SECURE ACCESS WITH TELTONIKA GPS TRACKING DEVICES FOR INTELLIGENT TRANSPORTATION SYSTEMS

Gheorghe-Romeo ANDREICA[1,2], Ciprian STANGU[2], Iustin-Alexandru IVANCIU[1], Daniel ZINCA[1], Virgil DOBROTA[1]
[1]*Communications Department, Technical University of Cluj-Napoca, Romania*
[2]*AROBS Transilvania Software Cluj-Napoca, Romania*
Romeo.Andreica@com.utcluj.ro; Ciprian.Stangu@trackgps.ro; Iustin.Ivanciu@com.utcluj.ro; Daniel.Zinca@com.utcluj.ro, Virgil.Dobrota@com.utcluj.ro

Abstract: GPS tracking devices are widely used in industries like logistics, transportation, and security. However, they are susceptible to cyber-attacks, including Man-in-the-Middle (MITM). This study focuses on Teltonika GPS tracking devices and examines the impact of MITM attacks on their operation. We propose implementing encryption protocols and other measures to enhance the security and resilience of Teltonika GPS tracking devices.

Keywords: *GPS Tracking, MITM Attack, TLS/SSL encryption.*

I. INTRODUCTION

Internet of Things (IoT) devices based on Global Positioning System (GPS) or other principles [1] enable organizations to track their assets and to monitor the movements of their employees and/or vehicles. However, these devices are vulnerable to cyber-attacks, which can compromise their accuracy, reliability, and security [2]. One of the most significant threats to the case of GPS tracking devices is represented by the Man-in-the-Middle (MITM) attack [3], [4]. This occurs when an attacker intercepts and alters the communications between two parties. In the context of this approach, it means that an attacker can intercept the GPS data sent by the device to the tracking platform and to modify it, to show a different location or route [5]. This can have serious consequences, such as lost productivity, stolen goods, or compromised personal safety [6].

One approach to protect against MITM attacks is to implement encryption protocols, such as Transport Layer Security (TLS) and Secure Sockets Layer (SSL) [4], [7]. These protocols can help to ensure that data transmitted between the device and the tracking platform is secure and cannot be intercepted or modified by the attackers. We investigated the impact of this type of attack and we proposed several mechanisms that can be integrated into the firmware of the devices. Thus, Teltonika GPS tracking devices can be made more resilient against cyber-attacks, enabling businesses and individuals to trust the accuracy and security of the GPS data [8].

The remainder of the paper is organized as follows: Section II describes the architecture of FMB122 IoT device, whilst Section III discusses the legacy insecure algorithm integrated within the firmware. Section IV presents the countermeasures to secure the firmware, and Section V concludes the paper.

II. TELTONIKA FMB122 IOT DEVICE

The Teltonika FMB122 solution (see Figure 1) is a compact and versatile GPS tracking device designed for various applications, such as fleet management, asset tracking, and personal tracking. The device features a powerful ARM Cortex-M3 processor, built-in GSM/ GPRS and GPS modules, and a range of I/O interfaces, including digital/ analog inputs, and digital outputs [9].

Figure 1. FMB122 hardware device.

Its firmware was written in C programming language and it is based on the FreeRTOS real-time operating system. The code was designed to control the device's hardware components, including the GPS and mobile communications modules, and to manage the exchange of information between the device and the tracking platform. The firmware can be updated Over-the-Air (OTA), allowing Teltonika to release updates to address security vulnerabilities and other issues [8], [10].

However, the FMB122 model uses encoding algorithms (i.e., Base64 or hexadecimal conversion) for data in transit, which provide less security than encryption. The legacy algorithms can be easily decoded, making them vulnerable to MITM attacks. Thus, without encryption, data transmitted between the device and the tracking platform can be intercepted, modified, or even replaced by attackers [11], [12]. For a better understanding of the processes of encoding and decoding, we summarized herein an algorithm used by Teltonika devices for telemetry data transmission.

Manuscript received March 29, 2023; revised May 27, 2023.

