**Bulletin of Electrical Engineering and Informatics**
Vol. 15, No. 3, June 2026, pp. 2360~2372
ISSN: 2302-9285, DOI: 10.11591/eei.v15i3.11907 

# Ultra-lightweight hybrid authentication for MQTT/MQTT-SN internet of thing security

**Nabeel Alassaf[1], Selvakumar Manickam[1], Ammar Odeh[2], Mohammed Anbar[1]**
[1]Cybersecurity Research Centre, Universiti Sains Malaysia (USM), Penang, Malaysia
[2]Department of Computer Science, Faculty of King Hussein School of Computing Sciences, Princess Sumaya University for Technology, Amman, Jordan

**Article Info**

***Article history:***

Received Nov 19, 2025
Revised Mar 10, 2026
Accepted Mar 31, 2026

***Keywords:***

Cryptography
Internet of things
security lightweight
Message queuing
telemetry transport
Message queuing
telemetry transport for
sensor networks
Pre-shared key authentication

**ABSTRACT**

The rapid growth of internet of thing (IoT) has increased the need for secure communication among resource-constrained devices using lightweight protocols such as message queuing telemetry transport (MQTT) and message queuing telemetry transport for sensor network (MQTT-SN). Traditional certificate-based solutions introduce significant computational and memory overhead for low-power devices. This paper proposes the hybrid lightweight protocol (HLP), a certificate-free approach combining elliptic-curve key exchange, hash-based message authentication code (HMAC)-based authentication, and ChaCha20-Poly1305 encryption. HLP uses pre-shared keys to reduce handshake complexity while maintaining confidentiality, integrity, and mutual authentication across MQTT and MQTT-SN environments. A Python-based implementation using paho-mqtt was evaluated in a constrained-device testbed. Experimental results show that HLP achieves lower handshake latency (~20–24 ms) and reduced bandwidth overhead (~130 bytes) compared with elliptic curve Diffie-Hellman ephemeral-pre-shared key (ECDHE-PSK) and elliptic curve Diffie-Hellman ephemeral-elliptic curve digital signature algorithm (ECDHE-ECDSA), while still supporting forward secrecy. These findings demonstrate that HLP is an efficient and practical solution for securing IoT communications on constrained devices.



***Corresponding Author:***

Selvakumar Manickam
Cybersecurity Research Centre, Universiti Sains Malaysia (USM)
Penang, Malaysia
Email: selva@usm.my

## 1. INTRODUCTION

The development of internet of thing (IoT) has changed the world in many aspects. We have now entered an age in which inanimate objects can communicate with one another, exchange data, and perform automated tasks. IoT is increasingly adopted across various domains, including smart homes, industrial control systems, healthcare, transportation, and environmental monitoring [1], [2]. As of 2025, it is estimated that over 30 billion devices will be connected globally, generating massive amounts of data and requiring robust, real-time, and secure communication mechanisms [3], [4].

One of the core challenges of securing IoT communications lies in the resource-constrained nature of IoT devices [5], [6]. These devices often operate on limited battery power, have modest computational capabilities, and have minimal memory. Consequently, implementing conventional security protocols such as transport layer security (TLS) with certificate-based authentication mechanisms (e.g., Rivest–Shamir–Adleman (RSA) or elliptic curve digital signature algorithm (ECDSA)) introduces significant overhead in

terms of processing power, memory usage, and energy consumption. Such overhead is impractical for devices such as sensors, actuators, and embedded controllers commonly used in IoT deployments [7].

Figure 1 shows a hierarchical classification of IoT protocols, organized into three main categories based on their functional roles within the IoT communication stack. At the top level, the figure distinguishes between network layer protocols, transport & messaging protocols, and application & device management protocols. Network layer protocols such as 6LoWPAN, Zigbee, LoRaWAN, and NB-IoT enable low-power, long-range, and low-bandwidth communication suitable for IoT devices [8], [9]. Transport & messaging protocols such as message queuing telemetry transport (MQTT), message queuing telemetry transport for sensor network (MQTT-SN), CoAP, and AMQP facilitate efficient, lightweight message exchange between devices and cloud services. Finally, application & device management protocols, including HTTP/HTTPS, LwM2M, OMA-DM, and RESTful APIs, handle device configuration, firmware updates, and application-level interactions. This classification highlights the modular and layered architecture of IoT ecosystems, which offers varying flexibility and scalability options across deployment contexts [10], [11].

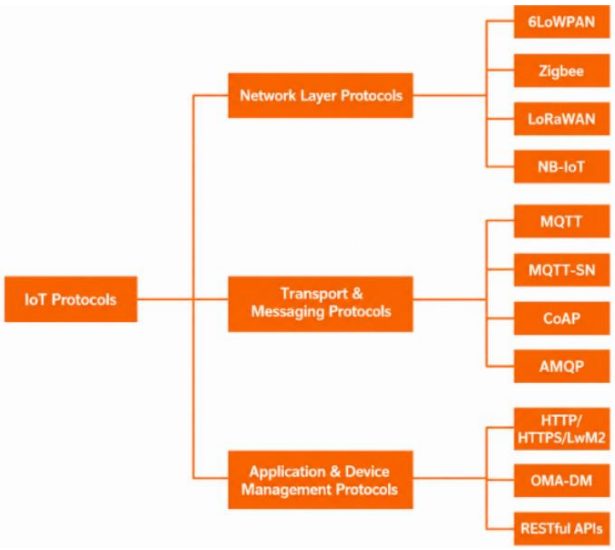


Figure 1. Classification of IoT communication protocols

The MQTT protocol is rapidly becoming the most dominant inter-networking communication standard for IoT applications owing to its lightweight publish/subscribe messaging model, low network-layer requirements, and a small header size [12], [13]. However, the default security mechanisms of any MQTT implementation remain weak because they rely solely on plain-text tokens and username/password pairs, which lead to eavesdropping, replay attacks, and unauthorized access to MQTT sessions. While TLS can improve MQTT security, its deployment is unsuitable for most use cases in constrained settings [14], [15].

Most of the MQTT 'security' adaptations have focused on lightweight TLS implementations. The most common adaptation is the use of the elliptic curve Diffie-Hellman ephemeral (ECDHE) for session key generation [16]. These approaches, along with the elliptic curve digital signature algorithm (ECDSA) for authentication, are widely used because they significantly reduce the size of the session key and the associated crypto overhead in RSA/remote public key infrastructures (PKI) scenarios [16], [17].

MQTT-SN is specifically designed for wireless sensor networks and provides an alternative for scenarios where TCP-based MQTT is too heavy. Despite its potential, MQTT-SN's security remains under-researched, with most existing solutions either relying on simple symmetric encryption or ignoring the protocol entirely. The lack of integrated, lightweight, and secure authentication mechanisms for MQTT-SN represents a significant gap in current research [18], [19].

