National College of Ireland

Lightweight Cryptography in Embedded IOT Systems

MSc Research Project
MSCCYBE_JANO23_O

Kamal Bassiouny Kamal Tawfik
Student ID: **x22189661**

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Kamal Bassiouny Kamal Tawfik		
Student ID:	X22189661		
Programme:	MSCCYBE_JANO23_O	**Year:**	2023
Module:	MSc Research Project		
Supervisor:	Ross Spelman		
Submission Due Date:	12/08/2024		
Project Title:	Lightweight Cryptography in Embedded IOT Systems		
Word Count:	5381 **Page Count:** 13		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Kamal Tawfik

Date: 11/8/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Lightweight Cryptography in Embedded IOT Systems

Kamal Bassiouny Kamal Tawfik
x22189661

Abstract

Internet of things (IOT) is a hot topic, and applications are increasing every day, IOT has been involved now in sport, healthcare, automotive, smart homes and etc. it depends on o creating a wide range of network that contains nodes(things) that are connected together to gateway and share data through internet. That nodes or things can by monitored, controlled, send and receive data through internet from anywhere. And with that huge increase of IOT application, securing that date became a challenge for the security researchers, on how to achieve the security of the data with complying with IOT limitation in terms of resources such as power consumption, processing speed, memory and etc. the researcher are trying to find the most optimized cryptographic method that enable developer to encrypt data in IOT restrict environments with a good balance of security and resources consumption.

Research has been studying different types and algorithms, and Market already has used some algorithm, maybe optimizing exist general one to IOT environment such AES and other tried to develop special designed algorithm for IOT environment such as PRESENT, SIMON, ChaCha, SIMON, KTANTAN etc. each method tries to meet the IOT constraints and achieve the security and go towards perfect secrecy.

In this research project I will try to do a comparison between most common used Light weight cryptography methods to find out the best recommendation and practices to reduce the overhead execution time, memory consumption and cost when using with Embedded system. The paper states advantage and drawbacks of each method in the study. That would help decision maker and engineers to choose the best method and apply the finding so that they can have a secure system with lower resource consumption. In this paper I looked into three different methods which are the most used based on research Papers in previous work and standards, AES, PRESENT, and ChaCha.

The methodology of the comparison is based on two resources of data First, the Literal review of previous research papers, and the data collected on real code execution on ESP32 MCU which is very popular MCU used in IOT applications.

The result shows that there is a trade-off between security and resource usage. Overall, you can chive more security, but you would need high computing power to overcome processing overhead for example AES algorithms. You can still achieve some good balance if you use SIMON algorithms, but you need to consider its limitation implementation of plain text size

Keywords: IOT, lightweight Cryptography, Microcontroller, Software, Hardware.

1 Introduction

The rapid increase of IOT devices across different fields, from simple home Automation to full smart cities, Medical, automotive and etc. these devices, are developed and designed for specific purposes or function for example a device to collect environment data from a room and apply controls based on that. typical products can be the smart air conditioner or smart security camera. that is why they're fail under embedded systems devices. The semi-conductor company turning to reduce the size and cost of such device to be more embedded into more applications. Also, the software developer tries to reduce the software development overhead such as processing time, memory consumption of these applications. The big challenge is how to secure the data collected or transferred by IOT devices to be secure enough against cyber security attack such as MAN in middle attack without increasing the resource overhead physical or software. The security developer has been trying to find a more optimized light wight cryptography that has a good balance between achieve good encryption level and meet resource limitations. So that, can be used with such devices to enhance the security and encryption of the data that will be shared among the devices (things) with consider the resource limitations so it can enhance the security without adding more overhead such as memory consumption, processing time, cost, board size and etc.

To understand why Lightweight cryptography is important in today's worlds. you should know that installed IOT devices around the world is estimated to be 75.44 billion in 2025 and number is expected to increase each year.

Main MCU in IOT device can vary from be 8-bit to 32-bit microcontroller with 1kb to hundreds of Kbytes of RAM, 1 MHz can be typica processing frequency, and such MCU can be the main processing unit for a node connected in IOT network. The traditional cryptographic method won't fit with such systems.

The primary focus for the developer is to secure such devices that will transfer and hold vulnerable data without effecting the efficiency of the devices and meet embedded IOT constrained resources.

