"TLS Cipher Suites-ECC & RSA for secure communication in IoT"

Privacy and Security in IoT(IIS3152)

Research Assignment

Submitted in the partial fulfilment of the requirement for the award of

Bachelor of Technology in

IoT and Intelligent Systems

By:

Shaurya Jain - 23FE10CII00161

Under the supervision of:

Dr. Rohit Singh

September 2025

IoT and Intelligent Systems

School of Computer Science and Engineering

Manipal University Jaipur

VPO. Dehmi Kalan, Jaipur, Rajasthan, India – 303007

Problem Statement

TLS(Transport Layer Security) is essential for secure communication of MQTT in IoT. Traditional RSA(Rivest-Shamir-Adleman) based cipher suites are very resource heavy resulting in performance overheads while processing for the IoT devices with low resource capacity. This creates the need for better performing lighter alternative cipher suites and need for performance analysis of different cipher suites to find an optimal solution is also seen in the researches.

Proposed Solution

The solution is to conduct a comparative performance analysis between two TLS Cipher Suites- Traditional RSA based and a lightweight ECC(Elliptic Curve Cryptography) based. The performance analysis will be implemented using a python script with simulating a client connection with a broker to represent a simulated IoT environment to obtain more directed results. The Comparison will be on the basis of average TLS handshake time measured in milliseconds(ms) and average percentage of CPU usage of the device. Cipher Suites used for comparison:-

>>RSA Cipher Suite ::
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

>>ECC Cipher Suite ::
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

Result

The python script was run various times. Observation was that the average handshake times for RSA and ECC cipher suites were very close in all the runs but mostly it was ECC based suite with lower handshake times and lower CPU usage compared to RSA based suite. The best result out of all the runs is presented below.

Cipher Suites	**Average TLS Handshake Time (ms)**	**Average CPU Usage (%)**
RSA:: **AES_256_GCM_SHA384**	6724	2.19
ECC:: **ECDHE_ECDSA_WITH_ AES_128_GCM_SHA256**	5638.47	1.79

TABLE 1: Average TLS handshake time and CPU usage

Performance Analysis Charts:

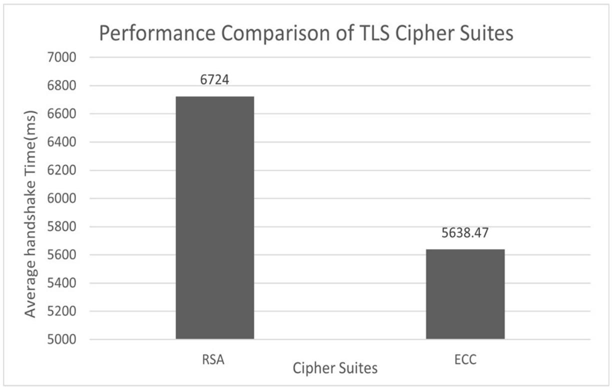


Fig 1: Comparison of Average Handshake Times

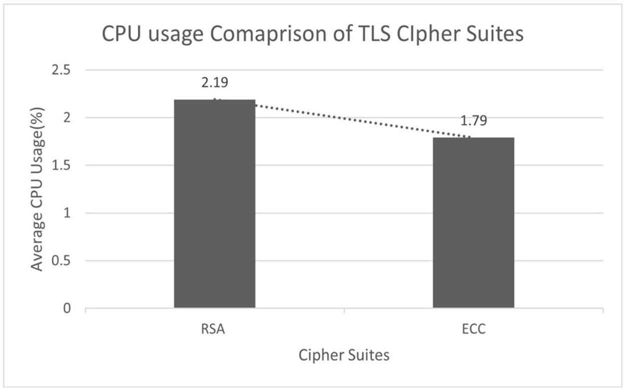
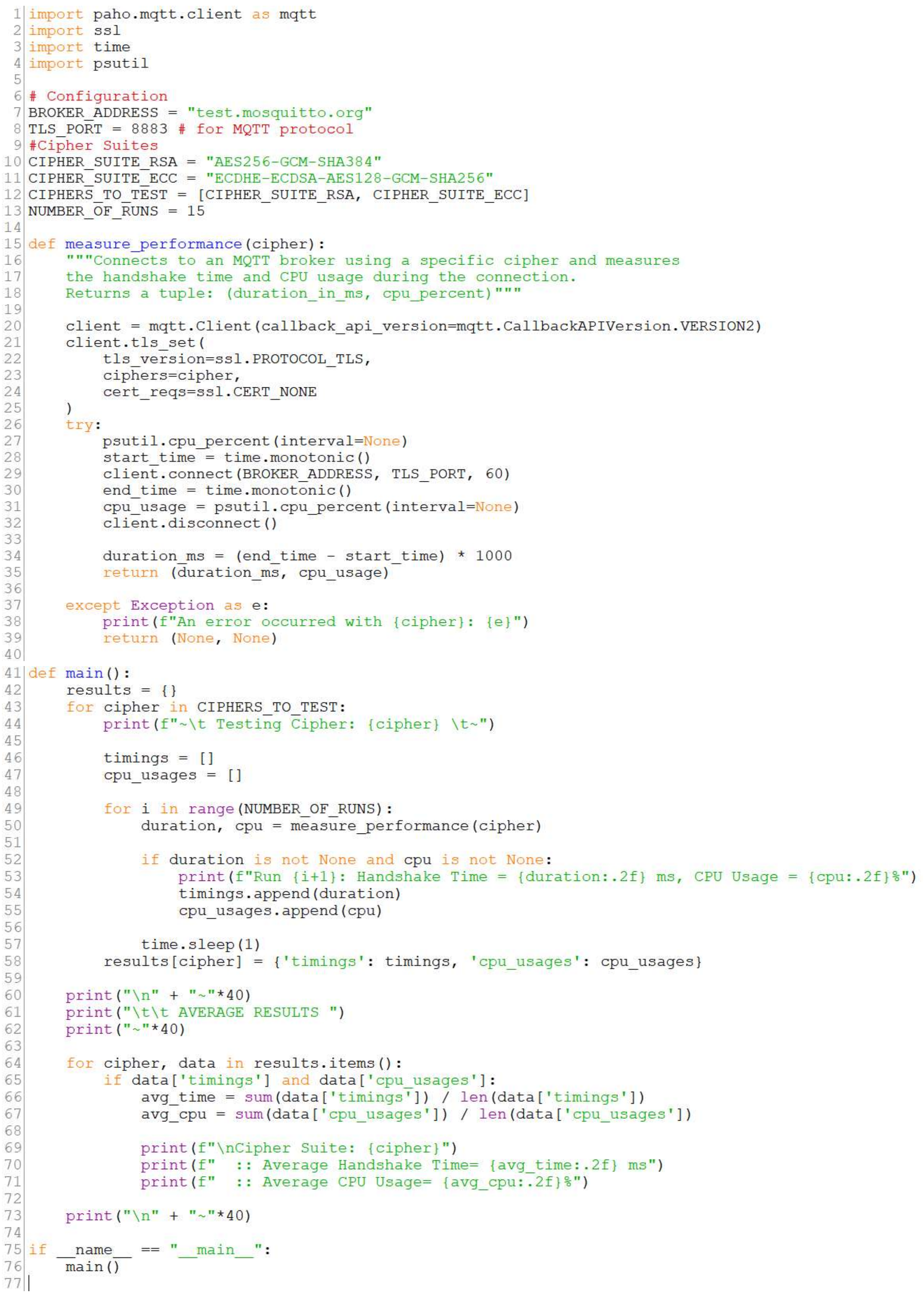