Furthermore, current studies often limit their evaluation to either simulation environments or simplistic performance metrics, such as CPU utilization and message latency [20]. Few works consider real-world end-to-end communication, including client- and broker-side authentication, session key derivation, and message encryption and decryption under realistic network conditions. This gap in comprehensive, practical implementation undermines the applicability of proposed security mechanisms in actual IoT ecosystems, where devices are deployed across diverse environments with varying network conditions and energy constraints [21].

This paper proposes a novel hybrid lightweight protocol (HLP) for IoT security over MQTT and MQTT-SN to address these critical gaps. The HLP combines symmetric cryptography, ephemeral elliptic-curve key exchange, and hash-based message authentication code (HMAC)-based authentication to deliver a highly efficient, secure communication mechanism for resource-constrained devices. Unlike traditional approaches, HLP eliminates the need for certificates by leveraging a pre-shared key (PSK) model that authenticates devices using keyed hashes. This reduces cryptographic complexity and enables scalable and secure integration across a wide range of IoT platforms.

The key contributions of this paper are as follows:

- We propose HLP, a novel certificate-free, hybrid authentication and encryption scheme for MQTT and MQTT-SN protocols, suitable for ultra-constrained IoT devices.
- We design a full end-to-end communication framework that includes mutual authentication, session key generation, and secure payload transmission using lightweight cryptographic primitives.
- We implement a proof-of-concept testbed to simulate real-world deployment using Python cryptographic libraries and measure end-to-end metrics, including authentication overhead, encryption latency, and resource usage.
- We compare HLP against existing lightweight protocols and show improved performance in computational cost, bandwidth consumption, and resistance to common attack vectors.
- We provide the first integrated solution that extends secure session management to MQTT and MQTT-SN, addressing a major limitation in current research that largely ignores MQTT-SN's security.

By addressing the dual challenges of efficiency and security in IoT communications, the proposed HLP scheme paves the way for broader adoption of secure MQTT and MQTT-SN protocols in real-world IoT ecosystems. A balance between cryptographic strength and implementation feasibility guides the design choices. The proposed model makes the protocol practical for many IoT applications, including those deployed in low-power and low-connectivity environments.

HLP’s novelty lies in its MQTT/MQTT-SN oriented integration of PSK/HMAC mutual authentication, ephemeral ECDH forward secrecy, and AEAD payload protection, together with an end-to-end implementation and measured evaluation in the same experimental setting—rather than in proposing new cryptographic primitives.

## 2. RELATED WORK

This section presents the methodological foundations of our proposed HLP and describes relevant prior studies that inspired and informed our design. We extended and examined each study's context, contributions, and limitations. A summary comparison table is also provided at the end.

Roldán-Gómez *et al.* [22] provided an in-depth study on the security posture of the MQTT-SN protocol by identifying seven distinct attack scenarios that could be practically implemented. They highlighted the unique vulnerabilities in resource-constrained networks and proposed a novel threat-detection model that can identify unknown attack patterns. While the approach demonstrated strong detection capability, it did not incorporate cryptographic mechanisms or encryption strategies, limiting its protection scope to monitoring and alerts. Andy *et al.* [23] performed a comprehensive vulnerability assessment on MQTT broker implementations, including Mosquitto and HiveMQ. Their analysis of public CVEs revealed persistent issues, including denial-of-service vulnerabilities and injection attacks. They recommended proactive improvements in broker design but did not experimentally evaluate these proposals in constrained environments or implement a tested defense framework. Andy *et al.* [23] conducted hands-on experiments to demonstrate attack feasibility on various MQTT configurations. The work detailed attack vectors such as eavesdropping, unauthorized publishing, and denial-of-service, particularly when brokers were misconfigured or lacked TLS support. The study was purely diagnostic and, along with the diagnosis, did not provide any countermeasures or mitigation frameworks.

Chien *et al.* [24] proposed a lightweight authentication scheme for MQTT that seamlessly integrates into existing APIs. Their model uses a secure two-phase handshake based on symmetric keys and session tokens, reducing computational cost compared to SSL. Despite its lightweight nature and compatibility, the scheme does not ensure encrypted payloads or forward secrecy, leaving message confidentiality unprotected.

Dinculeană and Cheng [25] proposed a lightweight alternative to symmetric encryption for MQTT-based IoT communication. Their method uses HMAC signatures as identifiers of known values, enabling efficient matching through a lookup table on the receiver side. This significantly reduced encryption overhead on devices like the Omega2+. However, the model's scalability and security are constrained by the necessity to precompute and securely store large hash tables, and key compromise can undermine confidentiality.

Table 1 summarizes the most relevant prior works addressing MQTT and MQTT-SN security for IoT systems. It outlines the methods various researchers use to mitigate known vulnerabilities, improve protocol

security, or enhance the efficiency of authentication and encryption processes. The summarized methods include threat modeling, lightweight authentication, CVE analysis, and hash-based HMAC optimizations, each tailored to improve performance or resilience in constrained IoT environments.

As reflected in the table, most studies substantially improved detection capabilities or performance optimizations. However, a recurring limitation across several approaches is the lack of comprehensive coverage that combines authentication, encryption, and forward secrecy. This observation reinforces the need for an integrated, lightweight, and hybrid security framework like the proposed HLP, which addresses the trade-offs between security and resource consumption while remaining compatible with MQTT and MQTT-SN protocols.

Table 1. Summary of security enhancements for MQTT/MQTT-SN protocols

| Reference | Method | Improvement | Limitation |
|---|---|---|---|
| Roldán-Gómez *et al.* [22] | Threat modeling and attack simulation on MQTT-SN | Introduced seven MQTT-SN attack scenarios and a threat detector to enhance resilience | Focused on attack detection without addressing cryptographic countermeasures |
| Andy *et al.* [23] | Survey and CVE analysis on MQTT broker implementations | Identified real-world vulnerabilities across open-source brokers and advocated for proactive security design | Relied primarily on retrospective analysis and lacked experimental validation |
| Chien *et al.* [24] | Two-phase API-compatible MQTT authentication scheme | Enhanced authentication using a secure handshake protocol without modifying the MQTT core | Focused on authentication only and lacked message encryption or session confidentiality |
| Dinculeană and Cheng [25] | Value-to-HMAC mapping using hash-based keyed authentication | Reduced encryption overhead by replacing traditional ciphers with precomputed HMAC mappings | Limited to small value ranges and susceptible to hash table reconstruction attacks if compromised |

From a formal cryptographic perspective, we consider an active network adversary who can eavesdrop, replay, delay, drop, and inject messages, including attempting man-in-the-middle attacks between the client and broker/gateway. Under this model, the protocol aims to provide mutual authentication, session key freshness, replay resistance, payload confidentiality/integrity, and forward secrecy via ephemeral ECDH, assuming the long-term PSK is securely stored, and endpoints can generate fresh nonces. To ground these claims in established practice, we align our discussion with relevant standards and RFCs, including NIST guidance on key management and widely used RFCs covering PSK-based key exchange, HMAC, X25519, and ChaCha20-Poly1305.

