Incorporating OCSP Stapling in EDHOC for Certificate Revocation in Resource Constrained Environments

by

Yousef AbdElKhalek

Matriculation Number 0474336

A thesis submitted to

Technische Universität Berlin
School IV - Electrical Engineering and Computer Science
Department of Telecommunication Systems
Service-centric Networking

Master's Thesis

January 15, 2023

Supervised by:
Prof. Dr. Axel Küpper

Assistant supervisor:
Aljoscha Schulte, M. Eng.

Industrial supervisor:
Shahid Raza, Associate Prof.

Eidestattliche Erklärung / Statutory Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed.

Berlin, January 15, 2023 Yousef AbdElKhalek

Acknowledgments

I would like to thank Shahid Raza and Göran Selander for making this Masters thesis possible, their guidance and support throughout the whole journey of performing this research was incredible and I've gained a gold mine of knowledge and experience from them.

I would like to thank the SNET Department and Prof.Dr. Axel Kuepper for agreeing to supervise this thesis.

My greatest thanks to my amazing supervisor Aljoscha Schulte, his continuous encouragement throughout the thesis and complete belief in my abilities is what made some of the dark days better. Thanks to him, the whole process of getting feedback and support was a breeze. Thank you again for being an amazing person.

Big thanks to Marco Tiloca for the beautiful technical discussions, the amount of knowledge I gained about the topic and surrounding research is unparalleled thanks to him. Big thanks to Rikard Höglund for helping with the measurements and code debugging and again for the lovely discussions.

Thanks to Joel Höglund for being a master in problem formulation and teaching me how to become one as well, my discussions with him contributed to setting the foundation for the research.

Thank you to my friends for bearing with me when I missed out on everything because I was too busy working on this.

My lovely girlfriend, I appreciate you and the endless support you have given me throughout this thesis journey.

Last but not least, a warm thank you to my family for always pushing me to be better, and bearing with my mood swings and supporting me throughout this research.

Abstract

Internet of Things (IoT) solutions are continuously gaining popularity, the rising adoption of the technology is accompanied by continuous research and development regarding costs optimization and sustainability. Having a plethora of devices transmitting sensor data and awaiting server commands is very costly in terms of processor utilization, each device has to maintain a connection session with the server. When it comes to scaling within IoT, the goal would be to have a large number of inexpensive nodes (devices). To achieve the mentioned inexpensive property; the characteristics of these nodes have to be scaled down, and the same goes for the networks built out of them. This introduces the concept of resource constrained devices, these are devices which aren't equipped with the characteristics that would normally be taken for granted in regular internet nodes. When the private key of an endpoint or certificate authority is compromised, any certificate signed by them is revoked. While access to certificate revocation information is somewhat handled outside the IoT, the acquiring of revocation information in resource constrained environments presents itself as a challenge.The proposed solution for certificate revocation in resource constrained environments successfully utilizes standard protocols to achieve the revocation functionality. By specifying integration vectors accompanied with each design, the proposed changes in the standard protocols maintain the current functionality of the latter and the integrity of the security mechanisms in place. The research firstly proposes a Lightweight OCSP profile, results of the latter are evaluated and found to give a reduction of $\approx$ 76.2% of the message size of the current profile of the OCSP response. The proposed certificate revocation approach utilizing EDHOC and OCSP Stapling proves to build a solid foundation for certificate revocation in IoT, with the approach being tested on a microcontroller in a resource constrained setting, the overhead in terms of power consumption on the microcontroller's side was calculated to be a 30% increase in total power consumption with respect to running plain EDHOC(EDHOC without Certificate Revocation), the 30% increase corresponds to 1.26 Watts when running EDHOC initiator on an nRF52840DK over 6LoWPAN.

Zusammenfassung

Lösungen für das Internet der Dinge (IoT) erfreuen sich immer größerer Beliebtheit. Die steigende Akzeptanz der Technologie wird von kontinuierlicher Forschung und Entwicklung im Hinblick auf Kostenoptimierung und Nachhaltigkeit begleitet. Eine Vielzahl von Geräten, die Sensordaten übertragen und auf Serverbefehle warten, ist in Bezug auf die Prozessorauslastung sehr kostspielig, da jedes Gerät eine Verbindungssitzung mit dem Server aufrechterhalten muss. Wenn es um die Skalierung innerhalb des IoT geht, besteht das Ziel darin, eine große Anzahl von kostengünstigen Knoten (Geräten) zu haben. Um die erwähnte kostengünstige Eigenschaft zu erreichen, müssen die Eigenschaften dieser Knoten verkleinert werden, und dasselbe gilt für die aus ihnen aufgebauten Netzwerke. Dies führt das Konzept der ressourcenbeschränkten Geräte ein, d. h. Geräte, die nicht mit den Merkmalen ausgestattet sind, die bei normalen Internetknoten als selbstverständlich angesehen werden. Wenn der private Schlüssel eines Endpunkts oder einer Zertifizierungsstelle kompromittiert wird, wird jedes von ihnen signierte Zertifikat widerrufen. Während der Zugang zu Zertifikatswiderrufsinformationen außerhalb des IoT in gewisser Weise gehandhabt wird, stellt die Beschaffung von Widerrufsinformationen in ressourcenbeschränkten Umgebungen eine Herausforderung dar Die vorgeschlagene Lösung für den Zertifikatswiderruf in ressourcenbeschränkten Umgebungen nutzt erfolgreich Standardprotokolle, um die Widerrufsfunktionalität zu erreichen. Durch die Angabe von Integrationsvektoren die mit jedem Entwurf einhergehen, behalten die vorgeschlagenen Änderungen an den Standardprotokollen die aktuelle Funktionalität der letzteren und die Integrität der vorhandenen Sicherheitsmechanismen bei. Die Forschungsarbeit schlägt zunächst ein leichtgewichtiges OCSP-Profil vor, dessen Ergebnisse ausgewertet werden und das eine Verringerung der Nachrichtengröße des aktuellen Profils der OCSP-Antwort um $\approx$ 76,2% ergibt. Der vorgeschlagene Ansatz für den Widerruf von Zertifikaten unter Verwendung von EDHOC und OCSP Stapling erweist sich als solide Grundlage für den Widerruf von Zertifikaten im IoT. Bei Tests des Ansatzes auf einem Mikrocontroller in einer ressourcenbeschränkten Umgebung wurde der Overhead in Bezug auf den Stromverbrauch auf Seiten des Mikrocontrollers als 30%ige Erhöhung des Gesamtstromverbrauchs im Vergleich zur Ausführung von einfachem EDHOC (EDHOC ohne Widerruf von Zertifikaten) berechnet, die 30%ige Erhöhung entspricht 1,26 Watt bei der Ausführung des EDHOC-Initiators auf einem nRF52840DK über 6LoWPAN.

Contents

1 Introduction

1.1 Motivation

IoT solutions are continuously gaining popularity, the rising adoption of the technology is accompanied by continuous research and development regarding costs optimization and sustainability. Having a plethora of devices transmitting sensor data and awaiting server commands is very costly in terms of processor utilization, each device has to maintain a connection session with the server. In the case of wireless networks, the power requirements are increased with RF antennas being introduced to the infrastructure.

Low-power Wide-area network (LPWAN)[1] introduced a group of technologies such as Narrow Band IoT (NB-IoT) and Low-range Wider-area network (LORaWAN). The technologies aim to offer reliable connectivity at optimised power requirements, thus addressing issues regarding sustainability of wireless network infrastructures in IoT.

When it comes to scaling within IoT, the goal would be to have a large number of inexpensive nodes (devices). To achieve the mentioned inexpensive property; the characteristics of these nodes have to be scaled down, and the same goes for the networks built out of them. This introduces the concept of resource constrained devices[2], these are devices which aren't equipped with the characteristics that would normally be taken for granted in regular internet nodes.

The concept of resource constrained devices introduces the challenge of developing communication protocols that can overcome the unavailability of characteristics present in normal internet nodes, such as large storage space and computation power. The challenge becomes a question of how to achieve desirable functionality and performance with limited available resources.

Constrained Application Protocol (CoAP)[3] is a great example, it provides a request-response interaction model between application endpoints while meeting the requirements of low overhead and simplicity for constrained environments.

Security protocols are known to require large buffer sizes and computation power, both of these properties are unattainable in constrained nodes. This introduces the challenge of developing lightweight versions of these security protocols that can run on these devices, while maintaining security integrity. Accordingly, constrained devices were classified according to available data size(ram) and code size(flash), where class zero devices having less than 10kb wouldn't be able to directly communicate with the internet in a secure manner [2].

The emergence of the lightweight protocol suites introduces room to collectively leverage them, in order to achieve desirable security properties in resource constrained environments that would otherwise be unattainable.

Ephemeral Diffie-Hellman Over Concise Binary object representation (CBOR) Object Signing and Encryption (COSE) (EDHOC)[4] introduces a lightweight authenticated secure key exchange protocol designed for highly constrained settings, EDHOC is used to establish high quality cyrptographic keys that would be used for other lightweight security protocols.

1.2 Problem Statement

Session establishment is the precursor to communication between two endpoints. Establishing a session starts with authentication, Public Key Infrastructure (PKI) allows issuing digital certificates that authenticate the identity of users, devices or services. The digital certificate binds a public-key to an owner, and the certificate itself can be trusted as it is signed with the private key of a trusted authority. When the private key of an endpoint or certificate authority is compromised, any certificate signed by them is revoked. While access to certificate revocation information is somewhat handled outside the IoT, the acquiring of revocation information in resource constrained environments presents itself as a challenge. Constrained nodes don't have the resources to neither store revocation information nor keep a connection alive with a querying protocol. The compromise of the private-key belonging to a node in an IoT infrastructure opens room for all kinds of cyber attacks, where the compromised node can now act as a malicious entity in the network. IoT infrastructures need to have access to certificate revocation information, the question is how can this information be transported securely in resource constrained environments.

1.3 Research Goals

The research will investigate the incorporation of certificate revocation in IoT public key infrastructures, specifically considering an IoT infrastructure that implements resource constrained nodes, with the aim to contribute research work that would serve as a solid foundation to the topic, and be part of contributing to the scalability of the IoT utilizing resource constrained nodes.

For the solution to be practical and well adopted by the scientific body of the research surrounding the topic, the author has defined certain criteria for the solution, the defined criteria acts as a subset of the goals defined by the research.

The criteria for the potential solution to the problem statement is defined by as follows:

- The proposed solution should utilize standard protocols to ensure practicality and applicability.
- Proposed changes to protocols should maintain the current functionalty of the latter and the integrity of the security mechanisms in place.
- The proposed solution should be able to handle the challenges imposed by the constraints of resource constrained environments.
- The proposed solution must maintain the security of the certificate revocation process.
- The overhead of the proposed solution should be minimal and within the capabilities of resource constrained environments.

1.4 Research Methodology

The text presents and follows a methodology to perform the research, where firstly the study of the surrounding background knowledge and related work relative to the problem statement is performed, the acquired knowledge is used to identify protocols to be analysed, with the goal to acquire insights from the analysis that would create a foundation for the design phase. The design phase uses the acquired insights to present the ideas that contribute to achieving a solution to the problem statement. After the design phase; the research will implement, evaluate and discuss an approach to contribute to a potential solution to the presented problem statement. The usefulness of the research is largely dependent on how practical its findings are to implement in production grade use cases and how widely they are adopted.

1.5 Delimitations

The research adopts a communication scenario between a constrained client and a non-constrained server, the scope of the research focuses on how the proposal affects the constrained client and doesn't go into studying the non-constrained server end. The protocol libraries used in collaboration with the approach presented by the thesis are moving targets and aren't fully optimised for resource constrained nodes, meaning that flash and ram occupied by the compiled binaries goes beyond ranges assigned for constrained nodes regarding these storage sizes. The implementation phase only implements the necessary parts needed to acquire a proof of concept the demonstration, the implementation should not be treated as a production grade solution.

1.6 Outline

Chapter 2 of the text begins by establishing a firm background regarding secure IoT frameworks, public key infrastructures and resource constrained environments. The chapter then discusses related work in the scope of certificate revocation in public key infrastructures.

Chapter 3 outlines the methodology followed to perform the research, the protocol analysis phase and the experiment design from which the results will be acquired.

In Chapter 4, the thesis presents its proposal regarding a lightweight profile for OCSP and the accompanying security considerations for the proposal.

In Chapter 5, the thesis presents its proposal on incorporating certificate revocation in EDHOC and the accompanying overhead and security considerations of the proposed approach.

Chapter 6 breaks down and discusses the acquired results, presenting insights and conclusions on the proposed approaches and the research. The chapter also proposes future work as a continuation to the work of this thesis.

2 Background

This chapter will present the compiled background study surrounding the problem statement, the chapter will then mention related work in the scope of certificate revocation in public key infrastructures.

2.1 Public Key Cryptography

One-way functions are mathematical functions that are easy to compute given any input, but hard to invert given the output of some random input to the function. In this context, Computational complexity theory defines the terms "hard" and "easy". In Public Key Cryptography (PKC), two endpoints Alice and Bob use cryptographic algorithms to generate a key pair. The cryptographic algorithms referred to here are those incorporating one-way functions, where the "hard-to-invert" property is leveraged to create a set of private and public keys which are mathematically related. The keys are then used for encryption to establish a secure communication channel.

2.2 Public Key Infrastructure

Digital certificates bind a public key to an owner, this relationship is verified by a trusted source (Certificate Authority (CA)) that issues and signs a certificate. An example of a digital certificate is one that follows the X.509[5] standard, and those are called X.509 certificates. Running public-key cryptography with raw keys opens room for man in the middle attacks leading an adversary to establish a communication channel with a node in the network, using their own key pair for encryption. This leads to the theft and loss of sensitive data, and in other cases sabotaging entire systems.

2.3 X.509 PKI

A public key infrastructure provides a framework of encryption standards for communication over public networks. The infrastructure is built on a trust that is established among clients, servers and certificate authorities by the generation, exchange and verification of certificates.

The X.509 public key infrastructure[5] incorporates X.509 digital certificates in order to verify that a public key belongs to the user, computer or service identity contained in the certificate. The X.509 defines Certificate Revocation list (CRL)s, when a private key of a node is compromised; the public key certificate of that node is added to the CRL, this is called certificate revocation. Each node in the network is required to have a fresh copy of the CRL, the period of updating the local copy of the CRL is governed by the operational policy set by the organisation. In session establishment, the certificate of the communicating entity is cross referenced with the CRL to ensure that the entity can be trusted and that its certificate has not been revoked. Depending on the communication scenario, the same is done for the responding entity.

2.4 The IoT and CRLs

In the scope of the IoT, the use of CRLs for revocation posed major problems with scalability due to variations in size of the CRLs and the requirement that they would be kept on the device memory. As IoT incorporates resource constrained devices[2], the variations in size would not be acceptable. Delta CRLs introduced the option to update an on memory CRL, the problem remained the same as the size of the CRLs would not be bounded.

