A Frame-Based Architecture for Enhanced Secure IoT Communication with Ascon-128a

Huu-Tu Hoang, Duc-Hung Le*

Faculty of Electronics and Telecommunications, The University of Science, Vietnam National University, Ho Chi Minh City, Vietnam
(*)Email: ldhung@hcmus.edu.vn

Abstract. IoT devices often face secure communication challenges due to limited computational power, thus increasing vulnerability to threats. In this paper, we propose a new secure communication framework using structured data frames for transmission in perceptual, gateway, and data processing layers. We integrated within the frame-based methodology the lightweight authenticated encryption algorithm Ascon-128a to improve end-to-end security. By applying the frame structure into our proposed IoT system, the tests show that: 99.99% Packet Delivery Ratio (PDR) over 120 minutes, 49.91 ms average latency, and 100% rejection of 70,000 replayed packets. Ascon-128a also achieved a better performance compare to AES on the ARM Cortex-M4 processor core. This work will provide a scalable and efficient approach, balancing security and performance for resource-limited IoT ecosystems.

Keywords: IoT security - Lightweight cryptography - Secure communication - Embedded systems

1 Introduction

The Internet of Things (IoT) has emerged as one of the leading drivers of today's technology, transforming healthcare, industrial automation, and smart infrastructure with cutting-edge solutions [1]. The majority of IoT devices nowadays, nevertheless, operate with low-processing power, energy-constrained, and unreliable network connectivity, which could simply reveal vulnerable security threats [2]. It not only increases the probability of data packet loss during transmission but also make the devices face many cyber threats, including unauthorized data interception, malicious data modification, and replay attacks, where valid data transmissions are captured and resent [3]. Traditional cryptographic algorithms, such as AES and RSA, have been proven to be effective in high-resource environments, but they may be inefficient on low-power IoT devices due to high computational costs [4]. As mentioned in [5], IoT security must be addressed at multiple layers, with particular emphasis on communication links between the perceptual layer (sensors and microcontrollers) and gateways, as well as between gateways and backend servers. The perceptual layer often uses serial communication protocols like UART and transmits data with simple structure;

hence, it can increase vulnerability to packet loss as well as unauthorized tampering [6]. Simultaneously, the gateway-to-server communication requires strong encryption and reliable key management to ensure data integrity and prevent unauthorized access [7].

To address these challenges, this paper presents a communications framework particularly designed for low-resource IoT systems with a focus on the utilization of structured data frames. Rather than send raw data, this method mandates that all exchanged data between each layers must be encapsulated within structured frames prior to any transmission. The frame structure contains some mandatory fields that must be verified before accessing the payload. In our design, we utilized the lightweight authenticated encryption algorithm Ascon-128a to generate a 16-byte authentication tag for each frame, thus ensuring the integrity and authenticity of the transmitted data. To add extra security, we introduce a novel safe counter mechanism to dynamically reset sequence numbers whenever the brute-force attack is detected. With the combination of 32-bit sequence number and synchronized safe counter updates, we reduced the success probability of such attacks to 0.000000023%. The validation of our IoT health monitoring system using STM32 microcontrollers at the perceptual layer and ESP32 gateways shows some impressive performance, proving the effectiveness of the framework. 99.99% packet delivery ratio for 120 minutes of continuous operation with 49.91 ms average transmission latency. Our system also achieved 100% rejection on 70,000 malicious packets during a brute-force attack on the server. However, some disadvantage in bandwidth utilization due to protocol overhead needs to be noticed and requires further research in the future. Our results suggest a new approach for IoT deployments with limited resources, giving a strong foundation for secure and reliable communication in emerging applications.

2 Background

2.1 The Ascon-128a Algorithm

We selected Ascon-128a as a lightweight authenticated encryption algorithm for its ability to provide both data confidentiality and integrity in resource-constrained IoT devices and embedded systems [8]. On ARM Cortex-M4 device without AES-NI accelerator, the Ascon-128a scheme performs better in term of runtime compared to AES-based method in software-only implementation. It also demonstrates resistance to side-channel attacks, making it very suitable for secure IoT communication. The algorithm using a key length $k \leq 160$, data block size r, and round numbers a and b, with parameters for authenticated encryption detailed in Table 1.

