



Internship at Centre for IoT

Department of Computer Science & Engineering

*PES University (Established under Karnataka Act no. 16 of 2013)
100ft Ring Road, BSK 3rd Stage, Hosakerehalli, Bengaluru – 560085*

“Benchmarking cryptographic algorithms on resource constrained ESP8266”

Submitted by:

Nagesh B S

PES1UG20CS816

Under the guidance of

Prof. Vadiraja Acharya

Assistant Professor,
PES University

8 Weeks Internship

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India



PES University



PESU Center for
**Internet
of Things**

(Established under Karnataka Act no. 16 of 2013)

100ft Ring Road, BSK 3rd Stage, Hosakerehalli,
Bengaluru - 560085

**Department of Computer Science &
Engineering**

Centre for IoT

Date: 17th June'22

To,
Nagesh BS
PES1UG19CS816
6th Sem, CSE
PESU-RR

We are pleased to offer you an internship position as Student in the Centre for Internet of Things for the duration of two months from **20th June 2022 to 20th Aug 2022**. On your first day report to **Mr. Vadiraja Acharya**. In addition to your duties as an intern, you are to fulfill tasks assigned to you by your supervisor and indulge in all the documentation pertaining to your internship. Failing which, you would not be entitled for claiming credits/certificates for the same. You will return all the documents, equipment and property of the Centre on completion.

Congratulations on your selection. Welcome aboard.

With Best Regards,

Vadiraja Acharya,
C-IoT, PESU

DECLARATION

We hereby declare that the project entitled “Benchmarking cryptographic algorithms using ESP8266” has been carried out at C-IoT, PESU by me under the guidance of Mr. Vadiraja Acharya, Assistant professor C-IoT, PESU and submitted in partial fulfilment of the credits for the degree of **Bachelor of Technology in Computer Science and Engineering of PES University, Bengaluru** during the academic semester for the duration of 8 weeks. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1UG20CS816

Nagesh B S

Nagesh B S

ACKNOWLEDGEMENT

I would like to express my gratitude to our guide Mr. Vadiraja Acharya, Assistant professor C-IoT, PESU for their continuous guidance, assistance and encouragement throughout the development of this project.

I am grateful to the internship coordinator Prof. Mahitha G, Dept. of Computer Science and Engineering, PES University for organizing, managing and helping out with the entire process.

I take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department.

I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way.

Finally, this internship could not have been completed without the continual support and encouragement I have received from my parents and my friends.

ABSTRACT

The **Internet of Things** is a network of smart devices that connect to each other in order to exchange data via the internet without any human intervention. As more and more IoT devices are emerging in today's world which are deployed in uncontrolled and hostile environments, it is important to safeguard these IoT devices which contain sensitive data.

In this project, the focus is to provide an experimental benchmark study that shows the cost (e.g., processing time of encryption and decryption algorithms) of applying different security protocols on restricted devices equipped with lightweight Arduino ESP8266 sensor platform.

The keys used for encryption and decryption are generated randomly without the use of built in function to ensure more security.

TABLE OF CONTENTS

Content	Page No.
1. Introduction	1
2. Brief Introduction of the Company	2-
3	
3. Internship Project Details	
3.1 Project Title	3
3.2 Introduce the project with respect to roles and responsibilities	3-4
4. Project abstract and scope	4
5. Project design details with technology used	
5.1 Hardware platform required for our project	4
5.2 Operating Systems and Versions	4
5.3 The ESP8266 board	4-
5	
5.4 NodeMCU ESP8266 Specifications	5-
6	
5.5 ESP8266 PinOut configurations	6
5.6 NodeMCU ESP8266 Block Diagram	6-
7	
5.7 Cryptographic Libraries	7-
13	
5.8 Random Number Generations	13-
15	
6. Code Implementation/Working	
6.1 AES	16
6.2 Speck	16
6.3 ChaCha	16
6.4 ChaChaPoly	16-
17	
6.5 Random Number Generation	17
7. Project Results/ Outcomes	17-
20	
8. Conclusion	20
9. References	20-
21	

1. Introduction:

The internet of things (IoT), is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. These devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analysed or analysed locally.

In today's world, as the number of connected devices increases and more information is shared between devices, the potential that a hacker could steal confidential information also increases. As most of the devices may be battery operated, due to less processing power the security and privacy is a major issue in IoT.

Internet of Things (IoT), works to provide various services by linking people on the one hand with sensors IT components and actuators on the other hand, these services are widely used. Also, the number of connected devices in the Internet of Things is increasing tremendously. Whereas in 2015, nearly 15 billion devices were connected, and in 2019 they reached nearly 26 billion, and the number may reach about 75 billion devices by 2025, and the Internet of things market around the world has doubled since 2016, and forecasts estimate. Whereby 2020 it could reach about \$ 457 billion. With the exponential growth of IoT, there is a growing problem of security threats. There are some reasons that make devices connected to the Internet of things vulnerable to these threats and attacks, including:

- An attacker can have physical access to these devices, as most of them operate without human intervention.
- Attackers can eavesdrop on these devices because they are linked to wireless networks among themselves.
- These devices do not support complex security algorithms due to the nature of their installation, which depends on a low power level, as well as for the low computational capabilities.

IoT technology can be one of three types:

- Directed via the Internet which, means that it works as an intermediary program.
- Object-oriented means that it provides the ability of sense.
- Semantic vector and this type allow access to knowledge, and this type depends on the working principles of a specific application. A mixture of types or just the independent Internet of Things can be used to build smart applications aimed at solving problems in everyday life.

To ensure IoT security, the following objectives must be met:

- Confidentiality: (protecting data from unauthorized disclosure).
- Safety: (ensuring cannot be modified that data if not obtained prior permission).
- Availability: (ensure access to data as needed).