Lightweight cryptography is a cryptographic algorithm used to encrypt data that is used in limited resource system and constrained environments such as: IOT, sensors, Radio frequency applications RFID, or any embedded system projects that has limited resources. It achieves efficiency of end-to-end communication in order to achieve end to end security for limited resources applications such as battery driven devices where you keen about low power consumptions. It also has small memory footprint which is a very key point in IOT or embedded system applications, so it requires less RAM, Less Power supply, less computing resource and processing time.

My research focuses on the main problem of the trade-off between security of IOT device in respect to cryptography method to move towards perfect secrecy and the limitation of IOT devices. The research states why normal general cryptography methods can't be used withing IOT devices. The paper compares and analysis few of the commonly used lightweight cryptography methods and how they meet or don't meet IOT limitations.

The comparison and data analysis are based on two methods: review of previous research papers, and the practical implementation of the cryptography methods on IOT MCU EXP32 which is very common in IOT applications.

At the end this paper should answer the research question how to achieve lightweight cryptography for IOT device. The methodology of the research is to evaluate the performance of the selected Lightweight Cryptographic methods (AES, PRESENT, ChaCha, SIMON and KTANTAN) in terms of Execution time, memory usage and computational overhead. Also

gives the recommendation of how to achieve the cryptographic methods with lower resources based on the result of the research.
However, the second path, implementing the Methods and record the data, is limited by resource and available libraries. Open-source libraries are used as a base of that method and due to this some libraries are not complete and maybe data gathered are affected by the open-source implementation.

The following section is the Related Work which provides a detailed review of previous research papers that explain the existing lightweight cryptography and its Application of IOT and highlighting the advantage and limitation of each method.

Section 3 is the methodology which describe ethe experimental setup and the implementation of Selected Lightweight cryptography algorithm on ESP32 microcontroller. The benchmarking results are described in section 4 to show the data collected from the ESP32 microcontroller showing the comparison between the applied cryptographic method in terms of execution time, memory usage and overall efficiency. Also summarize the key finding of the study and offers the recommendation based on the data and also provide the direction of the future work. At the end, section 5 provides the conclusion and future work.

2 Related Work

2.1 IOT devices Growth and need for data encryption

The huge increase of IOT devices has introduces big changing on how to secure such devices with the limitation of resource-constrained environments. The IOT devices may hold very critical data such as home devices measurements in smart homes or can hold critical information and sensitive data on the person itself in healthcare applications. Typical applications where a device is designed and developed to measure body measurements such as temperature, heart rate, pressure, sleeping time and other body data and other sensitive the measurement to the Doctors with the person personal information such as name, date of birth, address, and other sensitive data. With the connectivity of billions of IOT devices holding, transmitting and receiving sensitive data. It is crucial to encrypt such data and try to achieve the perfect secrecy against IOT typical threats.

However, IOT devices work in limited constrained environments compared to general purpose computer in terms of processing power, memory capacity and battery lifetime. This limited environment makes it impractical to use traditional cryptographic algorithm on such devices [1].

Lightweight cryptography tries to resolve the challenging of IOT devices security by providing more optimized method in order to achieve the security and meet the IOT environment criteria. The primary objective for the developer and researcher is to maintain an adequate level of security while minimizing the algorithm footprint in terms if memory usage. Processing cycles, and power consumption [2].

In 2003 a study made by Cisco Internet Business Solutions Group (IBSG) to measure how many devices are connected through internet [2] [3], taking into consideration the human population. The study shown that there were 500 million devices connected to the internet when human population was 6.3 billion. In 2010, this number increased to be 12.5 billion when human population was 6.8 billion. They predicted the number to be 50 billion by 2020.