Ascon-128a takes a 128-bit secret key K, 128-bit nonce N, associated data A, and plaintext P of arbitrary lengths as inputs, outputting an authenticated ciphertext C (matching P's length) and a 128-bit tag T to verify A and the encrypted message. It operates in three phases—initialization (mixing K, N,

Table 1: Parameters for recommended authenticated encryption

Bit Size				Rounds	
Key	Nonce	Tag	Data block	p^a	p^b
128	128	128	64	12	6

and parameters over 12 rounds), data processing (encrypting P over 6 rounds after handling A), and finalization (generating the 128-bit tag) using a 320-bit internal state for security.

2.2 The Elliptic Curve Diffie-Hellman (ECDH) Key Exchange

To keep an IoT system secure, a strong key exchange method is very essential. The secret key used for encrypting and decrypting data is a critical part of the system. A weak key exchange or secret key generation method can lead to the secret key being exposed or easily brute-force if the security level is too low. In our framework, we have chosen ECDH [9] as the key exchange method because it can create a shared secret over untrusted network safely. ECDH has been shown to provide the same level of security as Diffie-Hellman with much smaller key size [10], making it really suitable for IoT devices. Although ECDH requires a higher computational cost compared to timestamp-based or static key methods often used in low-resource devices, selecting suitable curve will make this trade-off worthwhile to ensure long-term protection against attacks.

In ECDH, each party generates a private key $d \in [1, n-1]$ and computes a public key $Q = d \cdot G$, where G is a base point. For parties like the gateway (d_A, Q_A) and server (d_B, Q_B), the shared secret is:

$$d_A \cdot Q_B = d_B \cdot Q_A = d_A \cdot d_B \cdot G.$$

We use the *sect163k1* curve over GF(2^{163}) recommended by SECG for its efficient arithmetic in the Koblitz family [11], reducing computational overhead. With an 80-bit security level, it suits resource-limited IoT devices for secure key exchange [12].

3 Proposed Method

3.1 IoT System Overview

We applied the frame structure to our specific IoT system. The IoT system was built based on the 4-layer architecture [13]. The perceptual layer collects sensor data, and the gateway layer transmits it to the server. The system operates in two main phases: key exchange and data transmission, all data when transmitting will be encrypted and authenticated.

In the key exchange phase, the gateway and server use the Elliptic Curve Diffie-Hellman (ECDH) to form the shared secret key. Because the data encryption is performed at the perceptual layer, the gateway must transmit this key

to the perceptual layer. To prevent this key from leaking, the gateway will first encrypt this key with a pre-shared key before transmission. The perceptual layer will decrypt the shared secret key, and use it for data encryption, ensuring that the key remains protected against any interception between the gateway and perceptual layers.

For the data transmission phase, the gateway will send a trigger command to the perceptual layer. After receiving the trigger, the perceptual layer will encrypt its collected data, construct the data frame, and then send it to the gateway. The gateway will check all the fields in the data frame; if it matches, the gateway will then forward it to the server. Once the server can successfully decrypt the data, it will send an acknowledgment (ACK) command to the gateway, completing a secure transmission cycle.

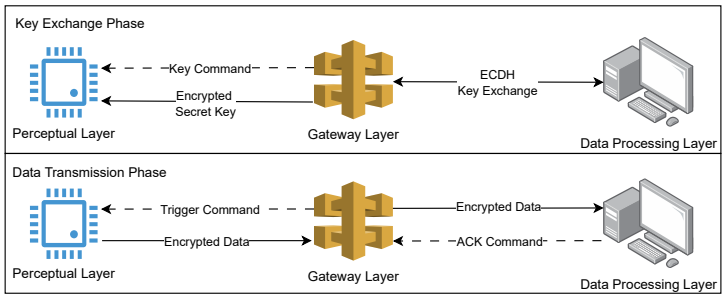

Fig. 1: The IoT system

3.2 Frame Structures for Secure Communication

This section details these frame structures, which are employed for all communications flows illustrated in Figure 1. Each frame consists of a common structure initiated by a 2-byte Start of Frame (SOF) to denote the transmission start. A 4-byte *Identifier ID* follows to indicate the target device and it was defined at the perceptual layer. The frame ends with a 2-byte *End of Frame* (EOF). The *data field* varies depending on the communication's purpose, with additional fields included as needed to improve security and robustness. For clarity, frames are denoted by their source and target; for instance, a frame from gateway to server is referred to as GS. For associated data (AAD), AAD_{data} is calculated and used during the data transmission phase, while AAD_{key} is predefined and used in the key exchange phase. We divided the frame into two types: those for communication between the perceptual and gateway layers, and those between the gateway and server layers.

