The previous diagram represents an overview of our FOTA system which includes:

**1- Cloud (Firebase)**

Responsible for the management of vehicle software release, and optionally to customize updates for every vehicle client based on OEM policies. It’s where the

users can get new updates for their systems or send diagnostics to get problems

solved.

**2-Telematics Unit(ESP32)**

It’s the bridge connecting the server and the whole system allowing you to send and receive multiple data through it. It can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI / SDIO or I2C / UART interfaces.

As an overview of our implemented system, we used:

• Google Firebase for the OEM server as it provides real-time database and free large storage.

• ESP32 as the telematics unit, as it has a Wi-Fi module which will make it easy to connect to the server and will be connected to gateway through a UART bus.

**Telematics Unit**

As we said before, the Telematics unit is the bridge connecting the server and the whole system allowing us to send and receive multiple data through it. We use the Telematics unit to communicate mainly with two other units, the server, and the gateway (STM32). We used ESP32 module as it has a WI-FI module which is easy to establish using a few lines of code through Arduino IDE which gives us access to multiple libraries and functions in the ESP32.

**The main function of the ESP32 is:**

1-Communicate to the server:

• Download the updated file.

• Decrypt downloaded update files.

• Send a request to download the update file.

• Send diagnostics to the server.

2- Communicate to the Gateway:

• Send the update file to the Gateway.

• Receive download request from the user.

**ESP32 Specs:**

• Single or Dual-Core 32-bit Microprocessor with clock frequency up to 240 MHz

• 520 KB of SRAM, 448 KB of ROM and 16 KB of RTC SRAM.

• Supports 802.11 b/g/n Wi-Fi connectivity with speeds up to 150 Mbps.

• Support for both Classic Bluetooth v4.2 and BLE specifications.

• 34 Programmable GPIOs.

• Up to 18 channels of 12-bit SAR ADC and 2 channels of 8-bit DAC

• Serial Connectivity includes 4 x SPI, 2 x I2C, 2 x I2S, 3 x UART.

• Ethernet MAC for physical LAN Communication (requires external PHY).

• 1 Host controller for SD/SDIO/MMC and 1 Slave controller for SDIO/SPI.

• Motor PWM and up to 16-channels of LED PWM.

• Secure Boot and Flash Encryption.

• Cryptographic Hardware Acceleration for AES, Hash (SHA-2), RSA, ECC and

RNG.

**3.2.4. Server**

For the server implementation, we used Google Firebase which is a Google backed application development software that enables developers to develop iOS, Android, and Web apps. Firebase provides tools for tracking analytics, reporting, and fixing app crashes, creating marketing and product experiment. It’s free and easy to use, and it has a real-time database.

**Google Firebase advantages:**

• **Real-time Database**

A real-time Database is a cloud-hosted database. Data is stored as JSON and synchronized continuously to each associated client.

• **Hosting**

Hosting is production-grade web content that facilities the developers. With Hosting, you can rapidly and effectively send web applications and static content to a Content Delivery Network (CDN) with a single command.

• **Authentication**

Firebase Authentication gives back-end development services, simple-to-use SDKs, and instant UI libraries to confirm clients over your application. It supports authentication using passwords, email ID, or username.

• **Storage**

It is built for application developers who need to store and serve user-generated content, for example, photos or videos. It gives secure document transfers and downloads for Firebase applications, regardless of network quality.

**Implementation**

**Firebase**

Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google’s infrastructure as well as easy to use Firebase is categorized as a NoSQL database program, that stores data in JSON-like documents.

**(a)Authentication: -**

Firebase Authentication makes it easy for developers to build secure authentication systems and enhances the sign-in and onboarding experience for users. This feature offers a complete identity solution, supporting email and password accounts, phone authentications, as well as Google, Facebook, GitHub, Twitter login and more

.

**(b)Real-time database:**

The Firebase Real-time Database is a cloud-hosted NoSQL database that enables data to be stored and synced between users in realtime. The data is synced across all clients in real-time and is still available when an app goes offline.

**(c) Performance:**

Firebase Performance Monitoring service gives developers insight into the performance characteristics of their iOS and Android apps to help them determine where and when the performance of their apps can be improved.

**(d)Storage:**

Firebaseallows us to store files like the software update to access them later on and download it.

**ESP with Firebase:**

One of Firebase’s key features is Firebase Cloud Storage, which lets you store arbitrary binary blobs which can be retrieved over the Internet using a simple HTTP request.

A manual way to upload a new binary to Firebase is to visit the Firebase console, click on “Storage” in the left nav bar, and then click on the “Upload file” button, but we upload the file from our website. Our website is linked with firebase and makes it easy to upload the file without a lot steps, with use ESP we can download the file from Firebase.

We created a Firebase project with a real-time database, and we used the ESP to store and read data from the database. The ESP can interact with the database from anywhere in the world as long as it is connected to the internet.

**Security**

**4.2. Advanced Encryption Standard (AES)**

**4.2.1. Introduction**

At FOTA, data is transmitted over air, network, and always susceptible to theft and exploration of what is inside. Did you remember our application? We transmit code files, very sensitive data. It can be stolen or edited and retransmitted. In case of edit may cause crisis. Imagine the scenario of code editing and retransmission at industrial environment, same thing and may kill people. Therefore, we are going to create our own encryption code using AES Algorithm in C.