2. Brief introduction of the Company:

PESU Centre for Internet of Things – PESU IoT Fully funded by PES University

Internet of Things (IoT), a network of things or devices connected to the internet, creates new customer experiences, formulates new business models, improves operational efficiencies, and thus it has received humongous attention recently. IoT is the lynchpin enabling augmented, virtual, and mixed reality to provide more natural and immersive ambient experiences within the digital world. With the advent of 5G and associated wireless technologies, the barriers of physical proximity have faded, enabling the easy collection, processing, storage, and visualization of the data with the aid of cloud computing. The McKinsey report estimates the IoT economy to be worth between four to eleven trillion USD by 2025. It is incumbent on us to seize the unprecedented opportunities and challenges presented by IoT. Consequently, PES University has fully funded and created the Centre for Internet of Things (C-IoT) to focus and work on the IoT models and research to overcome the above discussed issues.

Objectives:

- Provide students and faculty opportunities to work on IoT domains and understand the implementation challenges of greenfield and brownfield projects.
- Address the security of IoT platforms comprehensively working with Centre for Information Security (ISFCR).

- Form a research and development community with cross disciplinary collaboration, including engineering, non-engineering, communication, electronics, microsystems, information systems, and software, to focus on challenges in IoT issues.
- Pursue research in IoT technology, applications, services and Publish their work.

3. Internship project Details:

3.1 Project Title:

Benchmarking cryptographic algorithms on resource constrained ESP8266

3.2 Introduce the project with respect to your roles and responsibilities.

The spaces around us are becoming equipped with devices and appliances that collect data from their surroundings and react accordingly to provide smarter networks where they are interconnected and able to communicate with one another. These smart networks of devices and appliances along with the applications that utilize them build smart spaces known as Internet of Things (IoT). With the on growing popularity of such smart devices (e.g., smart cars, watches, home-security systems) and IoT, the need for securing these environments increases. The smart devices around us can collect private and personal information, and the challenge lies in maintaining the confidentiality of the collected data and preventing unsecured actions—from tapping into surveillance cameras to tracking someone’s daily schedule.

For example, digital health, devices that record personal data from blood pressure, heart rate, weight and daily activities sensors are storing the personal data of users for processing and monitoring and may give future recommendations. If such personal information reaches unwanted third parties who distribute or use the data without user consent or knowledge, they are attacking the user’s confidentiality. Therefore, selecting the appropriate security protocols and procedures is critical. The limited processing, storage and power capabilities.

In this project, the focus is to provide an experimental benchmark study that shows the cost (e.g., processing time of encryption and decryption algorithms) of applying

different security protocols on restricted devices equipped with lightweight Arduino ESP8266 sensor platform.

In addition to our project, we are generating random set of bits such as 128bits,192bits,256 bits in place of secret key. Random number generators (RNGs) are essential for cryptographic applications and form the foundation of security systems. For IoT devices, an RNG is generally implemented by incorporating hardware peripheral controllers, which are proving to be imperfect as a source for real randomness because they start with a deterministic input.

4. Project abstract and scope:

The project that we are undertaking will be helpful to other users who are searching to run their cryptographic algorithms to choose the board on which to run. The benchmarking tests that we have run will give a clear idea on the time it takes for encrypting and decrypting a message on various algorithms and how the time varies with respect to the length of the key and message on an ESP8266 board. The developer can then compare these values with those values that are already available online for the Arduino Uno and then select the board where he wants to run the cryptographic algorithm.

Also, the key generated from random numbers will be very helpful for security purposes as a hacker won't be able to guess what the encrypted message will be as the key for each encryption will be different and totally random.

5. Project design details with technologies used.

5.1 Hardware platform required for our project:

Arduino
ESP8266

5.2 Operating system and versions:

IoT Operating System #1: Linux
IoT Operating System #2: Windows 10 / 11

5.3 The ESP8266 Board:

The NodeMCU ESP8266 development board comes with the ESP-12E module containing the ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

NodeMCU can be powered using a Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.

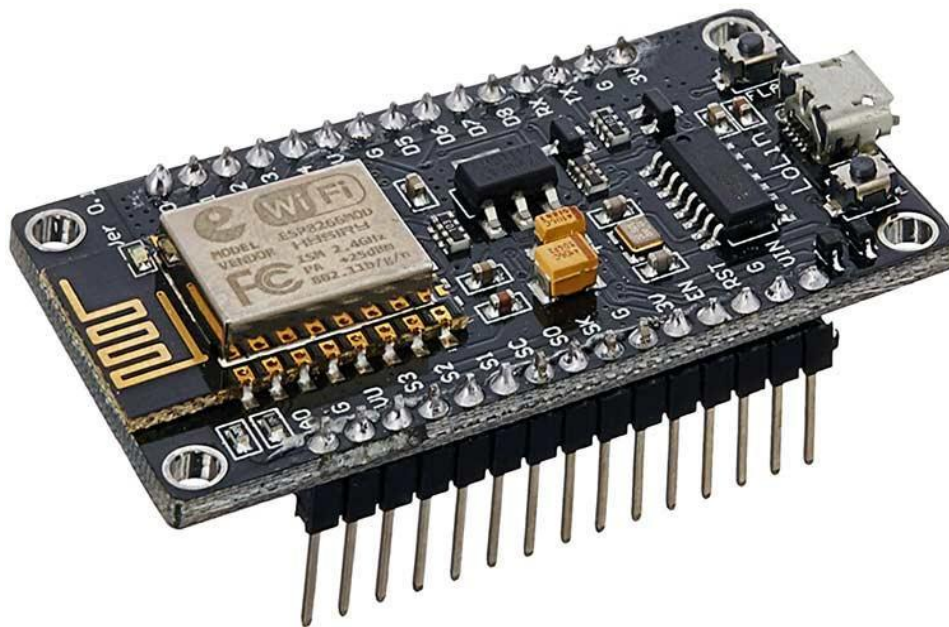


Fig2: The NodeMCU ESP8266 board

5.4 NodeMCU ESP8266 Specifications & Features:

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1

Benchmarking cryptographic Algorithms on resource constrained ESP8266

- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects.