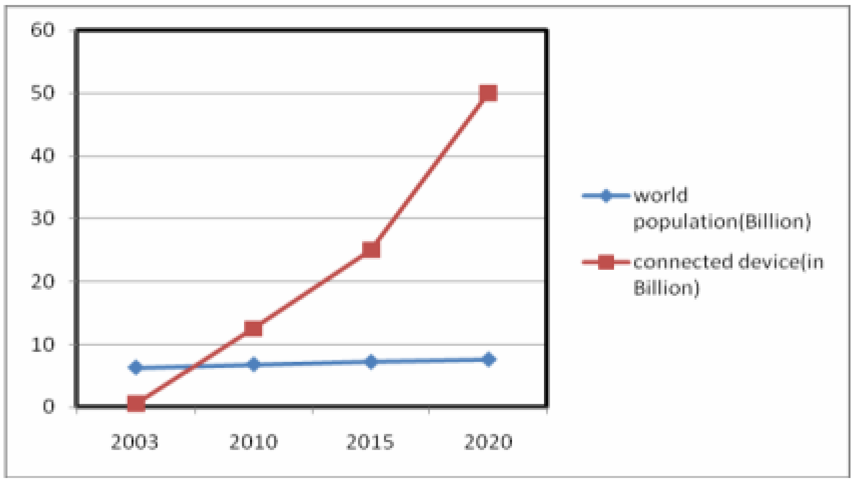


Fig.1: Year Vs world population and connected devices

Statista [4], which is a research website focused on publishing statics data, has published research regarding the number IOT connection worldwide from 2022 to 2023and the forecast for 2024. As you can see in figures only in IOT, there is currently 13.8 billion connections and number expected to increase with linear rate to achieve 39.6 billion connections by 2033.

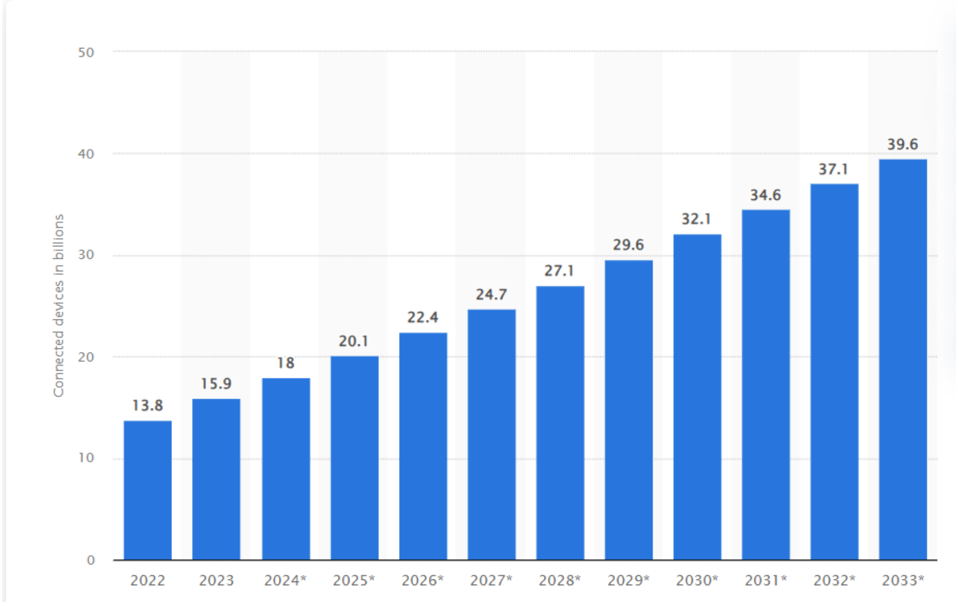


Fig 2: Number of IOT connections per year

Therefore, a well security design is required to achieve the security while moving towards automation and make everything connected to the internet. Encrypting the shared data is important key in achieving the security of IOT devices. Cryptography of the data in IOT, ensures the confidentiality, integrity and authenticity id data in IOT network, enabling secure communication and protect the devices from cyber attacker or security threats.

2.2 Limitation of IOT environments and need of lightweight cryptography

As we discussed that growth of IOT devices in today's world makes it crucial to encrypt the shared and transferred data between the IOT devices and connections. However, IOT devices work in constrained environments resulting with limitation need to be taken into consideration while developing a cryptography method that can be used in IOT applications. And from that we had the lightweight concept that makes new cryptography method that has lighter footprint. In this section let's discuss some of these limitations.

Resource Constrains: this is the most challenge in IOT devices, as it consists of low power micro controller (MCU) as the main processing unit, not like normal pc or laptops which have huge processing power. The low production cost is main feature of IOT and when going to use low-cost sensors and controllers that means they come with minimal footprints in terms of processing power or memory capacity. Also, power consumption is a crucial element as IOT devices designed to have long battery life to be embedded in long life applications such as wearable medical devices like smart watches [1][2].

Different flavours if IOT devices: IOT is used in wide range of applications such as smart homes, healthcare, Automotive, agriculture and etc. therefore, you can find different as a result of the range of application and each one has its own characteristics. So, what algorithm works in IOT automotive application won't fit in Smart home applications due too difference in standards, market, costumer behaviour, devices used and etc.

Scalability challenges: as mentioned above, IOT is rapidly increasing over time. The innovation of new technology in IOT field is rapidly changing, that add more challenge to update the algorithm and manage key distribution. Regular cryptography methods are not designed to be regularly update with high range or change so maybe new method or update is required to meet these criteria.

