

Secure MQTT Authentication with Dynamic Salt and TOTP for IoT Environments

A. Renuka Devi¹, M. Chandra Mohan²

¹Research Scholar, JNTU, Hyderabad, renuka.adibhatla@gmail.com

²Professor of CSE, JNTU, Hyderabad, c_miryala@jntuh.ac.in

With the rapid growth of internet of Things (IoT), ensuring the security of communication protocols has become increasingly critical. Message Queuing Telemetry Transport (MQTT), a lightweight messaging protocol, is widely adopted in IoT systems due to its low overhead and efficiency. However, its simplicity comes at the cost of security vulnerabilities, particularly in authentication mechanisms. This paper investigates the integration of Time-Based One-Time Password (TOTP) authentication into MQTT to address these challenges. In place of the regular username and password authentication, we used the hash of the salted TOTP as password along with the username for authentication, which increases the security. By leveraging TOTP in the MQTT protocol, we aim to enhance security while maintaining its lightweight characteristics, ensuring that it remains suitable for resource-constrained IoT environments. This provides a scalable and secure solution for IoT environments, offering enhanced protection for sensitive data transmission without significantly compromising performance. This approach is particularly valuable for IoT applications in critical sectors such as healthcare, smart cities, and industrial automation, where security and reliability are paramount.

Keywords: internet of Things, Message Queuing Telemetry Transport, Time-Based One-Time Password.

1. Introduction

The rapid expansion of the Internet of Things (IoT) has revolutionized various sectors, connecting physical devices to the internet and enabling real-time data exchange. IoT devices are now ubiquitous in critical applications such as healthcare monitoring, industrial automation, smart homes, and connected vehicles. As IoT ecosystems continue to scale, securing communication between devices has become a pressing concern, particularly

because of the resource-constrained nature of most IoT hardware. Efficient, lightweight communication protocols are essential to maintaining the functionality and scalability of IoT networks. One such protocol that has gained significant traction is Message Queuing Telemetry Transport (MQTT)[1].

MQTT is a lightweight, publish-subscribe messaging protocol designed for low-bandwidth, high-latency environments, making it ideal for IoT devices that have limited computational power and memory. Its simplicity and efficiency have made it a popular choice for IoT applications. However, MQTT's security features are limited, as it was originally designed with minimal overhead to support devices with constrained resources [2]. The default MQTT protocol uses simple username-password authentication, often transmitted over unsecured channels, which leaves it vulnerable to a variety of attacks, including credential theft, password replay, and man-in-the-middle (MITM) attacks.

In light of increasing threats to IoT networks, the need for stronger security measures in MQTT has become evident. Transport Layer Security (TLS) is often employed to secure MQTT communication, but it introduces significant computational and resource overhead, making it less feasible for constrained devices[3]. This creates a demand for lightweight yet effective authentication mechanisms that can enhance the security of MQTT without compromising performance or usability[4].

A promising solution to this challenge is Time-Based One-Time Password (TOTP) authentication. TOTP is a dynamic authentication mechanism that generates a new, time-sensitive password, based on a shared secret and the current time. This method significantly reduces the risk of password replay and brute-force attacks, as each password is valid for only a brief period. TOTP has been widely adopted in multi-factor authentication (MFA) systems but has not been extensively applied to IoT protocols like MQTT. Integrating TOTP into MQTT could provide a robust layer of security, offering protection against common attack vectors without introducing significant computational overhead or latency.

Two-factor authentication (2FA) is a security process that requires users to verify their identity using two distinct forms of identification before accessing an account. Typically, 2FA combines something the user knows (like a password) with something the user has (such as a smartphone or hardware token). By requiring both, 2FA makes unauthorized access much more difficult, even if one factor, like the password, is compromised.

In this research, we have used TOTP as password along with username in securing MQTT communication in IoT environments. This TOTP-based scheme that can be easily integrated into the MQTT protocol, enhancing its security while preserving its lightweight characteristics.