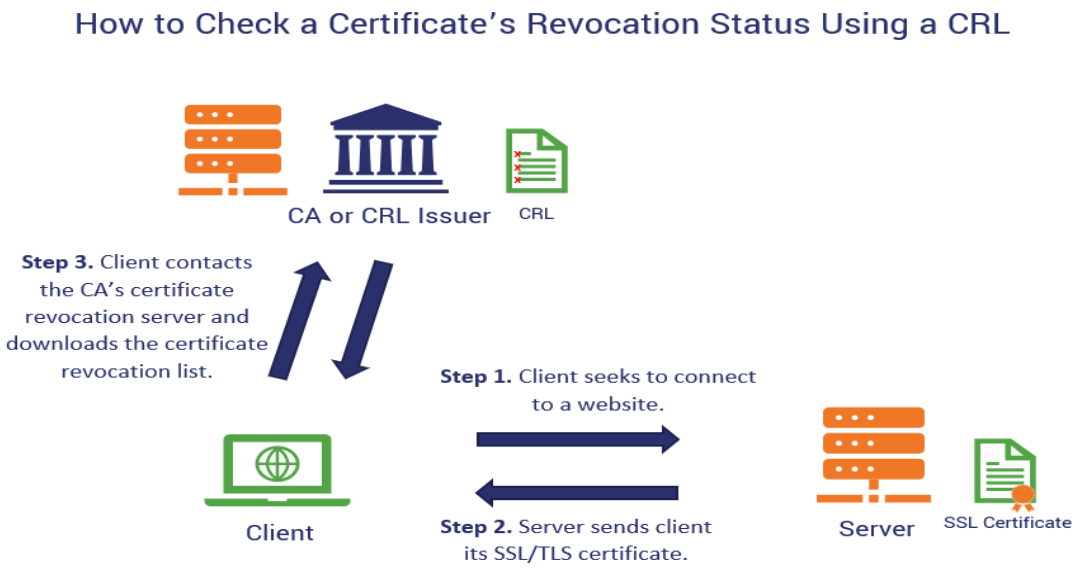

How to Check a Certificate's Revocation Status Using a CRL
CA or CRL Issuer
CRL
Step 3. Client contacts the CA's certificate revocation server and downloads the certificate revocation list.
Step 1. Client seeks to connect to a website.
Client
Step 2. Server sends client its SSL/TLS certificate.
Server
SSL Certificate

Figure 2.1: The process of a certificate revocation check when using a certificate revocation list [6]

Figure 2.1[6] shows the process of using a certificate revocation list to check the revocation status of the server's digital certificate, step 3 shows that the client contacts the CA's certificate revocation server and downloads the certificate revocation list. This approach would not work in the scope of the IoT and resource constrained devices.

2.5 The Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP)[7] introduced a request/response approach for certificate revocation, where the status of a certificate concerning revocation could be queried and a signed response would be generated regarding the validity of the certificate. An entity querying an OCSP server is called an OCSP client,the client is responsible to verify the signature of the OCSP response, using the OCSP responder's public key, the client must then verify the signature of the certificate under question using the appended public key of certificate authority. The OCSP server is also the OCSP responder, the latter can also be referred to as a

Validation Authority (VA).

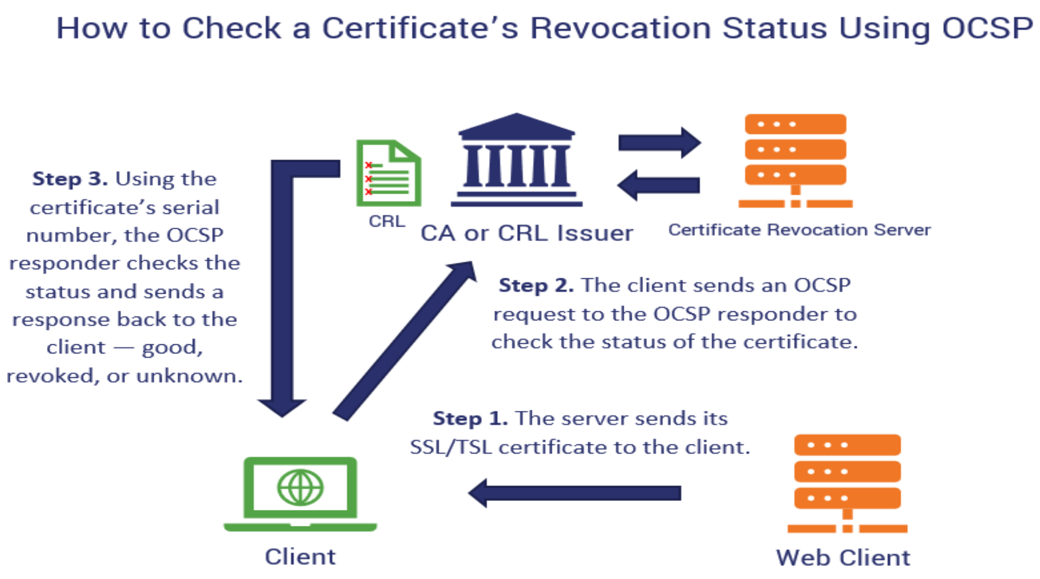

How to Check a Certificate's Revocation Status Using OCSP
Step 3. Using the certificate's serial number, the OCSP responder checks the status and sends a response back to the client — good, revoked, or unknown.
CRL
CA or CRL Issuer
Certificate Revocation Server
Step 2. The client sends an OCSP request to the OCSP responder to check the status of the certificate.
Step 1. The server sends its SSL/TSL certificate to the client.
Client
Web Client

Figure 2.2: The process of a certificate revocation check when using the OCSP[6]

Figure 2.2[6] shows the process of using the OCSP to check the revocation status of a digital certificate, in the case of this figure; the CA/CRL issuer is also the OCSP responder and the client performs an OCSP request using the server certificate's serial number. The client no longer downloads a CRL and is only required to process the OCSP response.

While the OCSP offers a significant improvement over the resource demands of CRLs, it has posed notable security and performance issues; (1) Due to the server-client model, OCSP responses are vulnerable to replay attacks. (2) if a relying party fails to establish a connection with the OCSP responder, a decision has to be made, which can either be to communicate anyway or to cease communication, both of which are not favourable cases regarding performance and security. OCSP also posed privacy concerns, as the client's metadata can be logged when the client performs an OCSP request, the metadata includes the client's IP address and time of visit(time of trying to establish a session with the server).

2.6 OCSP Stapling

The above issues with OCSP contributed to the emergence of OCSP stapling[8], by incorporating a Transport Layer Security (TLS) handshake extension, that allows relying parties to request a signed timestamped OCSP response along with the end entity's certificate when establishing a connection. This allows the end entity to forward either a cached OCSP response or a new one in the case of including a nonce in the OCSP request, this addresses the downfalls of plain OCSP. In the case of OCSP stapling, the OCSP response is referred to as the staple, and the staple is signed by an OCSP responder whose certificate is signed by a CA.

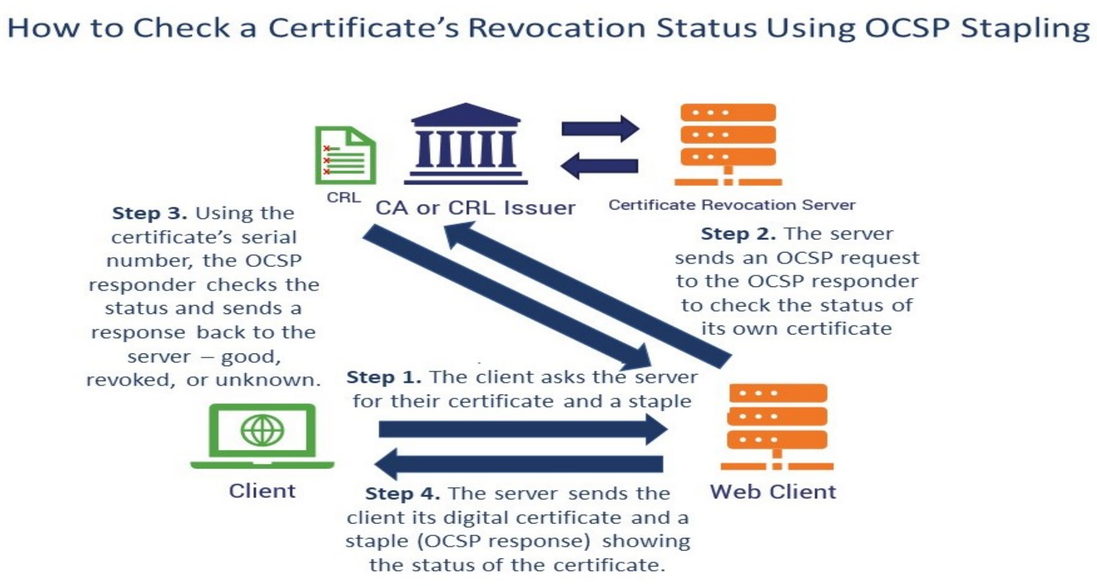

How to Check a Certificate's Revocation Status Using OCSP Stapling
CRL
CA or CRL Issuer
Certificate Revocation Server
Step 3. Using the certificate's serial number, the OCSP responder checks the status and sends a response back to the server – good, revoked, or unknown.
Step 2. The server sends an OCSP request to the OCSP responder to check the status of its own certificate
Step 1. The client asks the server for their certificate and a staple
Client
Web Client
Step 4. The server sends the client its digital certificate and a staple (OCSP response) showing the status of the certificate.

Figure 2.3: The process of a certificate revocation check when using OCSP Stapling - Adapted from[6]

Figure 2.3 - adapted from [6] - shows the process of acquiring certificate revocation status via OCSP stapling, the client sends a request for the server's certificate and a staple that includes the revocation status of the certificate. The server performs an OCSP request and includes the OCSP response along with the server certificate in the response sent to the client.

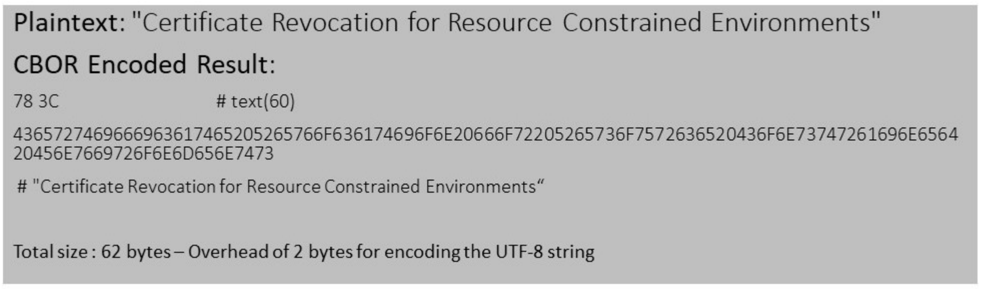
2.7 CBOR

Considering resource constrained nodes[2], the choice of encoding algorithm is a very critical one, constrained nodes should be able to encode and decode data structures without requiring a lot of computation power, and be able to do that without supporting a library that takes a significant amount of the flash size, referring here to the Abstract Syntax Notation 1 (ASN.1)[9]. The encoding rules defined by the encoding algorithm must also be optimised to yield relatively small message sizes compared to the case of non-constrained environments.

ASN.1 is used in X.509 certificates[5], the encoding is considered to be overly verbose for constrained IoT environments. At the time of writing this text; there is ongoing work regarding profiling X.509 certificates with CBOR encoding to reduce the certificate size significantly and achieve associated performance benefits.[10]

The CBOR[11] encoding is equipped with design goals which are very attractive for constrained settings, the design includes the possibility of extremely small code size and extensibility without the need to negotiate versions. CBOR allows map keys of any type, whereas JSON only allows strings as keys in object values, this already shows the flexibility of CBOR proving to be a preferred choice in resource constrained environments. CBOR is used by the lightweight security protocol suite including CoAP[3] and EDHOC[4].

CBOR uses a jump table for encoding depending on the structure/semantics of the item that will be encoded. An example for a CBOR encoding acquired using CBOR playground[12] is shown in Figure 2.4, the figure shows an example of encoding a Unicode Transformation Format (UTF)-8 string, the encoding only adds two bytes to the original length of the string, the first byte $0x78$ indicates that the item is a UTF-8 string and the second byte contains the length of the string.



Plaintext: "Certificate Revocation for Resource Constrained Environments"

CBOR Encoded Result:

```
78 3C                    # text(60)

43657274696669636174652052657766F636174696F6E20666F72205265736F7572636520436F6E73747261696E6564
20456E7669726F6E6D656E7473

 # "Certificate Revocation for Resource Constrained Environments“


Total size : 62 bytes – Overhead of 2 bytes for encoding the UTF-8 string
```

Figure 2.4: An example for encoding a UTF-8 string, created with [12]

2.8 EDHOC

EDHOC[4] is a lightweight authenticated key exchange protocol designed for use cases constituting resource constrained environments, EDHOC re-uses the lightweight security software suite of libraries; COSE[13] for cryptography, CBOR[11] for encoding and CoAP[3] for transport, this way additional code size is maintained to be low.

Datagram Transport Layer Security (DTLS)[14] is another protocol used to establish an authenticated, confidentiality and integrity-protected channel between two communicating peers, where TLS[15] requires a reliable transport channel which typically would be Transmission Control Protocol (TCP)[16], DTLS offers communication security protection for applications that use User Datagram Protocol (UDP)[17].

```
=====================================================================
 Flight                                     #1      #2      #3   Total
---------------------------------------------------------------------
 DTLS 1.3 - RPKs, ECDHE                    152     414     248     814
 DTLS 1.3 - Compressed RPKs, ECDHE         152     382     216     750
 DTLS 1.3 - Cached RPK, ECDHE              191     362     248     801
 DTLS 1.3 - Cached X.509, RPK, ECDHE       185     356     248     789
 DTLS 1.3 - PSK, ECDHE                     186     193      56     435
 DTLS 1.3 - PSK                            136     153      56     345
---------------------------------------------------------------------
 EDHOC - X.509, Signature, x5t, ECDHE       37     115      90     242
 EDHOC - X.509, Signature, kid, ECDHE       37     102      77     216
 EDHOC - RPK, Static DH, x5t, ECDHE         37      58      33     128
 EDHOC - RPK, Static DH, kid, ECDHE         37      45      19     101
=====================================================================
```

Figure 2.5: Comparison of message sizes of DTLS handhshake and EDHOC protocols in bytes[18]

A comparison between running EDHOC and DTLS with respect to message sizes shows how EDHOC is superior, Message size is a very important parameter to be considered when designing protocols that will operate in resource constrained environments, as the parameter directly correlates to round trip times and accordingly power consumption. Figure 2.5[18] shows message sizes when running both protocols with different methods for authentication, usingRaw Public Keys (RPK) and Pre-shared keys (PSK) in combination with Elliptic-Curve Diffie-Hellman Ephemeral (ECDHE).It is clearly seen that EDHOC is superior in terms of message sizes using any of the provided authentication options.

Figure 2.6 shows an EDHOC exchange, a client communicating EDHOC is referred to as an EDHOC initiator while the server is referred to as the EDHOC responder. The EDHOC messages contain External Authorization Data (EAD)fields, each message has its corresponding EAD field.

```
Initiator                                                   Responder
|                METHOD, SUITES_I, G_X, C_I, EAD_1                   |
+------------------------------------------------------------------>|
|                             message_1                              |
|                                                                    |
|      G_Y, Enc( ID_CRED_R, Signature_or_MAC_2, EAD_2 ), C_R          |
|<------------------------------------------------------------------+
|                             message_2                              |
|                                                                    |
|           AEAD( ID_CRED_I, Signature_or_MAC_3, EAD_3 )             |
+------------------------------------------------------------------>|
|                             message_3                              |
|                                                                    |
|                           AEAD( EAD_4 )                            |
|<- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                             message_4                              |
```

Figure 2.6: EDHOC message flow[4]

Figure 2.7[4] shows the structure of an EAD item in EDHOC.EAD items allow the integration of external security applications in EDHOC, where the processing of each item is defined in separate specifications as EAD items are opaque to EDHOC. The items are then added into the EAD fields of each EDHOC message, each EAD field is a CBOR sequence of EAD items where each of the latter is a CBOR sequence as well[4].

$$
\begin{aligned}
ead = (& \\
& ead\_label : int, \\
& ?ead\_value : bstr, \\
)&
\end{aligned}
$$

Figure 2.7: An Ead item[4]

EDHOC defines critical and non-critical EAD items, where the sign of the EAD label determines if an item is critical or not, a negative sign indicates that the EAD item is critical. Critical EAD items affect the state of the protocol, if a critical EAD item is received and the endpoint cannot recognize or process the item, it must send an error message and discontinue the protocol [4].

