.org/10.1109/SIOT.2015.15
8. Buchanan, W. J., Li, S., & Asif, R. (2017). Lightweight cryptography methods. *Journal of Cyber Security Technology*, *1*(3–4), 187–201. https://doi.org/10.1080/23742917.2017.1384917
9. Cheng, H., & Heys, H. M. (2008). Compact ASIC implementation of the ICEBERG block cipher with concurrent error detection.

Proceedings - IEEE International Symposium on Circuits and Systems, 2921–2924. https://doi.org/10.1109/ISCAS.2008.4542069

10. Christophe, D. C., & Preneel, B. (2006). Trivium Specifications. *Trivium Specification*, *507932*.
11. De Cannière, C. (2006). TRIVIUM: A stream cipher construction inspired by block cipher design principles. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4176 LNCS*, 171–186. https://doi.org/10.1007/11836810_13
12. Diedrich, L., Jattke, P., Murati, L., Senker, M., & Wiesmaier, A. (2016). *Comparison of Lightweight Stream Ciphers: MICKEY 2.0, WG-8, Grain and Trivium*. ftp://download.hrz.tu-darmstadt.de/pub/FB20/Dekanat/Publikationen/CDC/MickeyWg8GrainTrivium.pdf
13. Gong, W., Choi, P., Kim, B. C., & Kim, D. K. (2016). Analysis of masking effects on DPA countermeasure for lightweight cryptographic algorithms. *ISOCC 2015 - International SoC Design Conference: SoC for Internet of Everything (IoE)*, 315–316. https://doi.org/10.1109/ISOCC.2015.7401714
14. Hatzivasilis, G., Fysarakis, K., Papaefstathiou, I., & Manifavas, C. (2018). A review of lightweight block ciphers. *Journal of Cryptographic Engineering*, *8*(2), 141–184. https://doi.org/10.1007/s13389-017-0160-y
15. Hell, M., Johansson, T., & Meier, W. (2007). Grain: A stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing*, *2*(1), 86–93. https://doi.org/10.1504/IJWMC.2007.013798
16. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B. S., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., & Chee, S. (2006). HIGHT: A new block cipher suitable for low-resource device. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4249 LNCS*, 46–59. https://doi.org/10.1007/11894063_4
17. Hossain, M. M., Fotouhi, M., & Hasan, R. (2015). Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things. *Proceedings - 2015 IEEE World Congress on Services, SERVICES 2015*, *June*, 21–28. https://doi.org/10.1109/SERVICES.2015.12
18. Hosseinzadeh, J. (2016). A Comprehensive Survey on Evaluation of Lightweight Symmetric Ciphers : Hardware and Software Implementation. *ACSIJ Advances in Computer Science*, *5*(4), 31–41.
19. Kaps, J. P. (2008). Chai-tea, cryptographic hardware implementations of xTEA. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5365 LNCS*, 363–375. https://doi.org/10.1007/978-3-540-89754-5_28
20. Kramp, T., van Kranenburg, R., & Lange, S. (2013). Introduction to the Internet of Things. In *Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model* (Issue March, pp. 1–10). https://doi.org/10.1007/978-3-642-40403-0
21. Leander, G., Paar, C., Poschmann, A., & Schramm, K. (2007). New lightweight des variants. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4593 LNCS*, 196–210.
22. Lim, C. H., & Korkishko, T. (2005). MCrypton - A lightweight block cipher for security of low-cost RFID tags and sensors. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3786 LNCS*, 243–258.
23. Manifavas, C., Hatzivasilis, G., & Fysarakis, Konstantinos Papaefstathiou, Y. (2015). A survey of lightweight stream ciphers for embedded systems. *Security and Communication Networks*, *9*(December 2015), 1226–1246. https://doi.org/DOI: 10.1002/sec.1399
24. Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, *10*(7), 1497–1516. https://doi.org/10.1016/j.adhoc.2012.02.016
25. Muhammad Abdullah, A. (2017). Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data. *Cryptography and Network Security*, *June*. https://www.researchgate.net/publication/317615794
26. Patel, K. K., & Patel, S. M. (2016). Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application &amp; Future Challenges. *International Journal of Engineering Science and Computing*, *6*(5), 1–10. https://doi.org/10.4010/2016.1482
27. Rothke, B. (2007). A look at the Advanced Encryption Standard (AES). *Information Security Management Handbook, Sixth Edition*, 1151–1158. https://doi.org/10.1201/9781439833032.ch89
28. San, I., & At, N. (2012). Lightweight Hardware Architecture for XTEA Cryptographic Algorithm. *Conference: The 2012 International Conference on Embedded Systems and Intelligent Technology (ICESIT 2012)*.
29. Scott, M., Costigan, N., Abdulwahab, W., Goubin, L., & Matsui, M. (2006). Implementations of Cryptographic pairing on smart cards. *Cryptographic Hardware and Embedded Systems - CHES 2006*, *4249*(October), 134–147. https://doi.org/10.1007/11894063
30. Sethi, P., & Sarangi, S. R. (2017). Internet of Things: Architectures, Protocols, and Applications. *Journal of Electrical and Computer Engineering*, *2017*. https://doi.org/10.1155/2017/9324035
31. Shepherd, S. J. (2007). The Tiny Encryption Algorithm. *Cryptologia*, *31*(3), 233–245. https://doi.org/10.1080/01611190601090606
32. Singh, S., Sharma, P. K., Moon, S. Y., & Park, J. H. (2017). Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions. *Journal of Ambient Intelligence and Humanized Computing*, *0*(0), 1–18. https://doi.org/10.1007/s12652-017-0494-4
33. Standaert, F. X., Piret, G., Rouvroy, G., Quisquater, J. J., & Legat, J. D. (2004). ICEBERG: An involutional cipher efficient for block encryption in reconfigurable hardware. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3017*, 279–298. https://doi.org/10.1007/978-3-540-25937-4_18
34. Suzaki, T., Minematsu, K., Morioka, S., & Kobayashi, E. (2011). Twine: A lightweight, versatile block cipher. *ECRYPT Workshop Pn Lightweight Cryptography, LC11*, 146–169. http://www.nec.co.jp/rd/media/code/research/images/twine_LC11.pdf
35. Wheeler, D. J., & Needham, R. M. (1995). Tea, a tiny encryption algorithm. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1008*, 363–366. https://doi.org/10.1007/3-540-60590-8_29