Article

Not peer-reviewed version

Enhancing Cybersecurity in IoT Networks: Exploring Lightweight Key Establishment Using OSCORE and EDHOC

Mohammed El-Hajj * and Reneta Trifonova

Posted Date: 4 July 2024

doi: 10.20944/preprints202407.0427.v1

Keywords: IoT; CoAP; OSCORE; EDHOC; Hardware Acceleration; TLS; DTLS; Key Exchange

Article

Enhancing Cybersecurity in IoT Networks: Exploring Lightweight Key Establishment Using OSCORE and EDHOC

Mohammed El-Hajj [*,†] **and Reneta Trifonova** [†]

Department of Semantics, Cybersecurity & Services, University of Twente

[*] Correspondence: m.elhajj@utwente.nl

[†] These authors contributed equally to this work.

Abstract: The widespread adoption of IoT networks has revolutionized modern life thanks to their diverse range of applications. With their increased usage, the need to transfer data securely has never been more vital. Often IoT devices are battery-driven and possess limited processing and storage capabilities. In response to those limitations, there has been a shift towards the implementation of resource-efficient and lighter protocols. This research aims to evaluate how lightweight key establishment through Object Security for Constrained RESTful Environments (OSCORE) and Ephemeral Diffie-Hellman Over COSE (EDHOC) can enhance cybersecurity in IoT networks where resources are limited. The study aims to explore different key management approaches and the effects of certificates on security, efficiency, and power consumption. The research methodology consists of the practical implementation of the protocols in a physical system where factors such as the impact of hardware acceleration on computational overhead and overall power consumption will be assessed. Furthermore, the study will verify the cost of resources and overall efficiency, evaluating them against real-world scenarios. The expected contributions of this study are insights into the tradeoffs between security, power consumption, and efficiency in resource-limited IoT environments.

Keywords: IoT; CoAP; OSCORE; EDHOC; hardware acceleration; TLS; DTLS; key exchange

---

1. Introduction

The Internet of Things (IoT) refers to networks of connected devices that can communicate with each other or with the cloud [1]. Nowadays, these networks have become an essential part of our daily lives as we are always surrounded by smart products [2]. As the number of IoT devices continues to grow, they become more vulnerable to attacks. Therefore, it is vital to secure the transmitted data to prevent malicious activity that could compromise the entire network and cause serious consequences. To ensure secure communication in IoT networks, protocols for end-to-end security have been implemented. These protocols ensure that the messages' confidentiality, integrity, and availability remain uncompromised [3].

1.1. Background

In constrained environments, such as those found in IoT networks, lightweight security protocols play a critical role in ensuring secure communication without overburdening the limited resources of the devices [4]. These environments often consist of devices with minimal processing power, memory, and battery life, making it essential to use security protocols that are efficient and low overhead [5]. Lightweight protocols, such as OSCORE and EDHOC [6], are designed to provide robust security features, including confidentiality, integrity, and authentication, while maintaining minimal computational and energy costs. This efficiency is crucial for maintaining the performance and longevity of IoT devices, which are typically deployed in large numbers and expected to operate reliably over extended periods. By optimizing security mechanisms to fit within the constraints of these environments, lightweight protocols help ensure the overall security and functionality of IoT networks, enabling secure data transmission and protecting against various cyber threats without compromising device performance. The Constrained Application Protocol (CoAP) is a compressed version of the HTTP protocol designed for resource-constrained devices where compatibility and low power are essential [7]. While CoAP provides a high level of communications security, it remains vulnerable to various attacks, such as Man-In-The-Middle attacks and Denial-of-Service attacks. To mitigate such security vulnerabilities, the Internet Engineering Task Force (IETF) has published a new standard, Object Security for Constrained RESTful Environments (OSCORE), for application-level security. According to the protocol documentation, OSCORE provides end-to-end protection between endpoints communicating using CoAP or CoAP-mappable HTTP [8]. Moreover, it supports a wide range of proxy operations and translation between transport protocols. As it was designed for resource-constrained devices, its messages are as small as 11-13 bytes, and the protocol encrypts only the data part of the payload, thus decreasing security overhead and increasing bandwidth usage and battery life of the device [9]. To set up OSCORE, a security context needs to be established. The lightweight authenticated key exchange protocol Ephemeral Diffie-Hellman Over COSE (EDHOC) can be used for optimized setup. EDHOC was proposed by the IETF for its many advantages, one of which is that it significantly reduces the number of roundtrips needed for setting up the OSCORE security context [10]. The way the security context is set up can be seen in Figure 1.

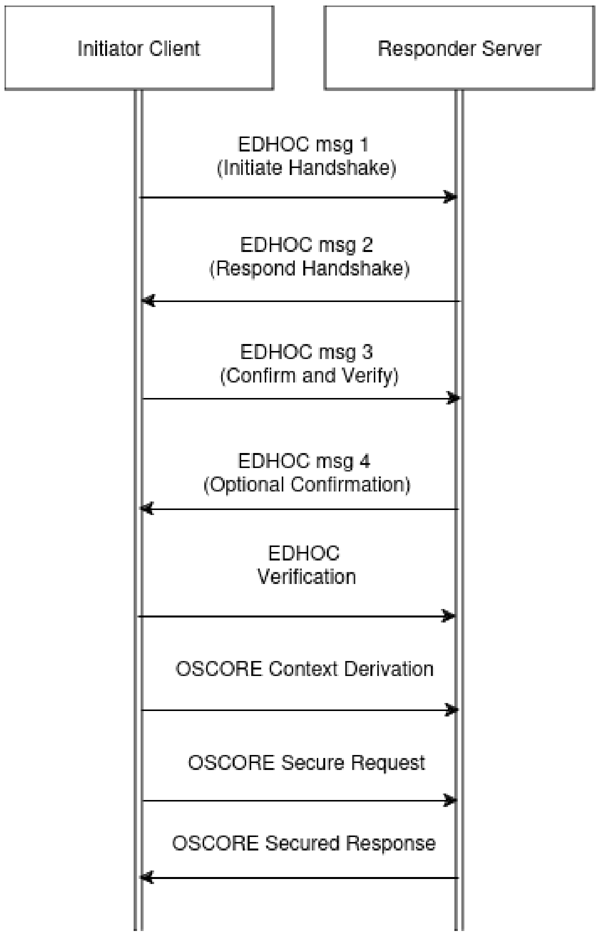

Figure 1. OSCORE-EDHOC secure context establishment.