3.2.1 Communication Between Perceptual and Gateway Layers

Figure 2 illustrated the frame use for transmission from gateway to perceptual layer. In this frame, we added a 1-byte *packet type* to specify whether the frame is a trigger command (GP2) to request data or a command to prepare the perceptual layer for receiving and decrypting a new secret key (GP1). For trigger packet type, the data field contains only a 16-byte AAD_{data}. For the key packet type, the data field includes a 16-byte *Nonce* and AAD_{data}, the perceptual layer will use this *Nonce* and AAD_{key} to decrypt the secret key. This is followed by a 1-byte *secret key length* and the encrypted secret key payload. A 16-byte *authentication tag* (Auth Tag), generated by Ascon-128a during secret key encryption, confirms no tampering occurred in transmission. The AAD_{data} in GP1 and GP2 is derived from specific fields of the GS2 frame, which the perceptual layer uses for data encryption. The GS2 frame is constructed before the gateway sends the GP1 or GP2 frame to the perceptual layer. Upon receiving the PG1 frame from the perceptual layer, the gateway extracts the *data field* and *Auth Tag*, incorporates it into the GS2 frame for transmission to the server:

$$AAD_{data} = \mathrm{SOF}_2 \,||\, \text{Identifier ID}_4 \,||\, \text{Sequence Number}_4 \,||\, \text{Nonce}_4 \,||\, \mathrm{EOF}_2$$

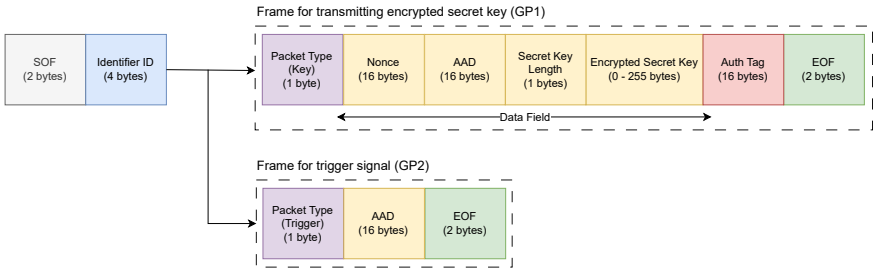

Fig. 2: The frame structure of transmission from gateway to perceptual

On receiving a trigger command from the gateway, the perceptual layer sends a PG1 frame (Figure 3) with encrypted data to the gateway. It includes a 16-byte *Nonce* for server-side decryption, an Ascon-128a-generated *Auth Tag* for authentication, and a 2-byte *Cyclic Redundancy Check* (CRC) to detect transmission errors in the encrypted data.

3.2.2 Communication Between Gateway and Data Processing Layers

The frame structure for communication between the gateway and server is depicted in Figure 4. Three packet types was implemented for this communication.

The GS1 frame is used for the key exchange process between the gateway and server, keeping common fields. Its *data field* contains a 16-byte *Nonce* for

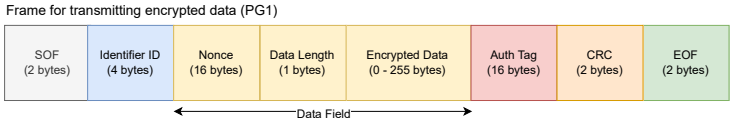

Fig. 3: The frame structure of the data transmission from perceptual to gateway

server to compute the *Auth Tag*, followed by a 1-byte *public key length* and an n-byte *public key payload*. A 16-byte *Auth Tag* is produced using Ascon-128a with a pre-shared key and AAD_{key} established between the gateway and server. To optimize computation, the plaintext is set to null, because we only need the *Auth Tag* for authentication, reducing processing time for both gateway and server while ensuring the public key remains untampered.