5.5 Node MCU ESP8266 Pinout Configuration:

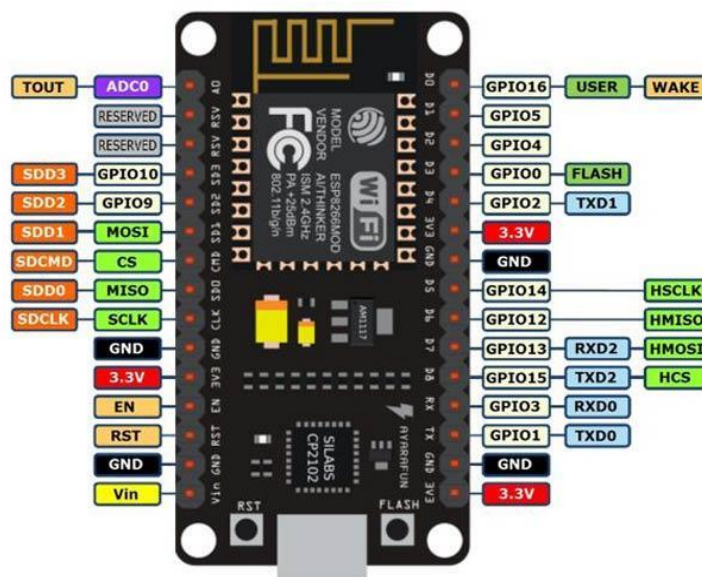


Fig3: Node MCU ESP8266 Pinout Configuration

5.6 NodeMCU ESP8266 Block Diagram:

ESP8266EX integrates antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters and power management modules. The compact design minimizes the PCB size and the external circuitry. ESP8266EX enables sophisticated features, such as:

- Fast switching between sleep and wake-up modes for efficient energy use;
- Adaptive radio biasing for low-power operation;
- Advanced signal processing;

- Spur cancellation;
- Radio co-existence mechanisms for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

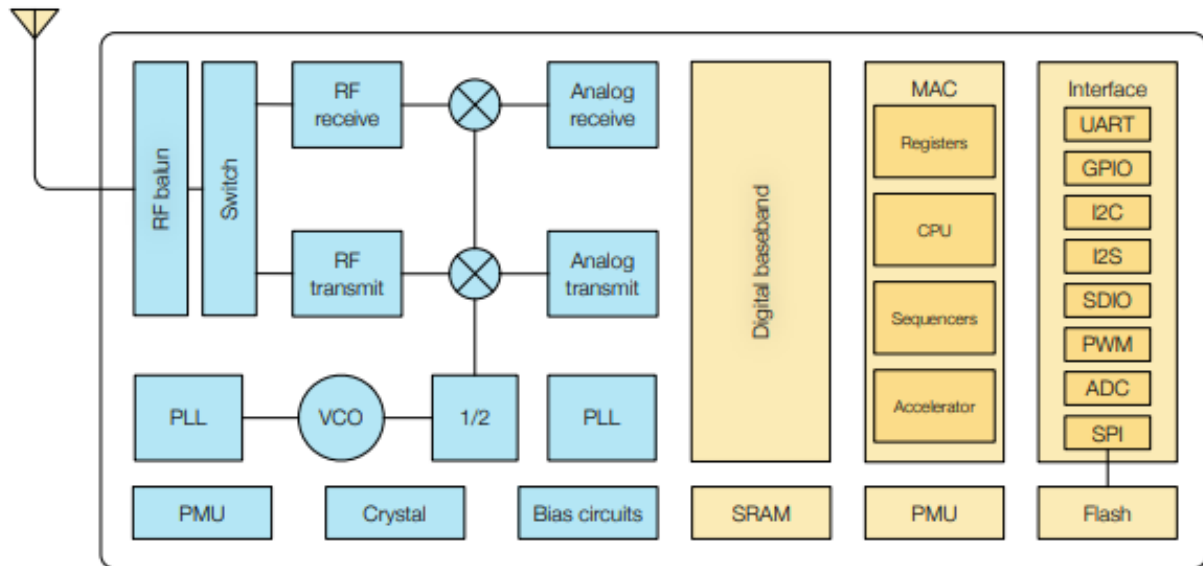


Fig4: NodeMCU ESP8266 Block Diagram:

5.7 ESP8266 Cryptographic Library:

The library is split into two main sections: Core and light-weight.

- **The Core Algorithms:**

1. **Authenticated encryption with associated data (AEAD): ChaChaPoly**

ChaCha20-Poly1305 is an authenticated encryption with additional data (AEAD) algorithm, that combines the ChaCha20 stream cipher with the Poly1305 message authentication code. Its usage in IETF protocols is standardized in RFC 8439. It has fast software performance, and without hardware acceleration, is usually faster than AES-GCM. The AEAD primitive is the most common primitive for data encryption, and is suitable for most encryption needs.

AEAD has the following properties:

- **Secrecy:** Nobody will be able to get any information about the encrypted plaintext, except the length.

