IoT Security: Implementation of Xtea, Simon/Speck Lightweight Block Ciphers.

Levent Ertaul, Arpit Chauhan
Computer Science, CSU, East Bay, Hayward, CA, USA.
levent.ertaul@csueastbay.edu, achauhan@horizon.csueastbay.edu

Abstract – The current technological advancements in the fields of internet, mobile devices, and machine to machine (M2M) communication can be regarded as the initial phase of the Internet of Things (IoT). As the IoT expands, it will integrate diverse technologies to enable novel applications by connecting various physical objects. In such an environment, it has become crucial to safeguard all the information. This research paper focuses on implementing lightweight encryption algorithms, such as Xtea, Simon, and Speck, using an ESP8266 microcontroller and the Arduino IDE. Additionally, the paper presents a performance analysis of these algorithms based on their computational cost and memory requirements. Thus, this study provides insights into the effectiveness of these three lightweight encryption algorithms for securing IoT devices.

Keywords: IoT Security, Lightweight Encryption Algorithms, Xtea, Simon & Speck.

I. Introduction

The emergence of IoT technology has created unparalleled opportunities for connecting not only humans but also facilitating communication between machines through M2M communication. This allows sensors and networks to directly communicate with each other, sharing information and creating an instrumented universe where accurate data is available to support decision making [1]. IoT devices can be found in various areas such as industrial automation, home area networks, personal area networks, and other networked devices. By 2023, it is estimated that around 25 billion devices will be connected to the Internet of Things (IoT), making security a critical concern for protecting users against eavesdropping and malicious attacks [10].

The architecture for IoT applications is based on the Embedded devices, Gateways, and Cloud (MGC) model, as illustrated in Figure 1. When nodes in wireless sensor networks are connected to the internet, they become a part of the Internet of Things, which raises concerns related to security, privacy, standardization, and power management [12]. The embedded devices are composed of microcontrollers, Wi-Fi chip boards, and other IoT boards. Gateways are devices that house these embedded chips and other microcontroller units. The Cloud in the MGC architecture refers to network devices that are used to connect to the outside world, such as different protocols like TCP/IP and 4G/5G. Collaborating embedded devices, gateways, and cloud leads to the development of an IoT application.

Figure 1: MGC Architecture.

As depicted in Figure 1, there are numerous connections among the three components, leading to an elevated risk of security breaches in IoT applications. Additionally, the storage repositories of IoT data pose a significant threat, as intruders can easily access and manipulate the data stored in them. The gateways connecting IoT devices to company and manufacturer networks also require security measures to be implemented to ensure the security of the devices and the networks. Therefore, it is essential to enhance the security of the gateways to enhance the overall security of IoT devices.

The complexity and distribution of the IoT system pose significant challenges for ensuring security. The intricate nature of the system makes it difficult to apply security measures, and this complexity is a major obstacle to the widespread adoption of IoT devices in homes [11]. There are an overwhelming number of differences in resources, ranging from 10^3 to 10^6 across all tiers, as well as a wide variety of languages, operating systems, and networks used, all of which add to the complexity of designing effective security measures for IoT devices.

The IoT devices are typically not designed to be used by a single user or owner. Instead, they may be owned by different entities with varying policies, management practices, and connectivity domains. As a result, these devices must be able to provide simultaneous, open access to data consumers and controllers while also maintaining privacy and exclusivity where necessary [3]. The use of lightweight encryption algorithms on the IoT devices can improve security and protect confidential information.

The 2nd section of the paper will further discuss various solutions which can be applied to provide security to the IoT devices. The 3rd section of the paper explains the detailed information about all the three lightweight encryption algorithms along with the block diagrams. The 4th section of the paper discusses the actual implementation of the lightweight encryption algorithms on the microcontroller ESP8266. In the 5th section the performance analysis of all the three lightweight encryption algorithms is given in terms of the memory usage, the execution time and the power usage. Finally, conclusion is given in section 6.

II. Solutions to IoT Security

There are several reasons why lightweight cryptography is preferred for IoT devices:

- Limited processing power: IoT devices are often limited in terms of processing power and memory, which makes it difficult to use resource-intensive cryptographic algorithms.
- Lightweight cryptography algorithms are designed to be efficient in terms of computation and memory usage, making them suitable for use on IoT devices.
- Limited energy resources: IoT devices are often powered by batteries or other limited energy sources, which makes it important to use algorithms that are energy-efficient.
- Limited communication bandwidth: IoT devices often have limited communication bandwidth, which makes it important to use algorithms that require less data to be transmitted.
- Security requirements: Despite the limitations of IoT devices, security is still a critical requirement for many IoT applications. Lightweight cryptography algorithms are designed to provide a high level of security while also being efficient enough to run on IoT devices. [9]

There exist various methods to ensure the security of IoT devices. However, there are primarily two approaches that need to be considered when choosing a security solution for such devices.