For data transmission from gateway to server, the GS2 frame is employed, it has the same structure as GS1 but adding a 4-byte *sequence number* to counter replay attacks [14]. This allows the server to detect and reject any re-sent packets with mismatched sequence numbers. The *data field* includes a 16-byte *Nonce*, a 1-byte *payload length*, an n-byte *encrypted payload* and a 16-byte *Auth Tag*, all extracted from the perceptual layer's PG1 frame. Once the server has received GS2, it will extract the necessary fields to compute AAD_{data} for decryption.

The SG1 frame is used for commands from the server to the gateway. In this work, two commands are implemented: an *ACK* command to confirm successful transmission and a *sequence update* command to update the sequence number if a brute-force attempt on the sequence number is detected.

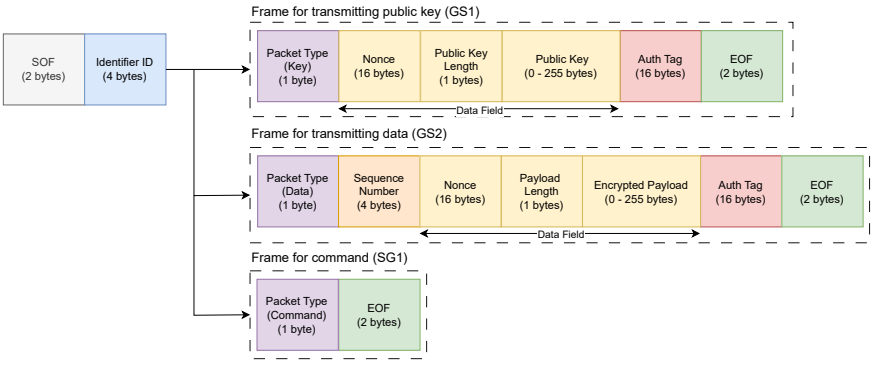

Fig. 4: The frame structure of transmission between gateway and server

3.2.3 The Frame Parsing Process

The frame parsing process is illustrated in Figures 5 and 6. This process is applied to all frames in the system. Here, we depict the parsing of the GS2 frame, the most complex frame, noting that other frames follow a similar process. The process begins by receiving the data frame from the target and then storing it in an internal buffer, prepare for the parsing process (Figure 5). Each byte in the buffer will be fed into the parsing function (Figure 6). Based on predefined conditions we set for each state, the parsing function will evaluate whether the conditions are met to transition to the next state. If the final state is reached correctly, the buffer is successfully parsed, finishing the frame parsing process and proceeding to the next step. The decryption process occurs at the Auth Tag state, where the encrypted payload is decrypted to produce the actual data and a 16-byte *Auth Tag*.

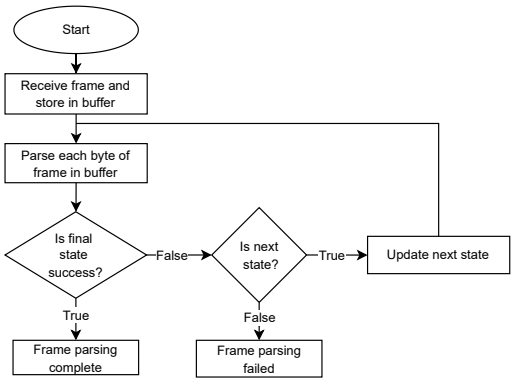

Fig. 5: The frame parsing process

4 Implementation Results and Analysis

4.1 Experimental Setup

To validate our framework, we built an IoT-based heart rate monitoring system. The STM32F411VET6 microcontroller (160 MHz) was used at the perceptual layer for gathering 3-byte data, including heart rate, SPO2, and temperature. An ESP32 module (240 MHz) aggregates and securely transmits the data to a Linux server. For the key exchange process, we set an interval of one minute each time for better observing how the overhead of ECDH can affect the system; it can be adjusted in practice based on specific computational, energy, or security requirements.