By addressing the security limitations of MQTT with a lightweight TOTP solution, this paper contributes to the growing body of research aimed at securing IoT networks. The proposed method not only strengthens the protocol against emerging cyber security threats

but also maintains its efficiency, ensuring its continued viability for large-scale IoT deployments in industries where security, performance, and scalability are crucial.

2. Literature Review

Yang et al.[5] highlighted that while MQTT is efficient for resource-constrained devices, its lack of inherent security measures such as encryption and dynamic authentication makes it highly susceptible to attacks, especially in sensitive IoT deployments like healthcare and industrial automation. Özlem ,Seker et al.[6] in their work that used a pre-shared secret and HOTP to authenticate and the client which wants to publish/subscribe. In the work [7] by PK Panda et al., the authors leverage elliptic curve cryptography (ECC) to achieve efficient and secure authentication, balancing security needs with the constraints of IoT hardware resources. Authors of [8] used HMAC-based one-time password (HOTP) for authentication and AES was used for payload encryption. Mosquitto auth plugin is added to provide access control with ACL by storing user name and password encrypted on the database.

While OTP scheme based on Elliptic curve cryptography based security was implemented by [9], [10], [11], the authors of [12],[13], [14] used OTP with block chain to further enhance the security.

Few studies have directly addressed the application of TOTP in securing IoT protocols, but there has been growing interest in using dynamic authentication mechanisms in IoT. Authors in [15],[16],[17] proposed using TOTP for securing IoT devices in environments with constrained computational resources. They demonstrated that TOTP could be implemented efficiently without introducing significant latency or overhead, making it suitable for IoT systems. Their study, however, did not focus on specific protocols such as MQTT.

In relevant studies by [18],[19],[20], the authors have discussed integrating two-factor authentication in MQTT for IoT networks. Although their focus was on hardware token-based authentication, they highlighted the need for lightweight, time-sensitive solutions such as TOTP in securing MQTT communication. They showed that adding dynamic authentication can effectively thwart attacks targeting static password systems, though they acknowledged that performance trade-offs must be carefully managed.

While the above studies have contributed significantly to securing IoT communication, several gaps remain, particularly in integrating lightweight authentication mechanisms like TOTP in MQTT. Most of the existing research has focused on general security enhancements for MQTT or explored TOTP in broader contexts such as user authentication. However, a comprehensive evaluation of TOTP's efficiency in securing MQTT without overwhelming resource-constrained devices is still lacking. Despite its benefits, TOTP faces challenges including synchronization issues between client devices and servers.

3. Proposed Methodology

System Components:

1. MQTT Clients (Publisher and Subscriber): Devices or applications that need to authenticate before publishing/subscribing.
2. MQTT Broker: The central server responsible for routing messages and handling authentication.
3. Salt Server: A dedicated server responsible for generating and distributing the salt to both the broker and MQTT clients.

Clients and brokers generate OTPs by appending the salt and system time, then the hash is computed and this computed hash is passed as password along with the username for authentication to the broker. The broker, on receiving the user name and this hash, checks with the generated hash. If both are same, it establishes the connection, otherwise rejects the device and notifies the same.

$TOTP = \text{SHA512}(\text{salt} + \text{system time in minutes})$. Authentication is done by sending this TOTP along with the username to the broker as shown in Fig 1.

