Lightweight Encryption Schemes for the Internet of Things: A Review

Sonxay Luangoudom*, Duc Tran, Nguyen Linh Giang
Hanoi University of Science and Technology, No.1, Dai Co Viet Road, Hai Ba Trung, Ha Noi, Viet Nam
Received: February 17, 2020; Accepted: June 22, 2020

Abstract

Nowadays, Internet of Things (IoT) enables many low resources and constrained devices to communicate, data analysis, control process and make decision in the communication network. Lightweight encryption schemes can be implemented in resource-constrained IoT devices with different cryptography primitives. However, in the heterogeneous environments for IoT, there are many challenges and issues for lightweight encryption suchlike power consumption, memory space, performance cost, and security. In this paper, we present and discuss performance of lightweight encryption algorithms integrated on IoT devices which are limited in resources, power and processing capacity and criteria to choose appropriate algorithm for each specific IoT application.

Keywords: IoT, security, lightweight encryption

1. Introduction

Cryptography is a process of protecting the communication data from unauthorized access by transforming the data into an unrecognizable form. The general cryptographic algorithms are designed sophisticatedly based on mathematical theory, making such algorithms hard to be cracked. However, the communication exchanged among limited-resource devices such as Internet of Thing (IoT) devices requires lightweight cryptography algorithms [1]. The reduction of the heaviness of cryptography algorithms has been linked to all performance aspects including memory, power, and energy consumption.

In IoT environment, it is necessary to secure communication information with a low power consumption on both hardware and software. Lightweight encryption schemes are designed for resource-constrained environments. Hence, these algorithms must be fast, consume less energy and store data more efficiently than conventional encryption and decryption algorithms [2]. To have an optimized lightweight encryption algorithm, it is necessary to balance between the performance, security, and computational cost.

It has been well-known that there is a trade-off between security and performance. Specifically, the shorter key length is the lower the security level is. Similarly, the smaller the number of rounds in the encryption process is the less security and performance are.

In this paper, we present a comparison between stream ciphers and block ciphers. The analyzed stream ciphers are CCM, GCM, Salsa20-Poly 1305 while the analyzed block ciphers are AES, DEA, 3DES, and Blowfish as shown on Fig. 1.

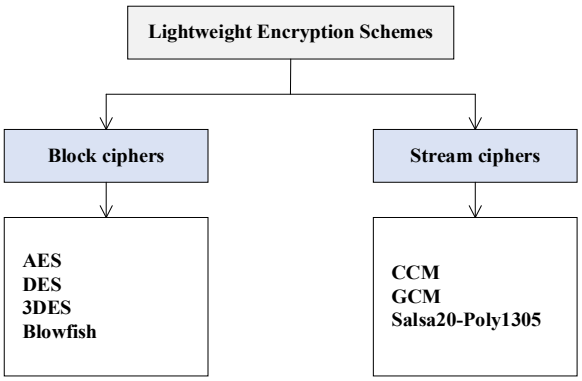

Fig. 1. Classification of lightweight encryption algorithms

The rest of the paper is organized as follows: Section 2 presents lightweight schemes. Section 3 provides a detailed discussion on the block ciphers. Section 4 analyzes stream ciphers. Finally, Section 5 is dedicated to conclusions and future works.

2. Lightweight encryption schemes

In IoT systems, implementing the traditional cryptography algorithm in the resource-constrained devices is not a trivial task. Hence, it is necessary to develop lightweight schemes for such devices. Lightweight schemes are specially designed for IoT and Wireless Sensor Networks (WSN). In general, these schemes can be categorized into two types: asymmetric encryption and symmetric encryption [3].

