Celal Bayar University Journal of Science

AES Encrypted Real-Time Video Stream and Image Transmission from ESP32-CAM

Pınar Savaştürk[1], Ömer Aydın[2*], Gökhan Dalkılıç[3]

[1] Gebze Technical University, Faculty of Engineering, Computer Engineering, Kocaeli, Turkey
[2] Manisa Celal Bayar University, Faculty of Engineering, Electrical and Electronics Engineering, Manisa, Turkey
[3] Dokuz Eylul University, Faculty of Engineering, Computer Engineering, İzmir Turkey
*omer.aydin@deu.edu.tr
*Orcid: 0000-0002-7137-4881

Received: 04.12.2020
Accepted: 7 November 2021
DOI: 10.18466/cbayarfbe.835945

Abstract

With the development of the technology, the demand for security has increased. Security comes to the forefront due to the increase in data produced and transmitted in real time. In this study, improvements in the security of video data transmitted in real time are proposed. Data encrypted with advanced encryption standard (AES) was transferred to the local server from the ESP32-Cam module. The encrypted data transferred was decrypted on the local server, so the user was able to access the video. In this way, the security level of the data transmitted live between the parties has been increased. Our motivation in this study is that the number of applications proposed on real-time systems is not too much.

Keywords: Arduino, AES, ESP32-CAM, encrypted image transmission, Nodejs, secure video transmission, security

1. Introduction

Along with the new technologies developed, the security of the systems installed with these technologies has also gained great importance. Security is critical, especially in systems with the live and continuous data flow. For example, systems with real-time data flow, such as images and videos, have gained importance. Therefore, a security layer has been added to increase security in the transfer of video and image data in this study. The security layer was realized on the ESP32-CAM module and the Web Server.

The main purpose of this study is to ensure that the video and image data transmitted between the parties over the wireless network are confidential. Two basic approaches were emphasized to create confidentiality.

The first approach is to send the image and video data from the ESP32-CAM module by shifting. In this way, attackers who want to receive data flowing on the Wi-Fi network can only access modified, pointless and unformed data. In this method, the transmitted data can be made to an extent incomprehensible. The Web Server, which receives shifted data, on the contrary, accesses the original data received by the camera when it performs the inverse shift operation. This way, the Web Server can process incoming data on localhost.

In the second approach, data sent from the camera module are encrypted with an encryption algorithm such as advanced encryption standard (AES). Then, encrypted data is sent to the other party. Received data is decrypted on the Web Server and processed on localhost. In this way, the data flowing through the ESP32-CAM become more incomprehensible than shifting. In this process, it is necessary to make sure that the webserver decrypts the encrypted data in accordance with the encryption method on the camera module.

In both of these approaches, the security of the data has been increased by the possibility to re-create the original image or video, because the algorithms used are lossless.

2. Related Works

Secure elements were used to secure four-quarter IoT device architecture, in a study presented by Urien [1]. This architecture includes a radio system on chip (SoC), a secure element that processes transport layer security (TLS) packets and acts as an identity module, a

sensor/actuator and a general-purpose unit (GPU). The system they recommend is a low cost, low energy and safe. This system can be developed in the future using the block chain structure. This is a structure that is not in the current architecture. It will offer the possibility to be used in a public blockchain structure.

One of the structures that offers high security [2] uses a separate architecture. According to this structure, TLS server running in a secure element is used to make the established communication channels safer on a wireless network. Matters such as mutual communication security and correctness are examined more carefully with this structure. Besides, since transmission control protocol (TCP), which is one of the layered structure protocols, is used, the whole system is gathered under a standard. In this way, the common channels of the system are fully communicated. One point that can be added to this structure is that it can be included in this structure is the live video stream. In this way, an important chain link may have been added to the system.

With the platform of developing IoT devices, the security part has become more and more important. Communication protocols, which are part of this whole [3], are of great importance. A structure, which contains the message queuing telemetry transport (MQTT) protocol, has been modelled and implemented to provide security with special encryption structures. A situation similar to the sender-receiver structure has been revealed. With this structure, which is defined as secure MQTT (SMQTT), the approach that security can be increased to higher levels has been revealed. Also, if this structure can be applied to live stream data, performance analysis can be made, and the architecture can be tested on different scenarios.