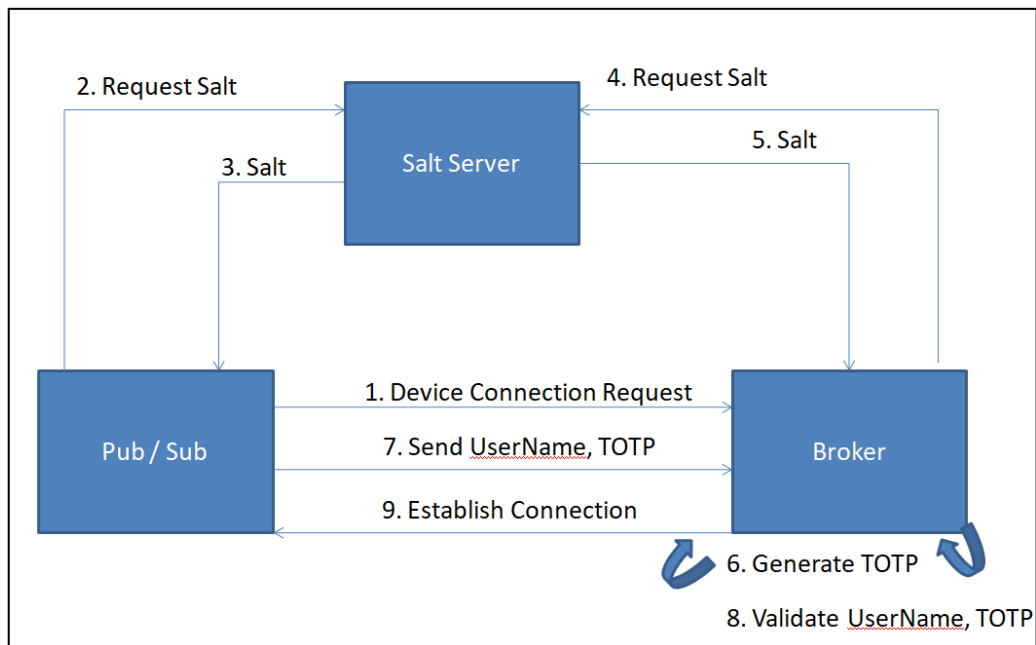


Fig 1: Publisher/Subscriber authentication with Username, TOTP

3.1 Implementation Steps

The implementation of secure MQTT connections through OTP-based authentication involves several key steps.

Step 1: Initial Setup. Each MQTT client, whether publisher or subscriber, is first assigned a unique shared secret that is securely stored on the broker and used in the OTP generation process. Alongside this, a Salt Server is configured to provide a dynamic salt to both the client and broker. While the salt does not directly influence the TOTP (Time-Based One-Time Password) generation, it enhances security by working in conjunction with the TOTP secret.

Step 2: Salt Generation and Distribution. When a client initiates a connection to the MQTT broker, both the client and broker request the salt from the Salt Server. This ensures that both parties use the same salt to generate the OTP. The salt, which is generated using a random number generator is then transmitted by the Salt Server to both the client and broker and is valid only for a specific session or time window.

Step 3: OTP Generation. On the client side, upon receiving the salt from the Salt Server, the MQTT client generates the OTP by first appending the salt and timestamp to the shared secret and current system time (Unix timestamp). The combined value is then hashed, typically using SHA-256. The broker, following the same procedure, receives the salt, combines the shared secret, salt, and system time, and generates the OTP. This OTP from the broker is then compared with the OTP sent by the client to verify the authentication.

Step 4: Authentication Process. In this phase, the client initiates the MQTT connection by sending an authentication request to the broker, which includes the client ID, the generated TOTP.. Upon receiving this, the broker retrieves the shared secret associated with the client, requests the same salt from the Salt Server, and generates an OTP following the shared procedure. The broker then checks if its generated OTP matches the client's OTP.

Step 5: Re-authentication for Continuous Sessions. If a client remains connected for an extended period, the broker periodically prompts re-authentication, requiring a new salt and OTP. The Salt Server periodically regenerates a new salt after each session ensuring that OTP generation remains dynamic and unpredictable for ongoing sessions, thus enhancing security over time.

This layered approach, combining shared secrets, dynamic salt, and TOTPs, establishes a secure framework for MQTT client-broker interactions.