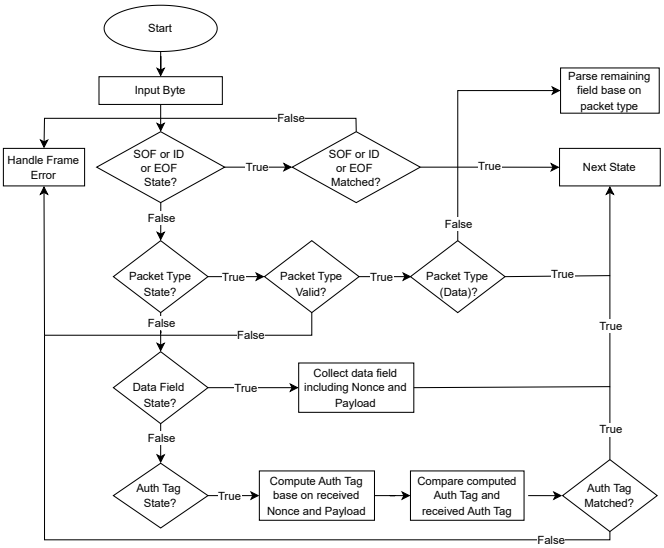

Fig. 6: The parsing function

4.2 Results and Analysis

We evaluated transmission over 120 minutes, with results shown in Figures 7 and 8, focusing solely on the performance of two constrained devices at the perceptual and gateway layers. At the perceptual layer, the STM32 required an average of 0.634 ms to construct the PG1 frame. Transmitting the PG1 frame via UART to the gateway averages 5.58 ms. The encryption process, using the lightweight Ascon-128a algorithm, averaged 0.328 ms, it easy to see that Ascon-128a has a significant improvement over the AES-based implementation using the X-CUBE-CRYPTOLIB on the Cortex-M4 processor, which averaged 0.936 ms. Parsing the GP1 frame averaged 0.374 ms, while the GP2 frame took 0.023 ms, the difference caused by secret key decryption and *Auth Tag* computation in GP1. At the gateway level, the ESP32 parsed the PG1 frame from the STM32 in an average of 0.35 ms. Constructing the GP1 frame took an average of 0.57 ms, while the GP2 frame required 0.014 ms. Transmitting the GP2 frame via UART has an average latency of 9.01 ms. The key exchange process, which includes generating the public/private key pair and computing the shared secret, averaged 335.08 ms and 335.46 ms, respectively. However, this operation was performed periodically by our setup, minimizing its impact on overall performance. The construction of GS1 and GS2 frames averaged 0.28 ms and 0.033 ms, respectively. Although GS2 has more fields than GS1, it only extracts the data field and *Auth Tag* from PG1 without additional computation, making it faster than GS1, which computes the authentication tag during frame construction. The ACK command from the gateway, confirming successful delivery, has an average latency of 29.58

ms, including the time for the GS frame to reach the server and the ACK to return. However, this latency depends on network speed and server performance, so it might have a slight difference on other systems. The total transmission time, defined as the duration from when the STM32 receives a trigger from the ESP32 until an ACK command is returned, averages 49.91 ms.

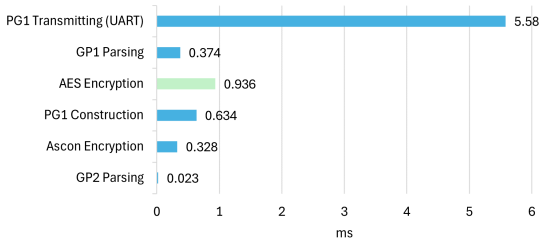

Fig. 7: Execution Time Analysis of STM32 Processes

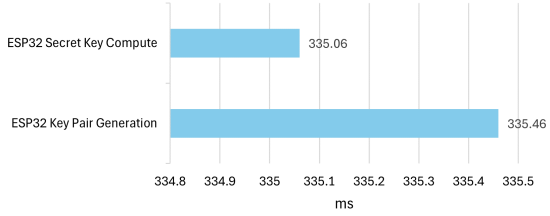

Fig. 8: Execution Time Analysis of ESP32 Operations and Key Exchange

During a 120-minute test session, the gateway transmitted a total of 147,682 packets. We define a successful transmission as a packet that is rightly encrypted and the server can decrypt it correctly. The PDR is a key metric for assessing system integrity and is computed every minute based on the following formula:

$$\text{PDR} = \frac{N_{\text{successful}}}{N_{\text{total}}} \quad (N_{\text{successful}} : \text{successful packets/min},\ N_{\text{total}} : \text{total packets/min})$$

Figure 9 presents the number of packets transmitted per 60-second interval over a 120-minute duration, with an average of 1231 packets per interval. Figure 10 depicts PDR evaluated every minute in the same 120-minute period. The system maintained a PDR of 100% for the majority of the intervals, with the remaining intervals maintaining stable values between 99.8% and 99.975%. The

overall average PDR was 99.99%, demonstrating high reliability and stability in data transmission throughout the experimental evaluation.