The structure is as follows: X is longitude; Y is latitude; Altitude in [meters above the sea level]; Angle in [degrees], where 0 is north, increasing 1 unit clockwise; Satellites in [number of visible satellites]; Speed in [km/h]. Note that the value 0x0000 is obtained when GPS data is invalid. Longitude and latitude are integer values constructed from degrees d, minutes m, seconds, and milliseconds according to the formula (d: degrees, m: minutes, s: seconds, ms: milliseconds, p: accuracy (10,000,000) [8], [9], [12]:

$$\left(d + \frac{m}{60} + \frac{s}{3{,}600} + \frac{ms}{3{,}600{,}000}\right) * p \qquad (1)$$

Figure 2 presents an example of 152 bytes sent using Codec 8 Extended (see the differences compared to Codec 8 in Table I). This does not provide an adequate level of security and it remains unsecured against MITM attacks [8], [12]

TABLE I. DIFFERENCES BETWEEN CODEC 8 AND CODEC 8 EXTENDED

	Codec 8	**Codec 8 Extended**
Codec ID	0x08	0x8E
AVL Data IO element length	1 Byte	2 Bytes
AVL Data IO element total IO count length	1 Byte	2 Bytes
AVL Data IO element IO count length	1 Byte	2 Bytes
VL Data IO element AVL ID length	1 Byte	2 Bytes
Variable size IO elements	Does not include	Include variable size elements

Adelson-Velsky and Landis (AVL) data element sizes in Codec 8 Extended were increased to 2 bytes length and new variable type added. For a detailed description see [12].

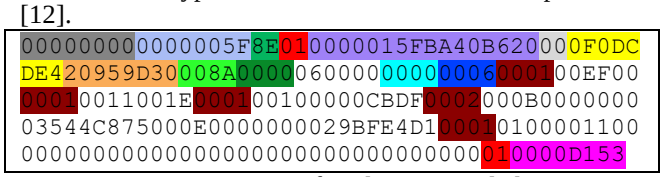

Figure 2. Use of Codec 8 Extended

The significance of these fields is the following:
00000000 - 4 zeros, 4 bytes
0000005F - data length, 4 bytes
8E – codec ID
01 - number of data (1 record)
0000015FBA40B620 - Timestamp in milliseconds (1510658324000)
GMT: Tuesday, January 14, 2023 09:18:10 AM
00 – priority
0F0DCDE4 – longitude 252562916 = 25, 2562916º N
20959D30 – latitude 546676016 = 54.6676016 º E
008A – altitude (138 meters)
0000 – angle (0)
06 – 6 visible satellites
0000 – speed 0 km / h
IO Element
0000 - IO element ID of Event generated (in this case when 0000 – data generated not on event)
0006 – 6 IO elements in record (total)
0001 – 1 IO elements, which length is 1 Byte
00EF – IO element ID = 239 (dec)
00 – IO element's value
0001 – 1 IO elements, which length is 2 Byte
0011 – IO element ID = 17 (dec)
001E – IO element's value
0001 – 1 IO elements, which length is 4 Byte
0010 – IO element ID = 16 (dec)
0000CBDF – IO element's value = 52191 (dec)
0002 – 1 IO elements, which length is 2 Byte
000B – IO element ID = 11 (dec)
0000000003544C875 – IO element's value
000E – IO element ID = 14 (dec)
0000000029BFE4D1 – IO element's value
01 – Number of data (1 record)
0000D153 - CRC-16, 4 bytes (the first two are always zeros)

AVL data packet is the same as with codec 8, except codec ID is changed to 0x8E. An example of data received by the server is presented in Figure 3.

```
00A1CAFE001B000F333536333037303432343431303
1338E010000013FEBDD19C8000F0E9FF0209A718000
690000120000
001E090100020003000400160147 03F0001504C8000
C0900910A00440B004D13004443155544000 0B5000B
B60005422E
9B180000CD0386CE000107C700000000F10000601A4
60000013C4800000BB84900000BB84A00000BB84C00
000000024E
0000000000000000CF00000000000000001
```

Figure 3. Data received by the server

The significance is the following:

- Data length: 00A1 or 161 bytes (not counting the first two data length bytes).
- Packet identification: 0xCAFE 2 bytes.
- Not usable byte: 00.
- Packet ID: 1B.
- IMEI length: 000F.
- Actual IMEI: 333536333037303432343431303133.
- Codec id: 8E.
- Number of data: 01.
- Timestamp: 0000013FEBDD19C8.
- Priority: 00.
- GPS data: 0F0E9FF0209A718000690000120000.