One of the advantages EDHOC offers is that it uses ephemeral keys, meaning that for each key establishment session, different keys are generated, ensuring there is Perfect Forward Secrecy in the communication channel [11]. The overall performance and cost of OSCORE and EDHOC-based implementations have been of interest to many researchers as it is crucial to evaluate metrics such as computational overhead, power consumption, and security when it comes to IoT devices. While there have been tests conducted to measure such metrics, there is limited data about how different key management strategies and certificates can impact performance. Moreover, many articles mention that hardware acceleration can have a high impact on the performance and memory requirements of the protocols; however, there is insufficient research to extract concrete quantitative data to support the claim.

IoT communication networks play a pivotal role in managing complex systems in various domains such as healthcare, smart homes, and industrial automation. As such devices have increased in complexity, new challenges associated with cybersecurity have emerged, especially in environments with limited resources. One of the largest IoT cyber attacks with severe consequences was the Mirai botnet distributed denial-of-service (DDoS) attack [12], where over 175,000 websites were affected. Such attacks, which take advantage of weak passwords, can be prevented by adopting strong authentication mechanisms. The combination of using OSCORE together with EDHOC ensures end-to-end encryption with verification of both parties before data is exchanged, preventing DDoS attacks. The need for robust authentication and encryption in IoT networks requires careful reflection on key management strategies (e.g., Pre-shared Keys (PSK), Raw Public Keys (RPK), or dynamically generated keys) as it is important to consider various metrics, such as latency, throughput, and security in the system design. However, at present, there is a lack of data that can help evaluate the suitability and trade-offs of these key management strategies, especially in the context of OSCORE and EDHOC-based systems. Moreover, such networks use certificate-based authentication, although a proper comparison of different digital certificates in terms of critical metrics like resource utilization has yet to be made. This is especially vital for devices with limited computational power, as certificate-based authentication can prove to be resource-heavy.

Other ways to improve computational overhead include the use of hardware accelerators, special-purpose hardware structures separated from the CPU [13], as they can enhance the cryptographic efficiency of algorithms such as Advanced Encryption Standard (AES), an algorithm used with OSCORE. However, there is a need for detailed documentation on the trade-offs between speed and power consumption when hardware acceleration is enabled, both crucial factors to consider for IoT networks.

1.2. Research Questions

Based on the above-mentioned information, the main goal of this work will be to answer the following question:
How do key management strategies, certificates, and hardware accelerators impact the latency, throughput, security, and energy efficiency of IoT devices secured by OSCORE and EDHOC protocols, and how does establishing the security context over DTLS compare?

The main question can be split into the following three sub-questions:

1. How do different EDHOC methods of operation impact the security of a system utilizing the OSCORE protocol and its performance metrics, such as latency and memory requirements in IoT networks?
2. How do hardware accelerators influence the computational overhead, energy consumption, and memory usage of IoT devices implementing OSCORE and EDHOC protocols, and what are the associated trade-offs?
3. How does establishing a security context for OSCORE over DTLS compare in terms of computational overhead to incorporating the EDHOC protocol in the design?

The rest of this paper is organized as follows: Section 2 provides an overview of related works. Section 4 presents the obtained data from performing tests on computational overhead with the help of an experimental setup together with an analysis of the security of the system. The following section 5 contains an analysis of the results, while the conclusions section 6 summarises the key points discussed and provides suggestions for future work.

2. Related Work

In the related work section, we plan to review existing cybersecurity solutions for IoT networks, comparing various lightweight key establishment protocols and highlighting their limitations. We will then discuss how OSCORE and EDHOC address these limitations to enhance security and efficiency in constrained environments. Then we will provide an overview of what is presented in literature regarding the implementation of OSCORE and EDHOC in IoT applications.

2.1. Review of Existing Cybersecurity Solutions for IoT Networks

The rapid growth of the Internet of Things (IoT) has brought significant benefits across various sectors, from smart homes to industrial automation. However, this growth also introduces substantial cybersecurity challenges. Traditional security solutions often struggle to meet the unique requirements of IoT devices, which are typically constrained in terms of power, processing capabilities, and memory [14]. In this section, we will review existing cybersecurity solutions for IoT networks, compare various lightweight key establishment protocols, and discuss the limitations of current approaches. We will also explore how OSCORE and EDHOC address these limitations.

2.2. Existing Cybersecurity Solutions for IoT Networks

Many existing cybersecurity solutions for IoT networks are adaptations of protocols and techniques used in traditional computing environments. These include:

- **Transport Layer Security (TLS) and Datagram TLS (DTLS):** TLS and DTLS provide robust security for communications over the internet by ensuring data confidentiality, integrity, and authenticity. While TLS is designed for reliable transport like TCP, DTLS adapts it for use with unreliable transport like UDP. These protocols are widely used but can be too resource-intensive for many IoT devices [15].
- **Internet Protocol Security (IPsec):** IPsec is a suite of protocols that provide security at the IP layer, ensuring secure data transmission between devices. Although IPsec is highly effective, it is often too complex and heavyweight for constrained IoT environments [16].
- **Message Queuing Telemetry Transport (MQTT) with SSL/TLS:** MQTT is a lightweight messaging protocol designed for low-bandwidth, high-latency networks. When combined with SSL/TLS, it can offer secure communication. However, the overhead introduced by SSL/TLS can be significant for constrained devices [17].
- **Lightweight Machine-to-Machine (LwM2M):** LwM2M is a protocol developed by the Open Mobile Alliance (OMA) for managing IoT devices. It includes security features such as bootstrap from smartcard, PSK, and RPK. While it is more suitable for IoT environments, it may not fully address the needs of all applications, particularly those requiring end-to-end security [18].

2.3. Comparison of Various Lightweight Key Establishment Protocols

To address the limitations of traditional security protocols in IoT environments, several lightweight key establishment protocols have been developed. These protocols are designed to provide secure communication while minimizing resource usage:

Table 1. Comparison of Lightweight Key Establishment Protocols.

Protocol	Security Level	Resource Consumption	Complexity	Latency
PANA	Medium	Low	Medium	Medium
IKEv2 with EAP	High	High	High	High
Zigbee APS	Medium	Low	Low	Low
CoAP with DTLS	High	Medium	Medium	High
OSCORE	High	Low	Low	Low
EDHOC	High	Low	Low	Low

2.4. Limitations of Current Approaches and How OSCORE and EDHOC Address Them

While the aforementioned protocols offer various levels of security, they also come with limitations that make them less than ideal for IoT environments:

2.4.1. Resource Consumption

- **PANA:** While it has low resource consumption, it relies on EAP methods, which can vary in complexity and resource requirements [19].
- **IKEv2 with EAP:** This protocol is highly secure but consumes significant computational and memory resources, which are often scarce in IoT devices [20].
- **Zigbee APS:** Although it has low resource consumption, its security mechanisms may not be robust enough for more demanding applications [21].
- **CoAP with DTLS:** DTLS adds significant overhead, affecting the performance of constrained devices [22].