Power consumption: the IOT devices need to be energy efficient, for example EI Electronics, which is Irish electronics company for alarm sensors, designs and develops low power alarm sensor that do GAS sensor and alarm sensor, and their main target and promotion is energy efficiency and low power consumption. You don't have electric socket or huge batter, but you can find the device has only 5 voltage battery as main power supply, hence you need to design the cryptographic method to use minimum power as much as possible.

Real time processing: due too the criticality of IOT applications, it requires real time processing which means all tasks and sensor need to be time specified and meet deadline criteria in respect to time. For example, IOT devices use watchdog timer to keep trace of all function and kill any stuck function to not affect system behaviour and other tasks. A lightweight cryptography algorithm that uses standard processing time and meet real time requirements of embedded IOT is mandatory when designing the algorithm. That is why some markets use hardware crypto rather than software. for example, Hardware security module (HSM) in Automotive is highly used as a standalone encryption and decryption unit.

Privacy concern: the IOT devices share sensitive data which require applying confidentiality. Attacker can use different threat vector to obtain the sensitive information. Cryptography must take into consideration the security of such sensitive data. When thinking of reducing resource usage, power consumption and minimum footprint, the developer must not reduce the security and must meet the security requirements in order to achieve confidentiality.

2.3 Current algorithm

Much research has been made into designing Lightweight cryptography methods that can achieve encryption and meet IOT device limitations.
The optimization of cryptographic algorithm to meet the IOT environment limitation either hardware based, or software based.

Hardware optimized Based Cryptography

In this option, the algorithm uses the advantage of hardware, so the optimization is more based on optimizing the hardware and some sometimes design detected hardware to do the encryption for example using FPGA, Field Programable Gate Array, like the case in automotive HSM.

Also, options in hardware that when designing the algorithm is to design it based on the available hardware so it is developed, and flashed to run on specific hardware that will give the algorithm the limitation requirements. And you need to adapt the algorithm if you change the hardware. For example, PRESENT which is a block cipher algorithm that is designed to use minimum gate equivalent. SIMON and Speck are developed by NSA, and they are families of block cipher.

Such algorithms are highly efficient in terms of processing power and execution time. However, they require specific hardware requirements which may lead to higher cost, higher hardware complexity, and increase board size which sometimes make it not suitable for some applications.

Software Based Cryptography

Software optimized algorithms focus more on software side. It takes the current general Cryptographic method and try to optimize the software to meet the IOT environmental requirements. Lightweight version of Advanced Encryption Standard (AES) is perfect example of such algorithms. It is optimized to use less memory, less processing time.

However, it reduces the security level of the algorithm so there is a trade-off between the optimization achieved and the overall security of the system. For example, lightweight version of AES tends to reduce the complexity and instruct sets of the algorithm in order to use lower memory and RAM consumption, and with lower instruction set you can increase the speed of the encryption.

Symmetric vs Asymmetric Cryptography

The cryptography algorithms in general might fail either into Symmetric key, or asymmetric key encryption.

Symmetric key is where same key is used in encryption and decryption. It is the one preferred in IOT application due to its efficiency and no ned to transmit key, so it is simpler as you have only one key that is prestored in all nodes and no ned for key calculations or transmission over the channel.

However **Asymmetric** key, where different keys are used on both sides, offer of key management and provide signature and more security over Symmetric key algorithm. Example Elliptic Curve Cryptography (ECC), but Asymmetric methods use intensive resources and researchers are trying to do more optimization but still can’t compete with Symmetric key algorithms in terms of resource consumption, hence Symmetric key algorithms such as AES, SIMON are more preferred in IOT applications.

3 Research Methodology

In this section I will try to evaluate and compare the performance of selected lightweight cryptography algorithm AES, PRESENT, Chacha, SIMON and KTANTAN. The comparison is to base on their usage in encrypting data in Internet of things (IOT) applications with its limited resources and constrains regarding power consumption, memory, processing time. The study is important to see how much security can achieved while provide optimized usage of the IOT device resources.