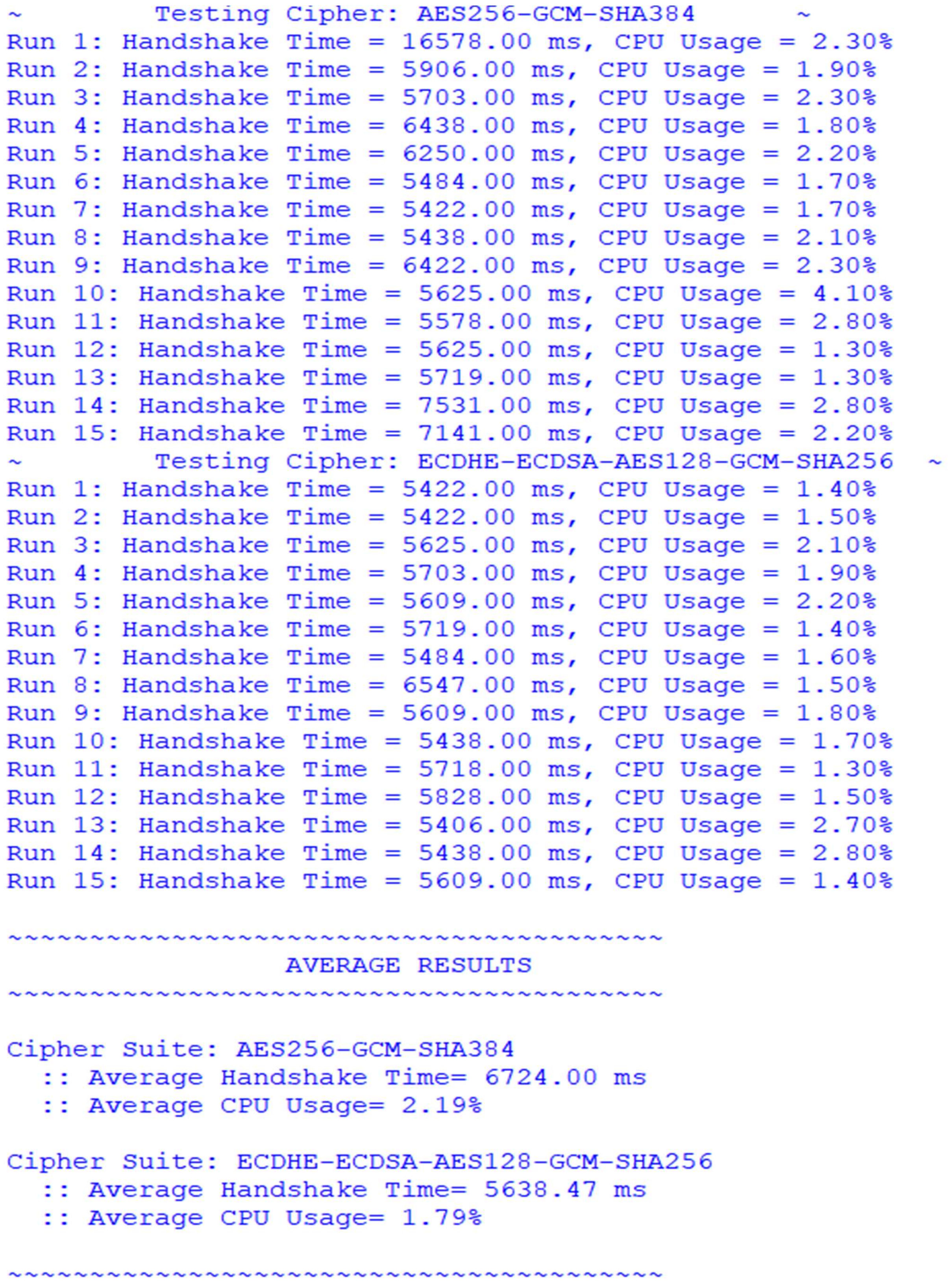
Fig 2: Comparison of Average CPU Usage (%)