**4.2.2. Encryption Algorithms**

Is a security technique used to protect data and communications by converting plaintext information into ciphertext using mathematical algorithms. Nobody can read or reuse ciphertext until convert it into plaintext. The receiver can easily convert this into plaintext by doing decryption process when anyone other does attack for conversion and it isn't easy depending on the strength of algorithm. Is used to ensure data confidentiality, integrity, and privacy.

**4.2.3 Algorithm Characteristics:**

 **Symmetric-Key Encryption**:

is a symmetric-key encryption algorithm, which means it uses the same secret key for both encryption and decryption of data. This simplicity allows for faster processing and is suitable for applications that require high-speed encryption and decryption.

 **Block Cipher**:

is a block cipher, which means it processes data in fixed-size blocks. The block size for is 128 bits.

 **Key Length Options/Fixable key length supports three key lengths**:

128 bits, 192 bits, and 256 bits. The key length directly affects the strength of the encryption. This key length contributes against brute force attack.

 **Security**:

is designed to be highly secure and resistant to various cryptographic attacks,

including brute-force attacks, differential cryptanalysis, and linear cryptanalysis.

 **Substitution-Permutation Network**:

employs a Substitution-Permutation Network (SPN) structure, which involves multiple rounds of substitution and permutation operations on the data. These rounds contribute to the confusion and diffusion properties, increasing its resistance against attacks.

 **Performance**:

is designed to be efficient and fast on a wide range of devices, including both software and hardware implementations.

**4.2.4. Encryption process**

Here, we describe the encryption process and decryption will be in the reverse direction. Here we have a box called a Round and is repeated box based on the key length and because we have 128-bit original key length, the number of rounds is 10. Each round has its own key derived from the Original Key. It's easy to develop code to receive 256 original key length. In the last round we skip MixColumns () and do only AddRoundKey () then generate block of output. Back again to file and generate another block of plain. You must know that we take only a block of plain and decrypt it separated than file, then out it to new file, then back and take another block. The black draw tells you how encryption applied in our application, we encrypt at PC and decrypt at node, at Microcontroller.

**Key Expansion**

It takes the original encryption key and generates a set of round keys,

which are used in subsequent rounds of the AES encryption algorithm

**1. SubByte (Byte substitution): an S-box substitution step.**

 Defines a 16 \* 16 matrix of byte values, called an S-box that contains a permutation of all possible 256 input values. (Hint) As we know from ASCII, we have 8 bits to represent the keyboard input, so we have 2 power 8 available values.

 Each individual byte of State, plain, is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value to select a unique 8-bit output value. Easy process, TRYIT.

**2. ShiftRows: a Permutation step.**

The bytes in each row of the state matrix are shifted to the left. The number of

positions each byte is shifted depending on its row number in the matrix. First row

(Row0) isn't shifted. Row1 shift by 1, Row2 shift by 2, and Row3 shift by 3. Easy process, TRYIT.

**3. MixColumns: a Matrix multiplication step.**

 It is designed to provide further diffusion and confusion, enhancing the overall security of the cipher. Each column of the state array is processed separately to produce a new column. The new column replaces the old one. The process is called Matrix Multiplication.

 In the MixColumns process, each column of the State matrix is treated as a polynomial, and a mathematical transformation called "polynomial multiplication" is performed on each column. Polynomial multiplication is done modulo a fixed irreducible polynomial in the Galois Field.

 This process helps to prevent patterns and correlations in the data from persisting through multiple rounds of encryption. It increases the resistance of the cipher against various cryptanalytic attacks.

4. **AddRoundKey**:

an XOR step with a round key, Ki. The 128 bits of State are bitwise XORed with the 128 bits of the round key.

**4.2.5. Decryption process**

In the previous section at encryption, the input is plaintext, clear text. At decryption the input is ciphertext and we go in reverse operation. The process aims to recover the original plaintext from the ciphertext using the same encryption key that was used during the encryption process. S-box be RS-box, ShiftRow be InverseShiftRow, and MixColumns be InverseMixColumns.

1. **KeyExpansion**:

The decryption process starts by expanding the original encryption key into a set of round keys. These round keys are used in reverse order during decryption, beginning with the last round key used during encryption.

1. **AddRoundKey(Last Round):**

The first decryption step is to apply the AddRoundKey operation using the last round key used during encryption. This step is identical to the AddRoundKey step I encryption.

1. **Inverse ShiftRows:**

The Inverse ShiftRows step is applied to the State matrix (cipher), reversing the shifting of rows performed during the encryption process. This step moves the bytes in each row to the right instead of left.

1. **Inverse SubBytes:**

The Inverse SubBytes step involves replacing each byte in the State matrix with its original value from the Substitution Box (S-box). This step reverses the substitution applied during encryption.

1. **Rounds: In each round, the following steps are performed:**

** Inverse MixColumns:**

The Inverse MixColumns step is applied to the State matrix, undoing the mixing done during encryption. It involves polynomial multiplication in the Galois Field.

 **Inverse** **ShiftRows**:

The Inverse ShiftRows step is applied again to undo the shifting of rows performed in the corresponding encryption round.

** Inverse SubBytes:**

The Inverse SubBytes step is performed once more to revert the substitution of bytes from the Substitution Box.

 **AddRoundKey**:

In each round, the State matrix is XORed with the corresponding round key in reverse order to undo the key mixing done during encryption.

1. **Final Round (Inverse AddRoundKey):**

The last decryption round involves applying the AddRoundKey operation using the first-round key used during encryption. This undoes the initial AddRoundKey operation in the first encryption round.