2.4.2. Complexity

- **PANA and IKEv2 with EAP:** Both protocols can be complex to implement and manage, especially in large-scale IoT networks [23].
- **CoAP with DTLS:** The complexity of managing DTLS sessions can be a barrier for IoT deployments [22].

2.4.3. Latency

- **IKEv2 with EAP and CoAP with DTLS:** The time required to establish secure communication can introduce latency, which is critical in time-sensitive applications [24].

2.5. How OSCORE and EDHOC Address These Limitations

OSCORE:

- **Efficient Security:** OSCORE secures the application data itself, making it resilient to attacks on intermediary nodes. This reduces the overhead associated with traditional transport-layer security [25].
- **Low Resource Consumption:** It uses COSE (CBOR Object Signing and Encryption) to provide confidentiality, integrity, and authenticity with minimal resource usage.
- **Simplicity:** OSCORE is simpler to implement and manage compared to traditional protocols like DTLS, making it more suitable for large-scale IoT deployments [26].

EDHOC:

- **Reduced Latency:** EDHOC significantly reduces the number of round-trips needed for key establishment, minimizing latency [26].
- **Flexibility:** It supports various authentication methods, including PSK, RPK, and certificates, providing flexibility while maintaining strong security properties like forward secrecy.
- **Low Resource Consumption:** EDHOC is designed to operate efficiently in constrained environments, ensuring secure communication without overloading the device's resources [27,28].

2.6. Literature Review

As secure communication is essential within IoT networks, many studies were conducted on advanced lightweight encryption algorithms. In the article [29], various challenges to IoT environments, such as power consumption of devices, limited battery, memory space, performance cost, and security, were listed. The same article pointed out that a constrained device's security can be subject to flexibility, emphasizing the importance of security metrics. The standard protocol for application-layer protocol is Hypertext Transfer Protocol (HTTP). However, it has proven to be heavy and inefficient when implemented on constrained, battery-powered devices [30].

Consequently, a simpler alternative, namely CoAP, was introduced for IoT nodes. There is a variety of methods to achieve secure communication for CoAP, one of which is applying OSCORE. As evaluated in the article [31], compared to other alternatives, such as Datagram Transport Layer Security (DTLS), OSCORE offers many advantages, including the possibility to deploy on non-trusted proxies through object security. Before working with the OSCORE protocol, a secure context needs to be established. This can be done through EDHOC, a lightweight authenticated key exchange protocol. One of EDHOC's advantages over Transport Layer Security (TLS) is the reduced amount of roundtrips [32]. Moreover, the EDHOC handshake process requires only three message with a fourth optional message, while through DTLS it can consist of up to 13 [33].

Evidence from a recent study [33] indicates that DTLS uses at least x7 times more time on air and x4 times more FLASH and RAM compared to EDHOC. After its standardization, OSCORE's performance, together with EDHOC, was compared to that of different protocols, demonstrating promising results. When OSCORE is used together with the key exchange protocol EDHOC, a times five reduction over DTLS has been demonstrated in terms of transmitted bytes [34]. While there are promising results, it is important thorough testing and valuation are done as the protocols were introduced recently.

A study by Kim [35] presented a detailed analysis of potential vulnerabilities of the EDHOC protocols, including susceptibility to man-in-the-middle attacks during the key exchange phase. Their findings suggest it is crucial to evaluate security against emerging threats continuously. Another study [36] delved into various key management methods, exploring their effectiveness and cost for IoT devices. Some strategies mentioned were PSK, RPK, and CBOR Web Tokens(CWT). The derived conclusions stated that PSK is commonly used as it is efficient and easier to implement, while RPK and CBOR Tokens have a higher computational demand and may be suitable for more powerful devices as they offer more robust security protection. In some studies about OSCORE and EDHOC protocols, [34], pre-shared keys were used for authentication. However, based on the documentation on the protocol, there is no support for PSK as static DH keys are sufficient [37]. It was explained that in initial versions, PSK was an option; however, in later drafts, the decision was retracted [38].

Although there are other ways to perform key sharing, the differences between these methods regarding security and resource usage need further evaluation. In [31,38], it is mentioned that hardware acceleration increases the performance of OSCORE and EDHOC. Additionally, the research done on Crypto-Hardware in the Low-end IoT by Santos [39] documents that by utilizing hardware acceleration for cryptographic operations, the overall energy cost and execution time can be significantly reduced. Nevertheless, there is still limited information about the exact impact hardware acceleration can make in terms of latency and memory requirements in a system implementing the OSCORE and EDHOC protocols.

An alternative approach for establishing a secure communication channel between IoT devices is Named Data Networking (NDN). NDN offers confidentiality by using authentication-based names instead of IP addresses for identification [40]. Its high reliability can be brought down to hop-wise transfers and network caches, reducing the need for retransmissions [41]. Therefore, in multiple-hop scenarios, NDN outperforms CoAP over DTLS and OSCORE [41]. Nevertheless, an OSCORE-oriented approach has advantages over NDN as it introduces a smaller security message overhead [41]. Figure 2 illustrates a better overview of the different approaches to secure IoT communications.

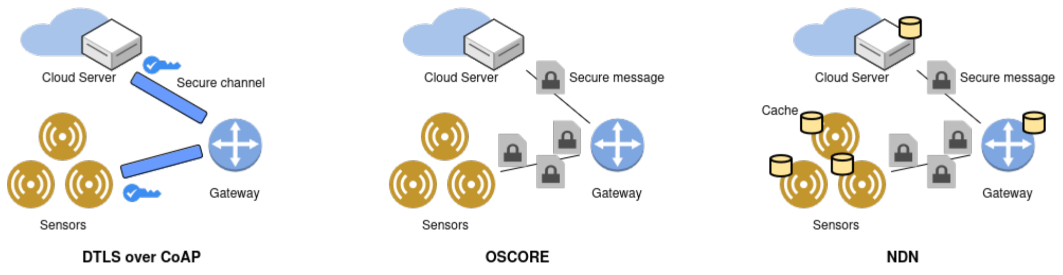

Figure 2. Comparison between security protocols.

While traditional cybersecurity solutions and existing lightweight key establishment protocols offer varying levels of protection for IoT networks, they also come with limitations in terms of resource consumption, complexity, and latency. OSCORE and EDHOC provide a promising alternative by delivering robust security tailored for constrained environments. Their efficient, flexible, and low-latency design makes them well-suited for the growing IoT landscape, ensuring secure and scalable communication.

3. Methodology

In this section, the research methodology will be discussed, addressing the hardware setup, software implementation, and used libraries together with the research metrics that will help answer the research questions.