The experiment is based on implementing the selected cryptographic algorithm on IOT devices. One of the most common IOT device is ESP32. ESP32 is a very common micro controller used in IOT applications.
The implementation of the encryption method will be based on the exist open-source libraries. However, some Libraries are not found so a simple implementation might be used. The idea is not to implement a new algorithm but more to compare the most common used

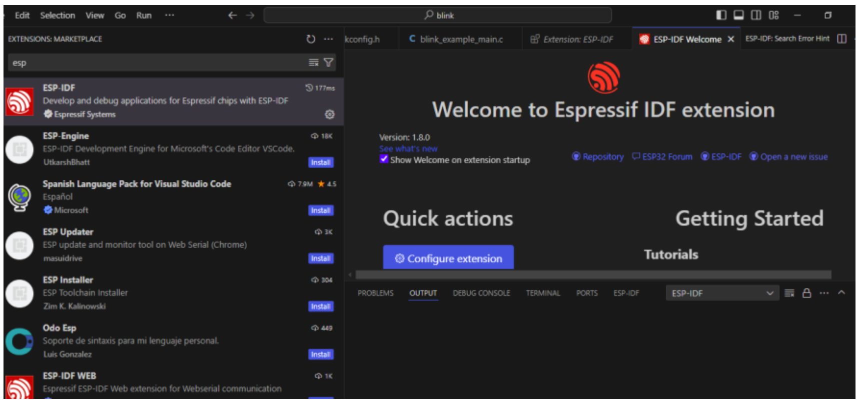
Algorithm in the IOT encryption and get a recommendation of how we can do more optimization in terms of processing time, power consumption, memory usage, and other resources if possible.

3.1 Development environment setup

The development environment setup is a very important part of the experiment as it shows which environment and constrains or limitation while doing the test and show the accuracy of the collected data. Also, it’s how if there was a variation or difference between the experiment and real case scenario.
In this section I will go through how I setup the environment and describe each element of the experiment.

3.1.1 Visual studio setup

Visual studio code is a software code editor used with software development. it is the selected one for being the primary integrated development environment (IDE) as it has good support for C/C++ development with a lot of extensions that can be adapted and included based on the project needs.
Installation: Visual studio code can be downloaded and installed from the official Vs Code website [7].
Extension installation: The “ESP-IDF” extension was installed within VS code extensions marketplace. The extension provides integration environment for IOT development on ESP32 by Expressif and enable development, debugging, monitoring and collecting data from the device [8].



ESP-IDF — 177ms
Develop and debug applications for Espressif chips with ESP-IDF
Espressif Systems

ESP-Engine — 18K
ESP-IDF Development Engine for Microsoft's Code Editor VSCode.
UtkarshBhatt — Install

Spanish Language Pack for Visual Studio Code — 7.9M ★ 4.5
Español
Microsoft — Install

ESP Updater — 3K
ESP update and monitor tool on Web Serial (Chrome)
masuidrive — Install

ESP Installer — 304
ESP Toolchain Installer
Zim K. Kalinowski — Install

Odo Esp — 449
Soporte de sintaxis para mi lenguaje personal.
Luis Gonzalez — Install

ESP-IDF WEB — 1K
Espressif ESP-IDF Web extension for Webserial communication

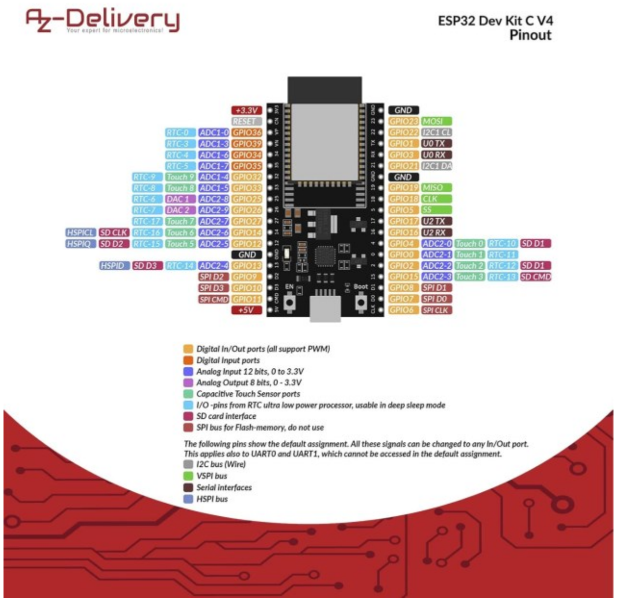
Figure 3: ESP-IDF with Visual studio code
Configuration of ESP-IDF: the ESP-IDF extension for VS code requires initial setup and configuration usually is done by following the setup and configuration wizard. I used the blink led example configuration s my basic configuration and built the crypto libraries over that configuration. Configuration Manual should have all configuration setup.