Yerlikaya and Dalkılıç [4] proposed the authentication and authorization mechanism in their study. Their scheme uses HMAC-based one-time password (HOTP) for device authentication addition to the open authorization (OAuth 2.0) protocol [5, 6].

Aydın and Erhan proposed a study of secure communication for ESP-Eye. This study presented a method of security for the camera, wireless communication module and face recognition. A comprehensive solution was not presented in this study [7].

A model containing protocol bases has been introduced in the structure [8] established to ensure secure video transmission. This mechanism provided remarkably useful results. In addition, the performance results obtained for the transmission of video packages reveal the efficiency of the study. If this mechanism can be provided with less costly modules, it will find more usage and application areas in practice.Video transfer on the DJI Tello drone and analysis of the images obtained with structures such as optical character recognition (OCR), single shot multibox detector (SSD) have revealed remarkable results [9]. According to these results, the images obtained are analyzed and meaningful structures are revealed. In addition, presenting and expressing the obtained structures in a meaningful way increase the importance of the study. It is very critical to receive, process and present the following image for the drones, which are remote control devices. These goals were successfully implemented in the study. In addition, if the security mechanism were added to the parts of the entire model, the transmitted images was secured.

In the literature, there are many studies on the encoding and transmission of pictures and moving picture experts' group (MPEG) videos [10-12, 14-16, 18]. It is also presented in various security and privacy related applications that are used and will work in such applications [13, 17, 19, 20].

3. Video or Image Transmission

Video transmission and display process on most systems are very important. In this study, it is aimed to make a secure video or image transfer system. To achieve this goal, the image must first be taken by ESP32-CAM module. This module acts as a data provider. The feature has been installed when customizing the module for this usage. To display the image taken from the module, it joined to a common network.

When the ESP32-CAM module is connected to a common network, it provides an Internet protocol (IP) address to broadcast. This network is a common network, including the Web Server. Within the scope of this study, the ESP32-CAM module and the desktop computer where the Web Server is installed are connected to the same access point. The user that we assume as an observer, connects to an access point via his/her computer. This access point is the same device that the ESP32-CAM and Web Server Provider Device connects to. In this way, both the observer and the camera module are connected to the same network. Once this partnership is achieved, the logs are observed in the web browser with the IP address provided by the ESP32-CAM. During this development, the necessary implementations for the ESP32-CAM module were made by Arduino IDE 1.8.12 version and Web Server was made by NodeJS with 10.16.3 version. A special-purpose library was used to show live streaming data while making the necessary improvements.

4. Shifted Data Transfer and Render

A model for sending the data by shifting is detailed as the first approach in the subsections. This model is vulnerable. The shifted data is sent, and the original data is obtained by a reverse process on the receiver side.

4.1 ESP32-CAM Module Side

The libraries used when developing the ESP32-CAM module are as follows: “esp_camera.h”, “WiFi.h”, “ArduinoWebSockets.h” and “camera_pins.h”. The camera model of the module has been chosen as CAMERA_MODEL_AI_THINKER. The service set identifier (SSID) and the password of the modem to which the camera module is connected are set by default in the Arduino code development. To send data from the module to the Web Server, the host and port number of the Websocket have been determined. The IP address to host the Websocket has been provided. While determining this IP address, "ipconfig" command was run on the device that is running a Web Server (this device is a Windows 10 desktop). The "IPv4 address" obtained as a result of executing “ipconfig” command is the address that hosts the Websocket. The specified Websocket server port number in the Arduino code is 8888. Subsequently, the client object for the Websocket is created in "WebSocket Client" type.

After necessary configuration settings and parameters are set for the camera module, a successful connection of the module to the web server is ensured. Subsequently, the websocket server host address (IPv4 address) and the specified websocket server port provide a connection to the websocket client. After the connection is successful, the camera is ready to send data to the server.

At this point, modifications of the data are required for shifting data transmission. The data type received from the camera module is in a special struct like “camera_fb_t”. The “buf” parameter of this struct corresponds to the pixel data received from the camera. The raw form of the data taken from the camera is in “char” pointer type.