2.9 Constrained environments

In the scope of the IoT, message overhead and energy consumption majorly contribute to the design considerations of the infrastructure. Scalability in IoT eventually means resource constrained environments, communication protocols running on constrained nodes always welcome any approaches that achieve a required functionality while decreasing overhead and maintaining the integrity of the security properties of the infrastructure.

The Internet Engineering Task Force (IETF) produced RFC7228[2] which provide succint terminology for different classes of constrained devices, RFC7228 mentions that a resource constrained device can be constrained in terms of memory, energy and the incorporated strategy regarding using power for communication.

Table 2.1 shows classes of constrained devices with respect to their RAM and Flash sizes, class `C0` devices are very constrained that they are very unlikely to communicate directly to the internet in a secure manner.

	RAM size	Flash size
Class 0 (C0)	$\ll 10KiB$	$\ll 100KiB$
Class 1 (C1)	$\sim 10KiB$	$\sim 100KiB$
Class 2 (C2)	$\sim 50KiB$	$\sim 250KiB$

Table 2.1: Classes of Constrained Devices (KiB = 1024 bytes) - adapted from RFC[7228][2]

Constrained devices beyond Class 2 exist, they are less demanding from a standards development perspective as they can use existing protocols unchanged due to the relatively larger memory. For a device to be referred to as a constrained device/node it does not have to be constrained in memory; it can be constrained in terms of available energy or power or a combination of all three. Table 2.2 shows the terminology for devices which are energy constrained with respect to their power source. Devices running on batteries are said to be period energy-limited, how limited they are depends on the the period of recharging or replacing the battery. `E9` devices have no limitations on available energy but can still be constrained by having limitations on available power.

	Type of energy limitation	Example power source
E0	Event energy-limited	Event-based harvesting
E1	Period energy-limited	Battery that is periodically recharged or replaced
E2	Lifetime energy-limited	Non-replaceable primary battery
E9	No direct quantitative limitations to available energy	Mains-powered

Table 2.2: Classes of Energy Limitation - adapted from RFC[7228][2]

Table 2.3 shows terminology denoting how a device uses power for communication. When wireless transmission is used, the radio constitutes a big portion of the total energy consumed by the device. Where `P9` is a device that is always connected and accordingly has no constraints on power, while `P0` is used in settings where the device is only required to be on in certain intervals. `P1` utilizes power saving methods to stay on while maintaining efficient power consumption on the cost of communication performance i.e., latency.

	Strategy	Ability to communicate
P0	Normally-off	Reattach when required
P1	Low-power	Appears connected, perhaps with high latency
P9	Always-on	Always connected

Table 2.3: Strategies of using power for communication - adapted from RFC[7228][2]

2.10 Related Work

Tanner Lindemer, Samuel, 2019; Digital Certificate Revocation [19] is a 2019 Masters Thesis that proposes ways to adapt the OCSP protocol for performing certificate revocation in IoT, the research firstly shows that certificate revocation can be partially implemented on a constrained device without changing the specification of the OCSP. They do this by running CoAP to contact an OCSP responder via a CoAP-HTTP proxy. Then they compare that approach with another vector where they propose a profile for both the OCSP response and request. [19] concludes that it is much better to standardize a new protocol like their proposed profile for OCSP instead of bringing in the current specification of OCSP into IoT PKI.

This research uses some of the arguments made by [19] when they were proposing the new OCSP profile. This research is different in the sense of proposing a profile for the OCSP but also presenting an integration vector to maintain backwards compatibility of the current specification of OCSP, accordingly bringing in that specification into IoT PKI. Downfalls in [19] from the point of view of this research would be the idea of not considering the difficulty of standardising a new protocol instead of proposing a non intrusive change to an existing standard. This research also introduces the idea of transporting the acquired revocation information in a manner considered more suitable for resource constrained devices.

An internet draft titled CBOR Encoded X.509 Certificates [10] currently active in 2022 looks into converting X.509 certificates into CBOR encoded ones referred to as C509 certificates. The work is directly related to certificate revocation in IoT PKI. The internet draft is focusing on specifying C509 profiles for Certificate Revocation lists and the OCSP. C509 greatly decreases certificate sizes and accordingly any protocol messages that include these certificates. At the time of writing this research text, the section titled "C509 Online Certificate Status Protocol" has no content, just a "TODO". This research is in contact with the authors and aims to contribute to the C509 OCSP section in that internet draft with the lightweight OCSP profile proposed in this research.

Boudagdigue et.al, 2020, Cluster-based certificate revocation in industrial IOT networks using Signaling game [20] investigates a new distributed certificate revocation protocol proposing cluster-based certificate revocation mechanism for IIoT networks that utilizes game theory. The proposal sets up an IoT infrastructure in a way that it is a community of member nodes, where there is a community leader who uses a form of a game to perform the renew of certificates for the well behaving member nodes and revoke the certificate of malicious member nodes. Boudagdigue uses the Perfect Bayesian Equilibrium, for the community leader to be able to quickly and accurately revoke the untrusted member nodes and accordingly enhancing the security of the infrastructure. This research thinks that mainly the idea of cluster-based revocation could be an interesting approach when employed for resource constrained environments, utilizing the community of the constrained nodes to perform the revocation information without the involvement of external entities or protocols.

Duan, Li and Li, Yong and Liao, Lijun, 2018, Flexible certificate revocation list for efficient authentication in IoT [21] proposes two novel lightweight CRL protocols that are equipped with maximum flexbility targetting constrained IoT settings. Their proposal use the concepts of Merkle hash tree and the Bloom filter, by leveraging the latter they were able to achieve lower RAM and bandwidth consumption compared to the usage of conventional CRLs. This research is aware of the problems with CRLs and constrained environments and finds that the findings of [21] aren't clear on how they handle the scalability factor of CRLs, as in the continuous expansion of the CRL database, which would not be possible to handle on a constrained device.

3 Methodology

The chapter firstly presents the approach taken to perform the research throughout this thesis project in section 3.1, then it describes the communication scenario adopted by the research in section 3.2. Sections 3.3, 3.4, 3.5 and 3.6 outline the protocol analysis phase of the research methodology with the goal to acquire insights that will then serve as the foundation for the Design phase. AS the research proposes changes to standard communication protocols, section 3.7 will discuss how the research specifies security considerations for its proposal.section 3.8 will demonstrate the experiment setup, hardware and technology stack, and the planned measurements to be acquired by the research.

3.1 Research Approach

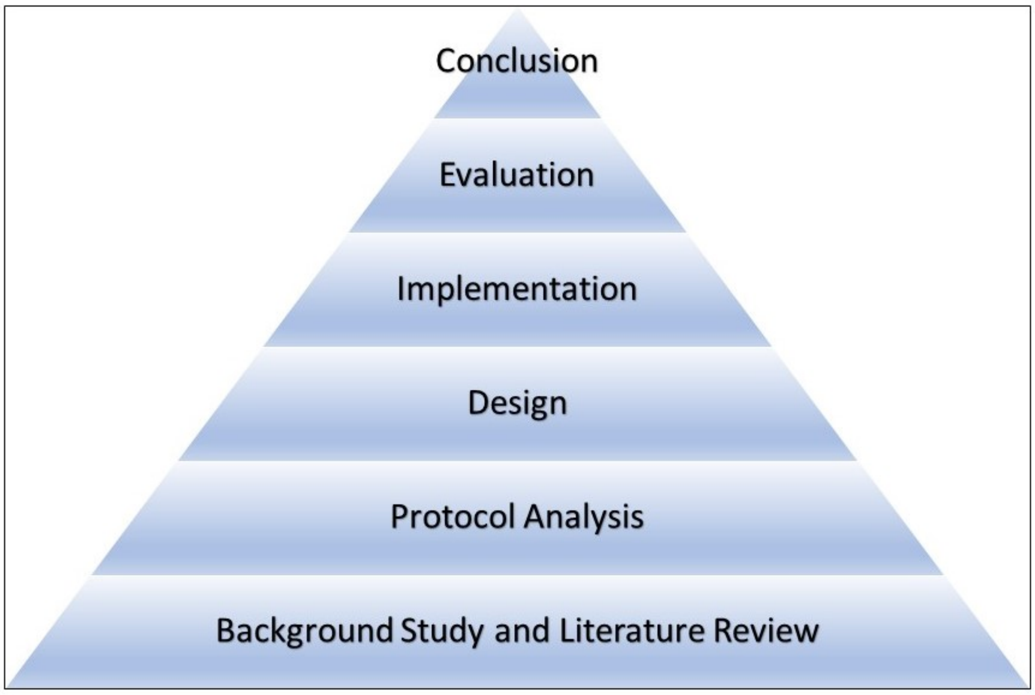


Figure 3.1: Approach taken to perform the research

Figure 3.1 shows the approach taken to perform the research. The research firstly performs a background study and literature review of surrounding work done relevant to the topic and problem statement. This study creates a foundation for the research work and contributes to solving the problem statement by compiling and presenting relevant information with respect to the topic. chapter 2 - "Background" will include the compiled knowledge.

Concluding the background study and literature review, the research identifies potential protocols to be analysed, the goal of the analysis is to derive insights which will then be used to propose an approach to solve the problem statement. The analysis is presented in section 3.3, section 3.4 and section 3.5. The research maintains the scope by considering a specific communication scenario discussed in section 3.2 - "Adopted communication scenario".

The research utilizes the acquired insights to design a proposal to solve the problem statement, in the design; the research will consider the incurred overhead and the practicality of the proposal. The research will specify security considerations for the proposed design, and will use RFC3552-Guidelines for Writing RFC Text on Security Considerations[22] as a guideline for that. The design is presented in chapters 4 and 5. The research will then implement the proposed design to acquire a working demonstration in preparation for evaluation. The implementation, file contents and structure is broken down in section 6.1

The research will set up an experiment in order to evaluate the implemented design, the evaluation will consider performance metrics proposed by the thesis, as well as the invasiveness and backwards compatibility of the proposed design with respect to the current state of the altered protocols. The experiment setup along with the planned measurements are discussed in this chapter as part of the methodology and data collection in section 3.8. The evaluation and results are discussed in section 6.2 and section 6.3

The research will then conclude on its findings and propose future work to move forward with the topic, hopefully using this research work as a solid foundation.

3.2 Adopted communication scenario

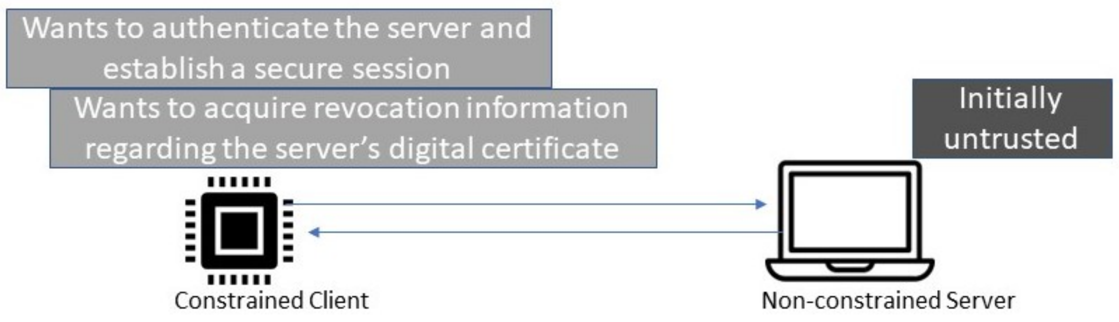

Wants to authenticate the server and establish a secure session
Wants to acquire revocation information regarding the server's digital certificate
Initially untrusted
Constrained Client
Non-constrained Server

Figure 3.2: Communication scenario in the scope of the research

Figure 3.2 shows the communication scenario considered by the research, the scenario in scope is that between a constrained client and a non-constrained server. The constrained client wants to establish a session with a non-constrained server. The constrained client aims to acquire revocation information regarding the server's certificate, to handle the case where a malicious entity can pose as a server using a compromised private key and a valid digital certificate due to the absence of mechanisms for acquiring certificate revocation information. Figure 3.2 will serve as a foundational figure for the upcoming sections.

3.3 Analysis of the OCSP

The background study has yielded the OCSP as the protocol to analyse when it comes to acquiring certificate revocation information the subsections below will expand on the analysis and present the role of the OCSP in the research.

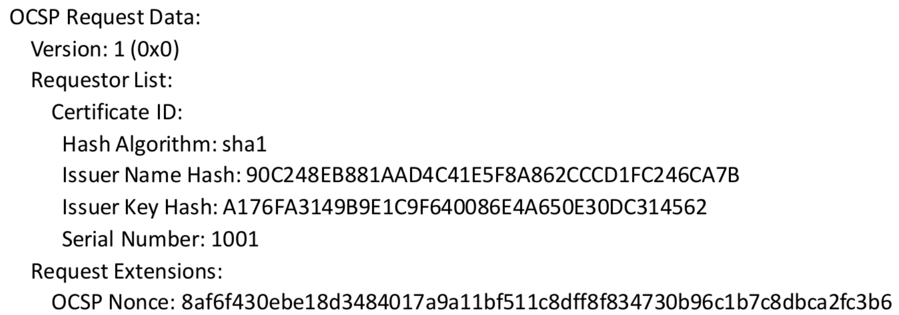
3.3.1 OCSP vs CRLs

The OCSP presents itself as an attractive option to acquire revocation information specifically in the scope of the IoT. IoT aims to employ resource constrained nodes in the solution infrastructure, when considering CRLs there is an uncapped storage requirement overhead to acquire revocation information. The constrained node needs to firstly store the CRL and then perform frequent updates to its own CRL database, due to limited flash and ram; the previously mentioned functionalities do not lie within the capabilities of a constrained node. There can also be other limitations such as power and energy as explained by [2].

The request/response functionality of the OCSP protocol makes it more suitable for transporting and relaying the revocation information, the OCSP response is a time-stamped signed response[7]. Accordingly, the response can be securely relayed to another endpoint in the case that the latter is equipped with means to process and verify the signature of the OCSP response.