- **Authenticity:** Without the key it is impossible to change the plaintext underlying the ciphertext undetected.
- **Symmetric:** Encrypting the message and decrypting the ciphertext is done with the same key.
- **Randomization:** The encryption is randomized. Two messages with the same plaintext will not yield the same ciphertext. This prevents attackers from knowing which ciphertext corresponds to a given plaintext.

2. Block ciphers: AES128, AES192, AES256

The Advanced Encryption Standard (AES) is an IoT encryption algorithm that's used by the US government's National Security Agency (NSA) – as well as many other large organizations.

The AES Encryption algorithm (also known as the Rijndael algorithm) is a symmetric block cipher algorithm with a block/chunk size of 128 bits. It converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the ciphertext.

It is based on a substitution-permutation network, also known as an SP network. It consists of a series of linked operations, including replacing inputs with specific outputs (substitutions) and others involving bit shuffling (permutations).

AES is extremely efficient when used in 128-bit form. However, it also uses keys of 192 and 256 bits for heavy-duty encryption.

Within the security world, AES is highly thought of and considered to be resistant to cyber-attacks. It's the world's most widely used encryption algorithm – and can be found in these IoT applications:

- Wi-Fi security
- Mobile app encryption
- Wireless security

- VPN
- Processor security and file encryption

What are the Features of AES?

1. SP Network: It works on an SP network structure rather than a Feistel cipher structure, as seen in the case of the DES algorithm.
2. Key Expansion: It takes a single key up during the first stage, which is later expanded to multiple keys used in individual rounds.
3. Byte Data: The AES encryption algorithm does operations on byte data instead of bit data. So it treats the 128-bit block size as 16 bytes during the encryption procedure.
4. Key Length: The number of rounds to be carried out depends on the length of the key being used to encrypt data. The 128-bit key size has ten rounds, the 192-bit key size has 12 rounds, and the 256-bit key size has 14 rounds.

3. Stream ciphers: Cha Cha

Developed by Daniel J. Bernstein, the ChaCha 20 stream cipher is a variant of the Salsa20 stream cipher. The design principles of this cipher are almost identical to that of Salsa20, however, there is increased diffusion per round. Reference [10] gives us an exhaustive explanation of the algorithm by D.J. Bernstein himself. It is a 256 bit stream cipher. The changes from Salsa20/8 to ChaCha8 are designed to improve diffusion per round, thus increasing resistance to cryptanalysis, while preserving and improving time per round. However, the extra diffusion does not add more operations when compared to Salsa20. A ChaCha round has 16 additions and 16 xors and 16 constant-distance rotations of 32-bit words. The parallelism and vectorizability of the ChaCha 20 algorithm are conformant with that of Salsa20.

Chacha20 is a cipher stream. Its input includes a 256-bit key, a 32-bit counter, a 96-bit nonce and plain text. Its initial state is a 4*4 matrix of 32-bit words. The first row is a constant string

“expand 32-byte k” which is cut into 4×32 -bit words. The second and the third are filled with 256-bit key. The first word in the last row are 32-bit counter and the others are 96-bit nonce.

It generate 512-bit keystream in each iteration to encrypt a 512-bit bolck of plain text. When the rest of plain text is less 512 bits after many times encryption, please padding to the left with 0s(MSB) in the last input data and remove the same bits unuseful data from the last output data. Its encryption and decryption are same as long as input same initial key, counter and nonce.

- **The Light-weight Algorithms:**

1. **Block ciphers: Speck, Speck Tiny, Speck small.**

Speck was proposed publicly in June 2013 by a group of researchers in the US National Security Agency's Research Directorate. Speck should be flexible enough to perform well on the full spectrum of constrained platforms, and this motivated us to choose the simplest components possible. But the simplicity of the designs had an added benefit: we ended up with algorithms that have exceptional performance on high-end platforms as well. As far as, Speck has the highest throughput on 64-bit processing of any block cipher implemented in software.

The desire for flexibility through simplicity motivated us to limit the operations used within Simon and Speck to the following short list:

- modular addition and subtraction, $+$ and $-$
- bitwise XOR, \oplus
- bitwise AND, $\&$
- left circular shift, S_j , by j bits
- right circular shift, S_{-j} , by j bits.

Speck includes a round counter in the key schedule. The designers state this was included to block slide and rotational cryptanalysis attacks. Still, rotational-XOR cryptanalysis has been used to find distinguishers against reduced-round versions of Speck. Though the authors don't describe standard key-recovery attacks based on their distinguishers, their best distinguishers on Speck32 and Speck48 in the known-key distinguishing attack model for certain weak key classes make it through slightly more rounds than the best differential distinguishers. One of the authors has said that his research was resource-constrained and that rotational-XOR distinguishers on more rounds are probably possible. However, this type of cryptanalysis

assumes the related-key or even the known-key attack models, which are not a concern in typical cryptographic protocols and solutions. The designers also state that Speck was not designed to resist known-key distinguishing attacks (which do not directly compromise the confidentiality of ciphers).

Block and key sizes:

Although the Speck family of ciphers includes variants with the same block and key sizes as AES (Speck128/128, Speck128/192, and Speck128/256), it also includes variants with block size as low as 32 bits and key size as low as 64 bits. These small block and key sizes are insecure for general use, as they can allow birthday attacks and brute-force attacks, regardless of the formal security of the cipher. The designers state that these block and key sizes were included for highly resource-constrained devices where nothing better is possible, or where only very small amounts of data are ever encrypted, e.g. in RFID protocols. Only the variant with a 128-bit block size and 256-bit key size is approved for use in U.S. National Security Systems.

2. Authenticated encryption with associated data (AEAD): Acorn128, Ascon128

Authenticated Encryption (AE) and Authenticated Encryption with Associated Data (AEAD) are forms of encryption which simultaneously assure the confidentiality and authenticity of data.