3.1. Hardware Setup

The testing environment consisted of the following components. The experiments utilize the setup shown in Table 2. The way the components were connected are illustrated in Figure 3:

Table 2. Devices for Hardware Setup.

Device	Flash Memory	SRAM
2x Raspberry Pi 4B	32GB	4GB
ESP32-S	4MB	520KB
Thinkpad P15v	500GB	16GB
AVHzY USB-Meter C3	/	/

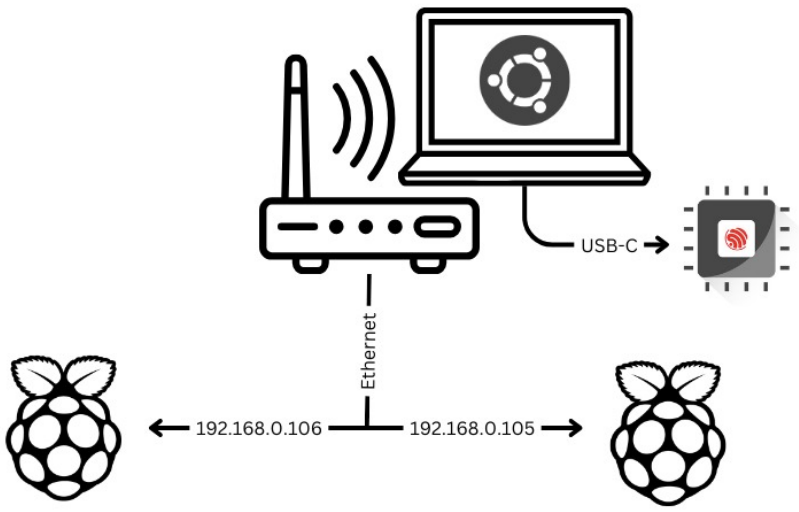

Figure 3. Hardware Setup.

The method used for measuring power consumption was taking the highest measured value throughout the program's execution. If there was a difference between multiple measurements, the average was presented.

3.2. Software Setup

3.2.1. uOSCORE-uEDHOC Library

The uOSCORE-uEDHOC library was installed on both Pis. The first Pi takes on the role of the initiator client, while the other one is a responder server. They use the library's corresponding sample implementations of both roles. Since the two Pis are connected via the same network, they were also assigned static IPs. There are four supported EDHOC methods: Signature keys for both initiator and responder, Static Diffie-Hellman keys for both initiator and responder and a combination of signature keys and Diffie-Hellman keys. Moreover, Zephyr OS was used to upload the project containing uOSCORE and uEDHOC protocol implementation on ESP32S so that the metrics could be evaluated for a microcontroller of a smaller size.

3.2.2. Libcoap Library

An additional library, namely libcoap, was used for its OSCORE implementation over DTLS. The library was installed on each Raspberry Pi 4 device, enabling them to communicate with each other via WiFi.

3.2.3. Edhoc and Libcoap Libraries

Libcoap, together with a third library called edhoc, was used for evaluating the performance of both OSCORE and EDHOC protocols on ESP32S as well as documenting the impact hardware acceleration has on the overall efficiency of the protocols. For the purpose of performing such tests, the project configuration options for hardware acceleration, namely "Use hardware AES acceleration", "Use hardware SHA acceleration", "Use hardware MPI (bignum) acceleration" were used. Currently, the edhoc library only supports EDHOC method 0 (EDHOC authenticated with asymmetric keys) and cipher suite 0 (AES-CCM-16-64-128, HMAC 256/256, X25519, EdDSA, Ed25519). It is important to mention that the edhoc library is only compatible with an older version of CoAP.

3.3. Research Metrics

A variety of metrics relevant to the computational overhead of resource-constrained devices were measured. As securing IoT communications is a pivotal goal in this research, metrics relating to the security of the implementation were evaluated. Below there is a list of relevant measurements related to the EDHOC and OSCORE protocols:

1. Power Consumption: The electric current (Amperes) and electric power (Watts) consumed by the devices [29].
2. Memory usage: the Flash or RAM requirements that need to be satisfied to use the protocols measured in bytes [31].
3. Computation Time: The time that certain processes take to execute [42]; in this study, it is measured in milliseconds or seconds, depending on the experiment.
4. Confidentiality: Ensuring no unauthorized individuals have access to the transmitted data [43].
5. Integrity: The data has not been altered by an unauthorized individual [43].
6. Authentication: The process of confirming the identity of a device or entity in the system [43].

4. Results

The setup mentioned in section 3.1 was used to evaluate the performance and security implications of using OSCORE and EDHOC protocols in resource-constrained IoT environments. The results are organized according to the research questions, providing insights into key management strategies, hardware accelerators' impact, and DTLS comparisons.

4.1. RQ1

The following section will summarise the findings related to answering the first research question 1.2. The first research question focuses on comparing the use of Diffie-Hellman Static Keys and Signature Keys as well as CBOR certificate authentication. The ESP32S device was used to obtain data related to the computational overhead of the authentication mechanisms on resource-constrained devices.

4.1.1. Power Consumption:

Between the three key management methods, there was no visible difference in terms of electric current and power; the measured values were consistently 0.025A and 0.129W. The power consumption was calculated based on the below Equation 1:

$$P = V \times I \tag{1}$$

where:

- P is the power in watts (W).
- V is the voltage in volts (V).
- I is the current in amperes (A).

4.1.2. Computation Time

After comparing the results from tests on the interaction between OSCORE and EDHOC with six different test vectors, two of which used Signature Keys, two which used Diffie-Hellman Static Keys, and two utilized CBOR certificates, the results showed a slight increase in latency when the Diffie-Hellman Static Keys were used. This can be explained by the fact that the Diffie-Hellman key exchange involves additional steps for generating and exchanging the keys in a secure way. In Table 3, the differences can be compared.

4.1.3. Memory Usage

It is important to note the total flash memory requirements for ESP32S were approximately 278528 bytes (or approximately 278 Kilobytes), which amounts to approximately 6,66% of the microcontroller's capacity of 4MB.

In Table 3, it can be seen the three methods have minimal differences when it comes to the usage of DRAM (Dynamic Random Access Memory) and IRAM (Instruction Random Access Memory).

4.1.4. Confidentiality

- Diffie-Hellman Static Keys: As DH keys provide a strong encryption mechanism that establishes communication over an insecure channel, even if attackers intercept a message, it will not be possible to decrypt the data. As the EDHOC suite uses the AES-CCM-16-64-128 algorithm, confidentiality is protected while keeping the protocols efficient in a constrained environment.
- Signature keys offer strong encryption, although they lack forward secrecy - the keys used to encrypt and decrypt information do not change, making the system vulnerable to retrospective attacks.
- Using PKI (Public Key Infrastructure), CBOR encoded X.509 certificates provide a strong level of confidentiality.