3.3.2 The role of the OCSP responder

In the approach proposed by this research, the OCSP is chosen as the means of acquiring revocation information for a digital certificate in question. Figure 3.3 shows the structure of an OCSP request, an OCSP client sends an OCSP request to an OCSP responder to query the revocation status of a certificate in question. To acquire revocation information via the OCSP, an OCSP responder is introduced to the communication scope.



```
OCSP Request Data:
  Version: 1 (0x0)
  Requestor List:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 90C248EB881AAD4C41E5F8A862CCCD1FC246CA7B
      Issuer Key Hash: A176FA3149B9E1C9F640086E4A650E30DC314562
      Serial Number: 1001
  Request Extensions:
    OCSP Nonce: 8af6f430ebe18d3484017a9a11bf511c8dff8f834730b96c1b7c8dbca2fc3b6
```

Figure 3.3: OCSP request including nonce - generated by the OpenSSL[23] command line interface

The OCSP request can include a nonce, this ties an OCSP response to a specific request. An entity verifying an OCSP response can cross reference an included nonce to an expected value, accordingly preventing replay attacks. It is important to note that an OCSP entity that receives a request including a nonce will return a non-cached response, which can increase round trip time. It is up to the governing operational policy to decide the period on which a non-cached response is preferred over a cached one. Deciding on the period induces a trade-off as a short period is more secure but requires more resources while a longer period can result in not updating frequently enough and missing a revoked certificate.[7]

3.3.3 Analysis of an OCSP response

This subsection analyses the current specification of an OCSP response, looking at the included headers and presenting insights regarding the importance of some headers and the redundancy of others. The discussion in this section motivates the proposed profile in chapter 4. The discussion also accounts for the ideas mentioned in [19]. The text then outlines the size of the OCSP response in its current ASN.1 profile by using the OpenSSL command line interface to perform an OCSP request and receive an OCSP response.

Figure 3.4 shows the structure of an OCSP response, the reader should refer to this figure when studying the arguments given by the underlying text. Headers surrounded by dashed lines indicate optional headers in the structure, while those surrounded by red lines indicate extensible ones.

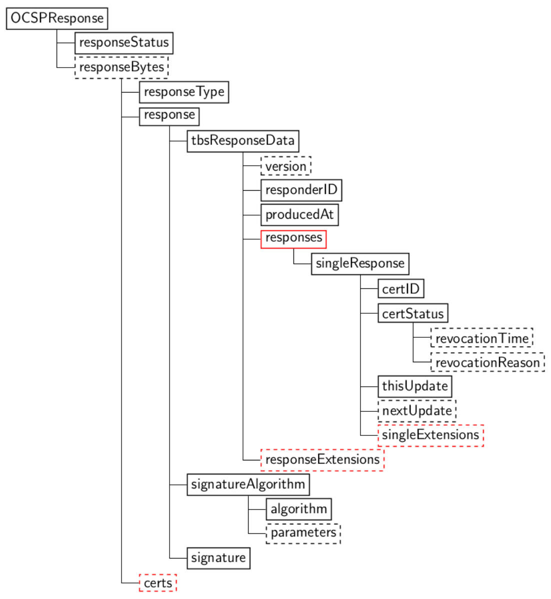


Figure 3.4: Illustration of the OCSP response structure[19]

Figure 3.4 illustrates the ASN.1 structure of an OCSP response, [19] has presented some arguments regarding the specification of the protocol, describing some of the included headers to be redundant, referring to the utilised encoding scheme to be inefficient. The research in this thesis agrees with the arguments and uses them as a solid foundation to deliver its proposal for a more efficient encoding scheme, it is to be noted that [19] considered a different usage of OCSP in their study, this difference will be highlighted in this analysis from the point of view of this research.

Discussing some points made by [19], the first one addresses the extensibility of OCSP, specifically referring to the extensible `responses` header, [19] mentions that despite the extensibility of OCSP, it is used almost exclusively to validate a single certificate. The mentioned thesis has both the OCSP request and response in the scope of its study, meanwhile this thesis only studies the structure of an OCSP response, due to the scope of the communication scenario under study.

Another argument made by [19] was concerning the presence of four different timestamp fields; `thisUpdate`, `nextUpdate`, `revocationTime` and `producedAt`. The update fields are made available for OCSP responders also known as VAs, if the trust policy of a client involves checking certificates every τ hours (where τ denotes a time period, meanwhile the VA only updates its information every 2τ hours, then adding the update fields doesn't provide any added functionality or security mechanism. For the scope of this thesis and the communication scenario under study, the client's checking period is smaller than the period where VA updates its information, therefore the update fields would not be included due to redundancy. For the communication scenario in question for this research, the client doesn't need to know the revocation time as no policy is in place to handle certificates which were revoked before or after a certain time. Accordingly, the `producedAt` header is sufficient for the client to be able to verify the freshness of the OCSP response over a governing policy regarding tolerance assigned to how fresh the OCSP response can be, an example would be a client that accepts an OCSP response that has a `producedAt` of one month ago, while another one only accepts a maximum of two days.

The last argument made by [19] which will be discussed in this thesis is one regarding the inefficiency of ASN.1 itself, as each header states the number of bytes contained in the next hierarchy level; the encoded message is riddled with byte counts which take a significant portion of the message size on their own. The latter also means that encoding has to be performed in reverse which is less efficient than a forward encoding scheme like CBOR, this is stated by [19] and is also agreed on by this research.

This research contributes more points to the analysis of the OCSP response structure; the header `revocationReason` is more like a nice-to-have than a need to have, firmware running on constrained clients is all about efficiency and minimizing code size, if a constrained client learns that a server's certificate is revoked it will immediately discontinue the connection and will not worry about the revocation reason.

Another optimization would be to remove the need of including the extensible `certs`, the header is an optional which is included to help a client verify the signature of an OCSP response, depending on the signer of the OCSP response, this extensible can contain either one certificate (case of signing by the root CA) or a certificate chain. By configuring the signer of an OCSP response out of band, client devices can be deployed with means to verify the signature, an example would be to load the the signer's public key in the device's trust store.

Performing an OCSP request with the OpenSSL command line interface for a single certificate yielded an OCSP response with a size of 1162 bytes, this number will be compared in subsection 6.2.1 with the message size results of the lightweight OCSP profile proposed by the research.

3.4 Choice of Encoding Algorithm

The background study has yielded CBOR[11] as a suitable encoding algorithm to be leveraged when entering the design phase of the research. CBOR will be used to propose a more lightweight version of the OCSP response, in combination with the arguments made in section 3.3.

3.5 Analysis of EDHOC

In the search for a method to transport certificate revocation information within resource constrained environments, the background study has yielded EDHOC as a solid option to be analysed, with the goal to acquire insights that will be used in the design process. This section discusses EDHOC[4] as a key exchange protocol employed in the research, the section elaborates on the role of EDHOC in the research and presents critical EAD items in preparation for their usage in the design discussed in chapter 5 - Certificate revocation in EDHOC.

3.5.1 The role of EDHOC in this research

Considering the communication scenario adopted by the thesis, now including EDHOC as the communication protocol used to establish the session. The two nodes are using EDHOC to agree on session keys, in this case the constrained client is referred to as the EDHOC initiator, while the the server is the EDHOC responder. With the goal of the thesis to introduce certificate revocation into IoT, EDHOC is used for means of transporting the revocation information. EDHOC is also used to demonstrate a successful session establishment in the case of a good certificate status, and also how the revocation functionality is successfully delivered when the responder's certificate is revoked and the client discontinues the protocol as a reason.

EDHOC includes EAD items which will be discussed more in the upcoming subsection, the EAD items are used inside EDHOC to transport the OCSP response (referred to as staple) relayed by the EDHOC responder. The thesis results will demonstrate the efficiency of this approach in terms of imposed overhead by the proposed functionality and how intrusive the proposal is to the current state of the protocol.

3.5.2 Critical EAD items

EDHOC defines critical and non-critical EAD items, where the sign of the EAD label determines if an item is critical or not, a negative sign indicates that the EAD item is critical. Critical EAD items affect the state of the protocol, if a critical EAD item is received and the endpoint cannot recognize or process the item, it must send an error message and discontinue the protocol [4].

Critical EAD items can be used to affect the state of the protocol, if an endpoint receives a critical EAD item which it doesn't understand or sends a request for a critical EAD item and doesn't receive it in the next message, the endpoint discontinues the protocol and sends an error message[4], Figure 3.5 shows the structure of the error message, it consists of an integer label for an error code and `ERR_INFO` can contain a String type to give more information about the error. Critical EAD items are leveraged in the design proposed by the thesis, chapter 5 will discuss this in further detail.

$$\begin{aligned} error = (& \\ & ERR\_CODE : int, \\ & ERR\_INFO : any, \\ ) & \end{aligned}$$

Figure 3.5: Structure of EDHOC Error Message[4]

3.6 Analysis of OCSP Stapling

The background study showed that OCSP stapling is worth looking into has it handles the downfalls of using OCSP without stapling, the upcoming subsections will discuss the benefit of using stapling in the communication scenario adopted by the thesis and present the role of OCSP stapling in this research.

3.6.1 Stapling or no stapling

The communication scenario in the scope of this research demonstrates a constrained client trying to establish a session with a non-constrained server, OCSP is utilised for the client to acquire revocation information regarding the server's certificate. For the client to perform the OCSP query itself, it would require that it performs an OCSP request to a known OCSP responder, then keep the connection alive. Resource constrained device can be limited in flash, Random Access Memory (RAM), energy or power[2]. Accordingly, if there is an option to remove the overhead from the constrained endpoint by delegating the task to another entity in the network; that would be the recommended approach, this way making room for the constrained endpoint to support other tasks and functionalities, or increase performance with larger allocated buffer sizes.

The aforementioned delegation can be achieved by OCSP stapling, where the constrained endpoint can ask the server to send its certificate and the OCSP response that includes the revocation status of the server's certificate, the OCSP response is referred to as the staple in this context, hence the term; OCSP stapling. This removes overhead on the constrained node to acquire revocation information.[8] The latter makes OCSP stapling the choice over no stapling in the scope of this research considering constrained endpoints.

3.6.2 The role of OCSP stapling

The research employs the stapling functionality of the OCSP stapling for the previously mentioned reasons. The constrained client sends a stapling request to the non-constrained server, the server accordingly performs the OCSP request and relays the OCSP response (Staple) to the constrained node. What the staple includes and how it is transported is covered in chapters 4 and 5.

3.7 Guidelines for specifying security considerations

The research proposes some changes to the current specification of some protocols in order to achieve an added functionality, in the case of this research; it is the functionality of certificate revocation. The specifications for internet protocols start off as an internet draft and then develops into an Request for comments (RFC). When developing a communication protocol; a very important aspect to consider is the security of the protocol. RFC3552 was created with the purpose to both encourage document authors to consider security in their designs and to inform the reader of relevant security issues[22].The research refers to the guidelines in [22] to specify security considerations for the proposed designs. The two design chapters; chapter 4 and chapter 5 will discuss security considerations for each respective design.

3.8 Experiment

This section will present the experiment setup created to acquire the thesis' results. The section will then discuss the planned measurements for evaluation. The project file structure and instructions to run the experiment will be outlined in sections 6.1 and 6.2.

3.8.1 Test platform and technology stack

This section will outline the hardware used throughout the research and motivation around the choices. The section then presents the tools and software libraries used in the development and implementation of the designs proposed by the research.

3.8.1.1 Hardware

To design an experiment that honours the constrained environment communication scenario, the research employs IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) as a relevant protocol used in the scope of the IoT[24]. When choosing a test platform, the approach was to select a microcontroller that is equipped with a module that supports 6LoWPAN and is accompanied with a well maintained software development kit.

Accordingly the nRF52840DK[25] was the chosen test platform, the board will act as the constrained node in the adopted communication scenario. The latter is equipped with Bluetooth Low Energy (BLE), giving the possibility to communicate via Internet Protocol Version 6 (IPV6) over BLE as a 6LoWPAN variant. The board is equipped with 1mb of flash memory and 256kb of ram.

The platform for the non-constrained server is a Linux PC, this PC is the same one used for the development environment, the latter is running Ubuntu 22.04. The PC is not equipped with a 6LoWPAN module, accordingly the BLE communication is required to be routed from the nRF board to the PC. A Raspberry Pi 3B+ running a router advertisement daemon is employed as a BLE router, the Raspberry Pi is running on Raspberry Pi OS with kernel version 5.15. The PC also runs the OCSP responder entity started from the OpenSSL command line interface.

The choice for measurement tool is the Joulescope JS110[26], this is a precision dc energy analyzer. The specifications include 250 kHz bandwidth with 2 million samples per second, 14-bit, simultaneous current and voltage. The equipped bandwidth ensures that a clear delta in power consumption can be observed when running the experiments with the approach proposed by the research vs without. As the main focus of the research with respect to evaluation is to report the overhead of its proposal.

The tool is accompanied with a software to visualize the measurements, as well as a GitHub repository[27] containing python scripts that can be used in post processing. The sensor ports of the Joulescope are electrically isolated from the USB ensuring there's no noise induced by the USB in the measurements. The Joulescope comes fully calibrated from the factory, and the calibration was verified as a pre-cursor to running the experiment.

3.8.1.2 Tools and software libraries

The programming languages used in the research are C, C++, and bash. These choices are enforced by the software libraries used when implementing the approach proposed by the thesis, bash is used for scripting on the Linux PC.

Library/Tool	version
Zephyr	3.1.0
Openssl	3.0.5
cmake	3.22.1
make	4.3
gcc	11.3.0

Table 3.1: Versions of tools and software libraries used in the research

Table 3.1 shows the versions of tools and software versions used throughout the development in this research. The EDHOC implementation used as a foundation in the research[28] runs EDHOC on Zephyr as an Real-time Operating System (RTOS) and uses Make and CMake buildsystems for building and GCC for compiling binaries. OpenSSL was chosen by the research to create CAs and client certificates, also to run OCSP responders, the choice was motivated by OpenSSL's open source code, and command line interface assisting in the development.

3.8.2 Experiment setup

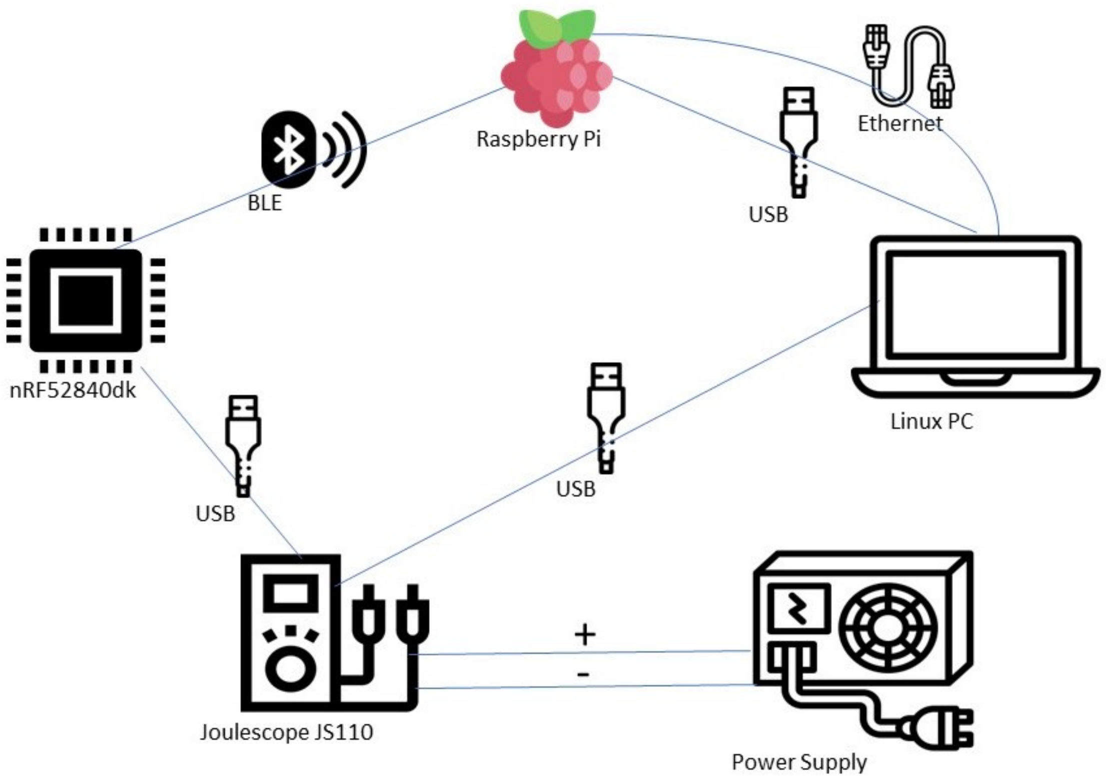


Figure 3.6: Illustration of the experiment setup

Figure 3.6 shows the hardware connections employed to run the experiment, the nRF is powered by the external 5v supply on the Joulescope, the Joulescope is connected to the PC. The Raspberry Pi is configured with a static IPV6 address and is connected via Ethernet to the PC, the nRF is configured with a static IPV6 address and is connected to the Raspberry Pi over BLE. The Raspberry Pi is powered via USB by the PC. The Joulescope is powered by the Power supply and has built-in circuit to route power to a connected device via USB. The Joulescope is also connected via USB to the PC for visualizing the measurements.