The decoder has two main parts: *(1) Decoder Handler*: it contains the logic that validates the decoding message, extracts general information such as IMEI, and it knows in case of successful decoding to send successful receiving messages (ACK messages) to the equipment; *(2) Decoder:* contains the decoding logic of the message data (AVL data) [12].

Codec 8 extended protocol sending over UDP. AVL data packet is the same as with codec 8, except codec ID is changed to 0x8E. IoT module sends the data to the server and the server must respond with acknowledgment. Number of data – number of encoded data (number of records). Codec ID is constant 0x8E. Data field length is the length of bytes [codec id, number of data 2]. Number of data 1 should always be equal to number of data 2 byte.

CRC-16 is 4 bytes, but first two are zeroes and last two are CRC-16 calculated for [codec id, number of data 2]. Minimum AVL packet size is 53 bytes (all IO elements disabled). Also communication with server is the same as with codec 8 protocol, except in codec8 extended protocol codec id is 0x8E. Server acknowledges data reception (2 data elements): 00000002.

Figure 4 shows the communication mode between the IoT GPS device and the server, respectively the encoded information using the Codec 8 Extended, which is a public and unsecured algorithm exposed to the MITM attack.

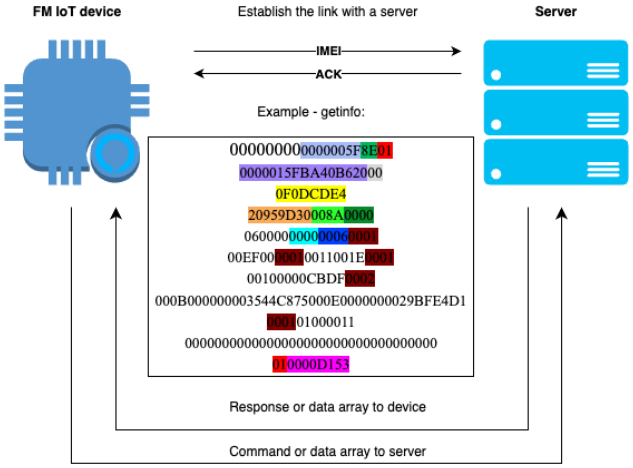

Figure 4. Data on transit between FM device and server.

This is how the FMB122 communicates with the server and receives a response, and how the information is encoded and decoded [8], [12].

III. LEGACY INSECURE ALGORITHM IN TELTONIKA FMB122 FIRMWARE

See in Figure 5 an example of source code for Teltonika FMB122 firmware, which is responsible for encoding and decoding telemetry data transmission using an insecure algorithm (publicly available) [12]. We used italics and red for the insecure parts.

```
//Encoding telemetry data for transmission
void EncodeTelemetryData(TelemetryData_t
telemetryData, uint8_t *encodedData)
{
    uint32_t pos = 0;
    uint8_t tmp8 = 0;
    uint16_t tmp16 = 0;
    uint32_t tmp32 = 0;
    uint8_t checksum = 0;
//Add start bits to the encoded data
    encodedData[pos++] = 0x01;
    encodedData[pos++] = 0x02;
    encodedData[pos++] = 0x03;
// Encode the longitude value
    tmp32 = (uint32_t)(telemetryData.longitude *
10000000);
    encodedData[pos++] = (tmp32 >> 24) & 0xFF;
[…]
//Encode the IMEI value
// Potential issue: IMEI is transmitted in plain
text. /MITM vulnerability: Longitude, latitude,
and other data are encoded in plaintext.
// An attacker could read or modify this data in
transit.
    for (uint8_t i = 0; i < 15; i++)
    {
        encodedData[pos++] =
telemetryData.imei[i];
    }
[…]
//Decode telemetry data received from the
tracking platform
// An attacker could read or modify this data in
transit.
void DecodeTelemetryData(uint8_t *receivedData,
uint32_t receivedDataLen, DecoderHandler_t
*decoderHandler)
{
[…]
//Decode the IMEI value
// Potential issue: IMEI is transmitted in plain
text.
// MITM vulnerability: IMEI is received in
plaintext, which could have been read or
modified by an attacker in transit.
for (uint8_t i = 0; i < 15; i++)
{
    decoderHandler->avlData.imei[i] =
receivedData[13 + i];
}
[…]
```

Figure 5. Insecure code using encoding algorithm.