When a 32-bit left shift operation is applied to this data, the payload (i.e. real-time streaming data) coming from the camera becomes incomprehensible. After the shift operation, the client is in binary type; the shifted payload and the size of this payload are sent.

4.2 Web Server Side

The Web Server side was developed on “Node.js”. Some components were used in the development process. These components are: ‘path’ [21], ‘express’ [22] and ‘ws’ [23]. Port number 8888 was determined as a Web Server port. On the other hand, 8000 was determined as hypertext transfer protocol (HTTP) port. After the Web Server made the necessary connections, it started to listen to the data from the camera module.
Here, the reverse shift operation was applied after the stream from the camera was handled. Then, the stream data ready for rendering were sent to the hypertext markup language (HTML) page by the client for display.

4.3 Rendering Page Side

The rendered page is a HMTL page [24]. This page creates the connection to the Web Server and waits for the messages from there. It casts the data from the buffer type to the blob type and assigns it to the object created with "img" tags so, the image can be seen by this HTML page.

4.4 Order of Operations

The process is in the following order:

1. The Arduino code is assigned to the ESP32-CAM module.
2. The file path containing the Node.js codes, created as a web server, is accessed from the "Node.js command prompt" screen.
3. In "Node.js command prompt" screen, the server is run using the "node <filename.js>" command.
4. The Chrome web browser runs on the computer where Web Server is running. Then, the "IPv4 address:8000/client" URL is accessed (IPv4 address is the address hosted on the web server).
5. Data flow is triggered by resetting the ESP32-CAM module.

5. Shifted Encrypted Data Transfer and Render

Under this section, it is suggested to add a security layer that is not included in the system mentioned in the previous subsections. Now let's explain this model and how to add the security layer.

5.1 ESP32-CAM Module Side

Some additional libraries have been used in the shifted encrypted data transfer and creation process for the ESP-32 Cam module side. These libraries can be listed as “AES.h”, “base64.h”, “iostream”, “stdio.h” and “string.h”. Everything is done as described in "shifted data" section (Section 4) until the data is taken from the camera module. After the data reaches, the processes differ.

Each of the received data is stored as "HEX" in "unsigned int" type. All of the stored data is too large to be encrypted. Therefore, the data are divided into 256- and 200-byte blocks.

Firstly, each divided block goes through base64 encoding operation by using the initial vector. The cipher text, secret key, and initial vector, encoded with base64, are subjected to 128-bit AES encryption. Parts of payload that are converted into encrypted form are sent to the Web Server one by one.

Celal Bayar University Journal of Science
Volume 17, Issue 4, 2021, p 447-452
Doi: 10.18466/cbayarfbe.835945

5.2 Web Server Side

In addition to the operations given in the "shifted data" section, "crypto-js" is used by the Web Server. On the server, the encryption key is shared with the ESP32-CAM module side.

Each block of the data coming from the camera module handles on the server side. Since the camera module sends the data in binary format, the incoming data is first converted to the "Base64" data type. The data converted to the appropriate format is decrypted with an initial vector and key. Since the streaming data comes piece by piece, it is brought together in accordance with order of arrival.

The type of incoming data must be in accordance with the JPEG data format. The header formats of the data, which are decoded and gathered, are checked to see if there is any interference between the data transmission. Finally, it is sent to the HTML page to be rendered.

The operations to be applied after this process is the same as the subsections "Rendering Page Side" and "Order of Operations" which are under the "Shifted Data Transfer and Render" section.

5.3 General Diagram

In Figure 1, you can see the process flowchart of the secure video transmission process with ESP32-CAM.