* Corresponding author: Tel.: (+84) 936.399.476
Email: s.luangoudom@cu.edu.la

Table 1. Block cipher based on the different indices like size of the key, block, rounds, speed and attacks [7]

Block cipher	Key length (bits)	Block length (bits)	Rounds	Speed (MB/sec)	Attacks
AES	128/192/256	64/128	10/12/14	61.01	Side channel attack, Man-in-the-middle
DES	64	64	14	21.34	Brute force attack, Man-in-the-middle attack
3DES	192	64	48	20.78	Theoretical attacks
Blowfish	448	64	16	64.386	Birthday attack, Known-plaintext attack

Asymmetric encryption relies on public and private keys to ensure the communication between the sender and receiver. The public key is used for encipherment, while the private key is used for decipherment. Asymmetric encryption can provide authentication, confidentiality, and integrity. It also offers a safety mechanism for key-sharing and supports various security services. However, the large key size in such method makes the encryption process slow and complex [4]. The most popular asymmetric algorithms are Rivest–Shamir–Adleman (RSA), Digital Signature Algorithm (DSA), Shamir-Adleman, Diffie-Hellman key exchange (DH), and Elliptic Curve Cryptography (ECC).

Symmetric encryption uses a single key for both encryption and decryption processes. This method is extremely secure and fast. It is able to guarantee the integrity and confidentiality but does not assure the authentication. The disadvantage of symmetric encryption is due to the key that must be shared between the communicating parties. If malicious parties get the key, the encrypted data will be compromised [4]. The symmetric encryption can be classified as block ciphers and stream ciphers [1, 5]. These ciphers will be analyzed and discussed in the following sections.

3. Block ciphers

In a block cipher, the message or plaintext is divided into blocks of data and the same key is used to encrypt each block. Block cipher has a fixed number of bits and different stages of transformation. These stages are determined by a symmetric. Block cipher algorithms are versatile and can be very helpful when deploying in the IoT systems [5]. The advantage of these methods is that the process has almost identical encryption and decryption methods. This implies that the implementation of the encryption and decryption processes will be reduced. Since the block ciphers have relatively low latency, they have been considered as an improved solution for IoT security [6]. There are different kinds of block ciphers, namely AES, DES, 3DES, Blowfish [7]. Table 1 shows the comparison of the various block cipher algorithms.

The Advanced Encryption Standard (AES) is a lightweight cryptography algorithm, which is standardized by NIST. Its key length can be 128/192/256 bits. AES relies on Substitution–Permutation Network (SPN) and operates using 4x4 matrices. This scheme has 10 rounds using 128-bit keys, 12 rounds using 192-bit keys and 14 rounds using 256-bit keys [8]. The output of the AES algorithm is a ciphertext, whose length is 128 bits. AES provides a good security, but its performance is not acceptable on resource-constrained devices because AES has large memory requirements to store s-boxes, large block, and key sizes [4]. AES has an advantage over 3DES and DES in terms of decryption time.

DES is also a block cipher encryption standard that has 64-bit plaintext, while the key length is 64 bits [9]. DES can be broken with a known-plaintext attack if the number of rounds is fewer than 16. DES is unsafe when being deployed in applications that require high security level. It is susceptible to linear cryptanalysis attacks, which raise a significant risk since the encrypted bulk data can be predicted with constant keys [11]. The DES algorithm also has the problem of simple relations in its key, which can potentially lead to a complementary relation between the resulting ciphertext [10]. DES can be cracked quickly because the same key is used for encryption and decryption process, hence, an attacker can get the original text by simply trying as many keys as possible. Motivated by the above reason, the 3DES cipher was developed in 1998. In 3DES, the 192-bit key is divided into three subkeys. Hence, each subkey has the length of 64 bits [12]. The procedure for encryption is the same as the regular DES. The data is encrypted and decrypted with the first and second keys and then encrypted again using the third key. Note that the 3DES algorithm is three times as secure as DES if three separate keys are used.