A. Data Security

By the end of the decade, it is anticipated that the number of IoT devices will surpass 50 billion, and these devices can potentially store significant amounts of private data. Although the manufacturers of IoT devices may prioritize data privacy, it appears that they view security and big data problems as less significant issues. To address this concern, new cryptographic computational models can be designed to facilitate secure data analytics and actuation on massive streams of real-time data from embedded systems. The ultimate objective of data security is to provide end-to-end protection. Encrypted data generated by the embedded devices can only be decrypted and viewed by the end application.

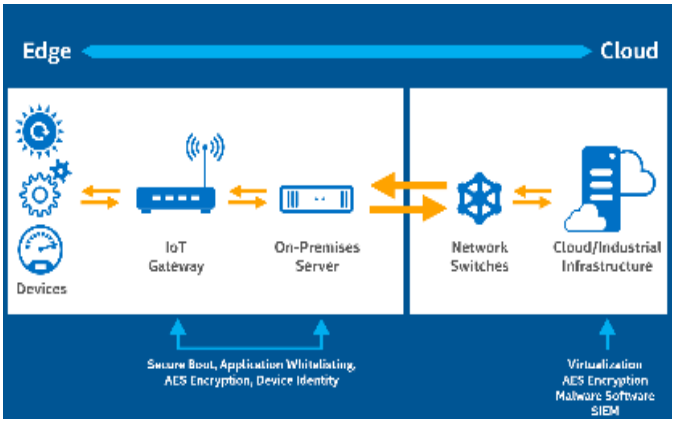

Figure-2 End-to-End Encryption

In Figure 2, we can observe the implementation of end-to-end security in IoT devices, which ensures that data remains secure from any intruders. The embedded device generates encrypted data that is transmitted via the IoT gateway, server, and network devices connected to the cloud. Despite being transmitted across various nodes, the data remains encrypted and cannot be tampered with by any unauthorized users. The application of such security measures can significantly enhance the overall security of IoT devices.

B. System Security

One way to ensure security in IoT applications is through system security, where a secure system or open framework is implemented to facilitate the building of IoT applications using new computational models [4].

The proposed solution to address the security concerns of IoT devices in this paper involves utilizing lightweight encryption algorithms. These algorithms are implemented on a microcontroller board during the design phase of the IoT devices. As a result, any data uploaded into the IoT device is automatically encrypted and transferred securely to any connected device. This ensures that confidential information remains safe from any potential intruders or hackers attempting to manipulate or access the data.

The next section of the paper provides detailed information about the Xtea, Simon/Speck lightweight encryption algorithms, including their structure, key size, and data flow from plaintext to ciphertext.

III. Xtea, Simon/Speck Lightweight Encryption Algorithms

The microcontroller chips present in IoT devices have limited memory and libraries, which means that lightweight encryption algorithms like Xtea and Simon/Speck must be chosen wisely for implementation. For example, Node MCU has 4Mb flash memory, 80k RAM, and a processor power of only 80 MHz. AES-128 algorithm, which is commonly used, is too complex and large for microcontrollers, even though there are fast software implementations available for 8-bit and 16-bit

microcontrollers. Therefore, implementing lightweight block ciphers on microcontrollers is a better option, as they are suitable for a wide range of hardware and software-based devices and can be easily implemented on 8-bit, 16-bit, and 32-bit microcontrollers. Hence, we can conclude that lightweight block cipher algorithms like Xtea, Simon & Speck, are more suitable for encrypting information on IoT boards.

A. Xtea Lightweight Encryption Algorithm

The Xtea algorithm is an extension of the Tiny Encryption Algorithm and is well-known for its simplicity, which makes it ideal for hardware implementations and widely used to provide lightweight security [6]. It is a 64-bit lightweight block feistel cipher that uses a 128-bit key value and a variable number of rounds, although it typically uses 64 rounds.