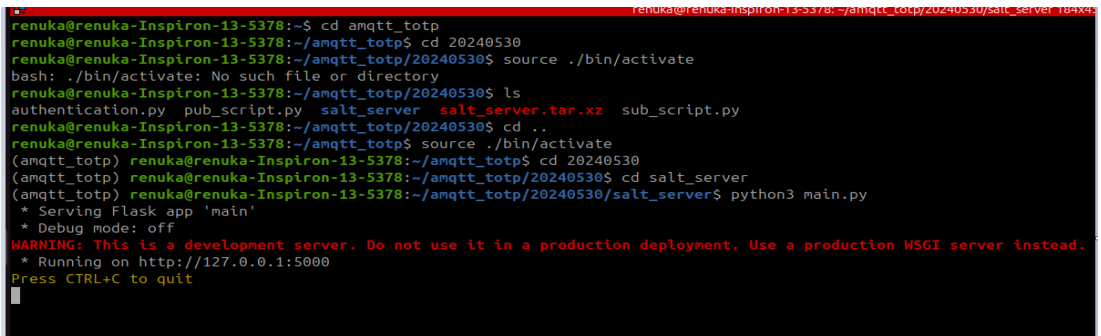
This approach includes a salt server that provides a dynamic salt to both the MQTT broker and clients (publishers/subscribers). The OTP is generated using the shared secret, the salt, and the current system time, and hash of the result is used along with the username for authentication thus preventing the man in the middle attack.

This dynamic OTP system improves security by ensuring that the authentication is resistant to replay attacks and other common security threats while remaining lightweight for resource-constrained IoT devices.

4. Results

If the TOTP's match, the client is authenticated, and the broker allows it to publish or subscribe to the requested topics otherwise the broker denies the connection and sends an authentication failure message to the client.

In our experiment, Salt server is done by setting up a Flask web server that generates and distributes a salt to MQTT clients and the broker as shown in the figure below.



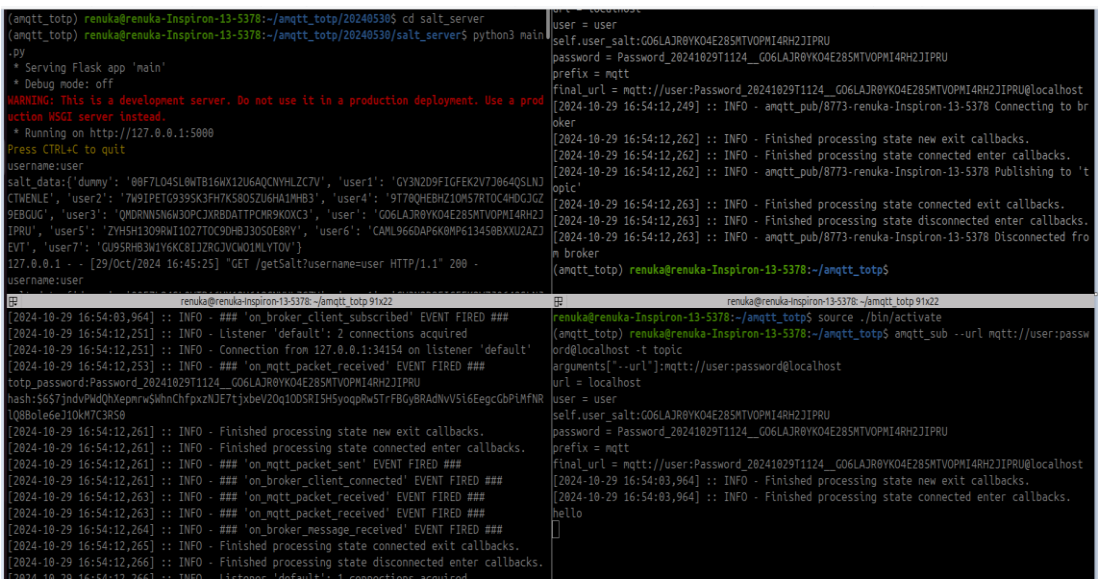
```

renuka@renuka-Inspiron-13-5378:~/amqtt_totp/20240530/salt_server$ ls
renuka@renuka-Inspiron-13-5378:~/amqtt_totp$ cd 20240530
renuka@renuka-Inspiron-13-5378:~/amqtt_totp/20240530$ source ./bin/activate
bash: ./bin/activate: No such file or directory
renuka@renuka-Inspiron-13-5378:~/amqtt_totp/20240530$ ls
authentication.py pub_script.py salt_server salt_server.tar.xz sub_script.py
renuka@renuka-Inspiron-13-5378:~/amqtt_totp/20240530$ cd ..
renuka@renuka-Inspiron-13-5378:~/amqtt_totp$ source ./bin/activate
(amqtt_totp) renuka@renuka-Inspiron-13-5378:~/amqtt_totp$ cd 20240530
(amqtt_totp) renuka@renuka-Inspiron-13-5378:~/amqtt_totp/20240530$ cd salt_server
(amqtt_totp) renuka@renuka-Inspiron-13-5378:~/amqtt_totp/20240530/salt_server$ python3 main.py
* Serving Flask app 'main'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit

```

Fig 2: Salt server is running and ready to provide salt

The salt server provides both the broker and clients salt when requested. This salt is combined with the system time then hash is calculated. This hash is the TOTP. This TOTP is sent as password along with the username for authentication. With successful connection with the broker, the subscriber and publisher connect to the broker. The subscriber receives the message “Hello” which is sent by the publisher is shown in Fig 3.



```

(amqtt_totp) renuka@renuka-Inspiron-13-5378:~/amqtt_totp/20240530$ cd salt_server
(amqtt_totp) renuka@renuka-Inspiron-13-5378:~/amqtt_totp/20240530/salt_server$ python3 main.py
* Serving Flask app 'main'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
username=user
salt_data={'dummy': '06f7L04SL6Wt816Wx12U6AQCNVhZC7V', 'user1': 'CV3N2D9F1GFEX2V7J0640SLN3
C2NE4LE', 'user2': '7W91PEIG0939K3FH7K5805ZU6HAIH83', 'user4': '9170QEBHZ10MS7RTOC4HDJCZ
9EBGUG', 'user3': 'QDRNNSNG30PCXVB0ATPCW9KXOC3', 'user': 'C06LAJRBVYK04E285MTVOPMI4RH2J
IPRU', 'user5': 'ZYHSH1309PM1027TOCDH83J05DE8RY', 'user6': 'CAML96G0AP6K0NP613450B4XU2AZJ
EVT', 'user7': 'Q95SRHB3W16KCB1JZRGJVO1M1LYTOV'}
127.0.0.1 - - [29/Oct/2024 16:45:25] "GET /getSalt?username=user HTTP/1.1" 200 -
username=user
renuka@renuka-Inspiron-13-5378:~/amqtt_totp/20240530/salt_server$ python3 main.py
user = user
self.user_salt:G06LAJRBVYK04E285MTVOPMI4RH2JIPRU
password = Password_20241029T1124_C06LAJRBVYK04E285MTVOPMI4RH2JIPRU
prefix = mqtt
final_url = mqtt://user:Password_20241029T1124_C06LAJRBVYK04E285MTVOPMI4RH2JIPRU@localhost
[2024-10-29 16:54:12,249] :: INFO - amqtt_pub/8773-renuka-Inspiron-13-5378 Connecting to br
oker
[2024-10-29 16:54:12,262] :: INFO - Finished processing state new exit callbacks.
[2024-10-29 16:54:12,262] :: INFO - Finished processing state connected enter callbacks.
[2024-10-29 16:54:12,262] :: INFO - amqtt_pub/8773-renuka-Inspiron-13-5378 Publishing to 't