Blowfish on the other hand, is a symmetric block cipher that can be treated as a replacement of the DES

algorithm [10]. It is unpatented, and thus, being free of cost for all usages [13]. Blowfish provides high speed, compactness, security and simplicity. Its rate of encryption is 26 cycles/byte on a 32-bit microprocessor. Blowfish requires less than 5 KB of memory space. Its block size is 64-bit and the key size is from 32 bits to 448 bits. The design and implementation of Blowfish rely on primitive operations, including lookup tables, XOR and addition [14]. In [7], Blowfish was observed to be the fastest algorithm as compared with AES, DES, 3DES and RC2. Similar observations can be found in [15], where the various block ciphers were executed on the Beagle Bone Black and Raspberry PI 3 for different file sizes ranging from 1 MB to 128 MB.

4. Stream ciphers

Stream ciphers use keys with the size that is equal to the size of the data. In stream ciphers, the ciphertext is obtained by bit operations on the plaintext. Particularly, a keystream that is generated using a key and an Initialization Vector (IV), is XORed with the plaintext to create ciphertext. Stream ciphers are potentially more compact, simpler, and faster as compared to the block ciphers [16]. In this section, the various stream ciphers are reviewed and discussed in detail.

CBC-MAC (CCM) stands for Cipher Block Chaining Message Authentication Code. CCM is originally designed to be used with 128-bit block ciphers but can be extended to be used with other block sizes [17]. CCM provides confidentiality and authenticity of data using an approved symmetric algorithm, whose block size is 128 bits with 12-byte nonce. CCM allows varying degrees of protection against unauthorized modifications by using variable-length authentication tags. In CCM, a single key to the block cipher must be established beforehand among the communication parties. For this reason, such scheme should be implemented within a well-designed key management structure. The security properties of CCM are much dependent on the secrecy of the pre-shared key.

Galois/Counter Mode (GCM) for authenticated encryption with associated data is constructed from an approved symmetric block cipher with a block size of 128 bits with 12-bytes nonce. GCM has two functions, i.e., authenticated encryption and authenticated decryption. GCM can provide data confidentiality with various counter modes of operation since its hash function is defined over a binary Galois field. The encryption and authentication of GCM is safe from the attack [18].

Salsa20 [19] is a stream cipher that was designed and introduced in 2005. Salsa20 has 256-bit keys. The 20-round stream cipher Salsa 20/20 is consistently faster than AES. The Salsa20/12 and Salsa20/8 are among the fastest 256-bit stream ciphers. In Salsa20, the key is a uniform random sequence of 32 bytes; The 24-byte nonce is never used for any other 32-byte messages that are exchanged between the source to the destination. The nonce is long enough to minimize the risk of collision. Salsa20 encryption function by hashing the key, nonce and block number and xor'ing the result with the plaintext [19].

Poly1305 authenticator is designed by D. J. Bernstein in 2005. Poly1305 is a one-time polynomial evaluation Message Authentication Code (MAC). It aims at providing fast authentication mechanisms on software platforms. Poly1305 is considered as a secure message authentication if AES is secure. It relies on a 32-byte secret key and a 16-byte nonce to compute the 16-byte authenticator of a given message. A popular implementation of Poly1305 can be found in NaCl library [20]. More importantly, the >100-bit security level of Poly1305 prevents forgery attack. The Poly1305 authenticator, which has been standardized in RFC 7539 [21], is designed to ensure that those forged messages are rejected with a probability of 1-(n/(2^102)), even after 2^64 legitimate messages have been sent. In other words, such method is unforgeable against chosen message attacks. Poly1305 is known to have consistent high speed, even when being run on many different Central Processing Units (CPUs).

Table 2. shows the comparison between lightweight stream ciphers based on the key size, block size, performance, number of rounds and the possible attacks [22]. CCM employs counter mode for encryption. However, reusing the same Initialization Vector (IV) with the same key is catastrophic. This potentially leads to an IV collision and the leakage of information in data packets. For this reason, it is inappropriate to use CCM with static keys. Additional measures would be needed to prevent the reuse of IV values with the static key.