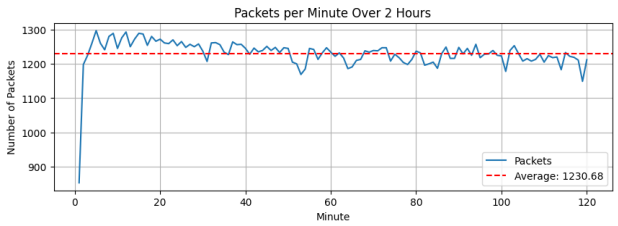

Fig. 9: Packets Transmitted Per Minute

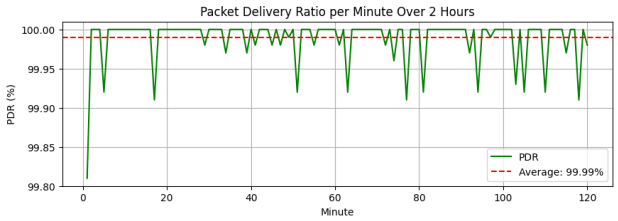

Fig. 10: Packet Delivery Ratio Per Minute

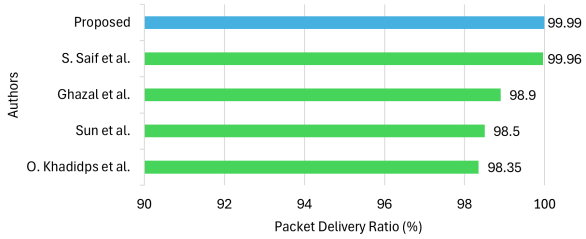

Fig. 11: Comparison with related works in terms Packet Delivery Ratio.

Figure 11 shows a comparative analysis of our framework's performance against recent IoT communication and security studies. Our approach achieves the highest packet delivery ratios, with an average PDR of 99.99% over a 120-minute test transmitting 147,682 packets. We compare it with four notable

works. Saif et al. (2024) [16] use a Timestamp-based Secret Key Generation (T-SKG) scheme with DES for IoT healthcare, achieving a PDR of 99.92–99.97% depending on the DES mode (ECB, CBC, CTR). Ghazal et al. (2021) [17] introduce the IoT-AIS for healthcare, leveraging WSNs and AI-driven encryption to secure patient data, and achieve a 98.90% PDR. Khadios et al. (2022) [18] present a Probabilistic Super Learning model with Random Hashing and ECC encryption, reaching a 98.35% PDR. Sun et al. (2021) [19] propose a mutual authentication scheme for IoMT device-to-server communication, achieving an average PDR of around 98.5% in a 30–60 minute simulation.

5 Throughput and Payload Efficiency Analysis

We evaluate the system's *throughput* and *payload efficiency* based on the GS2 frame, which is the most frequently used frame for data transfer over the network via MQTT. The total size of the GS2 frame is the sum of all fields and was calculated as follows:

$$\begin{aligned}\text{Size}_{GS2} &= \text{SOF (2)} + \text{ID (4)} + \text{Packet Type (1)} + \text{Seq. Number (4)} \\ &\quad + \text{Nonce (16)} + \text{Payload Length (1)} + \text{Encrypted Payload (3)} \\ &\quad + \text{Auth Tag (16)} + \text{EOF (2)} \\ &= 49 \text{ bytes}\end{aligned}$$

In our system, we only collect 3-byte data for transmission, resulting in the encrypted payload is 3-byte. Noticing that the total size of the GS2 frame can be varied if the system collects more data to transmit. As we measured in Section 4.2, the system sends approximately 147,562 GS2 frames in 120-minute, excluding key exchange packets. Therefore, the *throughput* is calculated as:

$$\text{Throughput} = \left(\frac{147,562 \times 49}{120 \times 60}\right) \times 8 \approx 8,033.93 \text{ bps}$$

In this evaluation, each GS2 frame consists of 49 bytes, of which only 3 bytes are used for actual payload data. This results in a *payload efficiency* of $3/49 \approx 6.12\%$. As a consequence, although the system achieves relatively high throughput, the *goodput*—defined as the rate of useful data successfully transmitted is significantly lower:

$$\text{Goodput} = \left(\frac{147,562 \times 3}{120 \times 60}\right) \times 8 \approx 491.87 \text{ bps}$$

From the above results, it is clear that the overhead when transmitting data is relatively high, the low *payload efficiency* leads to poor utilization of the available bandwidth. However, we can improve the *payload efficiency* by increasing the payload size. That means, if the system collects more data to transmit, the efficiency will be increased. As illustrated in Table 2, larger payloads can give a better ratio of useful data to total frame size, which improves bandwidth utilization.