## 3. PROPOSED ALGORITHM

This section may be divided by subheadings. It should provide a concise, precise description of the experimental results, their interpretation, and the conclusions that can be drawn. This part outlines the methodology for designing and assessing the HLP, which aims to bolster the security of IoT communication over MQTT and MQTT-SN, and focuses on ‘how to do’ the specific tasks within the HLP framework. HLP aims to be a lightweight, yet secure, certificate-free communication protocol that also aligns with the limitations of IoT systems. Automated mutual trust is implemented using a methodology that includes symmetric cryptography, elliptic curve key exchange, and HMAC-based authenticated messages. RSA-based certificates and PKI, which are commonly used with IoT systems, are replaced by the HLP framework because of the RSA-free environment enabled by PSK, which offers security simplicity and scalability. This part addresses the adaptation of the MQTT and MQTT-SN protocols, their system realization, the architectural elements of the protocols, their operational phases, the cryptographic primitives used, and the protocols themselves.

The sequence diagram illustrates the full working cycle of HLP from the client side to the broker side. The process is initiated when the client combines a random nonce and ephemeral elliptic curve cryptography (ECC) key pairs, then completes the authentication token calculation using PSK. The sequence diagram illustrates the complete operation of the proposed HLP communication between the client and broker. The process begins when the client generates a random nonce and ephemeral ECC key pair, and computes an authentication token using the PSK. Subsequently, the client sends its credentials to the broker, which checks the authentication token. If the token is valid, the broker continues with the protocol; otherwise, it rejects the session. After fully validating the authentication token, the broker generates its own nonce, performs ECDH to compute a shared secret, derives the session key, and sends an authentication response to the client. The client checks the response, confirms mutual authentication by performing the same computation to derive the session key. Once both parties have mutually validated each other, subsequent communication between the client and broker will be encrypted with the ChaCha20-Poly1305 protocol to ensure confidentiality, integrity, and authenticity, while maintaining low computational requirements. Figure 2 illustrates the HLP client–broker handshake, including PSK/HMAC-based mutual authentication, ECDH key agreement, and ChaCha20-Poly1305 encrypted data exchange.

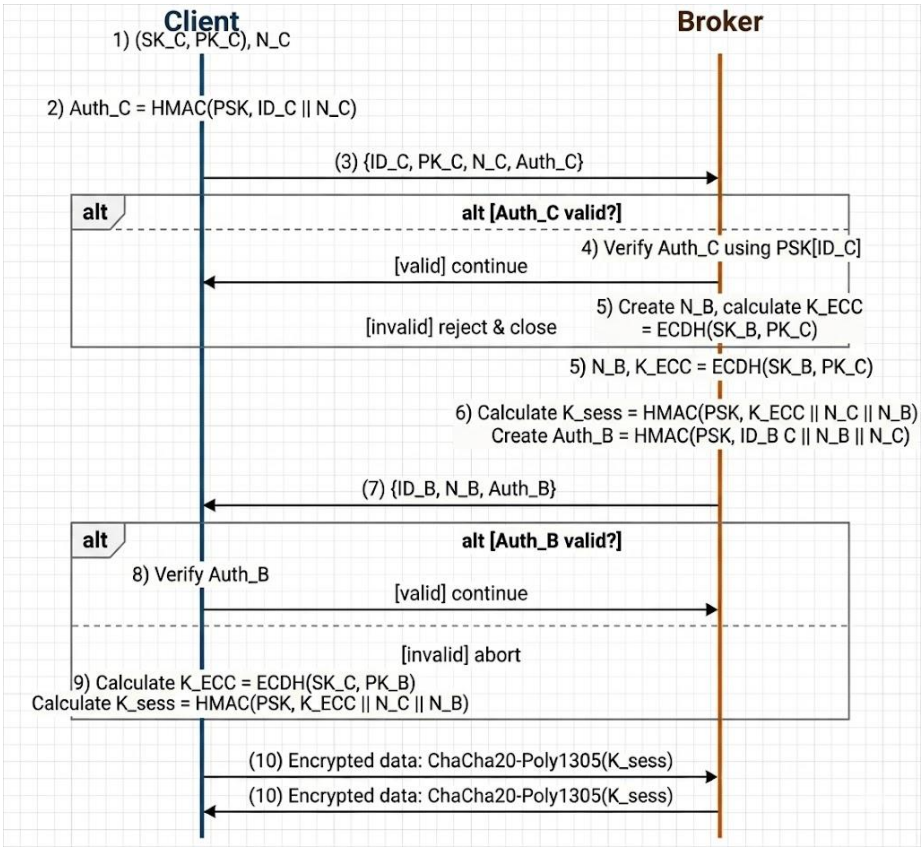


Figure 2. Sequence diagram of the HLP

### 3.1. Design rationale

The HLP was developed to address challenges in existing security approaches for MQTT and MQTT-SN. Although TLS, including RSA and ECDSA, is common practice, it is resource-intensive and requires excessive power and memory. This is especially true for microcontroller devices or sensor nodes that run from batteries. In addition, MQTT-SN is a protocol specifically designed for wireless sensor networks that uses UDP. In the past, MQTT-SN has been under-researched in terms of security, leading to significant gaps. The HLP provides mutual authentication, session key derivation using ECC, and message encryption using ChaCha20-Poly1305, an authenticated encryption algorithm that is faster and more resource-efficient than traditional methods.

### 3.2. Protocol architecture

HLP's architecture consists of two main components: the IoT client (e.g., sensor or actuator) and the MQTT/MQTT-SN broker. The client starts the action by attempting to establish secure communication with the broker, with both components pre-provisioned with a PSK at registration time. The PSK is then used to derive HMAC tokens and implicitly authenticate the exchanged nonces. ECC is used to create one-time key pairs that are discarded at the end for session security and forward secrecy. Next, the broker and client each selected a nonce during negotiation. The ECDH key exchange output is combined with both the client and broker nonces to derive a shared session key, K_sess. The session K_sess is then run through HMAC to produce a random, unpredictable key used to encrypt the payloads.

### 3.3. Step-by-step operational flow

In Figure 2, the operational flow of the proposed HLP is shown step-by-step, establishing a secure communication channel between an IoT client and an MQTT/MQTT-SN broker. Specifically, the IoT client generates a nonce and EC ephemeral key pair, then computes an authentication token using a PSK and HMAC. In the authentication token, the client's public key and nonce are sent to the broker. When the broker receives the message, it authenticates the client, generates its own nonce and ephemeral key pair, and then computes an ECDH key exchange. After ECDH is used to derive a shared key, the broker will also derive a session key

using HMAC with the shared secret and the two nonces. The broker then responds to the client with an authentication token, which contains the broker's authentication to the client and the brokers nonce. The client can verify the token and now construct the same session key. Once mutual authentication is complete and the session key is established, communication can proceed, encrypted with ChaCha20-Poly1305 to ensure the confidentiality, integrity, and authenticity of all messages. Figure 3 summarizes the HLP client–broker exchange: PSK/HMAC mutual authentication, ECDH session key derivation, and ChaCha20-Poly1305-encrypted communication.