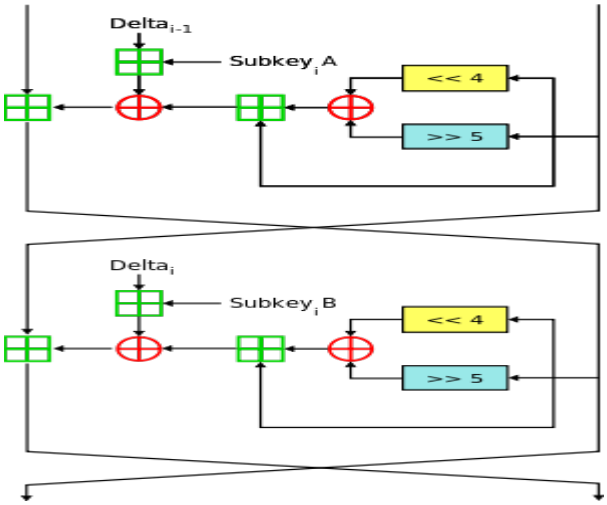

Figure 3: Two Feistel rounds of Xtea Lightweight Block Cipher Flow Diagram

The step-by-step process of two Feistel rounds of the Xtea lightweight encryption algorithm is illustrated in the provided figure above. The figure visually demonstrates how the plaintext undergoes a series of operations to be transformed into ciphertext. Xtea encrypts an 8-byte value and uses a 16-byte key. The figure demonstrates that the plaintext is divided into two equal halves, and the subkey is calculated from the round constant and the key. The subkey index value is used to address one of the 4-byte values contained in the 16-byte key. In each round, the right half of the plaintext is shifted left by four and then five times, and both values are XORed. The result is added to the original right side, and a delta value is calculated by adding 0x9e3779b9 to each round. Then, the delta value is added to the subkey for that particular round, and the sum is XORed with the modifications on the right side. Finally, the whole sum is added to the original left side, resulting in a new right side for the next round. The Xtea round encryption algorithm is simple and does not require any initialization, s-boxes, or permutations.

B. Simon/Speck Lightweight Encryption Algorithm

In The design of Simon and Speck lightweight block ciphers was aimed at making them suitable for implementation on resource-constrained devices such as Microcontrollers, RFID, and Wireless Sensor Networks. These ciphers were developed to be secure, efficient, flexible, and simple. Additionally, flexibility was considered in terms of block and key sizes, as different devices and applications have varying requirements. For instance, desktop computing may require block and key sizes of 64 and 128 bits, whereas electronic product code (EPC) applications may require typical block sizes of 48 or 96 bits [8].

block size	key sizes
32	64
48	72, 96
64	96, 128
96	96, 144
128	128, 192, 256

Figure 4: Simon Parameters.

The parameters of the Simon and Speck lightweight block cipher are shown in Figure 4. If the block size is 32, then the cipher requires a key size of 64 bits. However, the key size increases with the increase in block size.

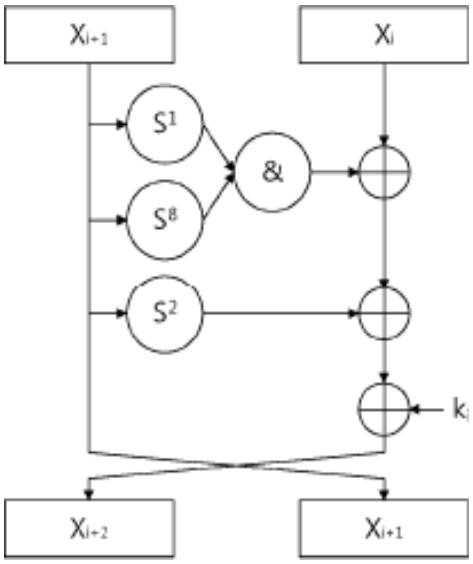

Figure 5: Simon Lightweight Block Cipher Design Flow

The Figure 5 illustrates the design flow of the Simon lightweight block cipher. In the diagram, S^1 represents a left shift operation performed once on the input, while S^8 represents a left shift operation performed eight times. Similarly, S^2 indicates a left shift operation performed twice.

On the left side of the diagram, the input undergoes left shifts of one and eight times. The results of these shifts are then subjected to an AND operation and XORed with the right side. The obtained result is further XORed with the input data left-shifted twice. Finally, this result is XORed with the required key size based on the block size specified in Figure 5, following the Simon parameters.

The same procedure is repeated based on the number of variable rounds indicated in Figure 6.

block size $2n$	key size mn	Simon rounds	Speck rounds
32	64	32	22
48	72	36	22
	96	36	23
64	96	42	26
	128	44	27
96	96	52	28
	144	54	29
128	128	68	32
	192	69	33
	256	72	34

Figure 6: Variable number of rounds for Simon/Speck lightweight block cipher.

The provided figure above presents the required number of rounds for the Simon/Speck lightweight block ciphers, along with the corresponding key sizes for different block sizes. Specifically, for a 32-bit block size, the Simon lightweight block cipher utilizes a 64-bit key size and executes 32 rounds, while the Speck lightweight block cipher adopts a 64-bit key size and performs 22 rounds.

Block size(2n)	Key size(mn)	Word size(n)	Key words(m)	Rot α	Rot β	Rounds (T)
32	*64*	*16*	4	7	2	22
48	*72* *96*	*24*	3 4	8	3	22 23
64	*96* *128*	*32*	3 4	8	3	26 27
96	*96* *144*	*48*	2 3	8	3	28 29
128	*128* *192* *256*	*64*	2 3 4	8	3	32 33 34

Figure 7: Speck Lightweight Cipher Number of Rounds.