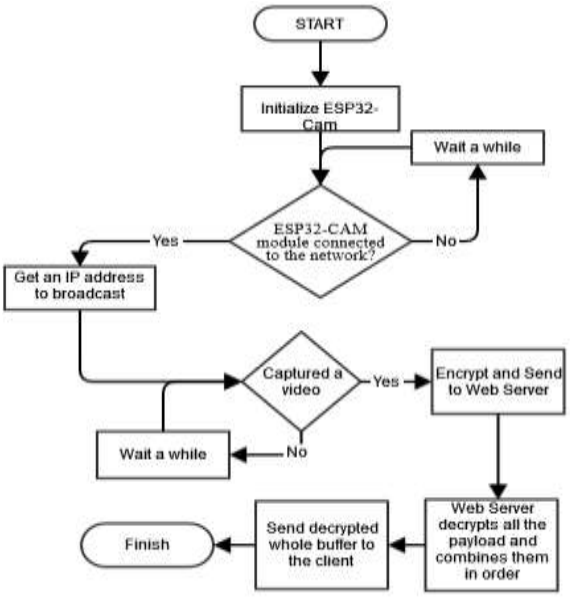

Figure 1. Flowchart of the secure video transmission process with ESP32-CAM.

Figure 2 gives an overview of all the process steps described in the previous sections.

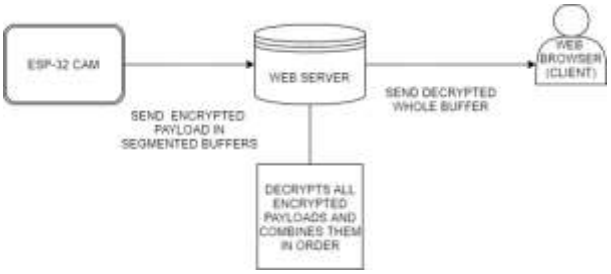

Figure 2. General diagram of the encrypted payload transfer.

The encrypted payload contained in the segmented buffer is sent to the Web server over the ESP-32 CAM. On a webserver, all encrypted payload is decrypted, and the webserver combines them in order. Finally, decrypted whole buffer is sent to the client (web browser).

6. Results and Discussion

You can find the general structure of the system and test results in the subsections below.

6.1 General Structure

In this study, the operations and experiments were carried out by using the ESP32-CAM IoT device.

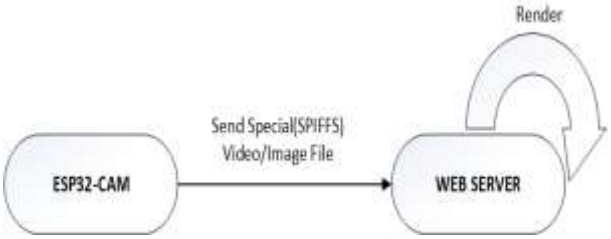

Figure 3. ESP32-CAM and web server transmission diagram.

On the system given in Figure 3, the ESP32-CAM module is acting on a server and the image was taken at the local host of the computer connected to the same network. In this state, the device sends received image to the web server and the browser renders the image. The structure established in this way has a security flaw. The image or video file is transmitted over the wireless network without a security layer. Since it does not have a security layer, it is vulnerable to malicious people. Another disadvantage of this structure is that only one client who wants to obtain the images or video from the camera can reach them. Another problem encountered in this structure is the format of the file sent from the ESP32-CAM module to the web server. It is not possible to encrypt and send the file and decrypt it by the web server. Because the IoT module and Web Server are programmed on the same device. Therefore, it is not possible to determine and apply a dividing line between them.

Celal Bayar University Journal of Science
Volume 17, Issue 4, 2021, p 447-452
Doi: 10.18466/cbayarfbe.835945

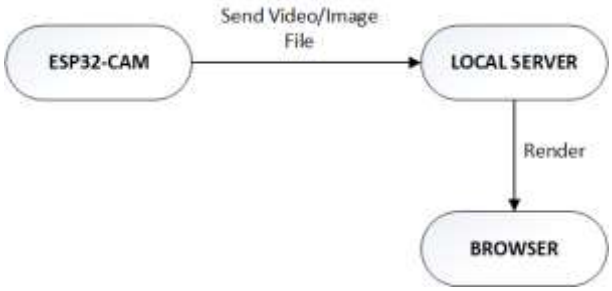

Figure 4. Separated ESP32-CAM and local server diagram.

Figure 5. ESP32-CAM and local server communication.