Implementation - Python Script:-



```python
import paho.mqtt.client as mqtt
import ssl
import time
import psutil

# Configuration
BROKER_ADDRESS = "test.mosquitto.org"
TLS_PORT = 8883 # for MQTT protocol
#Cipher Suites
CIPHER_SUITE_RSA = "AES256-GCM-SHA384"
CIPHER_SUITE_ECC = "ECDHE-ECDSA-AES128-GCM-SHA256"
CIPHERS_TO_TEST = [CIPHER_SUITE_RSA, CIPHER_SUITE_ECC]
NUMBER_OF_RUNS = 15

def measure_performance(cipher):
    """Connects to an MQTT broker using a specific cipher and measures
    the handshake time and CPU usage during the connection.
    Returns a tuple: (duration_in_ms, cpu_percent)"""

    client = mqtt.Client(callback_api_version=mqtt.CallbackAPIVersion.VERSION2)
    client.tls_set(
        tls_version=ssl.PROTOCOL_TLS,
        ciphers=cipher,
        cert_reqs=ssl.CERT_NONE
    )
    try:
        psutil.cpu_percent(interval=None)
        start_time = time.monotonic()
        client.connect(BROKER_ADDRESS, TLS_PORT, 60)
        end_time = time.monotonic()
        cpu_usage = psutil.cpu_percent(interval=None)
        client.disconnect()

        duration_ms = (end_time - start_time) * 1000
        return (duration_ms, cpu_usage)

    except Exception as e:
        print(f"An error occurred with {cipher}: {e}")
        return (None, None)

def main():
    results = {}
    for cipher in CIPHERS_TO_TEST:
        print(f"~\t Testing Cipher: {cipher} \t~")

        timings = []
        cpu_usages = []

        for i in range(NUMBER_OF_RUNS):
            duration, cpu = measure_performance(cipher)

            if duration is not None and cpu is not None:
                print(f"Run {i+1}: Handshake Time = {duration:.2f} ms, CPU Usage = {cpu:.2f}%")
                timings.append(duration)
                cpu_usages.append(cpu)

            time.sleep(1)
        results[cipher] = {'timings': timings, 'cpu_usages': cpu_usages}

    print("\n" + "~"*40)
    print("\t\t AVERAGE RESULTS ")
    print("~"*40)

    for cipher, data in results.items():
        if data['timings'] and data['cpu_usages']:
            avg_time = sum(data['timings']) / len(data['timings'])
            avg_cpu = sum(data['cpu_usages']) / len(data['cpu_usages'])

            print(f"\nCipher Suite: {cipher}")
            print(f"  :: Average Handshake Time= {avg_time:.2f} ms")
            print(f"  :: Average CPU Usage= {avg_cpu:.2f}%")

    print("\n" + "~"*40)

if __name__ == "__main__":
    main()
```