The measurement process starts with the NRF turned off, when the latter is turned on it performs a run of the EDHOC protocol and then goes to power saving, it is then manually turned off to prepare for the next measurement.

3.8.3 Planned measurements

The research will measuring the average power consumption in case of running EDHOC with revocation and without revocation, measuring deviation over 20 iterations of each, the average difference is the average overhead in power consumption. Graphs will show 20 iterations of each case.

The research uses the Joulescope to measure the average power consumption over 20 iterations in the case of running EDHOC with certificate revocation, and in the case of plain EDHOC(without revocation - without the added functionality proposed by the research). The number of iterations was chosen as 20 as a baseline, after looking at the measurements in their same respective cases, it could be seen that the variance was very small, making 20 iterations of each case to be sufficient.

The research will use the acquired measurements to report on the overhead incurred by the research's proposal to solving the problem statement. With the goal being to achieve the required functionality of certificate revocation without introducing a huge overhead with respect to message sizes, protocol time and average power consumption.

4 Design - Lightweight profile for OCSP

The chapter will introduce the Lightweight profile for OCSP proposed by the research, starting from the challenges to delivering the proposed profile. section 4.1 will present the ASN.1 to CBOR converter proposed by the research, and how the converter is integrated in a non intrusive manner to the OCSP in order to mantain backwards compatibility and the current specification of the OCSP. section 4.2 will present the details of the profile proposed by the research and discuss the choice of each header in the profile.subsection 6.2.1 will demonstrate an example of the lightweight profile and compare message sizes to the current specification of OCSP and related work.section 4.3 will specify security considerations with respect to using the new Lightweight profile in production grade infrastructures.

4.1 ASN.1 to CBOR converter

The section will outline the concept behind the ASN.1 to CBOR converter and discuss challenges surrounding integration into the te current specification of OCSP, the text will then illustrate the conversion algorithm and outline the conversion rules proposed by the research.

4.1.1 Extending the current specification of OCSP

The thesis proposes a lightweight profile for an OCSP response as part of the contributions of the research to achieve certificate revocation in IoT. The proposed profile is discussed in the upcoming section. To maintain the applicability and practicality of the proposal, the research aims to present a vector for integrating the new profile to be functional in the current implementations of OCSP. The research also considers backwards compatibility in its integration, maintaining the current functionality of the protocol while also extending it to include the proposed profile in the thesis.

4.1.2 Challenges surrounding integrating the converter

Studying the conversion problem, the research incorporates OCSP stapling into the key exchange protocol between a constrained client and a non-constrained responder.section 3.6 discusses OCSP stapling in more detail and describes its role in the research. The OCSP response is a signed response, in the case of OCSP stapling where a server performs the OCSP request and relays the OCSP response to the client, since the client doesn't trust the responder by nature of the communication scenario; the OCSP response needs to maintain the signature of the OCSP responder (depending on the implementation of OCSP the signer can also be different, an example would be to have the signer as the issuer of the OCSP responder's certificate), this accordingly means that the conversion of the OCSP response needs to happen somewhere in the implementation of the OCSP entity itself.

To maintain backwards compatibility, the request for the new OCSP response profile needs to be signaled to the OCSP entity, the thesis proposes two methods as options to achieve this; using Hypertext Transfer Protocol (HTTP) the content-type can be set to signal for a response structure that uses the new profile, where the OCSP entity can check the content type and accordingly set the logic to perform the conversion before signing the response. Another method to achieve this is through using `preferredSignatureAlgorithm` in the OCSP request extension[7], a new signature algorithm label can be registered to signal to the OCSP responder that the new profile of OCSP response is requested to be returned and not the current one.

The conversion is performed on the `tbsResponse` which is the OCSP response data structure that is going to be signed, the idea is to introduce the converter in a way that is least intrusive in terms of software libraries that implement the protocol. The `tbsResponse` contains ASN.1 Data structures, at the time of writing this thesis there aren't clear rules on converting an ASN.1 data structure to a CBOR one, this thesis proposes some conversion rules in the scope of the OCSP response headers that are chosen to participate in the new proposed profile.

4.1.3 The conversion algorithm

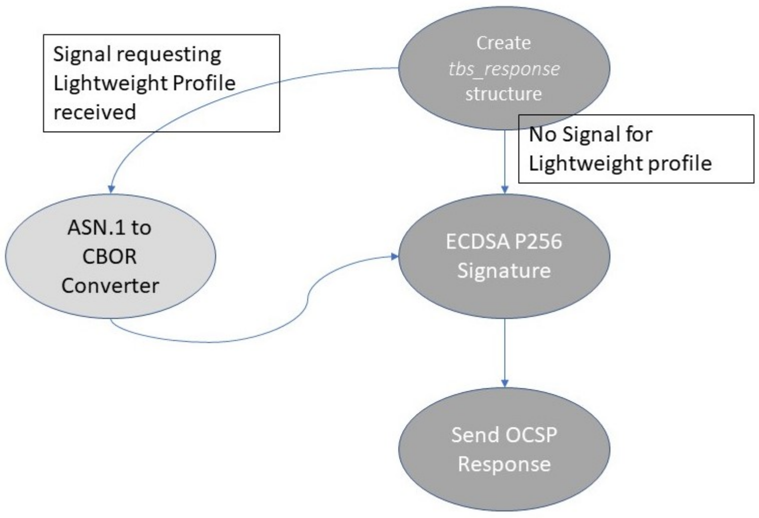


Figure 4.1: Illustration of the conversion algorithm in the case of signaling the OCSP responder for the lightweight profile

Figure 4.1 shows the conversion algorithm for converting the legacy OCSP response structure into the new Lightweight profile proposed by the research. The signaling is performed either via the HTTP content-type header specifying content-type with a new registered label referring to the lightweight profile, or using the OCSP request extension `preferredSignatureAlgorithm` by setting a new signature algorithm label that refers to the lightweight profile.

The research proposes some conversion rules for the OCSP response headers chosen to participate in the new profile, where the timestamp fields in `ProducedAt` are extracted from the ASN.1 structure encoded as CBOR time, `nonce` is extracted from the OCSP repsonse extensions and is encoded as a byte string, the fields inside certID are seperately extracted from the ASN.1 structure and encoded together as a CBOR array. The other chosen headers are considered by the research to be very trivial to convert into a CBOR encoded version of themselves.

4.2 OCSP Profile for resource constrained environments

$$
\begin{aligned}
tinyOCSP\_response = (& \\
& response\_type : unsigned\ int \\
& responderID : bytestring \\
& ProducedAt : CBOR\ Time \\
& nonce : bytestring \\
& certID : certID \\
& certStatus : unsigned\ int \\
)&
\end{aligned}
$$

Figure 4.2: Concise Data Definition Language (CDDL) of the Lightweight profile for the OCSP response proposed by the research

Figure 4.2 illustrates the Lightweight profile for the OCSP response proposed by the research, the illustration is motivated from the analysis phase of the research when the OCSP response was analysed and the acquired insights were utilized to propose the new profile.

The `responderID` is transported as a byte string and is necessary for the entity verifying the OCSP response to know which OCSP responder signed the response. `ProducedAt` is used for verifying that the time of producing the response is fresh with respect to a governing policy surrounding the operation of the verifying entity, in the scope of this research; the verifying entity is the constrained client. `nonce` is checked by the verifying entity in the case a fresh OCSP response was requested, if `nonce` doesn't match an expected value; the response can be treated as a form of replay attack. `certID` is checked by the verifying entity and is matched to the expected certID, in the scope of this research and adopted communication scenario; the expected certID would be that of the non-constrained server.`certStatus` contains the revocation status of the certificate, an entity processing this response checks the status of the certificate and discontinues session establishment in the case that the certificate is revoked. The research finds the proposed headers as the necessary requirements to achieve the certificate revocation functionality in the scope of resource constrained environments.

4.3 Security considerations

The section uses RFC3552 [22] to specify security considerations for the proposed lightweight profile. The upcoming sections will discuss the relevant potential attack vectors with respect to the proposed profile.

4.3.1 Denial of Service

Denial of service is always a potential attack vector, a malicious entity gaining knowledge about favourite OCSP responders in a network can launch a denial of service on the network's OCSP responders, denying the acquiring of certificate revocation information. A proxy between the entity performing the OCSP request and the OCSP responder can either deny the request from ever reaching the responder or do the same to the response with respect to the entity that performed the OCSP request. This attack vector is there in the current profile of the OCSP and is in no way accentuated by the lightweight profile proposed by this research.

4.3.2 Man in the Middle Attack

The attack here refers to a malicious entity that is proxying the OCSP request/response traffic, the malicious entity can act as an OCSP responder and gain access to the metadata of every device performing an OCSP request to it, but the attack stops there as the malicious entity would need to compromise the private key of the spoofed OCSP responder in order to be able to send back forged OCSP responses legitimately signed with the compromised private key. The proposed lightweight profile maintains the signature value in the response, an entity receiving the lightweight profiled response must first verify the signature and ensure that it is from a trusted OCSP responder.

4.3.3 Replay Attack

A replay attack is an attack where a request/response is saved for later use, when the saved message is reused in a communication; that is considered a form of a replay attack. In the scope of the OCSP, a replay attack can be a malicious entity saving an OCSP response containing a good certificate status and using it after the certificate in question has been revoked. The lightweight profile makes use of the `nonce` header, where a nonce ties a response to a specific request. Therefore the entity performing the OCSP request should include a well generated nonce in the request and should expect it in the OCSP response, if it's not present or if the header is present but the value is different then that should be treated as a form of replay attack.

5 Design - Certificate revocation using EDHOC

This chapter will present how the research Incorporates Certificate Revocation leveraging EDHOC as a lightweight key-exchange protocol. section 5.1 introduces EDHOC into the communication scenario adopted by the research. section 5.2 presents the design for incorporating certificate revocation into EDHOC. section 5.4 presents how the proposed design can be utilized to achieve an added functionality of synchronising the realtime clock of the constrained device. section 5.5 outlines the added requirements from the constrained device when utilizing EDHOC as proposed by the research.section 5.6 will specify security considerations with respect to how the research utilizes EDHOC in the overall design.

5.1 EDHOC communication scenario

In the scope of the EDHOC communication adopted by the research, the constrained node is an EDHOC initiator and the non-constrained node is an EDHOC responder. The constrained initiator wants to acquire revocation information regarding the responder's certificate. The accompanying challenges of the latter present themselves as the following: (1)acquiring revocation information in a secure manner (2)transporting revocation information (3)verifying revocation information.

The underlying text will present a vector to overcome the challenges and achieve the required functionality, the proposal leverages OCSP as a means to acquire certificate-revocation information and employs EDHOC for transport.

5.2 Incorporating certificate revocation

This section will present the staple-request and staple-response EAD items used for transporting revocation information in EDHOC, the text will then demonstrate how these items are used to transport the revocation information.

5.2.1 Staple-request in EDHOC

Figure 5.1 shows the expansion of an EAD item in CDDL[29] format with `staple-request` encoded as a byte string and placed in `ead_value`. As `staple-request` is the value of the EAD item, this EAD item is referred to by the text as the `staple-request` EAD item. The `staple-request` is a critical EAD item proposed by this research, it is added into EDHOC Message 1 by the EDHOC initiator, it contains a trusted OCSP responder list which is used by the EDHOC responder to determine which OCSP responder to query for the revocation information, it also contains a parameter called `fresh` to indicate whether a fresh OCSP response is required or if a cached one is sufficient. An EDHOC responder that receives `staple-request` with the `fresh` parameter set to `True` uses `G_X` (which in EDHOC is the initiator's ephemeral key) as a nonce in the OCSP request, accordingly the OCSP entity would include that nonce in the response. The OCSP response with the nonce would then be relayed by the responder in the staple-response item, the initiator can verify the nonce and thus protect itself against replay attacks.The integer label for the EAD item is negative signed to mark the latter as a critical EAD item.

$$
\begin{aligned}
ead = (&\\
&ead\_label : int,\\
&ead\_value : bstr(staple - request)\\
)&
\end{aligned}
$$

$$
\begin{aligned}
staple - request = (&\\
&responderID\_list : bstr,\\
&?fresh : CBOR\ True(0xF6)\\
)&
\end{aligned}
$$

Figure 5.1: CDDL of EAD value in the `staple-request` EAD item

The size of the trusted responder list determines the size of the `staple-request` item, as the `fresh` parameter is just a one-byte CBOR `True` or `False`.

In the case of a resource constrained setting; the trusted responder list can be transported as null to indicate a responder list that has been agreed upon out of band. Otherwise the list entries need to be formatted in a manner that the entity receiving the `staple-request` item can reach out to the responders on the list. By transporting the list as null, it is encoded as a CBOR Null and would take up one byte of space, making the total size of the CBOR-encoded `staple-request` EAD item to be 4 bytes (one byte for the integer label in `ead_label` and 3 bytes for `ead_value`.

5.2.2 Staple-response in EDHOC

The `staple-response` is a critical EAD item proposed by this research, it is added into EDHOC Message 2 by the EDHOC responder after receiving a `staple-request` EAD item in EDHOC Message 1, the responder constructs the `staple-response` item after acquiring an OCSP response from the OCSP responder. chapter 4 presented the lightweight profile of the OCSP response proposed by this research. The latter is then encoded as a bytestring and added into the `ead_value` field of the `staple-response` EAD item.

$$\begin{array}{l} ead = (\\ \quad ead\_label : int, \\ \quad ead\_value : bstr(staple - response) \\ \quad) \end{array}$$

$$\begin{array}{l} staple - response = (\\ \quad ResponseData : tinyOCSP\_response \\ \quad SignatureVal : bstr \\ \quad SignatureAlg : unsignedint \\ \quad) \end{array}$$

$$\begin{array}{l} tinyOCSP\_response = (\\ \quad response\_type : unsignedint \\ \quad responderID : byteString \\ \quad ProducedAt : cborTime \\ \quad nonce : bytestring \\ \quad certID : certID \\ \quad certStatus : unsignedint \\ \quad) \end{array}$$

Figure 5.2: CDDL of EAD value in the `staple-response` EAD item

Figure 5.2 shows the staple (OCSP response) after it has been encoded into the value field of an EAD item, the latter then becomes the `staple-response` EAD item, the integer label for the EAD item is negative signed to mark the latter as a critical EAD item. The CDDL[29] format is used to describe the expansion of the EAD item.