opic'
[2024-10-29 16:54:12,263] :: INFO - Finished processing state connected exit callbacks.
[2024-10-29 16:54:12,263] :: INFO - Finished processing state disconnected enter callbacks.
[2024-10-29 16:54:12,263] :: INFO - amqtt_pub/8773-renuka-Inspiron-13-5378 Disconnected fro
m broker
(amqtt_totp) renuka@renuka-Inspiron-13-5378:~/amqtt_totp
renuka@renuka-Inspiron-13-5378:~/amqtt_totp/20240530$ python3 main.py
[2024-10-29 16:54:03,964] :: INFO - ### 'on_broker_client_subscribed' EVENT FIRED ###
[2024-10-29 16:54:12,251] :: INFO - Listener 'default': 2 connections acquired
[2024-10-29 16:54:12,251] :: INFO - Connection from 127.0.0.1:34154 on listener 'default'
[2024-10-29 16:54:12,253] :: INFO - ### 'on_mqtt_packet_received' EVENT FIRED ###
totp_password:Password_20241029T1124_C06LAJRBVYK04E285MTVOPMI4RH2JIPRU
hash:56S7JndvPwQkXepnrwShnChfxp2N3E71xbeV2Qo10DSR15HYoqpw5TfBqyBRAdNvV516EegcGpIMfMR
1QB8olefe310M7C3R58
[2024-10-29 16:54:12,261] :: INFO - Finished processing state new exit callbacks.
[2024-10-29 16:54:12,261] :: INFO - Finished processing state connected enter callbacks.
[2024-10-29 16:54:12,261] :: INFO - ### 'on_mqtt_packet_sent' EVENT FIRED ###
[2024-10-29 16:54:12,261] :: INFO - ### 'on_broker_client_connected' EVENT FIRED ###
[2024-10-29 16:54:12,263] :: INFO - ### 'on_mqtt_packet_received' EVENT FIRED ###
[2024-10-29 16:54:12,263] :: INFO - ### 'on_mqtt_packet_received' EVENT FIRED ###
[2024-10-29 16:54:12,264] :: INFO - ### 'on_broker_message_received' EVENT FIRED ###
[2024-10-29 16:54:12,265] :: INFO - Finished processing state connected exit callbacks.
[2024-10-29 16:54:12,265] :: INFO - Finished processing state disconnected enter callbacks.
[2024-10-29 16:54:12,266] :: INFO - Listener 'default': 1 connections acquired
[2024-10-29 16:54:12,266] :: INFO - Finished processing state new exit callbacks.
[2024-10-29 16:54:03,964] :: INFO - Finished processing state connected enter callbacks.
hello