ESP32 Hardware

ESP32 Microcontroller is a very common in IOT applications. It is a system on a chip (SOC) that combines WI-FI and Bluetooth connectivity with sufficient processing capabilities to handle lightweight cryptographic operations. ESP32 has many options, I have used Esp32 Dev KitNodeMCU Wroom-32 and it comes with following features:

- 2.4 GHz dual-mode WI-FI
- TSMC Bluetooth chip
- 40nm low-power technology
- Dual high performance Xtensa 32-bit LX6 CPU cores
- Digital input and output pins

- Peripherals pins


AZ-Delivery
Your expert for microelectronics!
ESP32 Dev Kit C V4
Pinout
Digital In/Out ports (all support PWM)
Digital Input ports
Analog Input 12 bits, 0 to 3.3V
Analog Output 8 bits, 0 - 3.3V
Capacitive Touch Sensor ports
I/O -pins from RTC ultra low power processor, usable in deep sleep mode
SD card interface
SPI bus for Flash-memory, do not use
The following pins show the default assignment. All these signals can be changed to any In/Out port.
This applies also to UART0 and UART1, which cannot be accessed in the default assignment.
I2C bus (Wire)
VSPI bus
Serial interfaces
HSPI bus
Figure 4: ESP32 Dev kit Cv4 Pinout

3.1.2 Selection of Cryptographic algorithms

I tried to select the most common algorithm used in IOT applications. Some of the algorithm has open-source library and that is used on the implementation. Others couldn't find open-source libraries and sometimes it complexes to implement. So, for the complex ones was more review of the algorithm and maybe simple implementation of the basic elements to show how ESP32 will take to process and that might give indication of the footprint of the application itself.

AES (Advanced Encryption Standard): AES is a well know encryption algorithm and has many usages in computer security. Despite being more heavy weight algorithms but many IOT application has taken the advantage of AES by developing simpler version of AES to be used in IOT applications as a lightweight Cryptography algorithm. I used standard AES library comes with ESP-IDF extension with ESP environment on VS code.

PRESENT: PRESENT is the most common cryptography method. It is a block cipher developed by orange labs Ruhr University and technical University of Denmark. It is considered to be ultrahigh algorithm for extremely resource-constrained environment such as RFID, IOT devices. Its low footprint makes it identical choice for IOT application in the most cases.

Andrey Bogdanov et al has published first Ultra-Lightweight Block Cipher PRESENT [8] that can be suitable for IOT devices and resources constrained environment. It is based on SP-network and consists of 31 rounds, it encrypts 64-bit blocks with a key with a length of 80 or 128 bits [8]. I have chosen to test based on open-source library on Github [7]. Integrate the source and header file into my project and apply the encryption to measure the execution time of the algorithm.
PRESENT should be 2.5 smaller than AES and has lower power consumption. However, on the other hand it provides less security than AES.

ChaCha: is a stream cipher .it a lightweight method focuses on optimization in the software point of view to provide a good balance between security and performance [9].

SIMON: it is a lightweight cryptography method that is developed by National Security Agency (NSA) in June 2013. Its hardware is optimized to be integrated within IOT environments [6][10].

3.2 Implementation and Benchmarking strategy

I have used the advantage of already developed example of ESP32 by Espressif IDF extension with VS code. I used Blink Led example and add encryption APIs over it and also implement a simple benchmarking algorithm by calling each encryption method with different key size and different plain text sizes to measure the processing time and memory consumption for each function call.
I have implemented the code algorithms for the following:
AES: implemented using mbedTLS library.
PRESENT: I used open-source lib from GitHub. The algorithm us limited to a block size of 8 bytes Due to limitation of test environment.
ChaCha20: implementation is based on open-source GitHub library and the key size was 32 bytes, and padding is used in case ok keys smaller than 32 bytes.
Simon: the implementation is taken from GitHub and adapted to be used with ESP32 framework.

3.2.1 Benchmarking strategy

A benchmarking framework was developed to test the performance of each encryption algorithm.

The main concept is to try test different key size, if possible, with same plain text, and also test different plain text size with same key size for each encryption method and measure the execution time for each test case.
The test cases can be categorized to:
Vary the key size: Each algorithm was tested with key sizes of 16, 24, and 32 bytes (if possible).
Vary the plaintext size: Tests were executed with plaintext sizes of 8, 16, and 32 bytes.

The measure variable is **the Measure execution time**. The ESP32's built-in high-resolution timer was used to measure the execution time of each encryption operation, providing microsecond-level precision. Tera Term is used to, monitor the serial data generated by ESP32 and log the data into log file, any serial monitor can do the task.