5.3 The Certificate Revocation Process

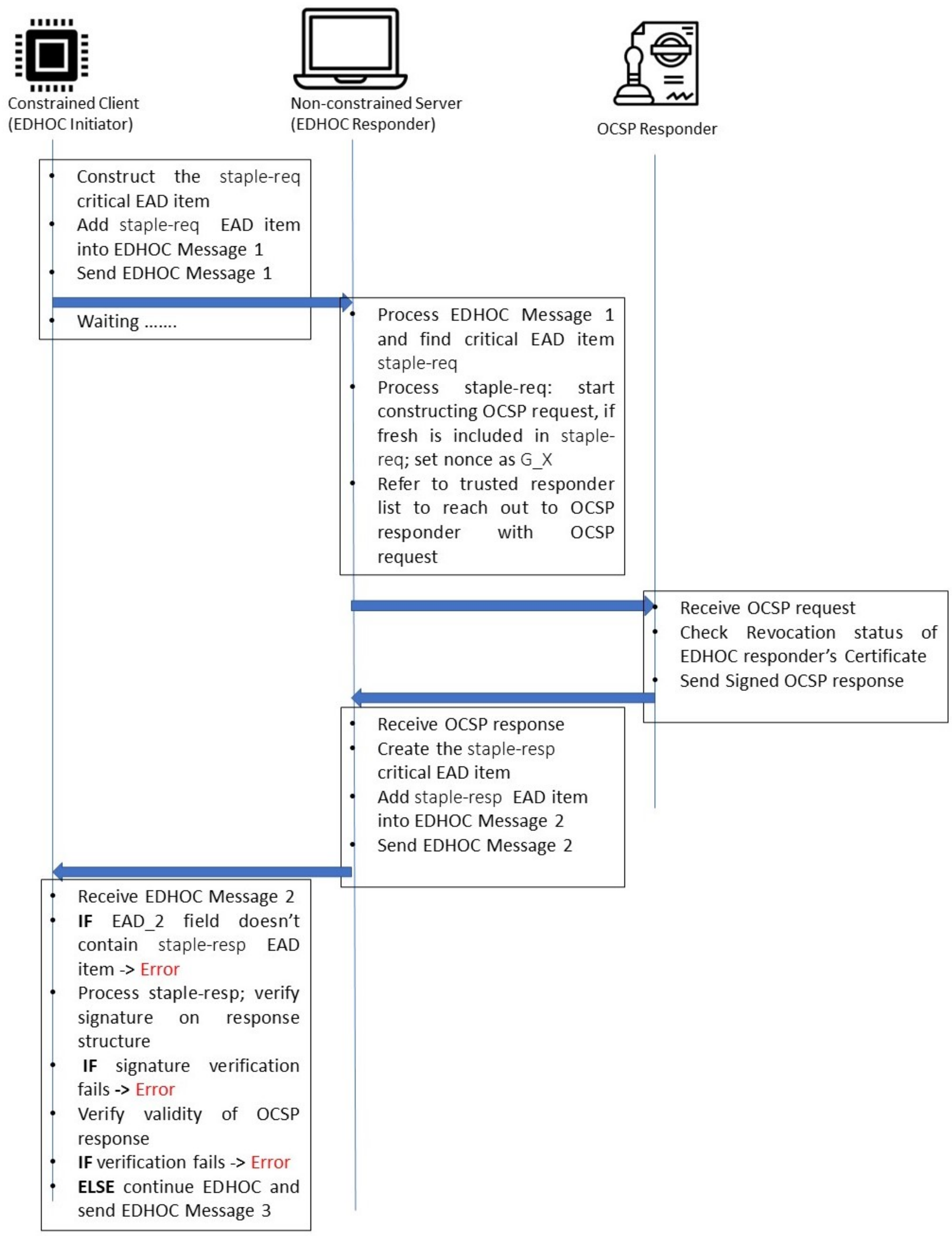

Constrained Client (EDHOC Initiator)
Non-constrained Server (EDHOC Responder)
OCSP Responder
• Construct the staple-req critical EAD item
• Add staple-req EAD item into EDHOC Message 1
• Send EDHOC Message 1
• Waiting ....
• Process EDHOC Message 1 and find critical EAD item staple-req
• Process staple-req: start constructing OCSP request, if fresh is included in staple-req; set nonce as G_X
• Refer to trusted responder list to reach out to OCSP responder with OCSP request
• Receive OCSP request
• Check Revocation status of EDHOC responder's Certificate
• Send Signed OCSP response
• Receive OCSP response
• Create the staple-resp critical EAD item
• Add staple-resp EAD item into EDHOC Message 2
• Send EDHOC Message 2
• Receive EDHOC Message 2
• IF EAD_2 field doesn't contain staple-resp EAD item -> Error
• Process staple-resp; verify signature on response structure
• IF signature verification fails -> Error
• Verify validity of OCSP response
• IF verification fails -> Error
• ELSE continue EDHOC and send EDHOC Message 3

Figure 5.3: Illustration of the certificate revocation process incorporated in EDHOC

Figure 5.3 shows the constrained EDHOC initiator and the non-constrained EDHOC responder, the process starts when the initiator constructs the `staple-req` EAD item and sends it in EDHOC Message 1. The proposal leverages OCSP stapling where the initiator asks the EDHOC responder for its certificate as well as a staple, the latter refers to an OCSP response that contains the revocation status of the responder's certificate. A staple request is transported in EDHOC Message 1, the responder receives the message and processes the staple request.

On processing the staple request, the EDHOC responder becomes an OCSP client and performs a request to an OCSP responder (VA) that is trusted by the EDHOC initiator, information regarding trusted OCSP responders is relayed in the staple request sent in EDHOC Message 1 by the initiator. The EDHOC responder accordingly receives a response from the VA that contains the revocation information required by the EDHOC initiator. The responder creates a staple response and transports it in EDHOC Message 2.

The constrained initiator receives EDHOC Message 2 which should contain a staple response, the initiator should return an error message and EDHOC fails in the case where EDHOC Message 2 doesn't contain a staple response. The initiator processes the staple response and acquires the revocation status of the responder's certificate, the initiator sends an error message and EDHOC fails in the case where the responder's certificate is revoked. The initiator proceeds to send EDHOC Message 3 after successfully processing the staple response and acquiring a good certificate status.

5.4 Realtime clock synchronisation for the constrained initiator

Using the `staple-request` EAD item, the initiator can synchronise its Realtime Clock (RTC). An unsynchronised RTC can expose the initiator to numerous replay attacks, in the case of certificate revocation; an attacker can use an old OCSP response that shows a good certificate status before the certificate in question was revoked, as a result an attacker would manage to authenticate itself as a responder to the constrained initiator.

The staple-request EAD item contains the fresh parameter which can be used by the initiator to ask for a fresh OCSP response, whereas otherwise it would get a cached one. As the OCSP response is a timestamped response, the initiator can use the timestamp to synchronize its RTC. Employing this method of time synchronization; an initiator running EDHOC with certificate revocation using the staple-request item can synchronize its RTC and add a layer of protection against replay attacks.

5.5 Processing the OCSP Staple

The EDHOC initiator constructs the `staple-request` and encodes it as a CBOR byte string, the trusted OCSP responder list is either loaded from the device's trust store or is then set to NULL to indicate a responder list that has been agreed upon out of band. The EDHOC initiator then constructs an EAD item setting `ead_label` to a registered negative signed integer label, and adding the `staple-request` bytestring as `ead_value`. The constructed EAD item is then added as a CBOR item to `EAD_1` field in EDHOC Message 1.

The initiator waits for EDHOC Message 2, `CIPHERTEXT_2` is decrypted and can process `EAD_2`. On processing `EAD_2`, if the `staple-response` EAD item is not found then the initiator should send an error message and discontinue the protocol, this is also considered as the specified behavior for dealing with critical EAD items in EDHOC. In the case the `staple-response` EAD item is found, the initiator will decode `ead_value` and perform the signature verification using signatureval and the responder's public key and the OCSP response structure. After verifying the signature, the responder must continue verifying the values present in other headers of the response structure, this is done as a security measure. `ProducedAt` must be within the range set by the governing policy with respect to accepting OCSP responses within certain time stamps. If `staple-request` contained was constructed with `Fresh` set to true then the nonce in the OCSP response structure must match g_x which is the responder's ephemeral public key. `certID` is checked to ensure that the received respsone refers to the certificate in question, in this case it would be the certificate of the responder. Lastly `certStatus` is checked, if the status has the value revoked then the initiator would send an error message and discontinue the protocol, otherwise it will move forward with the EDHOC exchange.

`ProducedAt` must be within the governing policy for the operation of the EDHOC initiator regarding acceptance period for revocation information. If the initiator included `fresh` in the staple-request; then `nonce` must equal `g_x` or the freshness criteria has not been satisfied and this can be treated as a form of replay attack. `certID` must match that of the EDHOC responder otherwise the revocation information should be considered invalid. `certStatus` must give a good certificate status, the initiator must discontinue the protocol if `certStatus` has the integer label signifying a revoked certificate.

The protocol can continue normally on a successful verification, the initiator constructs EDHOC Message 3 and sends it to the EDHOC responder. An error message is sent in the case of a verification failure and the protocol is discontinued.

5.6 Security considerations

The section uses RFC3552 [22] to specify security considerations for the proposed method to perform Certificate Revocation. The upcoming sections will discuss the relevant potential attack vectors with respect to the proposal.

5.6.1 Denial of Service

Denial of service can happen anywhere in the proposed certificate revocation vector, the session establishment can be denied by denying the EDHOC responder from receiving the OCSP response or performing the OCSP request. A denial of service can also come in the form of denying the communication between the EDHOC initiator and responder, as a result denying the session establishment. Due to the usage of critical EAD items, another denial of service would be a Man in the Middle Attack vector which is able to send an EDHOC message 2 that doesn't contain the critical EAD item and by design the initiator would then discontinue the protocol.

5.6.2 Man in the Middle Attack

An adversary proxying the communication between the EDHOC responder and initiator can view the contents of the `staple-request` EAD item in EDHOC message 1, from that the adversary can acquire knowledge regarding the usage of the `Fresh` parameter by the EDHOC initiator. The acquired knowledge can then be used as part of a vector for a Replay attack, where now the adversary knows how often the EDHOC initiator uses the `Fresh` parameter in the `staple-request`. This could be mitigated by specifying shorter periods for setting the `Fresh` parameter. The mitigation would be on the cost of increasing the total time of the EDHOC exchange, as the OCSP responder would need to create a new OCSP response with the new nonce and not a cached one.

5.6.3 Replay Attack

A replay attack vector with respect to the certificate revocation approach proposed by the research would be that where an adversary has acquired an OCSP response regarding the legitimate EDHOC responder's certificate, and then the adversary was able to compromise the private key of the EDHOC responder and can now use the saved valid OCSP response to pass the revocation check done by the EDHOC initiator. This is mitigated by the proposed method by setting the `Fresh` parameter in the `staple-request`. This asks the EDHOC responder to create an OCSP request using a nonce known by the EDHOC initiator, the EDHOC initiator then checks for the nonce in the received stapled OCSP response. If the nonce is not there, it will discontinue the protocol, mitigating the replay attack.

6 Results and Conclusion

This chapter will first outline the implementation results and explain the project file structure in section 6.1. section 6.2 will then present the steps for recreating and executing the experiments, and then discuss the acquired results. The research will then conclude and summarise on the results, and propose future work that would use this research as a solid foundation.

6.1 Implementation

The implementation was divided into 2 phases; the first phase was to implement the Lightweight profile presented in chapter 4, this was divided into two sub-phases of firstly implementing the conversion of the OCSP response to the new profile and then implementing the proposed integration vector that maintains backwards compatibility with the current specification of the OCSP protocol at the time of writing this Thesis.

The second phase; implementing the `staple-response` and `staple-request` EAD items presented in chapter 5 - Certificate Revocation using EDHOC. The implementation includes defining the EAD items and adding them into the EDHOC messages, and adding logic to process the latter. The implemented processing of the EAD items includes performing an OCSP request signaling for the lightweight profile, processing the OCSP response and appending the staple into the `staple-response` EAD item.

The goal of the previous implementation phases was to achieve a functional demonstration of the approach proposed by the research for performing Certificate Revocation in a resource constrained environment. The implemented demonstration is then utilized in the experiment outlined in section 3.8 to acquire the research's results.

```
├── openssl-tinyOCSP
└── uoscore-uedhoc
```

Figure 6.1: Highest level of the project implementation structure

The following sections will outline the results of each phase, Figure 6.1 shows the highest level of the project folder structure.

6.1.1 Lightweight Profile for OCSP

The approach outlined in chapter 4 is implemented as an extension to the OpenSSL library version 3.0.5, it uses the `content-type` method to signal the OCSP entity for the new proposed profile, and implements the presented conversion rules and algorithm. The implementation is a working proof of concept of the design proposed by the research and serves as a strong foundation to be refactored and turned into production code.

6.1.1.1 Conversion to the Lightweight Profile

```
└─ openssl-tinyOCSP
   └─ openssl-3.0.5
      └─ apps
         └─ ocsp.c
```

Figure 6.2: Path to the file changed in OpenSSL 3.0.5 to include the implementation of the OCSP conversion to the proposed Lightweight Profile

Figure 6.2 shows `ocsp.c` changed in the OpenSSL library to include the implementation of the OCSP response conversion to the Lightweight Profile proposed by the research. The `ocsp.c` governs the functionality of an OCSP responder that is run on the command line interface using the OpenSSL library. `ocsp.c` was the main file that was changed, some other files in the library were changed as a normal process of extending the library to support the added changes in `ocsp.c`.

Added code is marked with a `Lightweight-Profile` comment and additional comments inside the file explain the functionality of the added code. Appendix A.1 shows the implementation of the OCSP response conversion to the lightweight profile, the listing shows the main function used in the conversion.

The upcoming text will present the signaling implementation and provide an example of an OCSP response with the Lightweight profile proposed by the research, the response is acquired by calling the presented implemented functions.

6.1.1.2 Signaling a conversion request to the OCSP

```
└─ openssl-tinyOCSP
   └─ openssl-3.0.5
      └─ apps
         ├─ lib
         │  └─ http_server.c
         └─ ocsp.c
```

Figure 6.3: Path to the files changed in OpenSSL 3.0.5 to include the implementation of the signaling method for the proposed Lightweight Profile

Figure 6.3 shows `http_server.c` and `ocsp.c` changed in the OpenSSL library to include the implementation of signaling the OCSP to perform the Lightweight Profile conversion proposed by the research. The `http_server.c` contains added functions for setting `content-type` in the HTTP request, the file also contains some other workarounds to achieve the signaling while maintaining backwards compatibility. The file `ocsp.c` contains added flow control to act on whether a signal for the lightweight profile has been received or not. Added code is marked with a `Lightweight-Profile` comment and additional comments explain the functionality of the added code. `http_server.c` and `ocsp.c` were the main files that were changed to include this functionality, some other files in the library were changed as a normal process of extending the library to support the added changes in `ocsp.c`.

Listing 6.1 shows the code snippet added into the HTTP exchange flow of OpenSSL, it shows the usage of the HTTP `content-type` header, by setting it in the OCSP request; the implementation can check for it in the HTTP exchange. The return value 3 is then matched in `ocsp.c` to signal the conversion request to the OCSP responder entity.

Listing 6.1: Code snippet added into the HTTP exchange flow of the OpenSSL OCSP library

```

       //check for tiny response request via content type
        if (OPENSSL_strcasecmp(key, "Content-type") == 0)
        {
            if (OPENSSL_strcasecmp(value, "application/ocsp-request-tiny")
               == 0)
            {
                printf("Received Request for Tiny(Cbor) Response\n");
                ret=3;//signal for tiny request
            }

        }
```

Appendix A.2 shows an example of the Lightweight Profile of the OCSP response acquired as a result of performing an OCSP request with signaling for the conversion via the implemented functions.