The number of rounds employed by the Speck lightweight block cipher algorithm for various block sizes is depicted in the Figure 7 below. Additionally, the figure showcases the word size, which serves as an input parameter for encryption using the Speck lightweight cipher algorithm.

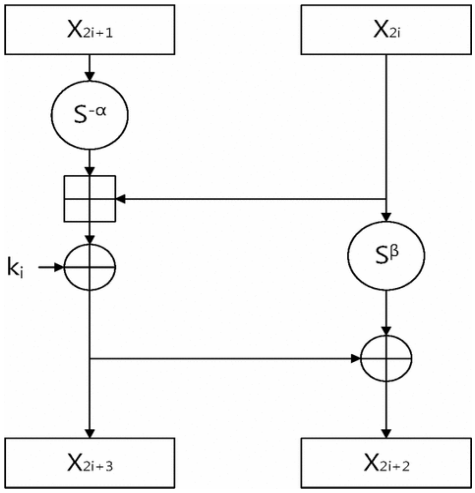

Figure 8: Speck Lightweight Block Cipher Design Flow

Figure 8 depicts the design flow for the Speck lightweight block cipher. The S-α and Sβ represent the number of left shifts that need to be applied, and their values depend on the block size, as shown in Figure 7. For a block size of 32, the value of α is 7 and β is 2, while for other block sizes, α is 8 and β is 3. The left half of the input data is shifted left based on the value of α, and the result is added to the right half. The addition result is then XORed with the key. Next, the right half is shifted left based on the value of β, and the result is XORed with the earlier addition result along with the XOR with the key. This process produces the ciphertext, which is used as the right half for the next round.

IV. Implementation of Xtea, Simon/Speck Lightweight Block Cipher.

Figure 9: Node MCU Esp-8266

The appearance of the Node MCU is shown in Figure 9. The Node MCU Esp-8266 is a microcontroller unit with a built-in

Wi-Fi module that is designed primarily for the Internet of Things (IoT) platform. Due to the limited amount of RAM, flash memory, and CPU speed on the board, it is suitable for IoT devices. The Node MCU has 128 Kbytes of RAM and 4Mb of flash memory, and is powered by a micro USB cable with the XTOS operating system. To upload code onto the Node MCU, the Arduino IDE is used, and the code for implementation is written in the C language within the Arduino IDE.

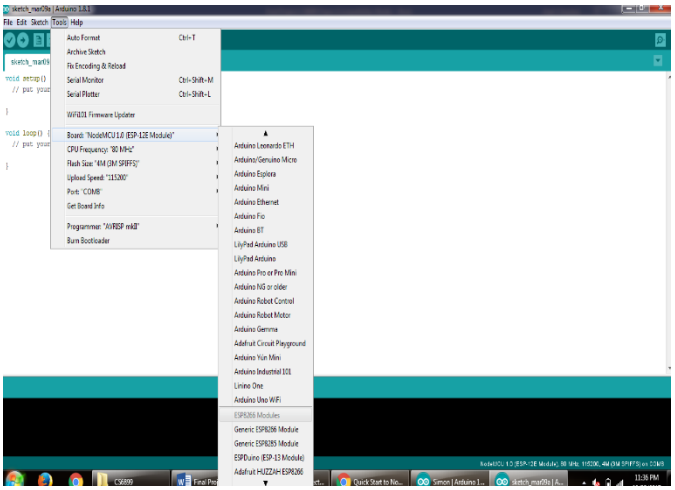

Figure 10: Setting Up Node MCU

To utilize the Esp-8266 with the Arduino IDE, the board needs to be selected as illustrated in Figure 10. The subsequent step involves opening the COM port to establish a connection with the board. Once the COM port is opened, the code can be uploaded onto the Arduino.

After establishing the connection, the implementation of the Xtea lightweight block cipher and Simon/Speck lightweight block cipher is carried out using the Arduino IDE in the C language. The code snippet for the encryption of both algorithms is shown in the figure.