Implementations of GCM mode often utilize short IV. This potentially results in the collision probability of random IV. The reuse of the GCM nonce/key combination also destroys the security guarantees and leads to the degradation of the confidentiality of a given plaintext. Because the GCM mode uses a variation of the counter mode to ensure confidentiality. As a result, it can be extremely difficult to deploy GCM securely when using static keys. In many cases, GCM has been proved to be faster than AES in CBC mode, especially when the hardware supports cryptographic engine [23]. AES-GCM is faster than AES-CCM. When it comes to performance, AES-GCM is a better alternative to be used in applications.

Table 2. Stream cipher based on the different indices like initialization vector (IV), size of the key, block, nonce and attacks [22]

Stream cipher	IV (bits)	Key size (bits)	Block size (bits)	Nonce (bytes)	Attacks
CCM	64	128	64/128	12	Unauthorized modifications
GCM	64	128	64/128	12	Chosen plaintext attack, replay attack
Salsa 20- Poly 1305	128	256	512	24	Forged attack

The Salsa20 stream cipher and Poly1305 authenticator were also evaluated by the CFRG. Based on such evaluation, the RFC7539 [21] and RFC7905 [24] have been established. Salsa 20 and Poly1305 have been designed for high-performance software implementations and to minimize leakage of information through side channel attacks.

Salsa 20 is simple and easy to setup. It can achieve a good overall performance and is selected as part of the eSTREAM portfolio of stream ciphers [21]. Poly 1305 is never used the same nonce for two different messages. Poly1305 has extremely high speed and low overhead. XSalsa20-Poly1305 is proved to be a well-suited algorithm that can be used to encrypt and decrypt data packets in a wide range of applications, where time and memory usage are considered as important factors. XSalsa20-Poly1305 is three times faster than AES-GCM on mobile devices. It spends less time on decryption and thus providing faster page rendering and better battery [25]. In [26], it was observed that GMC, CCM, SIV and EAX are not feasible to perform in the current swarm architecture and configuration. GCM and CCM are only feasible when risk is accepted. Overall, the best choice by far is XSalsa20-Poly1305. XSalsa20-Poly1305 should be a viable option in any scenario, where classified data is not being created or handled [26].

5. Conclusion

The security and privacy issues have drawn a lot of consideration, while other concerns such as availability, reliability, and performance of the constrained IoT devices still require more attention. In this paper, we provide a comprehensive discussion on the lightweight security solutions, i.e., stream ciphers and block ciphers for the IoT systems. Based on such discussion, we can conclude that there is no single best scheme that is able to meet the needs of the IoT applications. Block ciphers and stream ciphers achieve a good performance in terms of computational cost and improve the security level slightly. Future research is therefore dedicated to designing a lightweight cipher that can provide fast confusion and diffusion in a smaller number of rounds for block ciphers and extend the nonce for the stream ciphers.

Acknowledgements

This work is supported by the Centre for Technology Environment Treatment.

References

[1] Bansod, Gaurav, et al., An ultra-lightweight encryption design for security in pervasive computing, Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC). (2016) 79-84.

[2] Hammi, Mohamed Tahar and Livolant, Erwan and Bellot, Patrick and Serhrouchni, Ahmed and Minet, Pascale, A lightweight IoT security protocol, Cyber Security in Networking Conference (CSNet). (2017) 1-8.

[3] Dutta, Indira Kalyan and Ghosh, Bhaskar and Bayoumi, Magdy, Lightweight Cryptography for Internet of Insecure Things: A Survey, Annual Computing and Communication Workshop and Conference (CCWC). (2019) 475-481.

[4] Bhardwaj, Isha and Kumar, et al., A review on lightweight cryptography algorithms for data security and authentication in IoTs, International Conference on Signal Processing, Computing and Control. (2017) 504-509.

[5] Batina, Lejla, et al., Dietary recommendations for lightweight block ciphers: power, energy and area analysis of recently developed architectures, International Workshop on Radio Frequency Identification: Security and Privacy Issues. Springer, Berlin, Heidelberg. (2013) 103-112.