The framework was implemented in C and integrated into the ESP-IDF environment. The benchmark results were output to the serial console, allowing for easy collection and analysis of data.

3.2.2 Test Steps:

The following steps summarize the test procedure of the experiment:
Initialization: The ESP32 was initialized, and each encryption algorithm was prepared by setting up the necessary key, plaintext, and algorithm-specific parameters.
Execution: For each combination of key size and plaintext size, the encryption algorithm was executed, and the time taken for the operation was recorded.
Output: After each encryption operation, the resulting ciphertext and the execution time were output to the serial console for logging.
Repetition: This process was repeated for each algorithm, with different key sizes and plaintext lengths, ensuring comprehensive coverage of possible test cases.

3.2.3 Limitations

Despite implementing the algorithms but there were limitation comes from code limitation or test environment limitations. Those limitations are:
PRESENT Cipher: The algorithm's limitation to 8-byte blocks restricted its use to very small plaintexts. Larger plaintexts had to be executed, limiting the result range and data analysis and comparison to another algorithm.
ChaCha20 Key Size: ChaCha20's requirement for a 32-byte key meant that smaller keys had to be padded, and this might not be the case for example test cases were done to use 16- and 24-bits key, so they all treated as 32 bit key.
Simon Cipher: Although Simon supports a variety of block sizes, the implementation was limited to testing 16-byte plaintexts due to the block size constraints.
Hardware Constraints: The ESP32's computational limits and lack of hardware acceleration for some algorithms could impact Result and discussion as the result may vary if applied same experiment in different hardware platform.

4 Result and discussion

Test Case	Encryption Algorithm	Key Size (Bytes)	Plaintext Size (Bytes)	Execution Time (microseconds)	Source of Code
1	AES	16	8	26,471	mbedTLS library
2	PRESENT	10 (80-bit)	8	6,080	Custom, GitHub
3	ChaCha20	16	8	26,735	Custom, GitHub
4	Simon	16	8	5,500	Custom, GitHub
5	AES	16	16	48,698	mbedTLS library
6	PRESENT	10 (80-bit)	16 (limited to 8 bytes)	15,890	Custom, GitHub
7	ChaCha20	16	16	48,958	Custom, GitHub
8	Simon	16	16	10,800	Custom, GitHub
9	AES	16	32	93,142	mbedTLS library
10	PRESENT	10 (80-bit)	32 (limited to 8 bytes)	15,890	Custom, GitHub
11	ChaCha20	16	32	93,402	Custom, GitHub
12	Simon	16	16	10,800	Custom, GitHub
13	AES	24	8	26,476	mbedTLS library
14	ChaCha20	24	8	26,736	Custom, GitHub
15	Simon	24	8	5,600	Custom, GitHub
16	AES	24	16	48,697	mbedTLS library
17	ChaCha20	24	16	48,958	Custom, GitHub

18	Simon	24	16	11,000	Custom, GitHub
19	AES	24	32	96,006	mbedTLS library
20	ChaCha20	24	32	96,267	Custom, GitHub
21	Simon	24	16	11,000	Custom, GitHub
22	AES	32	8	27,256	mbedTLS library
23	ChaCha20	32	8	27,518	Custom, GitHub
24	Simon	32	8	5,700	Custom, GitHub
25	AES	32	16	50,173	mbedTLS library
26	ChaCha20	32	16	50,433	Custom, GitHub
27	Simon	32	16	11,200	Custom, GitHub
28	AES	32	32	96,006	mbedTLS library
29	ChaCha20	32	32	96,267	Custom, GitHub
30	Simon	32	16	11,200	Custom, GitHub

Table 1: Test results

Table 1 shows the summery of test result of the test cases. by looking to the table, we can see that **Simon Encryption** has faster execution times compared to AES and ChaCha20, especially with smaller plaintext sizes. Also, as the limitation **Simon** is Limited to a maximum plaintext size of 16 bytes due to its block size constraints.