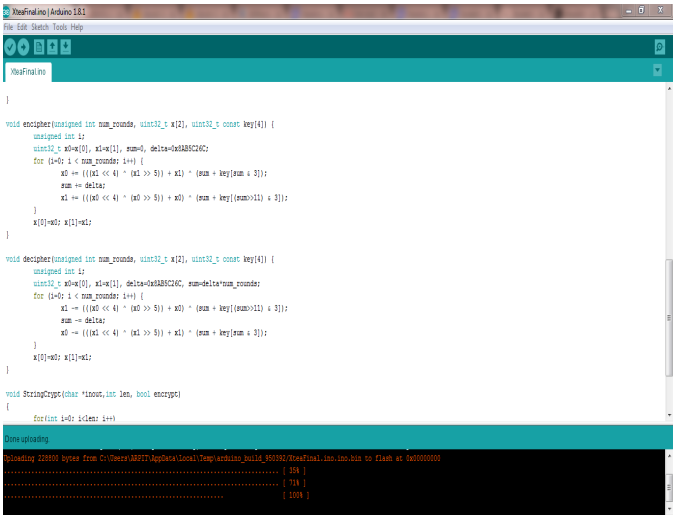

Figure 11: Xtea C code on Arduino along with uploading on Node MCU Esp-8266

The encipher block code for the Xtea lightweight block cipher is depicted in Figure 11 above. Once the code is written, it needs to be uploaded and compiled onto the Esp-8266. To view the output of the code, the serial monitor needs to be opened and the appropriate baud rate needs to be set. For this code, the baud rate used is 115200 baud, which determines the upload speed of the code. Figure 12 shows the output of the code, which displays the encrypted form of the text and the time taken for the encryption process is noted using a timer.

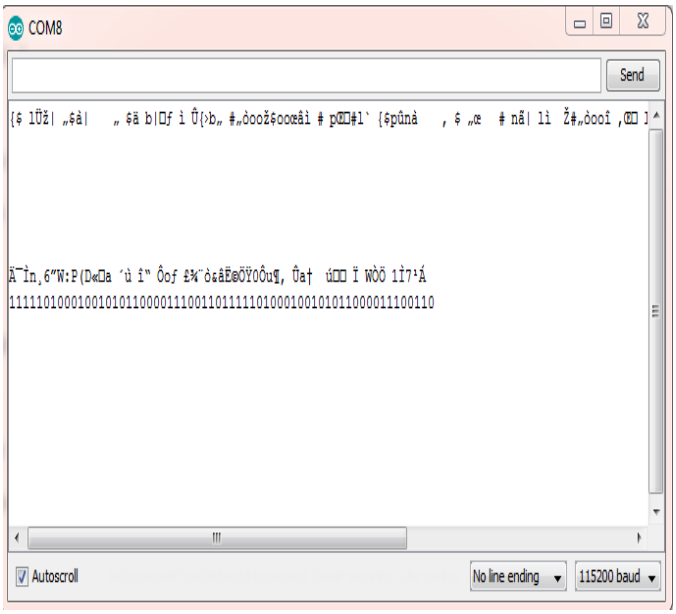

Figure 12: Output for Xtea on Node MCU Esp-8266

The same process as described earlier is followed to upload the code for both Simon and Speck lightweight block ciphers. The code is written in C language using the Arduino IDE and needs to be uploaded onto the Esp-8266, where it is compiled. The output of the code can be viewed in the serial monitor by setting the appropriate baud rate. The encryption results and the time taken to perform the encryption are noted using a timer, similar to the Xtea lightweight block cipher implementation.

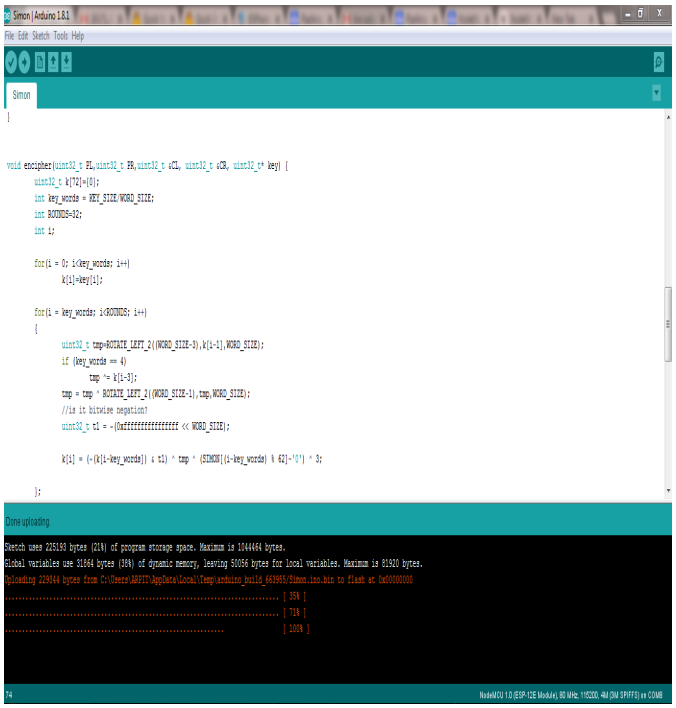

Figure 13: Simon C language code on Arduino along with uploading on Node MCU Esp-8266

The uploaded code for the Simon lightweight block cipher on the Node MCU Esp-8266, along with its compilation, is depicted in Figure 13.

