International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 01 | Jan 2021 www.irjet.net p-ISSN: 2395-0072

Cryptographic Communication between Two ESP32 Devices

Jenish Rudani[1], Daksh Khorana[2], Yagnesh Savaliya[3]

[1-3]*Student, Dharmsinh Desai University (DDU), Nadiad, India*

Abstract - *Embedded Systems are widely used in virtually every industry all over the world. The low footprint allows such systems to be tailor made for the set of specific tasks and are relatively cheap to manufacture. Contemporary research in security at application level has introduced multiple amelioration over traditional techniques. However, embedded system is a laggard in this regard. In this paper, we introduce communication technique between two ESP32 IoT devices, which is secured by RSA-256 cryptography algorithm.*

Key Words: Embedded Systems, ESP32, IoT, RSA256, Cryptography

1. INTRODUCTION

2020 decade is going to bring vast majority of IoT devices from handheld kitchen appliances, wearables to heavy industrial tools, and military duty devices able to withstand extreme Earth Conditions. With ever increasing amount of IoT device, the major concern is of privacy. Although many areas are now further secured by better, secure and reliable software systems, Embedded Devices are lagging behind due to limited set of specifications, which doesn't always allow such complex cryptosystems to be implemented on memory constraint embedded devices. This paper explores one of the cryptosystem RSA which is considered highly secure and reliable system on an inexpensive and famous IoT device ESP-32.

1.1 What is RSA?

RSA stands for "Rivest–Shamir–Adleman", it is a cryptographic algorithm used by most modern computers to encrypt and decrypt messages or information. RSA follows an asymmetric key pattern that allows users to have two different keys, a public key and a private key [1]. The public key can be given to anyone to encrypt the message, whereas the private key can be kept with the intended decryptor of the message.

Once the message has been encrypted with the public key, it can only be decrypted with a private key, due to the math behind the algorithm.

1.2 Where is RSA Algorithm used?

The RSA algorithm is not used to encrypt the messages directly as it is a heavy algorithm and requires more computational power than symmetric key encryption. So, RSA is used with some other encrypting algorithm to make use of its complexity. The file or message is generally encrypted with a symmetric key and the symmetric key is then encrypted with RSA.

RSA was one of the first keys to be widely used as public encryption keys and so it laid the foundation in secure communication we use today. It was first used in TLS and also the first algorithm to be used for PGP encryption.

These days RSA is majorly used for digital signature authentication, emails and VNC software. RSA plays an important role in VPN as there are multiple handshakes while initiating a secure connection in a VPN, and these handshakes are secured by RSA. RSA is quite often implemented in wolfCrypt, cryptlib, and a number of other crypto libraries.

2. The Algorithm

The algorithm works in a certain way of primes and modulus.

1. Two prime numbers p & q are to be selected.
2. We find the product n= p x q.
3. A totient is calculated as : $\phi(n) = (p - 1)(q - 1)$
    a. An integer is selected such that it lies between $1 < \epsilon < \phi(n)$ such that ϵ is co-prime.
    b. The ϵ is a public key exponent.
4. d is computed to satisfy the congruence relation. It is the private key exponent
    a. d is calculated as : $d = (1 + x(\phi(n))/\epsilon$: for some integer x .
    b. d should satisfy the condition $\gcd(\phi, \epsilon) = \phi(x) + \epsilon y = 1$ [2]
5. For encrypting the message m
    a. $c = m^{\epsilon} \bmod n$
    b. Here c is the encrypted message and m is the original message.
6. For decrypting the message c
    a. $m^{ed} = m \bmod pq$ (Applying Chinese remainder theorem we yield these two congruences.)
    b. $c^d = m \bmod n$

c. $m = c^d \bmod n$ (Here m is obtained as a decrypted message.)

Example:

Note for the ease of understanding we would be taking smaller prime numbers for p and q.