PRESENT Encryption has the fastest execution times for 8-byte plaintexts but cannot handle larger plaintext sizes effectively due to block size limitations. **AES** and **ChaCha20** Both show consistent and predictable scaling with increasing plaintext sizes.
The result shows the following consideration and key points when choosing Lightweight cryptography method from the selected methods:

Algorithm	Pros	Cons	Best Use Case	Security Level	Recommendation
PRESENT	- Extremely fast for small, fixed-size data blocks (8 bytes) - Low computational overhead	- Limited to 8-byte blocks - Not suitable for larger plaintext sizes	- Ideal for very small, fixed-size data (e.g., RFID tags, sensor data)	Low to Moderate	Recommended for applications with very small, consistent data sizes and tight resource constraints
Simon	- Efficient with low computational footprint - Good performance across	- Limited to 16-byte plaintext in the tested configuration	- Suitable for low-power, embedded systems with small and consistent data sizes (e.g., IoT	Moderate	Strong candidate for low-power and resource-constrained environment

	different key and plaintext sizes		devices, smart cards)		s with small data sizes
AES	- Standardized and widely used - Strong security guarantees - Flexible with key and data sizes	- Higher computationa l overhead compared to PRESENT and Simon	- Applications where security is paramount and computational overhead is acceptable (e.g., secure communications , data storage)	High	Recommended for high-security applications where computational cost is acceptable
ChaCha20	- Fast and secure - Flexible in key sizes - Easier to implement securely than AES in some environment	- Requires padding for smaller key sizes	- Secure streaming, mobile applications, or where performance is critical and high security is needed	High	Recommended for performance-critical applications with high security requirements

Table 2: test observation and recommendations

my recommendation based on the test is that **Simon** is a very good choice for having a good balance between security and efficiency. However, it restricts the developer to use small plain text if 8-16 bytes. If your objective is more into secure the system, you should consider Chacha or AES choice. for more security, in case the hardware and resource Is enough to handle processing overhead, so AES is better choice to achieve better level of security.

5 Conclusion and Future Work

The research studied the importance of having lightweight cryptography and stating the limitation of IOT devices which shows the important of developing a lightweight cryptography algorithm.
Part two of the research was based on practical implementation of selected exist algorithm and analysis and compare the performance if each selected algorithm with different key or plain text sizes.

The selected methods were PRESENT, SIMON, AES and Chacha. The implementation developed and tested on ESP32 platform which is a common Micro controller for IOT applications. The objective of the test is to compare the execution time for each encryption and try to find the best option to balance between security and efficiency.

The results showed that: Simon showed fast execution time with limitation of small plaintexts (8-16 bytes). It is highly efficient, making it an excellent choice for low-power, resource-constrained environments where data sizes are small and consistent. However, its security level is moderate, and its applicability is limited to smaller plaintext sizes. While PRESENT has the quickest execution times for 8-byte plaintexts, making it suitable for very small, fixed-size data encryption, such as in RFID tags or sensor data. However, it struggles with larger data sizes due to its fixed block size of 8 bytes, which significantly limits its scalability and range of applications.

AES and ChaCha20 provided solid level of security. AES is a great choice for achieving best security, but it comes with a higher computational overhead. ChaCha20 offers similar security benefits with slightly better performance in some scenarios.

The future work might be including more algorithm in the study such as Speck, KTANTAN for more inclusion. Also include more test scenarios and use more powerful microcontroller to apply more tests and make the full benefits of the algorithm.
The study didn't take into consideration the decryption process which is a very key point when comparing between cryptographic algorithms, future work shall include the encryption process to compare how secure each application. Also, how long or how many cycles need to be taken to decrypt the data for each algorithm.

References

[1] Okello, W.J., Liu, Q., Siddiqui, F.A. and Zhang, C., 2017, July. A survey of the current state of lightweight cryptography for the Internet of things. In 2017 International Conference on Computer, Information and Telecommunication Systems (CITS) (pp. 292-296). IEEE.
[2] Dutta, I.K., Ghosh, B. and Bayoumi, M., 2019, January. Lightweight cryptography for internet of insecure things: A survey. In 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0475-0481). IEEE.
[3] Soumyalatha, S.G.H., 2016, May. Study of IoT: understanding IoT architecture, applications, issues and challenges. In 1st International Conference on Innovations in Computing & Net-working (ICICN16), CSE, RRCE. International Journal of Advanced Networking & Applications (Vol. 478).
[4] https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/
[5] https://www.eielectronics.ie/
[6] Wetzels, J. and Bokslag, W., 2015. Simple SIMON: FPGA implementations of the SIMON 64/128 Block Cipher. *arXiv preprint arXiv:1507.06368*.
[7] https://github.com/Pepton21/present-cipher
[8] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y. and Vikkelsoe, C., 2007. PRESENT: An ultra-lightweight block cipher. In Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9 (pp. 450-466). Springer Berlin Heidelberg.
[9] https://github.com/983/ChaCha20
[10] https://github.com/983/ChaCha20