4.1.5. Integrity

Integrity in the system is maintained through the hashing algorithm SHA-256. Both DH-based key exchange and signature keys prevent the data from being tampered with by ensuring no modifications are made to the public keys used.

4.1.6. Authentication

The Diffie-Hellman key exchange poses a vulnerability of a man-in-the-middle attack. By using signature keys, authentication can be performed through digital signatures, confirming the identities of the parties trying to connect.

Table 3. Authentication Methods.

Type	EDHOC Method	Latency (sec)	IRAM (bytes)	DRAM (bytes)	FLASH (bytes)
Client request	Signature Keys	0.154	46656	8756	147456
Client request	Static DH Keys	0.164	46656	8756	147456
Server key derivation	Signature Keys	0.038	46656	8752	147456
Server key derivation	Static DH Keys	0.044	46656	8752	147456
Client authentication	CBOR certificate	2.827	46656	8808	229376
Server authentication	CBOR certificate	2.791	46656	8808	229376

These results indicate that the communication protocols exhibit low latency, which is crucial for time-sensitive IoT applications. The minimal time required for message transmission and acknowledgment highlights the efficiency of the OSCORE and EDHOC protocols in constrained environments.

4.2. RQ2

For the purpose of answering the second research question, the setup involving a laptop and microcontroller ESP32S was used. The microcontroller has support for the following types of hardware acceleration: AES, Random Number Generation (RNG) and SHA256. Previous studies have mentioned the possibility of improvement when such hardware acceleration is involved in the process; more information in section 2. The performance comparison between enabling and disabling hardware acceleration for the ESP32S in implementing the OSCORE and EDHOC protocols reveals minimal differences in performance, are illustrated in Figures 4,5,6 and 7. This observation can be attributed to several factors:

- Resource-Efficient Protocol Design: The EDHOC protocol is specifically designed for minimal computational and message overhead, as many resource-constrained devices do not have the capacity to perform intense operations. The protocol employs elliptic curve cryptography, which requires less computational effort and is thus efficient even without hardware acceleration.
- Optimized Library Implementation: The EDHOC library, edhoc,, available at Marco von Raumer's GitLab repository, utilizes the `mbedtls` library for cryptographic operations. This library is optimized for performance, leveraging efficient algorithms and data structures to minimize computational load. The use of `mbedtls_ecdh_calc_secret` together with other optimized functions ensures that the performance of resource-constrained devices remains high even without them having dedicated hardware acceleration.
- Minimal Message Overhead: The protocol's design minimizes the number of messages exchanged and their sizes, which reduces the time and computational resources required for each operation.

4.2.1. Computation Time:

- EDHOC Initialization (`edhoc_init_context`): Both configurations took nearly the same time (28 ms).
- Handling CoAP Message (`edhoc_handle_coap_message`): The average time with hardware acceleration enabled was slightly higher (98.33 ms) compared to when disabled (94.67 ms); however, such difference is negligible.
- OSCORE Context Creation (`create_oscore_ctx`): With hardware acceleration enabled, the average time was 54.75 ms, slightly more than the 52.8 ms without it.

4.2.2. Memory Usage:

Memory usage metrics can be observed in the following Table 4. There was consistency across both configurations for all operations with slight differences. The version where hardware acceleration was enabled consistently used less flash memory: 503872 bytes in comparison to 508758 for the server and 508707 compared to 513425. The difference is approximately 1%, which in such a system does not make a large impact.

4.2.3. Power Consumption:

Power measurements did not show a noticeable difference between the implementation with hardware acceleration and without it.

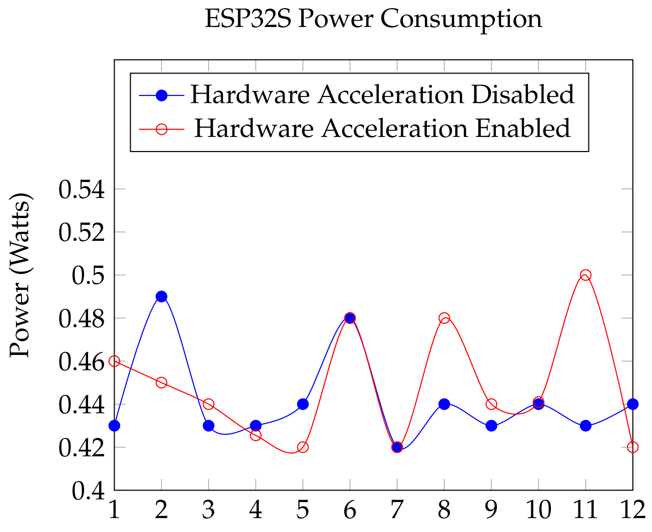

Figure 4. ESP32S Power Consumption (Watts)

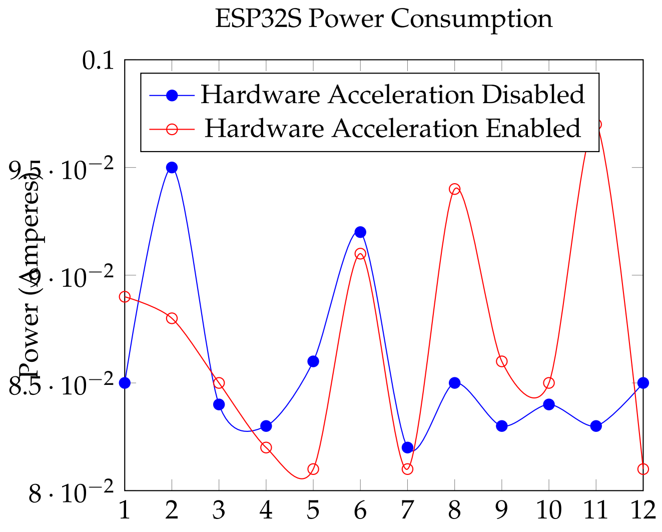

Figure 5. ESP32S Power Consumption (Amperes).