- p=3 , q=11
- Following above steps, we get:
  - n = 33
  - $\phi(n)= 20$
  - $\epsilon = 19$
  - d = 19
- The message m here is 3:
  - The cipher text would be $c = m^{\epsilon} \bmod n$
  - $c = 319 \bmod 33$
  - **c = 15 (Encrypted Message)**
- To decrypt the message, we use the following:
  - $c^d = m \bmod n$
  - $m = c^d \bmod n$
  - $15^{19} \bmod 33 = 3$
  - **m = 3 (Decrypted Message)**

We looked at a simple RSA algorithm to encrypt a number message "3" and the encrypted message we derived is 15, and as we further proceed with decryption, we finally calculated the original message to be 3. This example demonstrates a basic RSA Encryption/Decryption but in real world the keys are much longer, the prime values are much longer. So there are different RSA algorithms available, and depending upon system specification such as processing speed, RAM and ROM, we would choose one out of RSA-256, RSA-512, RSA-1024, RSA-2048 and so on. We have chosen RSA-256 algorithm for this paper[3].

2. What is IoT?

IoT or Internet of Things is a system of interrelated devices or gadgets that communicate with each other over wireless internet without intervention of a human. The things or gadgets here can be defined as devices that are capable of recording and transmitting data via embedded devices such as Kitchen appliances, cars, thermostats, Air Conditioners, Lights, Televisions, and many more.

The range of devices are not just limited to consumer products; these are more proficiently implemented in industries to automate the tasks. The emergence of IoT devices has been more prolific in this era of time because of the following things:

- Access to low cost and low power consuming sensors.
- Read access to internet
- A central hub to process the commands.

IoT is used to improve the life of humans and ease the redundant tasks as it helps the users have a all in one control, that would enable them to do the physical tasks such as setting up temperature, or have the lights turn on or off by detecting the presence of humans. The smart devices are connected to one central hub which processes and handles the command distribution to the node devices (IoT), and further using Machine Learning and Artificial Intelligence the use case scenario of these devices cannot be bound by any metrics.

2.1 Security Issue Faced in IoT

The devices connected in the IoT ecosystem are not equipped with efficient security measures making them vulnerable to security attacks such as data leak, override controls, and the authenticity of the message.

The IoT devices are not yet implemented to transmit data securely as the computational power and the bandwidth required by the devices to transmit data securely are not yet implemented successfully.

The preventive side to be taken where confidentiality of the data is at utmost priority is in the field of Home Automation, here we will discuss some of the security issues faced by home automation products.

- Privacy attacks
  - The smart home devices are the most susceptible platform for attackers as private information is shared in these networks, so an encrypted channel is a must for the devices being used in a home as there are a lot of private messages floating over the network.
- DDoS Attacks
  - The attacker can take down the network and infiltrate it by sending PoD(Ping of Death) messages to gain access to the network or even disable certain security devices.

- Masquerade
    - The attacker can masquerade as an authorized device as there is no system check maintained to check the authenticity of the device.

2.2. Why is ESP32 Preferred for consumer IoT?

The ESP32 is a low power, low cost SoC(System on a Chip) device which contains a built-in Wifi and Bluetooth module. The SoC comes in two variants, one is single core and one in dual core model to handle the processing of the incoming and outgoing data.

Here is a brief data sheet of ESP32:

ESP-32	DESCRIPTION
CORE	2 (Simultaneous Execution)
ARCHITECTURE	32 Bits
CLOCK	Tensilica Xtensa LX106 (160-240 Mhz)
BLUETOOTH	Yes - Classic & BLE
WIFI	IEEE802.11 B/G/N
RAM	520 to 8192 Kb
FLASH	Extern QSPI - 16MB
GPIO	22 Pins
ADC	18 Bit
INTERFACES	SPI-I2C-UART-I2S-CAN

Table -1: Specifications of ESP32

NETWORK - The embedded Networking module in the ESP32 provides for a more closed and monitored networking with the device, resulting in more secure and controlled connections over a network, which instead with an external networking adapter would lead to more possible point of intrusions.

CPU - The included CPU in the device makes it easier to process complex algorithms which we would be implementing to encrypt and decrypt the messages shared by using the ESP32.

P2P - The device can communicate directly with other ESP devices using the IoT P2P connectivity.

WEB - The module can directly access and read the pages written in XML or other developmental languages to verify the authenticity or configuration of the connected device.