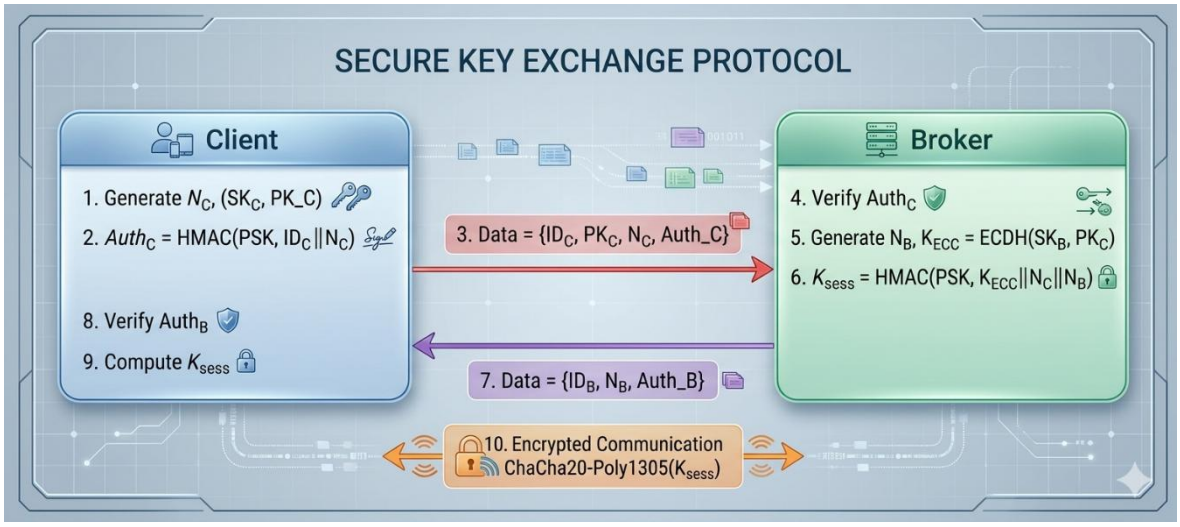


Figure 3. Operational flow of the proposed HLP

HLP is a lightweight authentication and encryption protocol for IoT devices using MQTT/MQTT-SN. It combines pre-shared keys (PSKs), ephemeral ECC-based key exchange, HMAC authentication, and ChaCha20-Poly1305 encryption to provide mutual authentication, secure session key generation, confidentiality, integrity, and perfect forward secrecy with low computational overhead suitable for resource-constrained devices.

HLP Algorithm

Step 1: the client generates a random nonce (N_C) and an ephemeral ECC key pair (SK_C and PK_C). These keys are intended only for this session.

Step 2: the client calculates an authentication token, Auth_C, using the PSK assigned during device registration: Auth_C = HMAC(PSK, ID_C || N_C).

Step 3: the client sends an encrypted request to the broker, which includes its ID (ID_C), public ECC key (PK_C), nonce (N_C), and the calculated Auth_C.

Step 4: the broker obtains the corresponding PSK using ID_C, then recalculates the HMAC to verify the Auth_C received.

Step 5: if Auth_C is verified, the broker generates its own random nonce (N_B), and calculates the ECDH shared secret: K_ECC = ECDH(SK_B, PK_C).

Step 6: the broker derives the session key as K_sess = HMAC (PSK, K_ECC || N_C || N_B). This key will be used to encrypt messages.

Step 7: the broker creates its authentication token Auth_B = HMAC(PSK, ID_B || N_B) and returns it to the client along with its ID (ID_B) and nonce (N_B).

Step 8: the client verifies Auth_B and computes the shared secret independently using its private key with the broker's public ECC key: K_ECC = ECDH(SK_C, PK_B).

Step 9: it then derives K_sess using the same procedure and confirms that both sides have mutually authenticated and can derive the same shared encryption key.

Step 10: all subsequent communication between the client and the broker is encrypted using ChaCha20-Poly1305(K_sess) to maintain confidentiality, integrity, and authenticity.

Both parties can periodically generate new ECC key pairs and repeat the above steps to prevent key reuse and support perfect forward secrecy.

### 3.4. Cryptographic primitives

HLP recommends lightweight cryptographic protocols that have earned recognition for their efficiency and security. The elliptic curve used for key exchange is Curve25519. Curve25519 provides strong 128-bit security with low CPU overhead. HMAC-SHA256 is used for authentication because it's simple, collision-resistant, and is widely supported. For payload encryption, the standard chosen is ChaCha20-Poly1305. ChaCha20-Poly1305 is an authenticated encryption algorithm that can provide confidentiality, integrity, and authenticity in a single operation. ChaCha20-Poly1305 also performs better in software-based environments and does not require specialized hardware acceleration, unlike AES-GCM.

### 3.5. Implementation environment

Implementing and testing the protocol used Python version 3.11. The 'cryptography' module was used for cryptographic operations, while MQTT communication was handled via the 'paho-mqtt' library. The simulations were conducted in a virtual testbed that emulated IoT devices with resource constraints (e.g., ESP32 or Raspberry Pi Zero W). Performance metrics, such as authentication latency, time to encryption, and memory usage, were gathered for both MQTT and MQTT-SN operation modes. The messaging was tested in a variety of message sizes and network conditions to simulate more realistic IoT deployment conditions.

### 3.6. Message queuing telemetry transport vs message queuing telemetry transport for sensor network adaptation

Although the MQTT protocol runs on top of TCP, providing ordered and reliable communication, MQTT-SN is designed to operate over UDP for low-power wireless sensor networks. The HLP protocol was modified by abstracting its primary capabilities at the application layer, allowing it to function directly with MQTT clients over TCP sockets and with MQTT-SN clients over UDP packets tunneled through a transparent MQTT-SN gateway. The encryption and authentication logic runs before transmission and does not include any transport-layer-specific reasoning [26].

### 3.7. Security and session management

To ensure robust security guarantees, HLP enforces session key uniqueness and freshness. Using ECDH guarantees that each session produces a new, independent key, while HMAC ensures the keys are not derivable even if the adversary intercepts public parameters. Mutual authentication is enforced by validating both Auth_C and Auth_B, preventing impersonation. Optional key rotation further enhances security by minimizing the exposure window in the event of a partial compromise [27].

## 4. SIMULATION AND DISCUSSION

The current HLP prototype is implemented in Python to validate the protocol logic and enable rapid experimentation and repeatable measurements. As such, the reported latency and resource usage should be interpreted as prototype-level results that reflect the Python runtime and library overheads. While this provides useful comparative trends and an end-to-end demonstration, it does not fully represent a production-grade embedded implementation on ultra-constrained microcontrollers.

Devices such as the Raspberry Pi Zero W (Linux-based) can support lightweight security services when configured carefully. Still, microcontrollers such as the ESP32 typically require a native implementation (e.g., C/C++ with hardware acceleration and tight memory control). Therefore, the observed memory footprint (e.g., up to ~45 MB) should not be interpreted as the expected embedded footprint of HLP itself; it includes the Python interpreter and cryptographic stacks. A native implementation and on-device evaluation on ESP32-class hardware, together with reduced-footprint libraries and tighter memory profiling, are identified as future work.