Table 2: Payload Efficiency with Different Payload Lengths

Payload Length	Total Frame Size	Efficiency
3 bytes	49 bytes	6.12%
10 bytes	56 bytes	17.85%
20 bytes	66 bytes	30.30%
50 bytes	96 bytes	52.08%

6 Security Analysis and Threat Mitigation

This section evaluates the robustness of the proposed IoT framework against common network-layer threats. We enumerate some typical attacks and how our system reacts against them below:

1. With our frame design, it makes sure that only devices within the same ecosystem can communicate. Other devices are unable to send data because of mismatches in the SOF, Identifier ID, and EOF fields, which are unique.
2. To secure the key exchange process, we incorporate a 16-byte Auth Tag when transmitting the public key. With the auth tag, it makes sure that even attackers try to send a fake public key to disrupt secret key computation. The system can be aware and reject the fake key immediately.
3. The ECDH key exchange uses the *sect163k1* curve with an 80-bit security level; it satisfies the NIST recommendation in security level [20]. Moreover, the secret key will be updated after a specific interval, making brute-force attacks on the secret key infeasible.
4. The 4-byte *sequence number* in the GS2 frame allows the server to detect and reject any retransmitted packet due to a mismatch in sequence number. Thus, we can ensure that no replay attacks can disrupt our system.
5. We also consider a scenario where attackers try to brute-force all 4-byte sequence number to bypass the system; therefore, a *safe counter* mechanism was used to counter such an attack. We set up a secret *safe counter* that increments with each transmission. If the server rejects packets due to sequence number mismatches beyond a set threshold, it sends a command via the SG1 frame, instructing the gateway to update the sequence number using that *safe counter* after T_{update} rejections (Algorithm 1). By setting T_{rejected} to 10, the probability of successfully brute-forcing the sequence number is approximately 0.000000023%, making such an attack virtually infeasible.
6. We tested our system by sending 70,000 malicious packets. It is obvious that if the attackers don't know the frame structure, they cannot send any data to our system. So in the test, we assume that the attacker knows the frame structure for each communication, aimed at disrupting data flow. The results still show that the server rejected all 70,000 packets, achieving a 100% rejection rate, demonstrating the system's robust protection against such attacks.

Algorithm 1 Safe Counter Update

Require: 32-bit Safe Counter SC, Secret Key K, Threshold T_{rejected}, Rejected Packet Count R, Current Sequence Number SN

$SC = (SC + ((SC \ll 3) \oplus (SC \gg 2) \oplus 7)) \mod 2^{32}$
if $R \geq T_{\text{rejected}}$ **then**
  $SN = (SC \oplus K) \mod 2^{32}$
  $R = 0$
end if
return SN

7 Conclusion

This paper presents a secure IoT communication framework based on Ascon-128a, ECDH key exchange, and structured frames, providing 99.99% PDR in 120 minutes, 100% filtering out 70,000 malicious packets, and a mean transmission delay of 49.91 ms. Ascon-128a has been shown to be suitable for most IoT device by having higher performance compared to AES 78.4% on ARM Cortex-M4 processor core. The future study will focus on ECDH overhead reduction through the optimization of the key exchange interval and continue improving the security of sequence numbers. Also, the high overhead of the protocol is still a problem that needs to be further researched to optimize the bandwidth utilization. Finally, we will evaluate the system's energy consumption to assess how well it can scale in real-world applications.