6.1.2 Certificate Revocation with EDHOC

The implementation uses Stefan Hristosov's `uoscore-uedhoc` library [28] as a foundation for implementing the concepts proposed by the research in chapter 5.

The implementation first started by making the EAD field of the EDHOC messages available to the application, this was done by extending the functions that run the EDHOC responder and EDHOC initiator to include a callback to a function that can be defined by the application level to process the EAD field in the EDHOC messages, application level here is the level above the EDHOC protocol. Listing 6.1 shows the main calling function where the callback function is added.

Listing 6.2: Extending `edhoc_responder_run()` to include a callback function `process_ead_1()` that can be defined by the application

```
enum err edhoc_responder_run_extended_ead_proc(
  struct edhoc_responder_context *c,
  struct other_party_cred *cred_i_array, uint16_t num_cred_i,
  uint8_t *err_msg, uint32_t *err_msg_len, uint8_t *ead_1,
  uint32_t *ead_1_len, uint8_t *ead_3, uint32_t *ead_3_len,
  uint8_t *prk_out, uint32_t prk_out_len,
  enum err (*tx)(void *sock, uint8_t *data, uint32_t data_len),
  enum err (*rx)(void *sock, uint8_t *data, uint32_t *data_len),
  enum err (*process_ead_1)(struct edhoc_responder_context *c_1, uint8_t *
     data_ead_1, uint32_t *data_ead_1_len))

{
  return edhoc_responder_run_extended_2(c, cred_i_array, num_cred_i,
              err_msg, err_msg_len, ead_1,
              ead_1_len, ead_3, ead_3_len,
              prk_out, prk_out_len, NULL, NULL,
              NULL, NULL, tx, rx, process_ead_1);
}
```

Appendix B.1 shows the definition of the `process_ead_1()` function, the function performs the processing of the `staple-request` item found in `EAD_1` and then accordingly generates an lightweight profiled OCSP request using the implementation done by the research, then the function encodes the received OCSP response as the staple and constructs `staple-response` and adds it into `EAD_2`.

There are other added changes in the library to support the implementation of the proposed design. Figure 6.4 shows some of the directories that have been changed to include functions that support the design, the reader can refer to the commented code for their understanding. The directory named `responder_extended` holds the implementation for the extended EDHOC responder that processes `staple-request` and constructs `staple-response`. The directory `initiator_extended` holds the implementation for the extended EDHOC initiator that sends `staple-request` and processes `staple-resp` to acquire the relayed certificate revocation information.

```
└─ uoscore-uedhoc
   └─ samples
      ├─ linux_edhoc
      │  └─ responder_extended
      └─ zephyr_edhoc
         └─ initiator_extended
```

Figure 6.4: Path to the folders including the implementations of EDHOC responder and EDHOC initiator that are run in the experiment and proof of concept demonstration

The implemented proof of concept demonstration shows the applicability of the certificate revocation approach proposed by the research. The demonstration starts by running an EDHOC exchange with the implemented certificate revocation approach in the case of a valid EDHOC responder certificate, the certificate is then revoked by the CA and the EDHOC exchange fails when run the second time due to a revoked EDHOC responder certificate.

6.2 Evaluation

6.2.1 Message Sizes - Lightweight Profile for the OCSP

Table 6.1 shows the OCSP response size of the lightweight profile proposed by the thesis compared to that of the current state of the OCSP. The lightweight profile achieves a reduction of 76.4% from the message size of the current profile of the OCSP response.

OCSP profile	OCSP Response size (bytes)
OCSP profile RFC[6960] [7]	1162
Lightweight Profile(This research)	274

Table 6.1: Comparison of the OCSP response message size in its current profile [7] vs the Lightweight profile proposed by the research

Table 6.2 shows the size utilized by each header in the Lightweight profile proposed by the research, the total size shown is the size before appending the signature calculated on the response structure.

Header	Size (bytes)
`response_type`	1
`responderID`	94
`ProducedAt`	21
`nonce`	34
`certID`	48
`certStatus`	1
Total Size	199

Table 6.2: Header sizes of the Lightweight OCSP response profile proposed by the research

Table 6.3 breaks down the elements of the total size of the OCSP response structure after encoding and appending the signature value and the signature algorithm. Encoding the structure as a byte string adds two bytes, adding the encoded signature adds 72 bytes and the signature algorithm label adds one byte, bringing the total size of the complete OCSP response structure to be 274 bytes.

`signatureAlgorithm`(1 byte)	`signatureVal` (72 bytes)	bytestring encoding (2 bytes)	lightweight structure (199)

Table 6.3: Total size of the proposed lightweight profile after appending the signature and encoding

6.2.2 Experiment

This section outlines the steps to recreate and execute the experiment. The section then presents the results of the experiment and accordingly the results of the undertaken research.

6.2.2.1 Experiment Execution

The OpenSSL[23] toolkit provides means of setting up a certificate authority using its command line interface, the certificate authority can then sign a certificate signing request for an OCSP entity's certificate, the OCSP entity is also created from OpenSSL's command line interface.

Certificates generated for all entities constituting this research contain 256-bit Elliptic-Curve (EC) keys using the prime256v1 algorithm. This is enforced for compatibility with the protocols used in the research. The prime256v1 algorithm is compliant with the NIST key size recommendations on Elliptic-Curve Digital Signature Algorithm (ECDSA)[30].

An OCSP responder that is set up with EC keys is equipped to sign OCSP responses outputting an ECDSA signature, accordingly an entity that is performing the function of verifying an OCSP response needs to be able to verify ECDSA signatures. The entity that will perform the verification in this scenario is the constrained initiator.

The nRF52840dk is flashed with the firmware residing in `./uoscore-uedhoc/samples/zephyr_edhoc/initiator_extended` and a Linux PC is made to run the program in `./uoscore-uedhoc/samples/zephyr_edhoc/initiator_extended`. These directories are found in the project files included with the research.

A version of OpenSSL that supports the lightweight profile is built by following the instructions included in `./openssl-tinyOCSP` and an OCSP responder entity is launched from the command line interface of the OpenSSL version that has just been built.

The hardware is connected as illustrated in Figure 3.6-subsection 3.8.2 and the experiment can be started by turning on the nRF52840dk and observing the output of the EDHOC responder entity running on the Linux PC. The nRF board is turned off after every complete EDHOC exchange and the measurements are logged on the Joulescope. The research performed this process for 20 iterations of EDHOC with the proposed certificate revocation approach. For running plain EDHOC, the research flashed the nRF board with the firmware residing in `./uoscore-uedhoc/samples/zephyr_edhoc/initiator` which doesn't contain the added funtions implemented by the research. The same is done in terms of measurement, the research ran 20 iterations of the case of running plain EDHOC. The acquired results are presented in the upcoming section.

6.2.2.2 Experiment Results

Figure 6.5 shows the average power consumption over the 20 iterations of running each case of the experiment, a box plot was chosen as the visualisation method to visualise the mean value over 20 iterations and use that for calculating the power consumption overhead. The results show that compared to plain EDHOC the average power consumption of the Certificate Revocation approach proposed by the research is only 0.46% more than running EDHOC without the certificate revocation function.

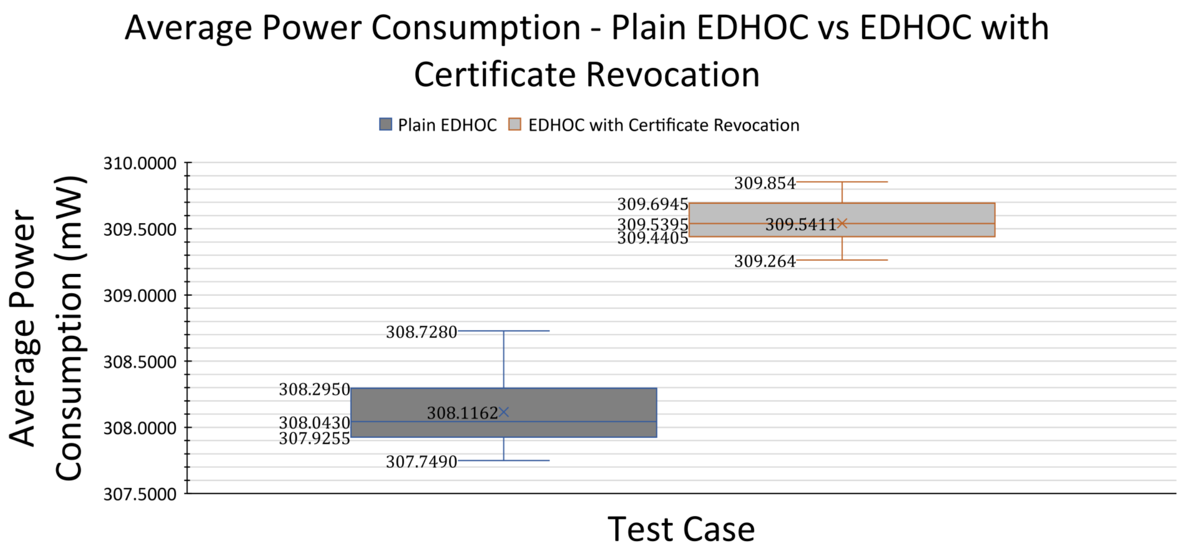


Figure 6.5: Comparison of the average power consumption when running Plain EDHOC vs EDHOC with the Certificate Revocation approach proposed by the research in chapter 5

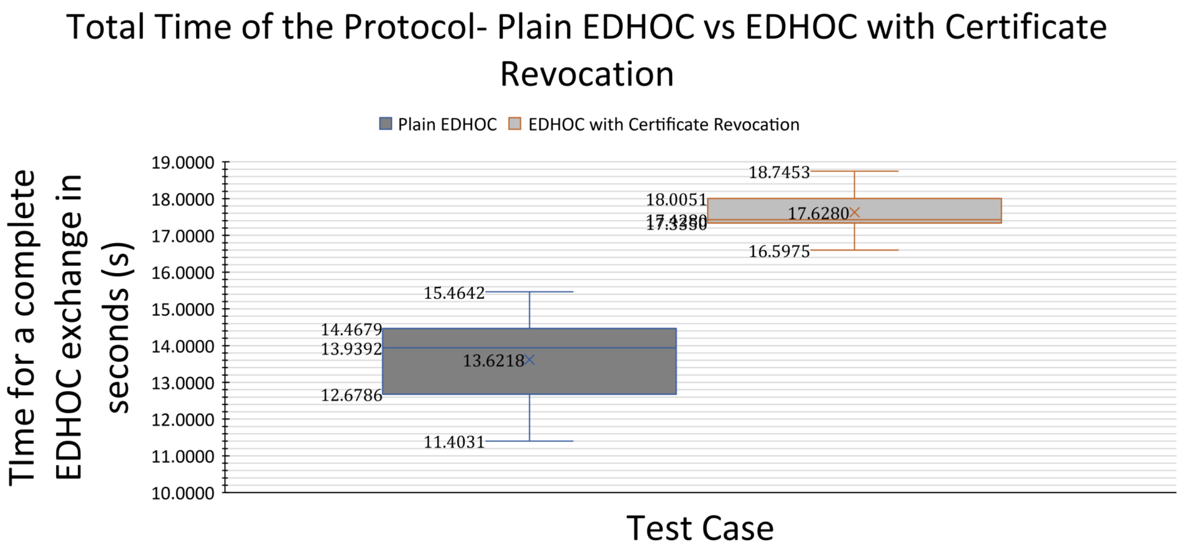


Figure 6.6: Comparison of the total time taken to run a complete exchange of Plain EDHOC vs EDHOC with the Certificate Revocation approach proposed by the research in chapter 5

Figure 6.6 shows 20 iterations of running plain EDHOC and EDHOC with the proposed Certificate Revocation approach, the plot shows the total time required for a complete EDHOC exchange for each iteration. Looking at the two box plots; it can be observed that the total time taken for a complete EDHOC exchange when running EDHOC with the Certificate Revocation approach proposed by the research is 29% more than that of when running plain EDHOC.

Plain EDHOC	EDHOC with Certificate Revocation (This research)
$\approx 4.2(Watts)$	$\approx 5.46(Watts)$

Table 6.4: Total Power Consumption when running plain EDHOC vs EDHOC with the Certificate Revocation approach proposed by the research

Utilizing the results of both box plots; the total power consumption of running EDHOC on the constrained client side with certificate revocation vs plain EDHOC can be calculated. Table 6.4 compares the total power consumption in both cases, the values were calculated by multiplying the mean value of 20 measurements of average power consumption in both cases by the mean value of 20 measurements of total time taken to complete the EDHOC exchange in each case.

6.3 Discussion

The section will firstly discuss the Experiment results presenting some insights around the reasoning behind the results, the text will then present some notes on the proposed Lightweight Profile surrounding the profiling and the acquired message sizes. Lastly the section will discuss the overhead incurred on the constrained initiator when performing Certificate Revocation using the approach presented in the research.

Looking at the experiment results, the difference in average power consumption between the two cases is very small. The research expected the difference to be larger to account for the extra signature verification process (verifying the signature of the staple in `staple-response`) performed on the constrained client, the research believes that the difference is small as the time taken to perform the signature verification is not long enough to affect the average power consumption measurement, therefore the difference between the two cases only shows up as a small value.

The difference in total time of the protocol can be explained, there are many factors that can affect the total time taken for a complete exchange in the case of using the proposed method for revocation. After the non-constrained responder receives the `staple-request` EAD item in EDHOC message 1, it first processes the request and then performs an OCSP request. Processing time and round trip time of the OCSP request is to be added to the total time of the protocol. Then sending and receiving EDHOC Message 2 which includes `staple-response` would also increase the total time, lastly the signature verification would contribute to increasing the total time, but from the acquired average power consumption results; it would seem that the signature verification only contributes a small portion to the increase in the total time of the protocol.

The proposal uses EAD items for transporting the request and response regarding revocation information. The size of the `staple-request` EAD item is only 4 bytes in the case of a `Null responderID_list`, the size can grow significantly if `responderID_list` is not `Null`, the growth would be determined by the structure used to store `responderID_list`.

Negative signed label (1 byte)	CBOR encoding of `staple-response` into `EAD_value`(3 bytes)	`staple-response` (274 bytes)	EDHOC Message 2 ($\approx$ 350 bytes)

Table 6.5: Total size of the proposed lightweight profile after appending the signature and encoding

Table 6.5 shows the overhead incurred by the use of the EAD field in the EDHOC Messages. The `staple-response` bytestring has a total size of 274 bytes, constructing the EAD item adds 2 bytes to encode the `staple-response` bytestring and 1 byte for the signed integer label, creating a total size of 277 bytes for the EAD item. Accordingly the size of EDHOC Message 2 goes up to around 700 bytes when using EDHOC with X.509 certificates and signature Elliptic-curve Diffie-Hellman (ECDH) keys. The table summarises the overhead with respect to the size of EDHOC Message 2 when employing EDHOC with this research's proposal for certificate revocation.