Authenticated encryption (AE) schemes provide confidentiality, integrity and authenticity at once. The classical way to achieve such a construction is to combine several cryptographic primitives, typically an encryption algorithm for confidentiality and a message authentication code (MAC) for integrity and authenticity.

A typical programming interface for an AE implementation provides the following functions:

- Encryption
 - Input: plaintext, key, and optionally a header in plaintext that will not be encrypted, but will be covered by authenticity protection.
 - Output: ciphertext and authentication tag (message authentication code or MAC).

- Decryption
 - Input: ciphertext, key, authentication tag, and optionally a header (if used during the encryption).
 - Output: plaintext, or an error if the authentication tag does not match the supplied ciphertext or header.

The header part is intended to provide authenticity and integrity protection for networking or storage metadata for which confidentiality is unnecessary, but authenticity is desired.

ACORN:

ACORN is a stream cipher based authenticated encryption with associated data (AEAD) algorithm. AEAD schemes allow to also include information that does not need to be encrypted but of which the integrity and authenticity needs to still be guaranteed. ACORN uses a 128-bit key, a 128-bit initialization vector (IV) and produces a 128-bit authentication tag.

Initialization. The initialization phase takes as input the encryption key and the IV. First, the entire state is initialized to zero.

Additional Data Processing. After the initialization step, the associated data is used to update the state. The state is updated for at least 256 steps, even if there is no associated data to process.

Encryption. At each step of the encryption, one bit from the plaintext is encrypted. The state is updated for at least 256 steps, even if there is no plaintext to encrypt.

Finalization. At the end, an n-bit authentication tag is computed. The state is updated 768 times and the tag consists of the last n keystream bits generated.

ASCON:

Ascon is an AEAD algorithm based on duplex sponge modes [4]. It uses a 128-bit key, a 128-bit IV and produces a 128-bit authentication tag. The internal state is 320-bit long and is represented by five 64-bit registers, noted x0 to x4 .

Initialization. The initial state is built from the encryption key and the initial vector before applying a = 12 rounds of the transformation p. Finally, the secret key is XORed to the 128 last bits of the state.

Additional Data Process. Each additional data block is absorbed into the state before applying b rounds of the transformation p. Then, even if there is no additional data to process, the last bit of the state is swapped to set up a domain separation in order to prevent attacks that change the role of plaintext and associated data blocks.

Encryption Each plaintext block is duplexed into the state, producing a ciphertext block. Except for the last block, the state is then updated by b rounds of the transformation p .

Finalization The encryption key is XORed to the internal state before applying $a = 12$ rounds of the transformation p . Finally, the n -bit tag consists of the last n bits of the state XORed with the key.

5.8 The Random number/bits Generation:

One solution to ensure secrecy in the Internet of Things (IoT) is cryptography. However, classical cryptographic systems require high computational complexity that is not appropriate for IoT devices with restricted computing resources, energy, and memory.

A random number generation is very important in computing devices which helps them to do task in random manner. The applications of random number generation can be found in shuffling the audio files in an audio player, in almost all kind of digital games, generating passwords etc.

There are so many algorithms which can generate the random numbers. The random number generating algorithms differ from other algorithms in an interesting way; they actually reads a random number from the hardware like the noise from the unconnected pins and then apply some calculations on it to generate a number which is inside a specified range.

Analog to Digital Converter or simply ADC is an amazing feature of most modern Microcontrollers and SoCs. An Analog to Digital Converter, as the name suggests, converts continuous Analog signals in to discrete Digital Values.

This is really important as all Microcontrollers are digital devices and work only on logic LOW and logic HIGH values. ADCs allow Microcontrollers and SoCs to interface with Analog devices (like Sensors) and build systems around them. Most modern microcontrollers already have an ADC Block built into their silicon so, you don't need to use an external (and dedicated) ADC IC.

The ESP8266EX SoC, which is the main processor in all the ESP8266 Boards, has a single channel ADC. The ADC in ESP8266 has a 10-bit resolution and is of Successive

Benchmarking cryptographic Algorithms on resource constrained ESP8266

Approximation Register (SAR) type. The 10-bit resolution means, the output values will be in the range of 0 to 1023.

If you refer to the datasheet of ES8266EX SoC, the ADC Pin is the Pin 6, also known as TOUT pin. The ADC can be used for basically two types of measurements. They are:

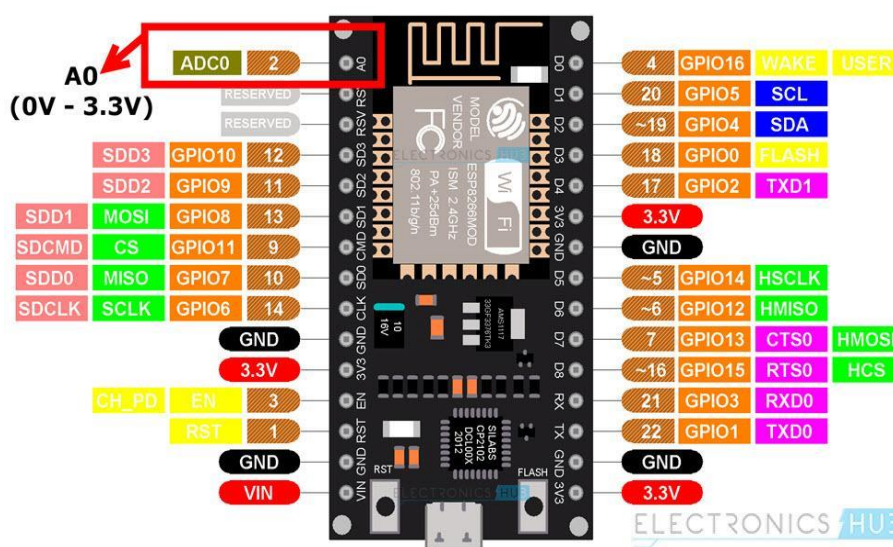
- Measure the Power Supply Voltage at Pin 3 and Pin 4.
- Measure the input voltage of Pin 6 i.e., TOUT Pin.