These peculiar features of the ESP32 make it a susceptible chip to be used for fast prototyping and suitable for consumer IoT. Also, the main reason behind using this particular device to demonstrate Communication Demonstrate by RSA-256 cryptography.

3. Flow Chart

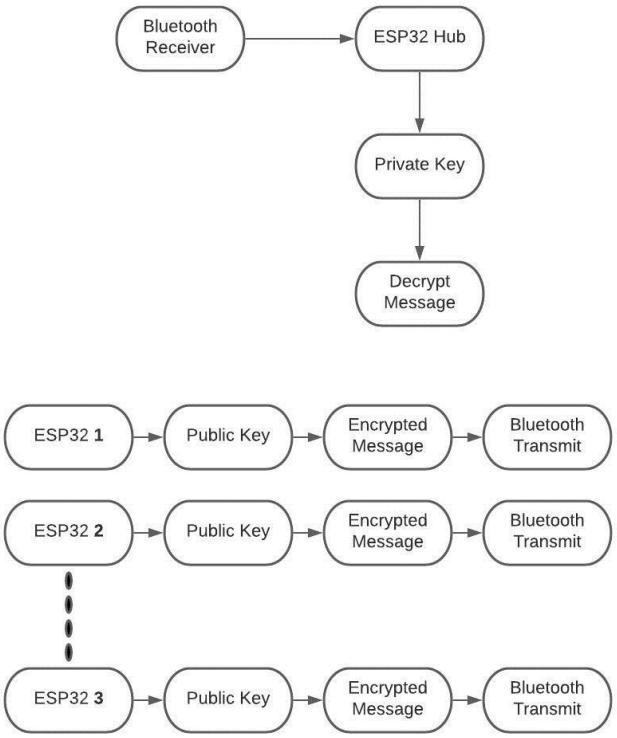

Fig-1: Flowchart of the system

Above diagram describes the working model of this system[4]. As per diagram, one ESP32 acts as a hub for the experiment and rest of ESP32s from 1 to n acts as nodes which actually communicates with the hub through Bluetooth interface. This setup is ideal for an example home automation application. Here, hub ESP32 first encrypts a message according to application and transmits the encrypted hash through Bluetooth peripheral and subsequently at the node, ESP32 would decrypt the message using embedded public key. Here, the message encrypted by hub ESP32 can only be decrypted by the public key of node ESP32. This ensures the security of the entire system as to even if the man in the middle attack happens, the attacker would not have any means of decrypting the message, except the brute force methods, which could take a long time.

However, there is one fault in this system, and that is one could reverse engineer the ESP32 itself, and by looking at the internal flash memory of ESP32, after careful analysis, it is quite possible to retrieve the public and private key and the security would be breached. To solve this problem, we used flash encryption [5] and that ensures that it would be extremely difficult to retrieve the public and private keys, which further increases the security.

4. CONCLUSION

This paper presents a way to ameliorate security at low overhead in Embedded Devices, specifically a famous IoT device ESP32 using RSA-256 algorithm. It describes a system which communicates under a secure, reliable and fast cryptographic algorithm-RSA256. We analyzed the working of RSA algorithm and understood it by an Example.

REFERENCES

[1] Wikipedia, (December, 2020). RSA (cryptosystem), Available: https://en.wikipedia.org/wiki/RSA_(cryptosystem)

[2] Machie, Edmond K. (29 March 2013). Network security traceback attack and react in the United States Department of Defense network. p. 167. ISBN 978-1466985742.

[3] Heninger, Nadia (February 15, 2012). “New research: There's no need to panic over factorable keys–just mind your Ps and Qs”. Freedom to Tinker, Available PDF : https://freedom-to-tinker.com/blog/nadiah/new-research-theres-no-need-panic-over-factorable-keys-just-mind-your-ps-and-qs

[4] RSA Algorithm Implementation C++ (November, 2020) Available Source Code:

https://github.com/espressif/arduino-esp32/blob/master/tools/sdk/include/mbedtls/mbedtls/rsa.h

[5] Espressif Documentation (November, 2020). “Flash Encryption Security in ESP32”, Available Source Code: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/flash-encryption.html