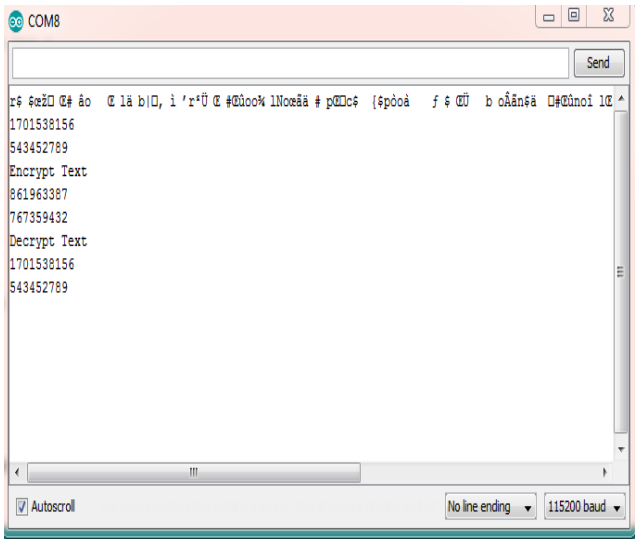

Figure 14: Output for Simon on Node MCU Esp-8266

The output on the serial monitor for the Simon lightweight block cipher, with a baud rate of 115200, is displayed in Figure 14. The output shows the encrypted text, and the time taken to perform the encryption is recorded using a timer.

The implementation of the Speck lightweight encryption algorithm is carried out in a similar manner using the Arduino IDE. The compilation procedure and output of the Speck algorithm are shown in Figure 15 and Figure 16 respectively.

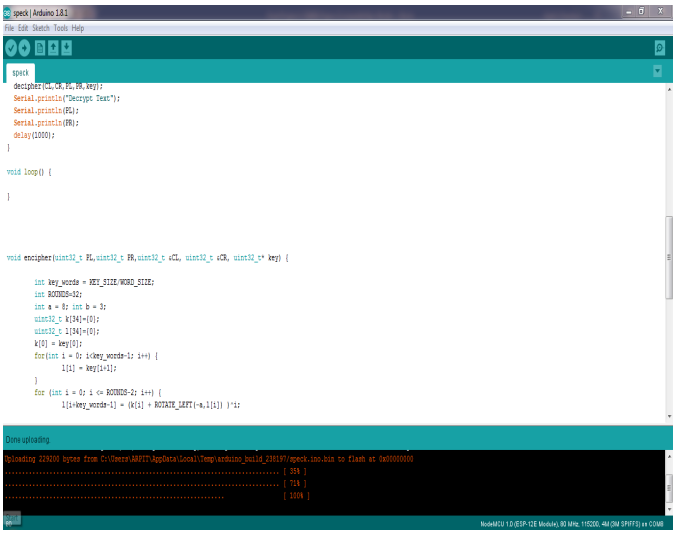

Figure 15: Speck C code on Arduino along with uploading on Node MCU Esp-8266

Figure 15 The process of uploading the code onto the Node MCU Esp-8266 and its compilation is illustrated in Figure 15. The baud rate used for uploading the code is set to 115200.

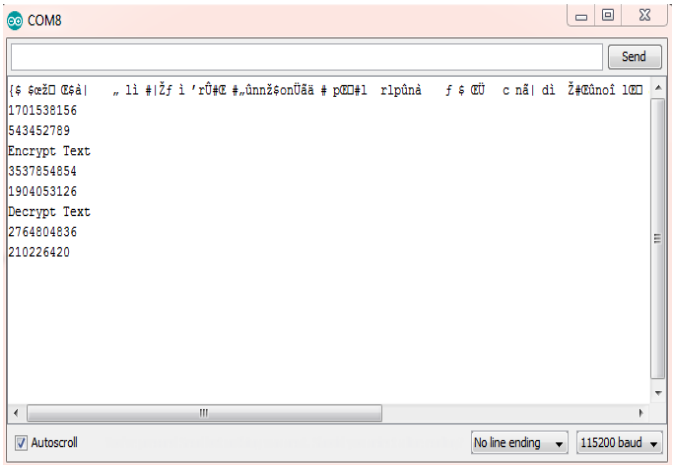

Figure 16: Output for Speck on Node MCU Esp-8266

Figure 16 The encrypted output for the Speck lightweight encryption algorithm is shown in Figure 16, which displays the encrypted text. The time taken for compiling the code for this algorithm is measured by setting up a timer.