```

Fig 3: publish/subscribe mechanism using TOTP

5. Conclusion

The implementation of a salt-based OTP authentication system for MQTT publish-subscribe mechanisms represents a significant advancement in securing IoT communications. By incorporating a Salt Server that dynamically provides salts to both the MQTT broker and clients, the system ensures that each OTP is unique and synchronized across the network. The use of a shared secret, combined with the salt and the current system time, generates a time-based one-time password (TOTP) that is both secure and practical. This approach enhances the authentication process by ensuring that OTPs are transient and cannot be reused, thus mitigating the risk of replay attacks. Additionally, the computational requirements for generating and verifying OTPs are minimal, making this system well-suited for resource-constrained IoT devices. This solution enhances the security and provides a scalable solution that can be adapted to various deployment scenarios. Future improvements could include integrating encryption for message payloads and exploring multi-factor authentication methods to further bolster the system's security posture. Overall, this methodology provides a balanced approach to secure MQTT communications, addressing key vulnerabilities and offering a practical solution for modern IoT environments.

References

1. Yassein, Muneer Bani, et al. "Internet of Things: Survey and open issues of MQTT protocol." 2017 international conference on engineering & MIS (ICEMIS). IEEE, 2017.
2. Hassija, Vikas, et al. "A survey on IoT security: application areas, security threats, and solution architectures." *IEEE Access* 7 (2019): 82721-82743.
3. Prantl, Thomas, et al. "Performance impact analysis of securing mqtt using tls." *Proceedings of the ACM/SPEC international conference on performance engineering*. 2021.
4. Ahmed, Waleed Kareem, and Rana Saad Mohammed. "Lightweight authentication methods in IoT: Survey." *2022 International Conference on Computer Science and Software Engineering (CSASE)*. IEEE, 2022.
5. Yang, Y., Wu, L., Yin, G., Li, L., & Zhao, H. (2017). A survey on security and privacy issues in Internet-of-Things. *IEEE Internet of things Journal*, 4(5), 1250-1258.
6. Şeker, Özlem, Gökhan Dalkılıç, and Umut Can Çabuk. "MARAS: Mutual authentication and role-based authorization scheme for lightweight Internet of Things applications." *Sensors* 23.12 (2023): 5674.
7. Panda, Prabhat Kumar, and Sudipta Chattopadhyay. "A secure mutual authentication protocol for IoT environment." *Journal of Reliable Intelligent Environments* 6.2 (2020): 79-94.
8. Yerlikaya, Özlem, and Gökhan Dalkılıç. "Authentication and authorization mechanism on message queue telemetry transport protocol." *2018 3rd International conference on computer science and engineering (UBMK)*. IEEE, 2018.
9. Lohachab, Ankur. "ECC based inter-device authentication and authorization scheme using MQTT for IoT networks." *Journal of Information Security and Applications* 46 (2019): 1-12.
10. Shivraj, V. L., et al. "One time password authentication scheme based on elliptic curves for Internet of Things (IoT)." *2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW)*. IEEE, 2015.
11. Hammi, B., Fayad, A., Khatoun, R., Zeadally, S., & Begriche, Y. (2020). A lightweight ECC-based authentication scheme for Internet of Things (IoT). *IEEE Systems Journal*, 14(3), 3440-3450.
12. Buccafurri, Francesco, and Celeste Romolo. "A blockchain-based OTP-authentication scheme

- for constrained IoT devices using MQTT." Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control. 2019.
13. Apriani, D., Devana, V. T., Sagala, A. P., Sunarya, P. A., Rahardja, U., & Harahap, E. P. (2023, May). Security using blockchain-based otp with the concept of iot publish/subscribe. In AIP Conference Proceedings (Vol. 2808, No. 1). AIP Publishing.
 14. Li, D., Peng, W., Deng, W., & Gai, F. (2018, July). A blockchain-based authentication and security mechanism for IoT. In 2018 27th International Conference on Computer Communication and Networks (ICCCN) (pp. 1-6). IEEE.
 15. Hsieh, C. F., & Chang, C. K. (2021, November). The security method in mqtt protocol for internet of things. In 2021 International Conference on Technologies and Applications of Artificial Intelligence (TAAI) (pp. 280-284). IEEE.
 16. Bangare, Pallavi S., and Kishor P. Patil. "Enhancing MQTT security for internet of things: Lightweight two-way authorization and authentication with advanced security measures." *Measurement: Sensors* 33 (2024): 101212.
 17. Sofian, Asyura Binti, et al. "Enhancing Authentication Security: Analyzing Time-Based One-Time Password Systems." *International Journal of Computer Technology and Science* 1.3 (2024): 56-70.
 18. Sadri, Mohammad Javad, and Maryam Rajabzadeh Asaar. "An anonymous two-factor authentication protocol for IoT-based applications." *Computer Networks* 199 (2021): 108460.
 19. Li, Shanshan, et al. "A secure two-factor authentication scheme from password-protected hardware tokens." *IEEE Transactions on Information Forensics and Security* 17 (2022): 3525-3538.
 20. Pahlevi, Rizka Reza, et al. "Secure two-factor authentication for iot device." 2022 10th International Conference on Information and Communication Technology (ICoICT). IEEE, 2022.