Both these measurements cannot be implemented at the same time. For Power Supply Voltage measurement, the TOUT Pin must be left floating. But when measuring external voltage at TOUT Pin, the input voltage range should be between 0V and 1V.

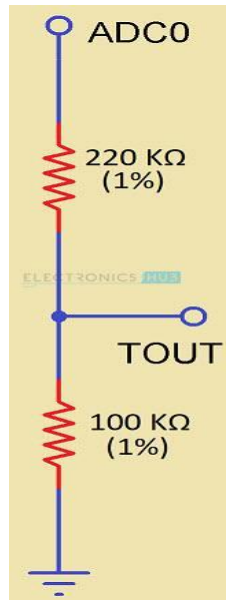
If you are building the firmware from the SDK, then refer to the datasheet and other relevant documentation, as there is a lot more information regarding the ADC and its configuration.

ADC in ESP8266 NodeMCU

The NodeMCU board, which is essentially based on the ESP-12E Wi-Fi Module, has a pin for ADC called A0. Additionally, NodeMCU also takes care of the input voltage range limit from the original 0V to 1V to more user friendly 0V to 3.3V.



If you take a look at the schematic of NodeMCU, the ADC0 pin in the above pinout image acts as an input to the voltage divider formed by 220 K Ω and 100 K Ω resistors with the output of the voltage divider given to the TOUT Pin of ESP8266EX SoC.



6. Coding/Implementation details (modules used to achieve the functionality)

We are preparing the benchmark for multiple algorithms, namely AES, Speck, ChaCha etc.

6.1. AES:

AES is one of the most widely used cryptographic algorithms in the world. Working of AES is quite simple. AES is made up of 4X4 matrices with each block of the matrix holding information of 1 byte, so totally making up 128 bits. `aes.set_key` function is called which sets the key that we have given in the function parameter. Then, when the encryption function is called with the output being the first parameter and input being the second, the key will be XORed after converting each input into hexadecimal form, flipping up the rows and columns and this will go on for multiple with finally the encrypted value being generated.

6.2 Speck:

Speck is a lightweight cipher which is also similar to AES in the way that both are block ciphers.

Speck supports a variety of blocks and each block contains two words. First in the program, the set Key function is called and the key is set. Later encryption is called and this is done by adding the right word of the block with the left word by rotation and then XORing the key to the left word and then XORing the right word to the left word. The decryption will be the opposite steps of that.

The output for Speck encryption and decryption along with the time taken for each process:

6.3 ChaCha:

ChaCha is a stream cipher. ChaCha's main core is the pseudo random number generator. The cipher text can be obtained by XORing the plaintext available with a pseudo random stream. The stream is generated by scrambling the blocks and by using quarter round. This is done with the help of left shift operators and so on and this scrambled value is XORed with the key and IV and the resultant encrypted value will be obtained. The decryption process is just the same as encryption. Meaning if the encrypted text is again encrypted with ChaCha, we get the original value back

The encrypt function of ChaCha takes the ciphertext and plaintext and the length of the ciphertext and the decrypt function takes the decrypted text and ciphertext and the length of the decrypted text as parameters.

6.4 ChaChaPoly:

In the encrypt operation, the message to be encrypted, Initial Vector (IV), Key and Authentication data are taken as inputs. First, using the Key and IV the message is encrypted using ChaCha. After the data is encrypted, the operation uses the key and IV to generate a secondary key which will then be used to generate a keyed hash of the Authentication data. The hash used in ChaChaPoly is Poly1305.

6.5 Random number generated used as Key:

Instead of using a hardcoded key, we are using the random value generated by the pin of the Arduino circuit, specifically the A0 pin. This takes analog values and converts it into digital which we shall then use. This value taken will then be subject to an AND operation with 1 and the result hence obtained will then be randomised further with the help of Von Neuman whitening method. Von Neuman whitening method is something like

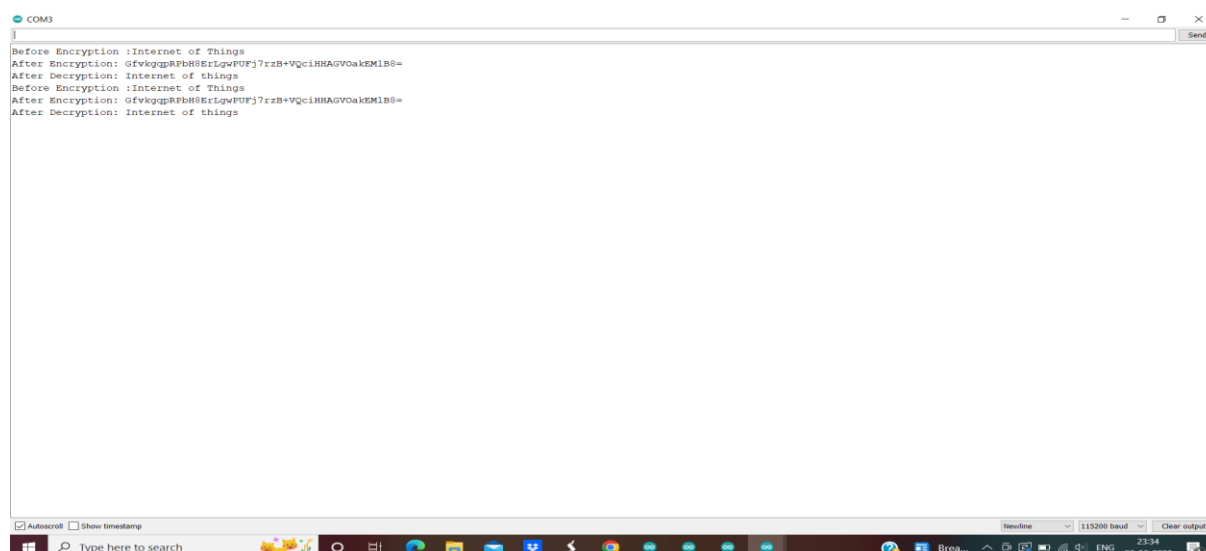
Benchmarking cryptographic Algorithms on resource constrained ESP8266