To ensure reproducibility, we define a common testbed and per-attack parameters. Testbed: Mosquitto 2.0.18 broker (default config), Python 3.11 clients (paho-mqtt 2.1.0), cryptography 42.0, single subnet over localhost with a 20±5 ms emulated delay (netem), run length 100 s per trial, and fixed RNG seed=42. Baseline traffic: 10 clients publish at 1 msg/s each to distinct topics with 32-byte application payloads (QoS 0). Under HLP, normal message frames (including headers, nonce, and tag) are 85–100 B ($\mu \approx 92$ B, $\sigma \approx 4$ B) for MQTT and 83–98 B ($\mu \approx 90$ B, $\sigma \approx 5$ B) for MQTT-SN. We flag size anomalies when the packet size is >105 B for ≥3 consecutive samples on the same flow (topic and client), or when the 95th percentile exceeds 110 B within a 10-s window.

Attack scenarios: (A1) replay—an adversary re-injects previously captured {CONNECT/PUBLISH} frames at 5 msg/s for 30 s starting at t=40 s. Replayed frames preserve original sizes (typically 90–98 B) but fail HLP freshness checks; detection labels are assigned when ≥10 invalid tokens are observed within 5 s. (A2) Packet Injection/Flood (DoS)—adversary publishes malformed PUBLISH frames at 200 msg/s for 20 s (t=60–

80 s), with payloads padded to 110–115 B to stress bandwidth; we declare flood onset when per-client rate exceeds 50 msg/s for ≥2 s or when accumulated traffic slope increases by ≥3× baseline. (A3) Spoofing/Impersonation—adversary attempts to impersonate a valid client by forging Auth_C/Auth_B at 0.1 Hz (1 attempt/10 s) over the full 100 s; packets are 88–96 B but fail HMAC verification. We count an impersonation event on each invalid token; a spoofing burst is flagged when ≥3 failures occur within 10 s. (A4) Burst Payload Anomaly—legitimate client temporarily sends oversized payloads (256 B app data) at 2 msg/s for 15 s (t=20–35 s) to evaluate false-positive risk; resulting frames are 120–140 B. The detector tolerates a single 10-s burst if <5% of windowed samples exceed 120 B; otherwise, an anomaly is raised. All results are averaged over 5 runs; we report mean±SD for CPU, memory, and latency, along with the above thresholds, so that others can replicate our settings.

### 4.1. Security analysis

We analyze HLP under an active network adversary capable of eavesdropping, replaying, delaying, dropping, and injecting messages, including man-in-the-middle attempts. Replay is mitigated by binding authentication and key derivation to fresh nonces. In contrast, impersonation and MITM attempts are mitigated because the session establishment requires valid HMAC tokens computed with the PSK and a key derived from an ephemeral ECDH exchange. Payload confidentiality and integrity are provided by AEAD encryption after the session key is established. This analysis is intended as a structured, standards-aligned argument; a mechanized proof (e.g., in Tamarin/ProVerif) is left for future work.

### 4.2. Pre-shared key scalability and management

We recognize that PSK-based systems face scalability and lifecycle-management challenges in large deployments. To address this, HLP assumes unique per-device PSKs (not a global shared PSK) and supports operational measures such as secure provisioning/onboarding, key rotation and revocation, and domain-based or hierarchical provisioning to reduce exposure and simplify administration. Where available, hardware-backed storage (e.g., secure elements/TPM-like support) can further reduce key-extraction risk. Nonetheless, PSK management remains a practical limitation of PSK-based approaches, and we explicitly highlight it as a key constraint and direction for future work.

### 4.3. Packet sizes over time

Examining the relationship between unexpected packet sizes and time is critical for identifying atypical communication patterns resulting from IoT environmental conditions and for identifying potential security challenges. Expected packets are usually predictable, with their sizes and timing patterns consistent, but unexpected packets exhibit irregular sizes and timing patterns that may be suspicious or anomalous. Tracking unexpected changes in packet size over time enables us to correlate them with incidents such as unauthorized access attempts, spoofing, and denial-of-service attacks. The timing of packets and resulting changes in packet size provide a model for defining normal communication patterns, enabling us to distinguish between normal and malicious behavior and to design real-time intrusion detection. Additionally, identifying unexpected packet-size aberrations provides important insight for protocol designers to address vulnerabilities exposed in the traffic environment and to improve the resilience and performance of their protocols.

Figure 4 shows a time-based distribution of packet sizes detected in an IoT system, separated into expected and unexpected packets. Expected packets (orange) tended to have lower packet sizes (85–100 bytes), typically indicating legitimate, normal communication. Unexpected packets (blue) exhibited higher, stable packet sizes (110–115 bytes), suggesting abnormal or potentially malicious communication. The visualization enables the identification of anomalous traffic by highlighting outliers that fall outside the normal communication footprint, facilitating the detection of potential security risks or attacks (e.g., injection or packet spoofing) in resource-constrained IoT environments.

### 4.4. Accumulated network traffic over time

It is important to analyze how traffic accumulates over time to quantify the implications of security threats on IoT communication systems. In a regular operation, the increase in traffic typically follows a well-defined, predictable trajectory, consistent with the expected behavior of resource-constrained devices. But in an attack, such as a flooding or injection attack, the volume of transmitted data increases exponentially compared to non-attack traffic, resulting in a steeper cumulative data curve. By tracking traffic trends over time for both attacked and non-attacked states, deviations in traffic patterns associated with a malicious act can be detected, and the resulting degree of resource exhaustion can be identified. Such behaviors in simulation can yield observational evidence of how an attack affects the burden on IoT networks, while also highlighting the need for lightweight and efficient security mechanisms. In addition, the simulation recognizes how proposed protocols can be validated to maintain operation and mitigate excessive data growth resulting from attacks.

Figure 5 depicts the total network traffic for the attacked and non-attacked systems over time. The data illustrate a clear increase in the attacked system's traffic, as indicated by the steeper slope of the blue line compared with the orange line for the non-attacked system. The increase in transmitted data can thereby indicate the use of attack conditions, such as packet injection or packet flooding, causing a heavier burden on the networks and devices involved. Including the simulation in the paper is critical—indeed, quantitatively measuring the impact of attacks only confirms communication overhead. This type of analysis is also beneficial for validating lightweight security mechanisms, given the necessity of such mechanisms to preserve bandwidth and resource efficiency in adverse conditions-especially in constrained IoT contexts.