As shown in Figures 4 and 5, a different approach has been taken to address the previously mentioned flaw. Instead of keeping the server on ESP32-CAM, server was designed as a separated computer connected on the same network, so the only task of the camera module is sending the video or picture. This computer is acting as a server and took over the task of rendering in the browser. In this way, the camera module is able to send the encrypted file. On the computer, the server set up as Node.js is able to decrypt and render the incoming file. As a result of this design, an extra security layer has been added to the system for a critical process such as an image or video transfer. Moreover, this system enables more than one client to receive images at the same time.

6.2. Test Results

In Table 1, a comparison table was given. Response times are shown when encryption is performed for image transfer and when encryption is not performed.

Each tuple in Table 1 represents images that mean packages of the same size. Each of these tuples represents different images. The average Wi-Fi speed of the modem used is 30 MBits/s. Encrypted data has the same size as plain data to avoid growth in transferring data.

The time when data is sent from the ESP32-CAM module is determined at the start time for each calculated time. On the other hand, the end time is obtained when the server sends the data to the HTML page. Within this period when encryption is proceeded, there are processes such as encryption, transmission, decryption of the payload and combination of the buffers that come with coordination. The image that is not encrypted has only the payload that is sent during the transfer and the server has to handle the incoming message.

Table 1. Comparison table for encrypted and non-encrypted image transfer durations.

Transferred Image Packages	Non-Encrypted Image Transfer and Render Durations (ms)	Encrypted Image Transfer, Decryption and Render Durations (ms)
Package_1	2169	4597
Package_2	203	5462
Package_3	512	5205
Package_4	756	4975
Package_5	650	4870
Package_6	924	5634
Package_7	712	4989
Package_8	618	4782
Package_9	569	5084
Package_10	878	4829

7. Conclusion

The number of IoT devices is increasing and their usage areas are expanding. These devices can capture images and videos. This image or video data obtained is transmitted over wireless networks. This situation brings along various security problems. There are many challenges in ensuring security, especially for live and streamed data. There are some studies in the literature to find a solution to this and to secure communication. In this context, this study has been put forward to ensure the secure transmission of video and images from IoT devices. ESP32-CAM was used in the study. ESP32-CAM has the ability to shoot video and images. It can also communicate by connecting to a network with a wireless connection. As the first approach, the image and video data obtained were shifted and transmitted in this way. There are problems in the structure of this model and in the security of the transmitted data. Therefore, a second model has been proposed. Structural changes have been made in this model and the transmitted data is encrypted with AES. In addition, the comparison of the transmission durations was given in order to compare the two models in the study.

As a result, with the structure established in the second model, a more secure system against attacks has been introduced. This model makes an important contribution to the security of IoT devices that are lightweight in terms of hardware resources.

Author's Contributions

Pınar SAVAŞTÜRK: Wrote the draft manuscript, prepare the system, and made the experiments.
Ömer AYDIN: Assisted in analysis on the structure, supervised the experiment's progress, made result

interpretation, helped in manuscript preparation, made the manuscript ready for the journal, took part in the journal submission and following the journal process.
Gökhan DALKILIÇ: Served as the consultant in the execution of the whole process. He supervised in the process of the preparation of the manuscript, revealed the main idea, made criticism, and made the proof reading in language.

Ethics

There are no ethical issues after the publication of this manuscript.

References

1. Urien, P. An Innovative Four-Quarter IoT Secure Architecture Based on Secure Element, In 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), Limassol, 25-27 June, 2018, pp. 1074-1080. Doi: 10.1109/IWCMC.2018.8450435

2. Urien, P. An innovative security architecture for low cost low power IoT devices based on secure elements: A four quarters security architecture, In 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, 12-15 January, 2018, pp. 1-2. Doi: 10.1109/CCNC.2018.8319309

3. Singh, M, Rajan, M, Shivraj, V, Balamuralidhar, P. Secure MQTT for Internet of Things (IoT), In 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, 746–751, 4-6 April, 2015. Doi: 10.1109/CSNT.2015.16

4. Yerlikaya, Ö, Dalkılıç, G. Authentication and authorization mechanism on message queue telemetry transport protocol. In 2018 3rd International conference on computer science and engineering (UBMK), 2018, pp. 145-150. IEEE.

5. Windley, PJ. 2015. API access control with OAuth: Coordinating interactions with the Internet of Things, *IEEE Consumer Electronics Magazine*, 4:52-58.