This example of a code demonstrates how telemetry data (longitude, latitude, altitude, IMEI, angle clockwise, and the number of visible satellites) is encoded and decoded for transmission between the FMB122 device and the tracking platform in an insecure way [8], [9]. The encoded data includes start and end bits, longitude, latitude, altitude, IMEI, angle clockwise, and the number of visible satellites, and a checksum is calculated to ensure the integrity of the data and can be easily decoded if data is intercepted by an attacker using MITM technique.

The decoding function verifies the start bits, data length, and checksum of the received data before decoding the telemetry data and storing it in the AVL data structure. The AVL data structure is used to store and transmit the telemetry data to the tracking platform [12].

Overall, this example code demonstrates how the FMB122 firmware handles the encoding and decoding of telemetry data for transmission, which is a crucial aspect of the device's operation and data security [12]. In conclusion, the code sections highlighted in red have been identified as vulnerable to MITM attacks and other potential security issues. Therefore, it will be replaced with SSL/TLS encryption algorithms and secure functions to ensure data security in transit, integrated at the firmware level.

IV. SECURING DATA TRANSMISSION WITH ENCRYPTION

While encoding is a useful technique for converting data into a more suitable format for transmission, it does not offer security in terms of protecting data against MITM attacks. We discuss herein the following solutions: (1) *Implementing the Transport Layer Security (TLS) or Secure Sockets Layer (SSL):* These protocols provide end-to-end encryption of data, ensuring that data transmitted between the device and the tracking platform cannot be intercepted or modified by attackers [7]; (2) *Encrypting the AVL data structure:* the data can be protected against unauthorized access or modification, ensuring the confidentiality and integrity of the data. The International

Mobile Equipment Identity (IMEI) is a unique identifier for mobile devices, thus FMB122 is included too [4], [13].

In Figure 6 we present an example of modified firmware for Teltonika FMB122, to secure the connection between the client device and the server.

```
//Initialize TLS session for telemetry data transmission
void InitTLSSession(char *serverIP, uint16_t serverPort)
{
[…]
}
// Encrypting and sending telemetry data using TLS
void SendEncryptedTelemetryData(TelemetryData_t telemetryData)
{
    uint8_t encodedData[ENC_DATA_LEN];
    uint32_t pos = 0;
// Encode the telemetry data
    EncodeTelemetryData(telemetryData, encodedData);
// Encrypt the encoded data using TLS
    pos = 0;
    while (pos < ENC_DATA_LEN)
    {
        int ret = mbedtls_ssl_write(&tlsCtx.sslContext, encodedData + pos, ENC_DATA_LEN - pos);
        if (ret < 0)
        {
//Error writing to the TLS context - handle the error
            return;
        }
        pos += ret;
    }
}
//Decrypt and receive the telemetry data using TLS
void ReceiveDecryptedTelemetryData(DecoderHandler_t *decoderHandler)
{
    uint8_t receivedData[RECV_DATA_LEN];
    uint32_t pos = 0;
// Receive the encrypted telemetry data using TLS
    pos = 0;
    while (pos < RECV_DATA_LEN)
    {
        int ret = mbedtls_ssl_read(&tlsCtx.sslContext, receivedData + pos, RECV_DATA_LEN - pos);
        if (ret < 0)
        {
            // Error reading from the TLS context - handle the error
            return;
        }
        pos += ret;
    }
// Decrypt the received data
    DecodeTelemetryData(receivedData, RECV_DATA_LEN, decoderHandler);
}
```

Figure 6. Secured code using encryption algorithm.

In this example, instead of encoding telemetry data, TLS encryption is used to protect the confidentiality and integrity of the data transmitted between the FMB122 device and the tracking platform.

The `InitTLSSession()` function is responsible for establishing a secure TLS connection with the tracking platform, while the `SendEncryptedTelemetryData()` function encrypts and sends telemetry data using the TLS context. The `ReceiveDecryptedTelemetryData)` function receives and decrypts telemetry data transmitted over the TLS connection [4], [7].

By using TLS encryption, the telemetry data transmitted between the FMB122 device, and the tracking platform is protected against MITM attacks, ensuring that the data remains confidential and cannot be modified or intercepted by attackers. The `SendEncryptedTelemetryData()` function encrypts the telemetry data using the TLS context, while the `ReceiveDecryptedTelemetryData()` function receives and decrypts the telemetry data, ensuring that the data is secure and confidential throughout the transmission process [7].