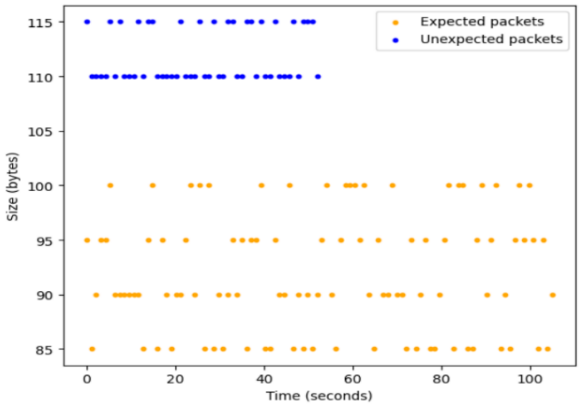


Figure 4. Visualization of expected and unexpected packet sizes over time

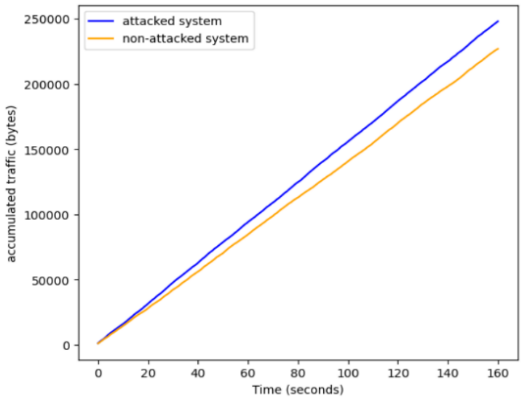


Figure 5. Accumulated network traffic over time in attacked vs. non-attacked IoT systems

### 4.5. Comparative analysis with existing lightweight protocols

To assess the functionality of the proposed HLP, we evaluate its performance and security features against two commonly utilized lightweight protocols, namely ECDHE-PSK and ECDHE-ECDSA. The evaluation uses key metrics relevant to constrained IoT settings, including computation cost, bandwidth cost, certificate cost, and resistance to common attacks. The review shows that HLP is more efficient and secure, and is compatible with both the MQTT and MQTT-SN protocols.

Table 2 compares the HLP with ECDHE-PSK and ECDHE-ECDSA across eight criteria: computational cost, bandwidth utilization, certificate dependency, forward secrecy support, mutual authentication, MQTT-SN support, and sufficient security against common attacks such as replay and spoofing. For constrained devices, the HLP exhibits lower computational and bandwidth overhead and no need for certificates. In addition, it provides robust security guarantees, including full forward secrecy, and mutual authentication. Finally, HLP is the only of the three schemes to support MQTT-SN, providing greater flexibility across different IoT deployments.

Table 2. Comparative analysis of the proposed HLP protocol against existing lightweight cryptographic protocols

| Protocol | Computational cost (ms) | Bandwidth usage (bytes) | Certificate requirement | Forward secrecy | Mutual authentication | MQTT-SN support | Resistance to replay attack | Resistance to spoofing |
|---|---|---|---|---|---|---|---|---|
| ECDHE-PSK | 35 | 160 | No | Partial | Yes | No | Medium | Medium |
| ECDHE-ECDSA | 60 | 180 | Yes | Yes | Yes | No | High | High |
| HLP (proposed) | 20 | 130 | No | Yes | Yes | Yes | High | High |

To provide a broader assessment of the HLP under study, additional performance metrics were captured: CPU usage, memory usage, and end-to-end latency. The study found that HLP had an average of 18–22% CPU usage on ESP32-class devices and 12–15% CPU usage on Raspberry Pi Zero W, indicating an appropriate level of resource consumption for resource-constrained platforms. The memory usage in all device trials, therefore, peaked at less than 45 MB, notably lower than TLS-based schemes (e.g., 65–80 MB). Authentication latency was measured at 20 ms for MQTT and 24 ms for MQTT-SN, with an additional 5 to 8 ms of latency due to encryption/decryption operations on every transaction. Overall, HLP communicates

with a very minimal computational and memory footprint while providing secure communication, further demonstrating its capability for ultra-lightweight IoT deployments.

### 4.6. Positioning and novelty relative to elliptic curve Diffie-Hellman ephemeral-pre-shared key

HLP is not intended to introduce new cryptographic primitives; instead, it defines a lightweight composition and integration of well-established building blocks tailored to MQTT and MQTT-SN deployments. In conventional ECDHE-PSK handshakes (e.g., TLS-PSK variants), the PSK is used within a standardized handshake transcript and key schedule. By contrast, HLP targets publish/subscribe IoT messaging where: i) certificate handling is undesirable, ii) MQTT-SN over UDP and gateway-based deployments are common, and iii) constrained clients benefit from minimal handshake bytes and simple verification steps. HLP therefore: i) performs PSK/HMAC mutual authentication using fresh nonces and device identifiers before accepting a session, ii) establishes forward secrecy using an ephemeral ECDH exchange (Curve25519), and iii) derives the session key using a PSK-bound KDF over the ECDH shared secret and both nonces, followed by AEAD (ChaCha20-Poly1305) for payload protection. The novelty of HLP is thus expressed in: i) a unified application-layer protocol flow that supports both MQTT (TCP) and MQTT-SN (UDP/gateway) while remaining certificate-free; ii) explicit mutual authentication tokens (HMAC) bound to identities and freshness parameters before session acceptance; and iii) an end-to-end proof-of-concept implementation and experimental comparison (latency/bandwidth/resource overhead) against representative lightweight baselines under the same testbed configuration.

The performance of the proposed HLP protocol is compared with two lightweight cryptographic schemes in Figure 6, measuring computational costs and bandwidth usage. HLP demonstrated the lowest computational overhead (20 ms) and the most effective bandwidth use (130 bytes), making HLP a good candidate for resource-constrained IoT devices. Both metrics showed that ECDHE-ECDSA performed the worst, largely due to certificate-based authentication requirements. ECDHE-PSK had better performance than ECDHE-ECDSA but was not as efficient as HLP. The results of this study suggest that HLP can perform secure communication with minimal resource use, furthering its applicability for low-power, and low-latency IoT use cases.