References

1. Coutinho, R. W. L., & Boukerche, A. (2020). Modeling and Analysis of a Shared Edge Caching System for Connected Cars and Industrial IoT-Based Applications. *IEEE Transactions on Industrial Informatics, 16*(3), 2003–12.
2. Satapathy, U., Mohanta, B., Jena, D., & Sobhanayak, S. (2018). An ECC-Based Lightweight Authentication Protocol for Mobile Phones in Smart Homes. *IEEE ICIINFS Conference Proceedings,* 303-308. https://doi.org/10.1109/ICIINFS.2018.8721417
3. Iwuanyanwu, U., Oyewole, O., Fakeyede, O., Okeleke, E., & Apeh, J. (2023). IoT Device Security Risks: A Comprehensive Overview and Mitigation Strategies. *Journal of Technology & Innovation, 3*, 38-43. https://doi.org/10.26480/jtin.01.2023.38.43
4. Ekwueme, C., Adam, I., & Dwivedi, A. (2024). Lightweight Cryptography for Internet of Things: A Review. *EAI Endorsed Transactions on Internet of Things.* https://doi.org/10.4108/eetiot.5565
5. Yousuf, T., Mahmoud, R., Aloul, F., & Zualkernan, I. (2015). Internet of Things (IoT) Security: Current Status, Challenges and Countermeasures. *International Journal for Information Security Research, 5*, 608-616.
6. Delvai, M., Eisenmann, U., & Elmenreich, W. (2003). Intelligent UART Module for Real-Time Applications. *WISES Conference Proceedings*, 177-185.
7. Rekha, Shashi & Lingala, Thirupathi & Renikunta, Srikanth & Gangula, Rekha. (2021). Study of security issues and solutions in Internet of Things (IoT). Materials Today: Proceedings. 80. 10.1016/j.matpr.2021.07.295.

8. Dobraunig, C., Eichlseder, M., Mendel, F., & Schläffer, M. (2021). Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *Journal of Cryptology, 34*(33). https://doi.org/10.1007/s00145-021-09398-9
9. Haakegaard, R., & Lang, J. (2015). The Elliptic Curve Diffie-Hellman (ECDH).
10. Siddiqui, R., & Mehrotra, S. (2011). A Review on Elliptic Curve Cryptography for Embedded Systems. *International Journal of Computer Science and Information Technology (IJCSIT),* 3, 84–103. https://doi.org/10.5121/ijcsit.2011.3307
11. National Institute of Standards and Technology. (2013). Digital Signature Standard (DSS). *Federal Information Processing Standards Publication.* Retrieved from https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.186-4.pdf
12. Certicom Research. (2010). Standards for Efficient Cryptography SEC 2: Recommended Elliptic Curve Domain Parameters. Version 2.0. Contact: Daniel R. L. Brown (dbrown@certicom.com).
13. Khan, H., & Singh, P. (2021). Issues and Challenges of Internet of Things: A Survey. *Journal of Informatics Electrical and Electronics Engineering (JIEEE), 2*(1), 1-8.
14. Awais, M., & Iqbal, J. (n.d.). Layered Architecture of Internet of Things: A Review.
15. Standards for Efficient Cryptography Group (SECG). (n.d.). SEC 2: Recommended Elliptic Curve Domain Parameters. Retrieved from https://www.secg.org/sec2-v2.pdf
16. Saif, S., Sohail, M., Das, P., Biswas, S., Khan, S., Haq, M. A., & Kovtun, V. (2024). A secure data transmission framework for IoT-enabled healthcare. *Heliyon, 10*, e36269. https://doi.org/10.1016/j.heliyon.2024.e36269
17. Ghazal, T. (2021). Internet of Things with Artificial Intelligence for Health Care Security. *Arabian Journal for Science and Engineering, 48.* https://doi.org/10.1007/s13369-021-06083-8
18. Khadidos, A., Shitharth, A., Khadidos, A., Sangeetha, K., & Alyoubi, H. (2022). Healthcare Data Security Using IoT Sensors Based on Random Hashing Mechanism. *Journal of Sensors, 2022*, 1-17. https://doi.org/10.1155/2022/8457116
19. Sun, J., Khan, F., Li, J., Alshehri, M. D., Alturki, R., & Wedyan, M. (2021). Mutual Authentication Scheme for the Device-to-Server Communication in the Internet of Medical Things. IEEE Internet of Things Journal, 1–1. doi:10.1109/jiot.2021.3078702
20. Mahto, D., & Yadav, D. (2017). RSA and ECC: A Comparative Analysis. *International Journal of Applied Engineering Research, 12*, 9053-9061.