The TLS protocol uses various mathematical algorithms and formulas to secure communication between the FMB122 device and the tracking platform. For a better understanding a brief summary of their principles is as following.

(1) Symmetric key encryption: this is based on a shared secret key, which is used to encrypt and decrypt data. The key is known only by the communicating parties, ensuring that the data remains confidential (see Figure 7).

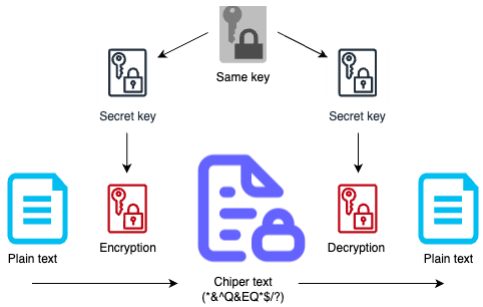

Figure 7. Symmetric encryption mechanism.

TLS uses various algorithms such as Advanced Encryption Standard (AES), Triple Data Encryption Standard (3DES), Rivest Cipher 4 (RC4) etc. [4], [14]. Using this method, the Codec 8 extended encoding algorithm is replaced at the firmware level by the TLS/SSL encryption algorithm, adapted and developed for the IoT device, ensuring a high level of security, preventing MITM attacks and data interception in transit for real-time protection, ensuring the integrity, confidentiality and availability of data in a GPS IoT monitoring devices for intelligent transportation systems. (2) *Public key encryption:* this is based on a pair of keys, a public key and a private key. The public key is used to encrypt data, while the private key is used to decrypt data (see Figure 8). TLS uses various public key encryption algorithms such as Rivest–Shamir–Adleman (RSA) and Elliptic Curve Cryptography (ECC) [15].

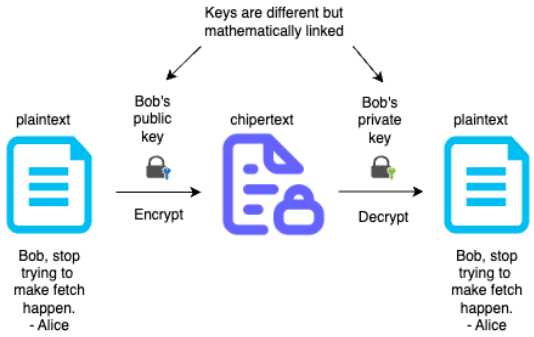

Figure 8. Public key cryptography mechanism.

(3) Hash functions: they are mathematical algorithms that convert input data into a fixed-length output called a hash (see Figure 9). TLS uses various hash functions such as SHA-256 (Secure Hash Algorithm 256) and SHA-384 to ensure that the transmitted data has not been tampered with [14].

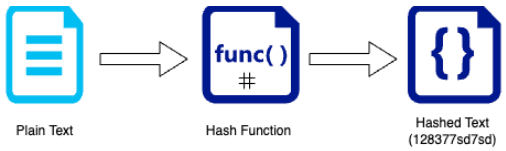

Figure 9. Hash algorithm.

(4) Key exchange algorithms: they are used to ensure that the symmetric key for encryption and decryption is securely shared between the communicating parties (see Figure 10). TLS uses various key exchange algorithms such as Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH) [13].

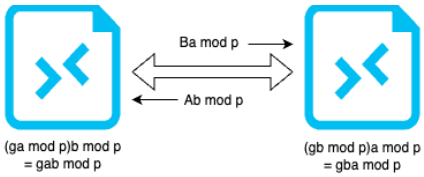

Figure 10. Diffie-Hellman key exchange algorithm.

By the time this paper was submitted, some ongoing field test were under progress, using the infrastructure of AROBS Romania (as the major beneficiary of this work, requested by this company). We expect overall, by performing these tests and analyzing the results, to get the evidence of the effectiveness of encryption over encoding in the Teltonika FMB122 GPS tracking device.