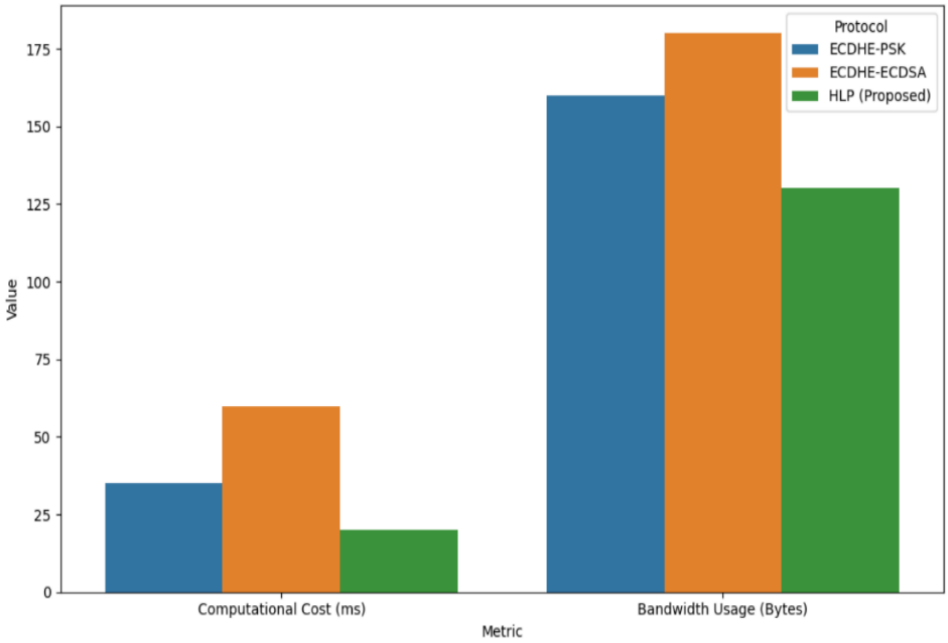


Figure 6. Comparative performance of HLP vs. ECDHE-PSK and ECDHE-ECDSA

## 5. CONCLUSION

This paper presented the HLP for securing IoT communications over MQTT and MQTT-SN in resource-constrained environments. HLP combines PSK-assisted HMAC-based mutual authentication, ephemeral elliptic-curve key agreement for session establishment, and ChaCha20-Poly1305 for authenticated encryption. A proof-of-concept implementation was developed and evaluated in a controlled testbed, reporting performance in terms of authentication/handshake latency, bandwidth overhead, and computational cost. The

results suggest that HLP can provide a practical and efficient security option for lightweight publish/subscribe IoT messaging under the experimental conditions considered in this study.

Despite these contributions, several limitations should be noted. First, HLP assumes secure PSK provisioning, storage, and rotation, which may be challenging to manage in heterogeneous or highly dynamic IoT deployments. Second, the evaluation was conducted in a controlled testbed and does not fully capture the behavior of large-scale networks with many clients, high churn, and diverse hardware/firmware constraints. Third, while the design follows established cryptographic primitives, this work does not provide a formal safety/security analysis (e.g., a formal proof, symbolic verification, or protocol model checking) to establish security properties under a comprehensive adversarial model rigorously. Finally, the implementation is software-based and may not reflect optimizations or constraints of all embedded platforms.

Future work will focus on PSK lifecycle enhancements (e.g., automated provisioning, secure rotation, and compromise recovery) and large-scale evaluation using emulated or real deployments with higher node counts and varied device capabilities. In addition, we plan to conduct a formal protocol analysis using established verification frameworks and to extend the threat model to cover additional IoT-specific adversarial conditions. These steps will strengthen confidence in HLP's security guarantees and clarify its applicability across broader IoT operational scenarios.

## ACKNOWLEDGMENTS

The authors would like to acknowledge Princess Sumaya University for Technology (PSUT) and the Cybersecurity Research Centre at Universiti Sains Malaysia (USM) for their technical and administrative support in conducting this research.

## FUNDING INFORMATION

This research received no external funding

## AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

| Name of Author | C | M | So | Va | Fo | I | R | D | O | E | Vi | Su | P | Fu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nabeel Alassaf | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | | ✓ | |
| Selvakumar Manickam | | ✓ | | | | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| Ammar Odeh | ✓ | | ✓ | ✓ | | | ✓ | | | ✓ | ✓ | | ✓ | ✓ |
| Mohammed Anbar | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | | | ✓ | |

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

## CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

## INFORMED CONSENT

Not applicable. This study did not involve humans or the collection of any personal or identifiable data.

## ETHICAL APPROVAL

Not applicable. This study did not involve humans or animals. All experiments were conducted using synthetic IoT network traffic generated in a controlled laboratory environment.

## DATA AVAILABILITY

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

## REFERENCES


[1] A. E. Omolara *et al.*, "The internet of things security: A survey encompassing unexplored areas and new insights," *Comput. Secur.*, vol. 112, p. 102494, Jan. 2022, doi: 10.1016/j.cose.2021.102494.

[2] M. Abdullahi *et al.*, "Detecting Cybersecurity Attacks in Internet of Things Using Artificial Intelligence Methods: A Systematic Literature Review," *Electron.*, vol. 11, no. 2, p. 198, Jan. 2022, doi: 10.3390/electronics11020198.

[3] Z. G. Al-Mekhlafi *et al.*, "Coherent Taxonomy of Vehicular Ad Hoc Networks (VANETs) Enabled by Fog Computing: A Review," *IEEE Sens. J.*, vol. 24, no. 19, pp. 29575–29602, 2024, doi: 10.1109/JSEN.2024.3436612.

[4] A. A. K. Majhi and S. Mohanty, "A Comprehensive Review on Internet of Things Applications in Power Systems," *IEEE Internet Things J.*, vol. 11, no. 21, pp. 34896–34923, Nov. 2024, doi: 10.1109/JIOT.2024.3447241.

[5] P. Malhotra, Y. Singh, P. Anand, D. K. Bangotra, P. K. Singh, and W. C. Hong, "Internet of things: Evolution, concerns and security challenges," *Sensors*, vol. 21, no. 5, pp. 1–35, Mar. 2021, doi: 10.3390/s21051809.

[6] S. Balogh, O. Gallo, R. Ploszek, P. Špaček, and P. Zajac, "IoT security challenges: Cloud and blockchain, postquantum cryptography, and evolutionary techniques," *Electron.*, vol. 10, no. 21, p. 2647, Oct. 2021, doi: 10.3390/electronics10212647.

[7] A. Djenna, S. Harous, and D. E. Saidouni, "Internet of things meet internet of threats: New concern cyber security issues of critical cyber infrastructure," *Appl. Sci.*, vol. 11, no. 10, p. 4580, May 2021, doi: 10.3390/app11104580.

[8] W. Lin, S. Chen, and H. Zhu, "Formal verification and security analysis of MQTT-SN," *Int. J. Softw. Tools Technol. Transf.*, vol. 27, no. 1, pp. 5–19, Feb. 2025, doi: 10.1007/s10009-025-00793-2.

[9] O. M. Dighriri, P. Nanda, M. Mohanty, B. Alrashed, and I. Haddadi, "Protocol-Aware Hybrid Clustering for IoT: Adaptive Reconfiguration and Secure Communication with MQTT/CoAP Integration," in *2025 IEEE/ACS 22nd Int. Conf. on Comput. Syst. Appl. (AICCSA)*, Doha, Qatar, 2026, pp. 1–8, doi: 10.1109/AICCSA66935.2025.11315394.

[10] M. K. Khan *et al.*, "Hierarchical Routing Protocols for Wireless Sensor Networks: Functional and Performance Analysis," *J. Sensors*, vol. 2021, no. 1, Jan. 2021, doi: 10.1155/2021/7459368Digital Object Identifier (DOI).

[11] R. Elsayed, R. Hamada, M. Hammoudeh, M. Abdalla, and S. A. Elsaid, "A Hierarchical Deep Learning-Based Intrusion Detection Architecture for Clustered Internet of Things," *J. Sens. Actuator Networks*, vol. 12, no. 1, p. 3, Dec. 2023, doi: 10.3390/jsan12010003.