6. Fremantle, P, Aziz, B. OAuthing: Privacy-enhancing federation for the Internet of Things, In 2016 Cloudification of the Internet of Things (CIoT), 2016, pp. 1-6, doi: 10.1109/CIOT.2016.7872911.

7. Aydın, Ö, Erhan, İİ. 2021. Video or Image Transmission Security for ESP-EYE IoT device used in Business Processes. *Yönetim Bilişim Sistemleri Dergisi*, 7(1): 1-9. Retrieved from https://dergipark.org.tr/tr/pub/ybs/issue/63606/857203

8. Radu, D, Cretu, A, Avram, C, Astilean, A, Parrein, B. Video content transmission in a public safety system model based on flying Ad-hoc networks. In 2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, 24-26 May, 2018, pp. 1-4. doi: 10.1109/AQTR.2018.8402713

9. Mirando Pinto, LG, Mora-Camino, F, de Brito, PL, Brandão Ramos, AC, Castro Filho, HF. A SSD – OCR Approach for Real-Time Active Car Tracking on Quadrotors, In 16th International Conference on Information Technology-New Generations (ITNG 2019), Las Vegas, 1-3 April 2019, pp. 471-476. Doi: 10.1007/978-3-030-14070-0_65

10. Maniccam, S, Nikolaos, G. 2004. Image and video encryption using SCAN patterns, *Pattern Recognition*, 37(4): 725-737. doi: 10.1109/30.920426

11. Yi, X, Tan, CH, Slew, CK, Syed, MR. 2001. Fast encryption for multimedia. *IEEE Transactions on Consumer Electronics*, 47(1): 101-107.

12. Tosun, AS, Feng, WC. Efficient multi-layer coding and encryption of MPEG video streams, In 2000 IEEE International Conference on Multimedia and Expo. ICME2000, New York, NY, 30 July-2 August, 2000, pp. 119-122. Doi: 10.1109/ICME.2000.869559

13. Choo, E, Lee, J, Lee, H, Nam, G. SRMT: A lightweight encryption scheme for secure real-time multimedia transmission, In 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07), Seoul, 26-28 April, 2007, pp. 60-65. Doi: 10.1109/MUE.2007.194.

14. Qiao, L, Nahrstedt, K. A new algorithm for MPEG video encryption, In First International Conference on Imaging Science System and Technology, Las Vegas, Nevada, USA, 30 June- 3 July, 1997, pp. 21-29.

15. Tang, L. Methods for encrypting and decrypting MPEG video data efficiently. In Fourth ACM international conference on Multimedia, Boston Massachusetts USA, November, 1996, pp. 219-229.

16. Sikora, T. 1997. MPEG Digital Video Coding Standard. *IEEE Signal Processing Magazine*, 14(5): 82-100. Doi: 10.1109/79.618010

17. Shannon, CE. 1949. Communication Theory of Secrecy Systems, *The Bell System Technical Journal*, 28(4):656-715. Doi: 10.1002/j.1538-7305.1949.tb00928.x

18. Spanos, GA, Maples, T. B. Security for real-time MPEG compressed video in distributed multimedia applications, In 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications, Scottsdale, AZ, USA, 27-29 March, 1996, pp. 72-78. Doi: 10.1109/PCCC.1996.493615

19. Adam, JA. 1992. Cryptography = privacy?. *IEEE Spectrum*, 29(8): 29-35. Doi: 10.1109/6.144533

20. Spanos, GA, Maples, TB. Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-Time Video, In Fourth International Conference on Computer Communications and Networks - IC3N'95, Las Vegas, NV, USA, 20-23 September, 1995. DOI: 10.1109/ICCCN.1995.540095

21. Internet: Nodejs.org, https://nodejs.org/api/path.html (accessed at 17.09.2021).

22. Internet: Fast, unopinionated, minimalist web framework for Node.js, https://expressjs.com/ (accessed at 17.09.2021).

23. Internet: ws: a Node.js WebSocket library, https://github.com/websockets/ws (accessed at 17.09.2021).

24. Graham, I. S. (1995). The HTML sourcebook. John Wiley & Sons, NY United States. ISBN: 978-0-471-11849-7