[6] M. A. Philip, A Survey on Lightweight Ciphers For IoT Devices, Int. Conf. Technol. Adv. Power Energy (TAP Energy). (2017) 1-4.

[7] Nadeem, Aamer and Javed, M Younus, A performance comparison of data encryption algorithms, international Conference on information and communication technologies. (2005) 84-89.

[8] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer, Strong Authentication for RFID Systems Using the AES Algorithm, in Cryptographic Hardware and Embedded Systems–CHESS Lecture Notes in Computer Science, Springer. (2004) 357-370.

[9] O.A. Hamdan, and B.B. Zaidan, New Comparative Study Between DES, 3DES and AES within Nine Factors, Journal Of Computing. 2 (2010).

[10] Y. Kumar, R. Munjal, and H. Sharma, Comparison of Symmetric and Asymmetric Cryptography with Existing Vulnerabilities and Countermeasures, International Journal of Computer Science and Management Studies. 11 (2011) 60-63.

[11] Mathur, Raghav and Agarwal, Shruti and Sharma, Vishnu, Solving security issues in mobile computing using cryptography techniques—A Survey, International Conference on Computing, Communication \& Automation. (2015) 492-479.

[12] Adhie, Roy Pramono and Hutama, Yonatan and Ahmar, A Saleh and Setiawan, MI, Implementation cryptography data encryption standard (DES) and triple data encryption standard (3DES) method in communication system based near field communication (NFC), Journal of Physics: Conference Series. 954 (2018) 012009.

[13] S.P. Singh, and R. Maini, Comparison of Data Encryption Algorithms, International Journal of Computer Science and Communication. 2 (2011) 125-127.

[14] A. Kumar, Comparative Analysis between DES and RSA Algorithm's, International Journal of Advanced Research in Computer Science and Software Engineering. 2 (2012) 386-391.

[15] Deshpande, Kedar and Singh, Praneet, Performance evaluation of cryptographic ciphers on IoT devices, International Conference on Recent Trends in Computational Engineering and Technologies. (2018) 1-6.

[16] Armknecht, Frederik, and Vasily Mikhalev, On lightweight stream ciphers with shorter internal states, International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg. (2015) 451-470.

[17] Whiting, D and Housley, R and Ferguson, N, RFC3610: Counter with CBC-MAC (CCM). (2003).

[18] McGrew, David and Viega, John, The Galois/counter mode of operation (GCM), submission to NIST Modes of Operation Process. 20 (2004).

[19] Bernstein, Daniel J, The Salsa20 family of stream ciphers, New stream cipher designs, Springer. (2008), 84-97.

[20] Bernstein, Daniel J, The Poly1305-AES message-authentication code, In International Workshop on Fast Software Encryption. (2005) 32-49.

[21] Y. Nir and A. Langley, ChaCha20 and Poly1305 for IETF Protocols, RFC 7539, https://rfc-editor.org/rfc/rfc7539.txt. (2015).

[22] https://libsodium.gitbook.io

[23] Bogdanov, Andrey and Mendel, Florian and Regazzoni, Francesco and Rijmen, Vincent and Tischhauser, Elmar, ALE: AES-based lightweight authenticated encryption, International Workshop on Fast Software Encryption. (2013) 447-466.

[24] A. Langley, W.-T. Chang, N. Mavrogiannopoulos, J. Strombergson, and S. Josefsson, ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS), RFC 7905, https://rfc-editor.org/rfc/rfc7905.txt. (2016).

[25] Islam, Maliha Momtaz and Paul, Sourav and Haque, Md Mokammel, Reducing network overhead of IoT DTLS protocol employing ChaCha20 and Poly1305, International Conference of Computer and Information Technology (ICCIT). (2017) 1-7.

[26] Thompson, Richard B and Thulasiraman, Preetha, Confidential and authenticated communications in a large fixed-wing UAV swarm, IEEE 15th International Symposium on Network Computing and Applications (NCA). (2016) 375-382.