Output



```
~        Testing Cipher: AES256-GCM-SHA384      ~
Run 1: Handshake Time = 16578.00 ms, CPU Usage = 2.30%
Run 2: Handshake Time = 5906.00 ms, CPU Usage = 1.90%
Run 3: Handshake Time = 5703.00 ms, CPU Usage = 2.30%
Run 4: Handshake Time = 6438.00 ms, CPU Usage = 1.80%
Run 5: Handshake Time = 6250.00 ms, CPU Usage = 2.20%
Run 6: Handshake Time = 5484.00 ms, CPU Usage = 1.70%
Run 7: Handshake Time = 5422.00 ms, CPU Usage = 1.70%
Run 8: Handshake Time = 5438.00 ms, CPU Usage = 2.10%
Run 9: Handshake Time = 6422.00 ms, CPU Usage = 2.30%
Run 10: Handshake Time = 5625.00 ms, CPU Usage = 4.10%
Run 11: Handshake Time = 5578.00 ms, CPU Usage = 2.80%
Run 12: Handshake Time = 5625.00 ms, CPU Usage = 1.30%
Run 13: Handshake Time = 5719.00 ms, CPU Usage = 1.30%
Run 14: Handshake Time = 7531.00 ms, CPU Usage = 2.80%
Run 15: Handshake Time = 7141.00 ms, CPU Usage = 2.20%
~        Testing Cipher: ECDHE-ECDSA-AES128-GCM-SHA256  ~
Run 1: Handshake Time = 5422.00 ms, CPU Usage = 1.40%
Run 2: Handshake Time = 5422.00 ms, CPU Usage = 1.50%
Run 3: Handshake Time = 5625.00 ms, CPU Usage = 2.10%
Run 4: Handshake Time = 5703.00 ms, CPU Usage = 1.90%
Run 5: Handshake Time = 5609.00 ms, CPU Usage = 2.20%
Run 6: Handshake Time = 5719.00 ms, CPU Usage = 1.40%
Run 7: Handshake Time = 5484.00 ms, CPU Usage = 1.60%
Run 8: Handshake Time = 6547.00 ms, CPU Usage = 1.50%
Run 9: Handshake Time = 5609.00 ms, CPU Usage = 1.80%
Run 10: Handshake Time = 5438.00 ms, CPU Usage = 1.70%
Run 11: Handshake Time = 5718.00 ms, CPU Usage = 1.30%
Run 12: Handshake Time = 5828.00 ms, CPU Usage = 1.50%
Run 13: Handshake Time = 5406.00 ms, CPU Usage = 2.70%
Run 14: Handshake Time = 5438.00 ms, CPU Usage = 2.80%
Run 15: Handshake Time = 5609.00 ms, CPU Usage = 1.40%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                 AVERAGE RESULTS
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Cipher Suite: AES256-GCM-SHA384
  :: Average Handshake Time= 6724.00 ms
  :: Average CPU Usage= 2.19%

Cipher Suite: ECDHE-ECDSA-AES128-GCM-SHA256
  :: Average Handshake Time= 5638.47 ms
  :: Average CPU Usage= 1.79%

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Summary of References

IoT devices have a very low resource capacity. Privacy and Security in these devices and communication amongst them is a challenge. TLS(Transport Layer Security) protocol is the standard used for securing communication in IoT, but research has proved the problem that TLS is resource heavy and results in performance overheads and higher latency. The main aim of these researches is to build lightweight methods for securing communication in IoT. These researches show some lightweight ciphers to achieve the aim like one approach shows that for the existing TLS 1.3 protocol full implementation is not always necessary. Instead a limited but a secure subset of the crypto algorithms can be selected to create more reliable and efficient cipher suite as per the requirements or use case of the embedded system. The research uses a X25519 and AES_128_GCM composed cipher suite for key exchange and encryption in an ESP32 module. In the other researches ,we see a new lightweight cipher: M-SPECK designed for low power environments and requires hardware and security balancing. Another approach uses older cipher AES with using a faster subkey generation SHA3 making it more efficient for encrypting small amounts of data. There can be seen in these researches is that there is a need for performance analysis of different cipher suites to find the most efficient solution with balance in security and low resource for secure communication in IoT in like MQTT protocol.

REFERENCES

1. J. Pajkos, E. Kupcova, M. Pleva and M. Drutarovsky, "ESP32 Microcontroller based Lightweight TLS 1.3 Client for IoT Applications," 2025 35th International Conference Radioelektronika (RADIOELEKTRONIKA), Czech Republic, 2025, pp. 1-6, doi: 10.1109/RADIOELEKTRONIKA65656.2025.11008381. keywords: {Performance evaluation;Quantum computing;Protocols;Embedded systems;Microcontrollers;Libraries;Web servers;Generators;Cryptography;Internet of Things;ESP32 Microcontroller;Internet of Things;Embed-ded Systems;Lightweight TLS 1.3 client},

2. M. Nadeem, R. Mustafa, P. E. Abi-Char and R. Singh Tucker, "A Study of Security Threats in IoT Network Layer using MQTT and TLS," 2025 12th International Conference on Information Technology (ICIT), Amman, Jordan, 2025, pp. 161-166, doi: 10.1109/ICIT64950.2025.11049210. keywords: {Simulation;Network architecture;Time measurement;Encryption;Sensors;Security;Internet of Things;Reliability;Standards;Monitoring;IoT;cybersecurity;network layer;three-layer architecture},

3. K. Zhang, M. Deng, B. Gong, Y. Miao and J. Ning, "Privacy-Preserving Traceable Encrypted Traffic Inspection in Blockchain-Based Industrial IoT," in IEEE Internet of Things Journal, vol. 11, no. 2, pp. 3484-3496, 15 Jan.15, 2024, doi: 10.1109/JIOT.2023.3297601.

keywords: {Industrial Internet of Things;Cryptography;Inspection;Middleboxes;Blockchains;Security;Cloud computing;Blockchain;encrypted traffic inspection;Industrial Internet of Things (IIoT)},

4.. R. Mohanapriya and V. Nithish Kumar, "Modified SPECK (M-SPECK) Lightweight Cipher Architecture for Resource-Constrained Applications," in IEEE Access, vol. 13, pp. 88993-89002, 2025, doi: 10.1109/ACCESS.2025.3570727.

keywords: {Ciphers;Hardware;Security;Encryption;Internet of Things;Field programmable gate arrays;NIST;SIMO;Schedules;Pipelines;Lightweight cryptography;SPECK;ARX;FPGA;M-SPECK;performance analysis},

5. A. Soni and S. K. Sahay, "SEASHA3: Secure and Efficient Encryption for the IoT Data by Replacing Subkeys of AES with SHA3," 2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Kuching, Malaysia, 2024, pp. 1998-2003, doi: 10.1109/SMC54092.2024.10831459. keywords: {Ciphers;NIST;Encryption;Internet of Things;Standards;Cybernetics},