In Figure 11, the experimental result (MITM Attack on Unsecured Encoded Data) is presented, along with the way data can be intercepted in Wireshark and our custom decoding script output, using the encoding/decoding mechanism in the unsecured previous code (Figure 5).

```
No.     Time       Source                Destination
Protocol Length Info
      1     0.000000                    192.168.1.2
192.168.1.3             TCP       66      54123 →
54545 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256
SACK_PERM=1
INFO: TCP Handshake detected
...
INFO: Initiating ARP spoofing...
...
INFO: ARP spoofing successful...
      2     0.003345                    192.168.1.2
192.168.1.3             TCP       162     Telemetry
Data

Data:
    Start Bits: 01 02 03
    Longitude: 23.6236
    Latitude: 46.7712
    Altitude: 320
    IMEI: 356307042441013
    Clockwise: 0
    Number of Visible Satellites: 10
    Checksum: 0xD4
    End Bits: 0D 0A

Alert: Possible privacy leak detected. IMEI number
and precise location data intercepted.
```

Figure 11. MITM Attack on Unsecured Encoded Data

The data is intercepted by the MITM and easily decoded due to the absence of encryption. In this experiment (MITM Attempt on SSL/TLS Encrypted Data), we use the same setup but with the device transmitting the telemetry data secured with SSL/TLS, as is shown in Figure 12.

```
No.            Time                    Source
Destination           Protocol Length Info
      1    0.000000                  192.168.1.2
192.168.1.3           TLSv1.3  219    Client
Hello
INFO: TLS Handshake detected
...
INFO: Initiating ARP spoofing...
...
INFO: ARP spoofing successful...

      2    0.003456                  192.168.1.2
192.168.1.3                    TLSv1.3    275
Application Data

Attempting   to   decrypt   TLS   Application
Data...

ERROR: Unable to decrypt data. RSA key not
found. Invalid or missing private key!

Data:
    Encrypted Data: "2qV4fG2aA1rR..."
```

Figure 12. MITM Attempt on SSL/TLS Encrypted Data

Even though the data was intercepted using the same MITM techniques, the encryption provided by SSL/TLS ensures that the content is unreadable without the corresponding decryption key. This clearly demonstrates the fundamental difference in security between unencrypted (encoded) and encrypted data transmission. Even if a MITM attack is successful in intercepting the data, without the correct decryption keys, the content remains secure when encrypted with SSL/TLS.

In Figure 13, a diagram of the experimental MITM attack can be seen, both unsecured using encoding and secured using SSL/TLS encryption.

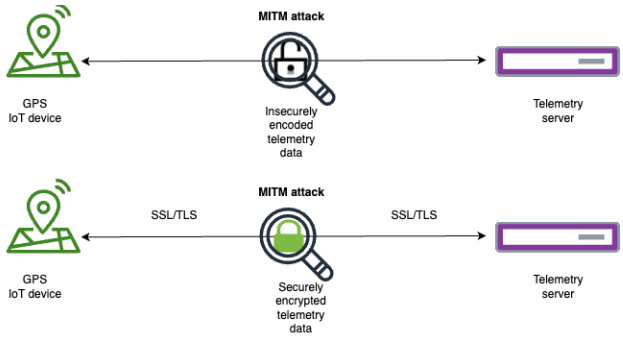

Figure 13. Unsecured / Secured Data Transit Diagram

In RSA encryption, keys are created using two primes, p and q, to compute n = pq, serving as the modulus for both encryption and decryption. The encryption key, e, is a coprime with $1 < e < \varphi(n)$, where $\varphi(n) = (p-1)(q-1)$, while the decryption key, d, is e's modular inverse mod φ(n). A plaintext, m, is encrypted to ciphertext, c, via $c \equiv m^e \pmod n$ and decrypted by $m \equiv c^d \pmod n$. In SSL/TLS, telemetry data, m, is encrypted with the server's public key (n, e) to yield c, which, without the private key d, can't be decrypted if intercepted. The server decrypts c back to m using its private key (n, d). RSA's security relies on the computational difficulty of factoring n, the product of large primes p and q [4], [7].

The choice of encryption solutions is driven by various factors and motivations. One significant aspect is the objective of ensuring robust data security, encompassing confidentiality and integrity, particularly when handling sensitive information. The selection process takes into consideration established encryption algorithms with proven security standards and suitability for the existing system architecture. Additionally, computational efficiency is a key consideration to minimize processing overhead. Alternative encryption solutions could involve asymmetric encryption algorithms like RSA or elliptic curve cryptography, each offering distinct trade-offs between security and performance. Furthermore, alternative protocols such as IPsec and VPN could be explored, tailored to specific system requirements. Ultimately, the selection of encryption solutions is contingent upon the desired security level, available resources, and compatibility with the prevailing infrastructure. This decision-making process aligns with scientific principles by considering established cryptographic principles, system requirements, and the pursuit of optimal security measures.