V. Performance Analysis

To conduct a fair comparison between the Xtea and Simon/Speck lightweight ciphers, their performance is analyzed based on their execution time, memory requirements, and power consumption. To ensure a neutral comparison, both algorithms are evaluated using a block size of 64 bits and a key size of 64 bits for Xtea and 96 bits for Simon/Speck.

A. Performance based on Execution Time

The comparison and analysis results of the Xtea and Simon/Speck lightweight ciphers are shown in the figure below..

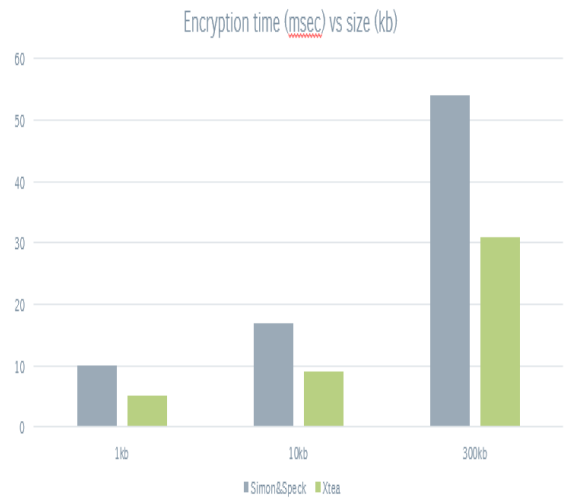

Figure 17: Performance based on Execution Time.

Figure 17 illustrates the performance of Xtea and Simon/Speck lightweight ciphers in terms of execution time and data size. The data is encrypted with 64-bit block size and a 64-bit key for Xtea, and a 96-bit key for Simon/Speck. The graph shows that Xtea performs faster than Simon/Speck, with Xtea taking 5 msec, 9 msec, and 31 msec to encrypt 1kb, 10kb, and 300kb of data, respectively, while Simon/Speck takes 10 msec, 17 msec, and 54 msec to encrypt the same data sizes. This is because Simon/Speck has more operations in each round function, including 3 rotations, 1 AND operation, and 3 XOR operations, while Speck has only 2 rotations, 1 addition operation, and 2 XOR operations. Additionally, Simon has more rounds than Speck. In contrast, Xtea performs only 2 rotations and 2 XOR operations, resulting in faster encryption as compared to Simon/Speck lightweight block cipher.

B. Power Consumption

To calculate the power consumption of the lightweight cipher algorithm, the approach used takes into account the operating voltage of the CPU and the average current drawn by clock cycles for carrying out the encryption. The power consumption (***E***) is determined using the following formula:

$$E = I * N * \tau * VCC$$

Where, ***I*** represents the average current in amperes,
N represents the number of clock cycles,
τ represents the clock period
VCC represents the supply voltage.

For the Node MCU, the average current is 0.08 amperes. Based on the calculation of operations and rotations taking place, the number of cycles (N) for an average size of 10kb for Xtea is 1349 cycles and for Simon/Speck is 1553 cycles. The clock period for the Node MCU is 0.0125, and the supply voltage used is 3.30 volts.

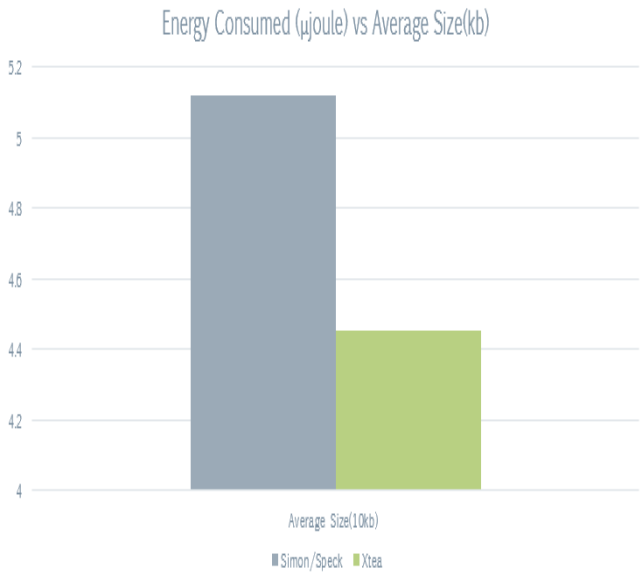

Figure 18: Energy Consumed (µjoule) for Xtea, Simon/Speck lightweight encryption algorithm.

Figure 18 shows that the power consumption of Xtea (4.4517µjoule) is lower than that of Simon/Speck (5.12 µjoule) lightweight block cipher. This is because Simon/Speck requires more clock cycles for the addition and rotation operations, leading to higher power consumption.

.