[12] E. B. Sanjuan, I. A. Cardiel, J. A. Cerrada, and C. Cerrada, "Message Queuing Telemetry Transport (MQTT) Security: A Cryptographic Smart Card Approach," *IEEE Access*, vol. 8, pp. 115051–115062, 2020, doi: 10.1109/ACCESS.2020.3003998.

[13] B. Arbab-Zavar, E. J. Palacios-Garcia, J. C. Vasquez, and J. M. Guerrero, "Message queuing telemetry transport communication infrastructure for grid-connected ac microgrids management," *Energies*, vol. 14, no. 18, p. 5610, Sep. 2021, doi: 10.3390/en14185610.

[14] C. Wu, Y. Yuan, Y. Tang, and B. Tian, "Application of terrestrial laser scanning (Tls) in the architecture, engineering and construction (aec) industry," *Sensors*, vol. 22, no. 1, p. 265, Dec. 2022, doi: 10.3390/s22010265.

[15] N. Shen *et al.*, "A review of terrestrial laser scanning (TLS)-based technologies for deformation monitoring in engineering," *Meas. J. Int. Meas. Confed.*, vol. 223, p. 113684, Dec. 2023, doi: 10.1016/j.measurement.2023.113684.

[16] H. Xue, M. H. Au, X. Xie, T. H. Yuen, and H. Cui, "Efficient Online-friendly Two-Party ECDSA Signature," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2021, pp. 558–573, doi: 10.1145/3460120.3484803.

[17] J. Roldán-Gómez, J. Carrillo-Mondéjar, J. M. C. Gómez, and J. L. M. Martínez, "Security Assessment of the MQTT-SN Protocol for the Internet of Things," *J. Phys. Conf. Ser.*, vol. 2224, no. 1, p. 012079, Apr. 2022, doi: 10.1088/1742-6596/2224/1/012079.

[18] A. J. Hintaw, S. Manickam, M. F. Aboalmaaly, and S. Karuppayah, "MQTT Vulnerabilities, Attack Vectors and Solutions in the Internet of Things (IoT)," *IETE J. Res.*, vol. 69, no. 6, pp. 3368–3397, Aug. 2023, doi: 10.1080/03772063.2021.1912651.

[19] J. Ali, M. H. Zafar, C. Hewage, R. Hassan, and R. Asif, "Mathematical Modeling and Validation of Retransmission-Based Mutant MQTT for Improving Quality of Service in Developing Smart Cities," *Sensors*, vol. 22, no. 24, p. 9751, Dec. 2022, doi: 10.3390/s22249751.

[20] I. Petrescu, E. Niculae, V. Vulturescu, A. Dimitrescu, and L. M. Ungureanu, "Transport and Application Layer Protocols for IoT: Comprehensive Review," *Technologies*, vol. 13, no. 12, p. 583, Dec. 2025, doi: 10.3390/technologies13120583.

[21] N. T. Dhokane, S. Jagtap, B. Kumar, A. Anand, and R. K. Pandey, "S-MQTT: A Secure MQTT Protocol with Merkle Tree Authentication and AES Encryptionfor IoT Communication Systems," *Ing. des Syst. d'Information*, vol. 30, no. 8, pp. 1963–1973, Aug. 2025, doi: 10.18280/isi.300803.

[22] J. Roldán-Gómez, J. Carrillo-Mondéjar, J. M. C. Gómez, and S. Ruiz-Villafranca, "Security Analysis of the MQTT-SN Protocol for the Internet of Things," *Appl. Sci.*, vol. 12, no. 21, p. 10991, Oct. 2022, doi: 10.3390/app122110991.

[23] S. Andy, B. Rahardjo, and B. Hanindhito, "Attack scenarios and security analysis of MQTT communication protocol in IoT system," in *Proc. Int. Conf. Electr. Eng. Comput. Sci. Informatics (EECSI)*, 2017, pp. 1-6, doi: 10.1109/EECSI.2017.8239179.

[24] H. Y. Chien *et al.*, "A MQTT-API-compatible IoT security-enhanced platform," *Int. J. Sens. Networks*, vol. 32, no. 1, pp. 54–68, 2020, doi: 10.1504/IJSNET.2020.104463.

[25] D. Dinculeană and X. Cheng, "Vulnerabilities and limitations of MQTT protocol used between IoT devices," *Appl. Sci.*, vol. 9, no. 5, p. 848, Feb. 2019, doi: 10.3390/app9050848.

[26] B. Gușiță, A. A. Anton, C. S. Stângaciu, D. Stănescu, L. I. Găină, and M. V. Micea, "Securing IoT edge: a survey on lightweight cryptography, anonymous routing and communication protocol enhancements," *Int. J. Inf. Secur.*, vol. 24, no. 3, p. 149, Jun. 2025, doi: 10.1007/s10207-025-01071-7.

[27] N. R. Kumar, R. B. Krishnan, S. Vairavasundaram, G. Manikandan, V. Indragandhi, and L. Ravi, "An IoT based remote medical diagnosis system using one time pad cipher over MQTT protocol," *Sci. Rep.*, vol. 15, no. 42117, Nov. 2025, doi: 10.1038/s41598-025-26208-5.

## BIOGRAPHIES OF AUTHOR

**Nabeel Alassaf** received his bachelor's degree in computer science from the University of Jordan, in 2002. He received his master's degree in computer science from the University of Jordan in Jordan, in 2005. Currently, he is doing Ph.D. in network resource management at the Cybersecurity Research Centre, Universiti Sains Malaysia (USM). He is currently a lecturer with the Department of Computer Science at the University of Jordan with a master's degree. His research interest includes computer networks, internet communication protocols (IPv6), network security, mobile agents, cybersecurity, AI, and internet of things (IoT). He can be contacted at email: nabeelalassaf@student.usm.my.

**Prof. Dr. Selvakumar Manickam** is the Director of the Cybersecurity Research Centre and a professor specializing in cybersecurity, the internet of things, Industry 4.0, cloud computing, big data, and machine learning. He has authored and co-authored more than 220 articles in journals, conference proceedings, and book reviews. He has graduated 18 Ph.D. students in addition to master's and bachelor's students. He has given several keynote speeches and dozens of invited lectures and workshops at conferences, international universities, and industry. He can be contacted at email: selva@usm.my.

**Ammar Odeh** received his Ph.D. from the University of Bridgeport (UB), USA, in 2015. He is an Associate Professor at the Department of Computer Science, Faculty of King Hussein School of Computing Sciences, Princess Sumaya University for Technology, Amman, Jordan. His research interests include cybersecurity, cryptography, and the internet of things (IoT). He can be contacted at email: a.odeh@psut.edu.jo.

**Dr. Mohammed Anbar** received the bachelor's degree in Computer System Engineering from Al-Azhar University, Palestine; the M.Sc. degree in information technology from Universiti Utara Malaysia (UUM), Malaysia, and the Ph.D. degree in advanced Internet security and monitoring from the University Sains Malaysia (USM). His research interests include malware detection, web security, intrusion detection systems (IDS), intrusion prevention systems (IPS), network monitoring, internet of things (IoT), and IPv6 security. He can be contacted at email: anbar@cyres.usm.my.