a = random() || random() << 1 // random() being the function which returns the ANDed value of the input generated from the A0 pin.

```
for(;;){
    if a==1 return 0; // 1 to 0 transition log a zero bit
    else if a==2; return 1 0 to 1 transition log a one bit
return 0;
```

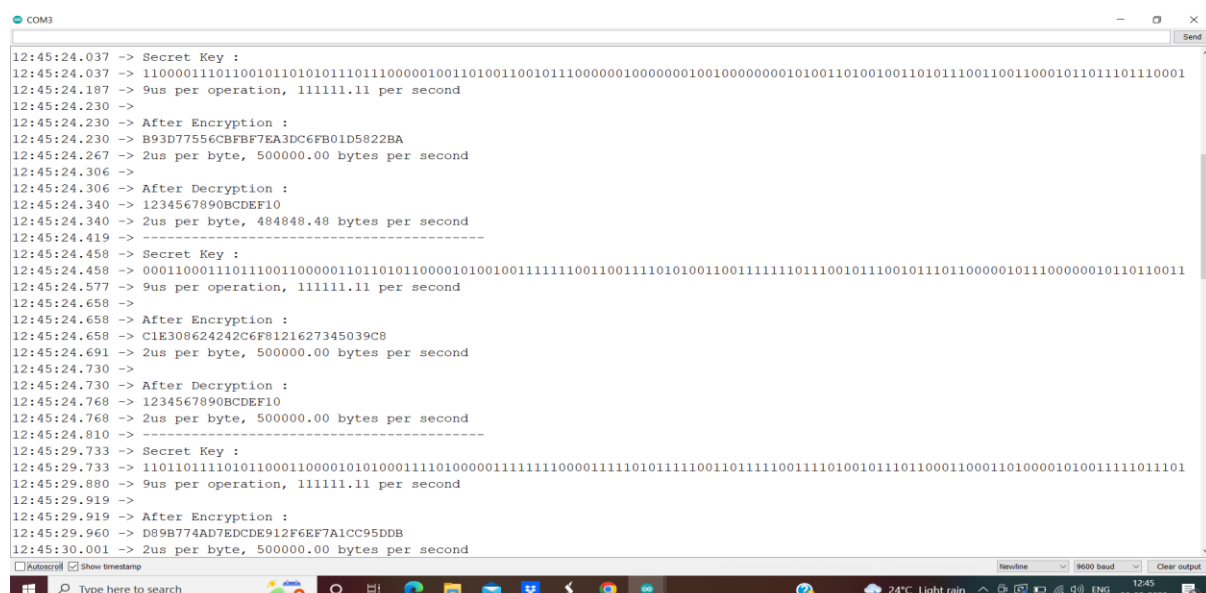
7. Project results/Learning outcomes

AES:



```
COM3
Before Encryption :Internet of Things
After Encryption: GfvkgqRRFbH8ErlGwPufj7rzb+VqcIHRAGVOakEMlB8=
Before Decryption: Internet of things
After Encryption :Internet of Things
After Decryption: GfvkgqRRFbH8ErlGwPufj7rzb+VqcIHRAGVOakEMlB8=
Before Decryption: Internet of things
```

ChaCha:



```
COM3
12:45:24.037 -> Secret Key :
12:45:24.037 -> 1100001110110010110101110111000001001101001100101110000001001000000010011010010011010111001100110001
12:45:24.187 -> 9us per operation, 111111.11 per second
12:45:24.230 ->
12:45:24.230 -> After Encryption :
12:45:24.230 -> B93D7756CBFB7EA3DC6FB01D5822BA
12:45:24.267 -> 2us per byte, 500000.00 bytes per second
12:45:24.306 ->
12:45:24.306 -> After Decryption :
12:45:24.340 -> 1234567890BCDEF10
12:45:24.340 -> 2us per byte, 484848.48 bytes per second
12:45:24.419 -> -----
12:45:24.458 -> Secret Key :
12:45:24.458 -> 000110001110111001100000110110101100000100100111111100110011111101110010111011000001011100000010110110011
12:45:24.577 -> 9us per operation, 111111.11 per second
12:45:24.658 ->
12:45:24.658 -> After Encryption :
12:45:24.658 -> C1E308624242C6F8121627345039C8
12:45:24.691 -> 2us per byte, 500000.00 bytes per second
12:45:24.730 ->
12:45:24.730 -> After Decryption :
12:45:24.768 -> 1234567890BCDEF10
12:45:24.768 -> 2us per byte, 500000.00 bytes per second
12:45:24.810 -> -----
12:45:29.733 -> Secret Key :
12:45:29.733 -> 1101101111011000110000010101000111101000011111110000111110101111100110111100110111001101100011000010100011111011101
12:45:29.880 -> 9us per operation, 111111.11 per second
12:45:29.919 ->
12:45:29.919 -> After Encryption :
12:45:29.960 -> D89B774AD7EDCDE912F6EF7A1CC95DDB
12:45:30.001 -> 2us per byte, 500000.00 bytes per second
```