C. Memory Requirements

The memory requirements for the implementation of the lightweight block ciphers on Node MCU Esp-8266 are determined by measuring the amount of space used by each cipher. The results are shown in Table 1 below:

Table 1: Memory Usage Comparision of Xtea, Simon/Speck lightweight encryption algorithm.

Lightweight Block Cipher	Memory Used (kbytes)
Xtea lightweight block cipher	190.2kb
Simon/Speck lightweight block cipher	229.34kb

Table 1 shows that Xtea lightweight block cipher requires less storage space compared to Simon/Speck lightweight

cipher. This is because Xtea has fewer rotations and addition operations compared to Simon/Speck, making it more efficient in terms of storage.

VI. Conclusion

The use of lightweight ciphers such as Xtea, Simon/Speck has become increasingly important in the context of microcontrollers with limited capacity. Among these ciphers, Xtea has shown to be a better fit than Simon/Speck, as it has demonstrated low execution time, low power consumption, and low memory requirements compared to Simon/Speck. The demand for strong encryption in IoT devices is critical to ensure data security. In this context, lightweight block ciphers offer a viable solution for keeping data secure. Based on the performance tests conducted, it is clear how these lightweight encryption algorithms perform on constrained devices, and which lightweight cipher is best suited for carrying out the encryption. The use of lightweight ciphers such as Xtea and Simon/Speck offers a practical approach to ensuring data security while minimizing the resource consumption on constrained devices.

In 2018, NIST [13] commenced a standardization process to identify and assess potential lightweight cryptographic algorithms suitable for diverse applications. This process involved soliciting proposals from the cryptographic community and conducting comprehensive evaluations to select algorithms that fulfill the requirements of resource-constrained environments. The ongoing NIST Lightweight Cryptography Standardization Process seeks to establish a collection of standardized algorithms that offer both security and efficiency for lightweight applications.

Recently, after careful consideration, NIST made the decision to standardize the ASCON [14] family of algorithms for lightweight cryptography applications due to its capability to meet the requirements of various use cases.

As the NIST did not select Xtea and Simon/Speck algorithms as the LWC standard, our plan is to implement the ASCON family algorithms. We aim to participate in the NIST Lightweight Cryptography Workshop 2023 [15] to further explore the advancements in ASCON family algorithms.

References

[1] Salim Elbouanani, My Ahmed El Kiram, Omar Achbarou "Introduction to the Internet of Things security: Standardization and researches challenges", http://ieeexplore.ieee.org/document/7492741/authors

[2] Ben Dickson "Why IoT Security is so Critical", https://techcrunch.com/2015/10/24/why-iot-security-is-so-critical/

[3] "Securing the Internet of Things: A Proposed Framework" http://www.cisco.com/c/en/us/about/security-center/secure-iot-proposed-framework.html#4

[4] Philip Levis "Secure internet of Things Project(SITP)", http://iot.stanford.edu/workshop14/SITP-8-11-14-Levis.pdf

[5] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, Louis Wingers, "Simon and Speck: Block Cipher for the Internet of Things" http://csrc.nist.gov/groups/ST/lwc-workshop2015/papers/session1-shors-paper.pdf

[6] Mehran Mozaffari-Kermani, Kai Tian, Reza Azarderakhsh, Siavash Bayat-Sarmadi, "Fault-Resilient Lightweight Cryptographic Block Ciphers for Secure Embedded Systems", http://ieeexplore.ieee.org/document/6936334/

[7] Prof.Alan Kaminsky, Ryan Sparlin, Peter Maresca, Justin Kelly, "Xtea Block Cipher" https://people.rit.edu/rab3106/CryptoReport.pdf

[8] Security Challenges in the Internet of Things(IoT), http://resources.infosecinstitute.com/security-challenges-in-the-internet-of-things-iot/#gref

[9] IoT Security Solutions, https://www.digicert.com/internet-of-things/

[10] Mike Hine, "Complexity of IoT may slow adoption", https://www.infosecurity-magazine.com/news/mwc15-complexity-of-iot/

[11] B.Vinayaga Sundaram, Ramnath M, Prasanth M,Varsha Sundaram J, "Encryption and hash based security in Internet Of Things", http://ieeexplore.ieee.org/document/7219926/

[12] Ariel Amster "Internet of Things: Big Data and Data Security Problems", http://www.dataversity.net/internet-things-big-data-data-security-problems/

[13] NIST, LWC https://csrc.nist.gov/Projects/lightweight-cryptography

[14] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, Martin Schläffer, "ASCON: Authenticated Encryption and Hashing", https://csrc.nist.gov/CSRC/media/Presentations/ascon-v1-2-analysis-of-security-and-efficiency/images-media/session2-mendel-analysis-of-security.pdf

[15] NIST Lightweight Cryptography Workshop 2023, https://csrc.nist.gov/events/2023/lightweight-cryptography-workshop-2023