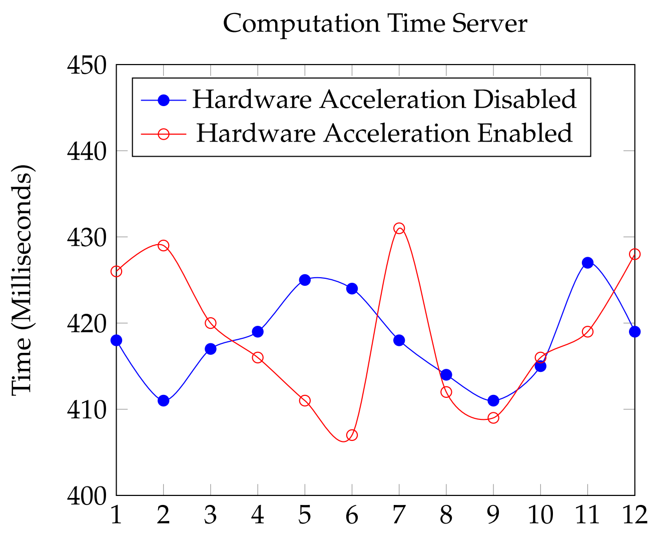

Figure 6. Computation Time for Server (Milliseconds).

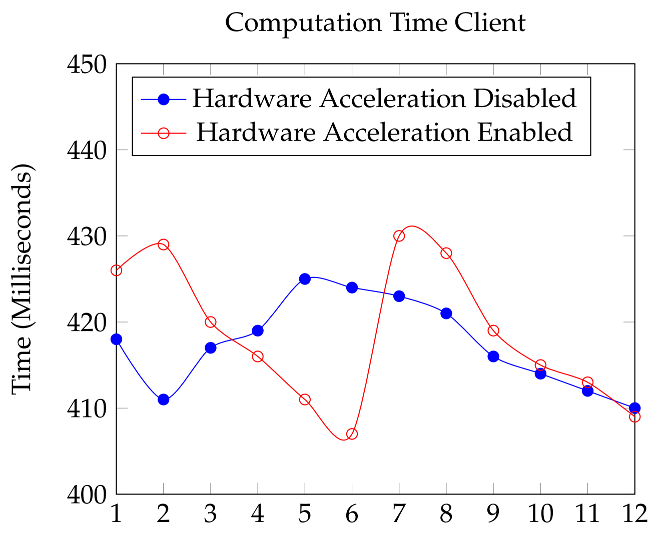

Figure 7. Computation Time for Client (Milliseconds).

Table 4. Average Flash Memory used in bytes.

Mode	Flash Memory	Flash Memory (Compressed)
Server HWA	785536	503872
Server no HWA	790608	508758
Client HWA	792624	508707
Client no HWA	797760	513425

4.3. RQ3

To compare the security context establishment over DTLS and EDHOC, both the libcoap and uOSCORE-uEDHOC libraries were used. Currently, the libcoap library does not support the EDHOC protocol, and it uses DTLS to establish the security context, while the uOSCORE-uEDHOC library has implemented the EDHOC protocol. The two libraries' implementations were compared through a scenario where the following actions happened: encrypting a request, sending it, receiving a response, decrypting it, and validating the results. The setup used for the tests included the two Raspberry Pi 4 devices, which were connected via WiFi; more information can be found in the methodology Subsection 3.1.

4.3.1. Power Consumption:

The maximum power consumed by the libcoap client side was 0.395 Amperes or 2.235 Watts. The power consumed by the other library's client was less, 0.301 Amperes or 1.536 Watts.

4.3.2. Computation time:

After using timing operations within the two systems, the measurements revealed it takes 2.04 seconds to establish secure context through libcoap, while with uOSCORE-eEDHOC, the operations and message exchange across the two devices takes 46 milliseconds.

4.3.3. Memory Usage:

The DRAM and SRAM usage by the libcoap client measured a total of 7049 KB, while that of the uSOCORE-uEDHOC client was significantly less, 2732 KB.

4.3.4. Confidentiality:

DTLS uses encryption of the entire communication channel, ensuring no unauthorized access is made. EDHOC ensures the confidentiality of individual messages by offering end-to-end encryption [37].

4.3.5. Integrity:

In DTLS, integrity is achieved by a Hash-based Message Authentication Code (HMAC) on the receiver side, allowing the detection of any altercation to the records. [44]. Conversely, EDHOC makes use of CBOR Object Signing and Encryption (COSE) for cryptography and identification of credentials [37], ensuring message integrity.

4.3.6. Authentication:

Authentication in DTLS is certificate-based, while EDHOC has more options, such as RPK or public-key certificates. In DTLS, there are three types of handshake: unauthenticated, server-authenticated, and fully authenticated. As the handshake phase includes multiple packet exchanges, it introduces significant communication and energy consumption overhead [45]. In contrast, EDHOC's handshake requires only three messages, making it a more compact alternative for constrained environments.

5. Discussion

The results from the previous Section 4 related to performance and security are analyzed for the purpose of providing a better understanding of the trade-offs and implications for enhancing cybersecurity in resource-constrained IoT environments.

5.1. Answering RQ1

The three authentication methods have certain advantages and possible disadvantages. DH static keys provide benefits such as forward secrecy and small message sizes. Given that the DH authentication mechanism is weaker and can be vulnerable to MiTM attacks, robust authentication should be ensured.

Authentication can be strengthened through digital signatures by utilizing signature keys. CBOR certificates reduce the risk of devices being impersonated in the network; however, as seen from the experimental data, the latency is significantly higher than that of the other methods. Moreover, the message overhead can increase, posing a burden for some resource-constrained devices.

Considering these factors, authentication mechanisms should be chosen according to the system's requirements. Hybrid approaches may be suitable in order to leverage the strengths and mitigate any potential weaknesses.

5.2. *Answering RQ2*

These results from the performed experiments suggest that the EDHOC protocol's design, coupled with the efficient implementation of the libcoap library, ensures high performance even without hardware acceleration. The similarity in performance metrics is proof of the protocol's suitability for constrained environments, providing robust security without imposing significant computational or energy overheads. While there is a slight difference between the memory usage between the two implementations, this can be explained by the fact that hardware acceleration offloads cryptographic computations to dedicated hardware, reducing the CPU and software load and the memory requirements.

By incorporating these findings, the study highlights that while hardware acceleration can enhance performance, the efficient design of the EDHOC protocol and its implementation ensures that IoT devices can maintain secure and efficient communication without supporting it. This is particularly beneficial for resource-constrained IoT environments where hardware acceleration may not always be feasible.

To achieve more conclusive results, it would be beneficial to perform more thorough tests on different microcontrollers with different types of hardware acceleration, as ESP32S is only one example.

5.3. *Answering RQ3*

The DTLS and EDHOC protocols offer a similar degree of security and available features. As the protocols are suitable for both IPv6 and IPv4, they are accessible for a variety of devices. Based on the measured results in the previous section, using uOSCORE-uEDHOC demonstrated a significant improvement from libcoap together with OSCORE with DTLS secure context establishment. It is important to keep in mind that the latter implementation also contained an improved version of OSCORE and EDHOC, which was designed specifically for constrained devices. The experiments highlight that the computational overhead was much smaller, which could be explained by the fact that securing CoAP messages through DTLS introduces significant delay as it requires decrypting the messages at proxies and a significant number of roundtrips.