Benchmarking cryptographic Algorithms on resource constrained ESP8266

Speck Along with the random key generated:

```
COM3
Secret Key:
111001010111001100000101100000000100111011000101010101010101110010100001101000111010010011100000111000100010011101101010001
Set key
26us per operation, 38461.54 bytes per second

After Encrypting
721968384021D92FCCBC2F510F4B55E
1.25us per byte, 800000.00 bytes per second

After Decrypting
6C617669757165207469206564616D20
1.25us per byte, 800000.00 bytes per second
```

ChaChaPoly:

```
COM3
After Encrypting
4C616469657320616E642047656E746C
After Decrypting
78C961FE9BD53281ADB4C3DE240F
After Encrypting
6C616469657320616E642047656E746C
After Decrypting
58C961FE9BD53281ADB4C3DE240F
```


Benchmarking cryptographic Algorithms on resource constrained ESP8266



Benchmarking of non-Lightweight algorithms (AES, ChaCha):

Encryption Algorithms	Encryption (per byte)	Decryption (per byte)	Key Setup	State Size (bytes)
CHACHA POLY	2.10us	1.94us	5.46us	240
AES (128)	4.77us	4.16us	70us	296
AES (192)	4.77us	5.12us	68us	296
AES (256)	4.16us	6.33us	89us	296
CHACHA (128)	0.46us	0.46us	9.36us	136
CHACHA (256)	0.67us	0.67us	4.46us	136

Benchmarking of Lightweight algorithms (Speck, ASCORN):

Encryption Algorithms	Encryption (per byte)	Decryption (per byte)	Key Setup	State Size (bytes)
SPECK (128)	1.21us	1.24us	24.88us	288
SPECK (192)	1.24us	1.27us	26.30us	288
SPECK (256)	1.27us	1.30us	27.74us	288
SPECK_SMALL (128)	2.13us	2.10us	19.26us	80
SPECK_SMALL (192)	2.19us	2.17us	20.56us	80
SPECK_SMALL(256)	2.26us	2.24us	21.89us	80
ACORN(128)	0.52us	0.51us	118.22us	64
ASCON (128)	2.27us	2.25us	34.95us	72

We can observe from the above tables that ChaCha is most efficient in terms of time saving for encrypting and decrypting and as we know, since the encryption and decryption process for ChaCha are exactly the same, it takes the same amount of time.

AES takes a lot of time compared to the others. Speck is also pretty fast as it is a lightweight algorithm.

8. Conclusion

Through this internship, I have managed to have a deeper understanding of the different cryptographic algorithms and the efficiency in terms of time taken between them. We can see from the table that ChaCha takes the least time out of all the other algorithms considered and AES takes the most time. Also, a set of 128 bit random numbers were generated using the A0 pin of the ESP8266 board and this was used as key for the encryption and decryption which made our code more secure as an attacker will not be able to guess the key as it keeps on changing randomly after every cycle.

9. References/Bibliography

- [1]Hoomod, Haider & Rokan, Jolan & Ahmed, Israa. (2020). A new intelligent hybrid encryption algorithm for IoT data based on modified PRESENT-Speck and novel 5D chaotic system. Periodicals of Engineering and Natural Sciences (PEN). 8. 2333.
- [2]Park, BoSun & Song, JinGyo & Seo, Seog. (2020). Efficient Implementation of a Crypto Library Using Web Assembly. Electronics. 9. 1839. 10.3390/electronics9111839.
- [3]Yuliana, Mike & Wirawan, Iwan & Suwadi, Suwadi. (2019). An Efficient Key Generation for the Internet of Things Based Synchronized Quantization. Sensors. 19. 2674. 10.3390/s19122674.
- [4]Anurag Rawal, Gaurav Chhikara, Gaganjot Kaur, & Hitesh Khanna. (2019). Cryptography Algorithm. Journal of Analog and Digital Communications, 4(1), 31–38. <https://doi.org/10.5281/zenodo.2606230>
- [5]Abdullah, Ako. (2017). Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data.
- [6]Gupta, Anjula & Walia, Navpreet. (2014). Cryptography Algorithms: A Review. International Journal of Engineering Development and Research 2321-9939. 2. 1667.
- [7]El-hajj, Mohammed & Maroun, Chamoun & Fadlallah, Ahmad & Serhrouchni, Ahmed. (2018). Analysis of Cryptographic Algorithms on IoT Hardware platforms. 1-5. 10.1109/CSNET.2018.8602942.
- [8]Kölbl, S. (2017). *Design and analysis of cryptographic algorithms*. Technical University of Denmark. DTU Compute PHD-2016 No. 434

- [9]L. E. Kane, J. J. Chen, R. Thomas, V. Liu and M. Mckague, "Security and Performance in IoT: A Balancing Act," in IEEE Access, vol. 8, pp. 121969-121986, 2020, doi: 10.1109/ACCESS.2020.3007536.

- [10]Advanced Encryption Standard (AES) (FIPS 197), NIST

- [11]Secure Hash Standard (SHS) (FIPS 180-4), NIST 5. Standards for Efficient Cryptography SEC 1: Elliptic Curve Cryptography, Certicom Research

- [12]Recommendation for Block Cipher Modes of Operation: Methods and Techniques (NIST SP 800-38A), NIST

- [13]Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality (NIST SP 800-38C), NIST

- [14]Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC (NIST SP 800-38D), NIST

- [15]Recommendation for Random Number Generation Using Deterministic Random Bit Generators (NIST SP 800-90A), NIST

- [16]Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography (NIST SP 800-56A), NIST