V. CONCLUSIONS AND FUTURE WORK

In the context of IoT devices, it was important to implement encryption algorithms instead of encoding ones to prevent Man-in-the-Middle attacks. The preliminary results can be used by the manufacturer of the GPS tracking devices (Teltonika in this particular case) to improve the security of its products. Moreover the companies working in intelligent transportation systems can get reliable and effective protection against attacks.

We plan to extend the solution with encryption mechanisms using AES-256 and security solutions (IPS/ IDS), investigating also Artificial Intelligence-based schemes for countermeasures. These features are needed to mitigate Denial-of-Service and MITM attacks. These could be aligned also with the use of new generation of devices and 5G/ B5G technologies to protect against cyber threats.

REFERENCES

[1] M. Won, "Intelligent Traffic Monitoring Systems for Vehicle Classification: A Survey," in *IEEE Access*, vol. 8, pp. 73340-73358, 2020, doi: 10.1109/ACCESS.2020.2987634.

[2] S.P. Potluri, B.P. Rao, S.K. Deshpande, "Security challenges in the IoT world: a comprehensive survey", *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 7, pp. 2923-2944, 2020.

[3] N.M. Naveen, N.K. Shet, M.K. Jayanthy "A Framework for Mitigating Man-in-the-Middle Attacks in IoT", *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1663-1668, 2018.

[4] B. Seshasayee, S. Machiraju, R. Bhatia, "Securing IoT Devices with TLS and DTLS", *IEEE Communications Magazine*, vol. 55, no. 10, pp. 78-85, 2017.

[5] J.J. Kim and D.G. Han, "AI-Driven IoT Security: Challenges, Solutions, and Future Directions", *IEEE Access*, vol. 9, pp. 16477-16491, 2021.

[6] S. Sudharsan, S.S. Sathya "A Comparative Analysis of IoT Security Threats and Mitigation Techniques", *International Journal of Advanced Science and Technology*, vol. 28, no. 10, pp. 224-231, 2019.

[7] J. Abreu, M. Alves, M. Almeida, L. Nunes, "Secure Communication for IoT Devices: TLS/DTLS with AES Encryption", *8th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pp. 121-126, 2017.

[8] G.R. Andreica, L. Bozga, D. Zinca and V. Dobrota, "Denial of Service and Man-in-the-Middle Attacks Against IoT Devices in a GPS-Based Monitoring Software for Intelligent Transportation Systems," *2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, Bucharest, Romania, 2020, pp. 1-4, doi: 10.1109/RoEduNet51892.2020.9324865.

[9] M. Strumskienė , T. Martišauskas "IoT Security: Teltonika Case Study", 2019, *International Scientific Conference "Society. Integration. Education"*, pp. 468-478, 2019.

[10] H. Dwivedi, "IoT Security: Protecting Your Connected World", *Apress*, 2018.

[11] S.K. Upadhyay, R.P. Mahapatra, A.K. Nayak "IoT Security: A Review on Threats, Vulnerabilities and Countermeasures", *International Journal of Innovative Technology and Exploring Engineering*, vol. 10, no. 9S, pp. 133-140, 2021.

[12] Teltonika, "Teltonika Data Sending Protocols", *Teltonika Docs*, 2023. [Online]. Available: https://wiki.teltonika-gps.com/view/Teltonika_Data_Sending_Protocols [Accessed: February 15, 2023].

[13] P. Pedamkar, "Diffie Hellman Key Exchange Algorithm" *Educba*, 2023. [Online]. Available: https://www.educba.com/diffie-hellman-key-exchange-algorithm/ [Accessed: February 10, 2023].

[14] Quantum Backdoor, "Symmetric-key algorithm in Cryptography", *Medium*, 2020. [Online]. Available: https://medium.com/@.Qubit/symmetric-key-algorithm-in-cryptography-3d839bba8613.

[15] K. Robinson, "What is a public key cryptography?", *Twilio Blog*, 2018. [Online]. Available: https://www.twilio.com/blog/what-is-public-key-cryptography [Accessed: January 4, 2023].