When it comes to the security offered by both protocols, although achieved by different means, both ensure the communication between IoT devices is tamper-proof. Consequently, both protocols have been optimized for different usages and have considerable strengths. While DTLS may not be as tailored to constrained devices as EDHOC, its benefits are notable as its standardization is wider, meaning integration into existing technologies can be simpler. Furthermore, it is highly suitable for the continuous transfer of large volumes of data where the robust protection of an entire communication session is crucial. Nevertheless, in relation to resource-limited systems, EDHOC remains more lightweight for its efficiency and low overhead.

6. Conclusions

This research explored the performance of the OSCORE and EDHOC protocols for resource-constrained IoT devices. Different key management strategies, including Diffie-Hellman static keys, signature keys, and CBOR certificates, offer their respective strengths and weaknesses. Using DH static keys allows a balance between security and performance to be met; however, the system becomes vulnerable without proper authentication. Although there is no forward secrecy, signature keys offer strong encryption and integrity, and CBOR certificates, despite their high security, can introduce a significant delay in the program. These findings emphasize selecting authentication mechanisms based on specific application requirements is of key importance.

The impact of hardware acceleration on the EDHOC protocol revealed minimal impact in terms of computation time, memory usage, and power consumption. This efficiency of the protocol makes it suitable for environments where hardware acceleration is not possible, demonstrating that secure communication can be feasible with limited resources.

A comparison between two implementations, one using OSCORE and EDHOC and the other using only OSCORE with DTLS security context establishment, revealed higher power consumption and computational overhead when the latter was used. Consequently, the lightweight security protocols have proven to be better suited for devices with limited resources, although DTLS can have considerable strengths in continuous data transfer scenarios.

Overall, the research provides insights into the trade-offs between security, latency, and resource utilization in IoT networks using OSCORE and EDHOC protocols, highlighting the need for additional investigation into possible optimizations.

6.1. Future Work

This demonstration indicates that it is crucial to explore the possibilities of OSCORE and EDHOC as it would be beneficial to have wider standardization and use of lightweight protocols in IoT networks. Evaluating the protocols for multi-hop scenarios would provide insight into possible packet loss. Future improvements to the EDHOC and OSCORE implementation may include adding a caching mechanism similar to NDN's for better performance in multi-hop scenarios. Moreover, it can be beneficial to see what impact crypto-accelerators can make when different microcontrollers are used instead.

References

1. Berte, D.R. Defining the IoT. *Proceedings of the International Conference on Business Excellence* **2018**, *12*, 118–128. doi:doi:10.2478/picbe-2018-0013.
2. Fneish, Z.A.A.M.; El-Hajj, M.; Samrouth, K. Survey on iot multi-factor authentication protocols: A systematic literature review. 2023 11th International Symposium on Digital Forensics and Security (ISDFS). IEEE, 2023, pp. 1–7.
3. El-Hajj, M.; Fadlallah, A.; Chamoun, M.; Serhrouchni, A. Secure PUF: Physically unclonable function based on arbiter with enhanced resistance against machine learning (ML) attacks. 2020.
4. Khan, M.N.; Rao, A.; Camtepe, S. Lightweight cryptographic protocols for IoT-constrained devices: A survey. *IEEE Internet of Things Journal* **2020**, *8*, 4132–4156.
5. Haterd, R.v.d.; El-Hajj, M. Enhancing Privacy and Security in IoT Environments through Secure Multiparty Computation. *Proceedings of International Conference on Intelligent Systems and New Applications* **2024**, *2*, 64–69. doi:10.58190/icisna.2024.92.
6. El-Hajj, M.; Beune, P. Decentralized zone-based PKI: A lightweight security framework for IoT ecosystems. *Information (Basel)* **2024**, *15*, 304.
7. Shelby, Z.; Hartke, K.; Bormann, C. The Constrained Application Protocol (CoAP). Internet Engineering Task Force (IETF), 2014.
8. Selander, G.; Mattsson, J.; Palombini, F.; Seitz, L. RFC 8613: Object Security for Constrained RESTful Environments (OSCORE), 2019.
9. Elhajj, M.; Jradi, H.; Chamoun, M.; Fadlallah, A. Lasii: Lightweight authentication scheme using iota in iot platforms. 2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet). IEEE, 2022, pp. 74–83.
10. Palombini, F.; Tiloca, M.; Höglund, R.; Hristozov, S.; Selander, G. Using Ephemeral Diffie-Hellman Over COSE (EDHOC) with the Constrained Application Protocol (CoAP) and Object Security for Constrained RESTful Environments (OSCORE). Internet-Draft draft-ietf-core-oscore-edhoc-11, Internet Engineering Task Force, 2024. Work in Progress.
11. Cottier, B.; Pointcheval, D. Security Analysis of the EDHOC protocol, 2022, [arXiv:cs.CR/2209.03599].
12. Malwarebytes. What was the Mirai Botnet? Malwarebytes, 2023.
13. Peccerillo, B.; Mannino, M.; Mondelli, A.; Bartolini, S. A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives. *Journal of Systems Architecture* **2022**, *129*, 102561. doi:https://doi.org/10.1016/j.sysarc.2022.102561.
14. El-Hajj, M.; Fadlallah, A.; Chamoun, M.; Serhrouchni, A. A survey of internet of things (IoT) authentication schemes. *Sensors* **2019**, *19*, 1141.

15. Fossati, T. RFC 7925: Transport Layer Security (TLS)/Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things, 2016.
16. Raza, S.; Duquennoy, S.; Höglund, J.; Roedig, U.; Voigt, T. Secure communication for the Internet of Things—a comparison of link-layer security and IPsec for 6LoWPAN. *Security and Communication Networks* **2014**, *7*, 2654–2668.
17. Paris, I.L.B.M.; Habaebi, M.H.; Zyoud, A.M. Implementation of SSL/TLS security with MQTT protocol in IoT environment. *Wireless Personal Communications* **2023**, *132*, 163–182.
18. Singh, D.; Singh, R.; Gupta, A.; Pawar, A.V. Message queue telemetry transport and lightweight machine-to-machine comparison based on performance efficiency under various scenarios. *International Journal of Electrical and Computer Engineering* **2022**, *12*, 6293.
19. Sanchez-Gomez, J.; Garcia-Carrillo, D.; Marin-Perez, R.; Skarmeta, A.F. Secure authentication and credential establishment in narrowband IoT and 5G. *Sensors* **2020**, *20*, 882.
20. Aboelfotoh, R.M.A. Quality of service and privacy in internet of things dedicated to healthcare. PhD thesis, Université d'Avignon; American university in Cairo, 2021.
21. Gavra, V.; Pop, O.A.; Dobra, I. A Comprehensive Analysis: Evaluating Security Characteristics of Xbee Devices against Zigbee Protocol. *Sensors* **2023**, *23*, 8736.
22. Kumar, P.M.; Gandhi, U.D. Enhanced DTLS with CoAP-based authentication scheme for the internet of things in healthcare application. *The Journal of Supercomputing* **2020**, *76*, 3963–3983.
23. Pagliusi, P.S.; Mitchell, C.J. PANA/IKEv2: an Internet authentication protocol for heterogeneous access. International Workshop on Information Security Applications. Springer, 2003, pp. 135–149.
24. Bhattacharjya, A.; Zhong, X.; Wang, J.; Li, X. CoAP—application layer connection-less lightweight protocol for the Internet of Things (IoT) and CoAP-IPSEC Security with DTLS Supporting CoAP. *Digital twin technologies and smart cities* **2020**, pp. 151–175.
25. Gunnarsson, M.; Brorsson, J.; Palombini, F.; Seitz, L.; Tiloca, M. Evaluating the performance of the OSCORE security protocol in constrained IoT environments. *Internet of Things* **2021**, *13*, 100333.
26. Hristozov, S.; Huber, M.; Xu, L.; Fietz, J.; Liess, M.; Sigl, G. The cost of OSCORE and EDHOC for constrained devices. Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy, 2021, pp. 245–250.
27. Jacomme, C.; Klein, E.; Kremer, S.; Racouchot, M. A comprehensive, formal and automated analysis of the {EDHOC} protocol. 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 5881–5898.
28. Fraile, L.P.; Fournaris, A.; Koulamas, C. Design and performance evaluation of an embedded EDHOC module. 2021 10th Mediterranean Conference on Embedded Computing (MECO). IEEE, 2021, pp. 1–6.
29. Singh, S.; Sharma, P.; Moon, S.; others. Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions. *Journal of Ambient Intelligence and Humanized Computing* **2024**, *15*, 1625–1642. doi:10.1007/s12652-017-0494-4.
30. Levä, T.; Mazhelis, O.; Suomi, H. Comparing the cost-efficiency of CoAP and HTTP in Web of Things applications. *Decision Support Systems* **2014**, *63*, 23–38. 1. Business Applications of Web of Things 2. Social Media Use in Decision Making, doi:https://doi.org/10.1016/j.dss.2013.09.009.
31. Gunnarsson, M.; Brorsson, J.; Palombini, F.; Seitz, L.; Tiloca, M. Evaluating the performance of the OSCORE security protocol in constrained IoT environments. *Internet of Things* **2021**, *13*, 100333. doi:https://doi.org/10.1016/j.iot.2020.100333.
32. Bruni, A.; Sahl Jørgensen, T.; Grønbech Petersen, T.; Schürmann, C. Formal Verification of Ephemeral Diffie-Hellman Over COSE (EDHOC). Security Standardisation Research; Cremers, C.; Lehmann, A., Eds.; Springer International Publishing: Cham, 2018; pp. 21–36.
33. Fedrecheski, G.; Vučinić, M.; Watteyne, T. Performance Comparison of EDHOC and DTLS 1.3 in Internet-of-Things Environments. IEEE Wireless Communications and Networking Conference; , 2024.
34. Höglund, R.; Tiloca, M.; Selander, G.; Mattsson, J.P.; Vučinić, M.; Watteyne, T. Secure Communication for the IoT: EDHOC and (Group) OSCORE Protocols. *IEEE Access* **2024**, *12*, 49865–49877. doi:10.1109/ACCESS.2024.3384095.
35. Kim, J.; Duguma, D.G.; Lee, S.; Kim, B.; Lim, J.; You, I. Scrutinizing the Vulnerability of Ephemeral Diffie–Hellman over COSE (EDHOC) for IoT Environment Using Formal Approaches. *Mobile Information Systems* **2021**, *2021*, 1–18. doi:10.1155/2021/7314508.

36. Keoh, S.L.; Kumar, S.S.; Tschofenig, H. Securing the Internet of Things: A Standardization Perspective. IEEE Internet of Things Journal, 2014. doi:10.1109/JIOT.2014.2312291.
37. Selander, G.; Mattsson, J.P.; Palombini, F. Ephemeral Diffie-Hellman Over COSE (EDHOC). Internet Requests for Comments, 2024. doi:10.17487/RFC9528.
38. Hristozov, S.; Huber, M.; Xu, L.; Fietz, J.; Liess, M.; Sigl, G. The Cost of OSCORE and EDHOC for Constrained Devices. Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy; Association for Computing Machinery: New York, NY, USA, 2021; CODASPY '21, p. 245–250. doi:10.1145/3422337.3447834.
39. Santos, T.P.; Chaves, R.; Homsirikamol, E.; Rogawski, M.; Adamo, O. Efficient Implementation of Lightweight Cryptographic Algorithms for Securing IoT Devices. Cryptology ePrint Archive, Report 2021/058, 2021.
40. Zhang, Z.; Yu, Y.; Zhang, H.; Newberry, E.; Mastorakis, S.; Li, Y.; Afanasyev, A.; Zhang, L. An Overview of Security Support in Named Data Networking. *IEEE Communications Magazine* **2018**, *56*, 62–68. doi:10.1109/MCOM.2018.1701147.
41. Gündogan, C.; Amsüss, C.; Schmidt, T.C.; Wählisch, M. IoT Content Object Security with OSCORE and NDN: A First Experimental Comparison. Proc. of 19th IFIP Networking Conference; IEEE Press: Piscataway, NJ, USA, 2020; pp. 19–27.
42. Harris-Birtill, D.; Harris-Birtill, R., Chapter 12 Understanding Computation Time: A Critical Discussion of Time as a Computational Performance Metric; Brill: Leiden, The Netherlands, 2021; pp. 220 – 248. doi:10.1163/9789004470170_014.
43. Stallings, W.; Brown, L. *Computer Security: Principles and Practice*; 2007.
44. Kothmayr, T.; Schmitt, C.; Hu, W.; Brünig, M.; Carle, G. DTLS based security and two-way authentication for the Internet of Things. *Ad Hoc Networks* **2013**, *11*, 2710–2723. doi:https://doi.org/10.1016/j.adhoc.2013.05.003.
45. Kwon, H.; Park, J.; Kang, N. Challenges in Deploying CoAP Over DTLS in Resource Constrained Environments. Information Security Applications; Kim, H.w.; Choi, D., Eds.; Springer International Publishing: Cham, 2016; pp. 269–280.