The other source of overhead is that which is incurred on the constrained initiator itself, as the constrained initiator must verify the signature on the Staple (OCSP response). The constrained initiator is already assumed to be able to verify ECDSA signatures as this functionality is one of the prerequisites to running EDHOC in this configuration. The overhead is introduced with this additional signature verification on the OCSP response(Staple) which is then considered to be on the application layer as EAD items and the processing of them is opaque to EDHOC as previously mentioned. The OCSP responder is configured to signs OCSP responses using ECDSA P-256. The EDHOC implementation employed [28] by the research uses Mbed TLS as a TLS/SSL toolset, accordingly it uses the latter for signature verification. The nRF board contains a crypto cell CC310, but there's currently no support for using CC310 to accelerate Mbed TLS.

Running the proposed approach on a microcontroller with a different crypto cell could probably yield better results in terms of time taken for signature verification, although it is unclear if the observed total protocol time increase is only due to larger message sizes or if it is also the time required for signature verification of the OCSP response staple, and constructing and processing the introduced EAD items in EDHOC.

6.4 Conclusion

The proposed solution for certificate revocation in resource constrained environments successfully utilizes standard protocols to achieve the revocation functionality. By specifying the integration vector for the lightweight OCSP profile and using EAD items in EDHOC in a manner non intrusive to the protocol, the proposed changes maintain the current functionality of the latter and the integrity of the security mechanisms in place. Setting up the Experiment and measurement with constrained configurations shows the potential of the proposed method to handle the challenges imposed by resource constrained environments. The research specifies security considerations for its proposal, accordingly maintaining the security of the certificate revocation process.

The comparison of message sizes in subsection 6.2.1 concludes that the Lightweight OCSP profile proposed by the research delivers a a reduction of $\approx$ 76.2% of the size of the current profile of the OCSP response, while delivering the certificate revocation information and maintaining the integrity of the protocol's security.

The proposed certificate revocation approach utilizing EDHOC and OCSP Stapling proves to build a solid foundation for certificate revocation in IoT, with the approach being tested on a microcontroller in a resource constrained setting, the overhead in terms of power consumption on the microcontroller's side was calculated to be a 30% increase in total power consumption with respect to running plain EDHOC(EDHOC without Certificate Revocation), the 30% increase corresponds to 1.26 Watts when running EDHOC initiator on an nRF52840DK over 6LoWPAN.

This research work was presented in the IETF 115 meeting under the LAKE working group, the meeting concluded with a recommendation from the chairs of the working group to write an internet draft outlining the Certificate Revocation method proposed by this research. Additionally, a paper discussing the proposed method is being drafted at the time of writing this text.

6.5 Future work

Performing extensive power profiling of the proposed method in terms of analysing the signature verification process of the staple individually, and analysing the processing of the proposed EAD items and their inclusion into the EDHOC messages would give insight to potential vectors for decreasing the overhead of the certificate revocation process.

The proposed solution is independent of the hardware platform used, profiling the performance on different microcontrollers that utlize different crypto chips would yield interesting results that would contribute to the scientific body of the research surrounding the topic.

Widening the scope by looking into neighbouring communication scenarios would be a good approach, for example; a scenario that considers two constrained nodes trying to establish a session would be interesting in terms of its requirements and security considerations.

List of Tables

List of Figures

Listings

Acronyms

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
ASN.1	Abstract Syntax Notation 1
BLE	Bluetooth Low Energy
CA	Certificate Authority
CBOR	Concise Binary object representation
CDDL	Concise Data Definition Language
CoAP	Constrained Application Protocol
COSE	CBOR Object Signing and Encryption
CRL	Certificate Revocation list
DTLS	Datagram Transport Layer Security
EAD	External Authorization Data
EC	Elliptic-Curve
ECDH	Elliptic-curve Diffie-Hellman
ECDHE	Elliptic-Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic-Curve Digital Signature Algorithm
EDHOC	Ephemeral Diffie-Hellman Over COSE
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IoT	Internet of Things
IPV6	Internet Protocol Version 6
LORaWAN	Low-range Wider-area network
LPWAN	Low-power Wide-area network
NB-IoT	Narrow Band IoT
OCSP	Online Certificate Status Protocol

PKC	Public Key Cryptography
PKI	Public Key Infrastructure
PSK	Pre-shared keys
RAM	Random Access Memory
RFC	Request for comments
RPK	Raw Public Keys
RTC	Realtime Clock
RTOS	Real-time Operating System
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UTF	Unicode Transformation Format
VA	Validation Authority

Bibliography

[1] S. Farrell, “Low-power wide area network (lpwan) overview,” Internet Requests for Comments, RFC Editor, RFC 8376, May 2018.

[2] C. Bormann, M. Ersue, and A. Keranen, “Terminology for constrained-node networks,” Internet Requests for Comments, RFC Editor, RFC 7228, May 2014, http://www.rfc-editor.org/rfc/rfc7228.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7228.txt

[3] Z. Shelby, K. Hartke, and C. Bormann, “The constrained application protocol (coap),” Internet Requests for Comments, RFC Editor, RFC 7252, June 2014, http://www.rfc-editor.org/rfc/rfc7252.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7252.txt

[4] G. Selander, J. P. Mattsson, and F. Palombini, “Ephemeral diffie-hellman over cose (edhoc),” Working Draft, IETF Secretariat, Internet-Draft draft-ietf-lake-edhoc-18, November 2022, https://www.ietf.org/archive/id/draft-ietf-lake-edhoc-18.txt. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-lake-edhoc-18.txt

[5] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, “Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile,” Internet Requests for Comments, RFC Editor, RFC 5280, May 2008, http://www.rfc-editor.org/rfc/rfc5280.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5280.txt

[6] C. Crane. (2022, 12) Crl explained: What is a certificate revocation list. [Online]. Available: https://www.thesslstore.com/blog/crl-explained-what-is-a-certificate-revocation-list/

[7] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, “X.509 internet public key infrastructure online certificate status protocol - ocsp,” Internet Requests for Comments, RFC Editor, RFC 6960, June 2013, http://www.rfc-editor.org/rfc/rfc6960.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6960.txt

[8] Y. Pettersen, “The transport layer security (tls) multiple certificate status request extension,” Internet Requests for Comments, RFC Editor, RFC 6961, June 2013, http://www.rfc-editor.org/rfc/rfc6961.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6961.txt

[9] P. Hoffman and J. S. and, “New asn.1 modules for the public key infrastructure using x.509 (pkix),” Internet Requests for Comments, RFC Editor, RFC 5912, June 2010, https://www.rfc-editor.org/info/rfc5912. [Online]. Available: https://www.rfc-editor.org/info/rfc5912

[10] J. P. Mattsson, G. Selander, S. Raza, J. Höglund, and M. Furuhed, “Cbor encoded x.509 certificates (c509 certificates),” Working Draft, IETF Secretariat, Internet-Draft draft-ietf-cose-cbor-encoded-cert-04, July 2022, https://www.ietf.org/archive/id/draft-ietf-cose-cbor-encoded-cert-04.txt. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-cose-cbor-encoded-cert-04.txt

[11] C. Bormann and P. Hoffman, “Concise binary object representation (cbor),” Internet Requests for Comments, RFC Editor, STD 94, December 2020.

[12] (2022, 12) Cbor playground - online cbor encoder/decoder. [Online]. Available: https://cbor.me/

[13] J. Schaad, "Cbor object signing and encryption (cose): Structures and process," Internet Requests for Comments, RFC Editor, STD 96, August 2022.

[14] E. Rescorla, H. Tschofenig, and N. Modadugu, "The datagram transport layer security (dtls) protocol version 1.3," Internet Requests for Comments, RFC Editor, RFC 9147, April 2022.

[15] E. Rescorla, "The transport layer security (tls) protocol version 1.3," Internet Requests for Comments, RFC Editor, RFC 8446, August 2018.

[16] W. Eddy, "Transmission control protocol (tcp)," Internet Requests for Comments, RFC Editor, STD 7, August 2022.

[17] J. Postel, "User datagram protocol," Internet Requests for Comments, RFC Editor, STD 6, August 1980, http://www.rfc-editor.org/rfc/rfc768.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc768.txt

[18] J. P. Mattsson, F. Palombini, and M. Vučinić, "Comparison of coap security protocols," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-lwig-security-protocol-comparison-06, December 2022, https://www.ietf.org/archive/id/draft-ietf-lwig-security-protocol-comparison-06.txt. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-lwig-security-protocol-comparison-06.txt

[19] S. Tanner Lindemer, "Digital certificate revocation for the internet of things," Master's thesis, KTH Royal Institute of Technology in Stockholm, 2019.

[20] C. Boudagdigue, A. Benslimane, and A. Kobbane, "Cluster-based certificate revocation in industrial lot networks using signaling game," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[21] L. Duan, Y. Li, and L. Liao, "Flexible certificate revocation list for efficient authentication in iot," in *Proceedings of the 8th International Conference on the Internet of Things*, ser. IOT '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3277593.3277595

[22] E. Rescorla and B. Korver, "Guidelines for writing rfc text on security considerations," Internet Requests for Comments, RFC Editor, BCP 72, July 2003, http://www.rfc-editor.org/rfc/rfc3552.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3552.txt

[23] OpenSSL. (2022, 12) The openssl toolkit. [Online]. Available: https://www.openssl.org/

[24] M. Seliem and K. Elgazzar, "Ioteway: A secure framework architecture for 6lowpan based iot applications," in *2018 IEEE Global Conference on Internet of Things (GCIoT)*, 2018, pp. 1–5.

[25] N. Semi. (2022, 12) nrf52840 dk. [Online]. Available: https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dk

[26] Joulescope. (2022, 12) Joulescope js110. [Online]. Available: https://www.joulescope.com/products/joulescope-precision-dc-energy-analyzer

[27] JetPerch. (2022, 12) Joulescope scripting examples repository - github repository. [Online]. Available: https://github.com/jetperch/pyjoulescope_examples

[28] S. Hristozov. (2022, 12) uoscore-uedhoc - edhoc implementation - github repository. [Online]. Available: https://github.com/eriptic/uoscore-uedhoc

[29] H. Birkholz, C. Vigano, and C. Bormann, "Concise data definition language (cddl): A notational convention to express concise binary object representation (cbor) and json data structures," Internet Requests for Comments, RFC Editor, RFC 8610, June 2019.

[30] E. B. (NIST) and Q. D. (NIST), "Sp 800-57 part 3 rev. 1 recommendation for key management, part 3: Application-specific key management guidance," 2015-01-22 2015.

Appendices

Appendix A

1 Function implementation for converting the OCSP response to the Lightweight Profile proposed by the research

Listing 1 will show the function implemented for converting the OCSP response to the lightweight profile proposed by the research, according to the design presented in chapter 4.

Listing 1: Function for converting to the Lightweight Profile proposed by the research

```c
/*OCSP_RESPONSE* resp: The OCSP response in ASN.1 structure
 X509* issuer: the CA certificate
 char* rkeyfile: OCSP responder key file
 X509 *cert: The client certificate
*/
/*The function performs the conversion from the OCSP response to the lightweight
     profile proposed by the research*/
/*The function returns a pointer to bytestring included the lightweight profiled
     response*/

uint8_t *OCSP_convert_to_tiny(OCSP_RESPONSE* resp, X509* issuer, char* rkeyfile,
     X509 *cert)
{
    OCSP_BASICRESP *bs = NULL;
    EVP_MD *cert_id_md = NULL;
    const ASN1_OCTET_STRING *pid;
    const X509_NAME *pname;
    int  status;
    int  reason;
    ASN1_GENERALIZEDTIME *revtime;
    ASN1_GENERALIZEDTIME *thisupd;
    ASN1_GENERALIZEDTIME *nextupd;
    const ASN1_GENERALIZEDTIME *producedat;
    EVP_PKEY *rkey = NULL;
    EVP_MD_CTX *mdctx = NULL;
    char *passinarg=NULL, *passin=NULL;
    uint8_t *sig;

    STACK_OF(X509) *issuers = NULL;



    //Certificate paths //in a refactor these should be passed by the
       application
    char *CAfile="/usr/lib/ssl/demoCA/certs/ca.pem";
    char *respkeyfile="/usr/lib/ssl/ocsp_ec.key";
    char *client_certfile="/usr/lib/ssl/client.pem";

    cert = load_cert(client_certfile, FORMAT_UNDEF, "certificate");
    if (cert == NULL)
        {
            printf("failed to load client cert\n");
            return NULL;
        }

    issuer = load_cert(CAfile, FORMAT_UNDEF, "issuer certificate");
    if (issuer == NULL)
        return NULL;
    if (issuers == NULL) {
        if ((issuers = sk_X509_new_null()) == NULL)
            return NULL;
    }
    //add issuer to issuers stack
    if (!sk_X509_push(issuers, issuer))
        return NULL;


    if (issuers == NULL) {
        if ((issuers = sk_X509_new_null()) == NULL)
            return NULL;
    }
    //add issuer to issuers stack
    if (!sk_X509_push(issuers, issuer))
        return NULL;
    unsigned long verify_flags;
    verify_flags |= OCSP_TRUSTOTHER;

     if (cert_id_md == NULL)
        cert_id_md = (EVP_MD *)EVP_sha1();
    char* outfile=NULL;
    BIO* out;


    //OCSP_response_get1_basic() decodes and returns the OCSP_BASICRESP
       structure contained in resp.
    bs = OCSP_response_get1_basic(resp);
    if (bs == NULL) {
        printf("failed to parse response\n");
        return NULL;
    }
    OCSP_CBOR_RESPONSE *tiny_resp;
    tiny_resp=tiny_response_item(); //get ptr to cbor struct


    const OCSP_BASICRESP* bs_const=bs;

    //get responderID---------------------
    int err= OCSP_resp_get0_id(bs_const,&pid,&pname);
    if(!err)
        return NULL;
    char *name=X509_NAME_oneline(pname, NULL, 0);
    tiny_resp->responderID= malloc(strlen(name)+2); //+2 bytes for cbor encoding
    uint8_t *temp=tiny_resp->responderID;
    *temp=one_byte_n_bs;
    temp++;
    *temp=strlen(name);
    temp++;
    string2ByteArray (name,temp);


    //get client id-----------------------------------
    STACK_OF(X509) *resp_certs;
    OCSP_CERTID *client_id = OCSP_cert_to_id(cert_id_md, cert, issuer);
    ASN1_OCTET_STRING *piNameHash; ASN1_OBJECT *pmd; ASN1_OCTET_STRING *
       pikeyHash; ASN1_INTEGER *pserial;
    OCSP_id_get0_info(&piNameHash,&pmd,&pikeyHash,&pserial,client_id);
    int serial_len= pserial->length;
    temp=tiny_resp->certID.sn;
    *temp=BS_SMALL+serial_len;
    temp++;
    memcpy(temp, pserial->data, serial_len);

    temp=tiny_resp->certID.issuer_kh;
    *temp=BS_SMALL+sha_1_hsize;
    temp++;
    memcpy(temp, pikeyHash->data, sha_1_hsize);

    temp=tiny_resp->certID.issuer_h;
    *temp=BS_SMALL+sha_1_hsize;
    temp++;

    memcpy(temp, piNameHash->data, sha_1_hsize);

    tiny_resp->certID.hashAlg=1;

    // get certStatus---------------------------------------------
    if(!OCSP_resp_find_status(bs, client_id, &status,
                      &reason,
                      &revtime,
                      &thisupd,
                      &nextupd))
        return NULL;
    if (status==V_OCSP_CERTSTATUS_GOOD) //the status is an integer constant and
       the pointer just points to one of them
        {
            tiny_resp->certStatus=good_cert;

        }
    else if(status==V_OCSP_CERTSTATUS_REVOKED)
        {
            tiny_resp->certStatus=revoked_cert;
        }
    //------------------------------------------------------------

    //get producedAt

    producedat= OCSP_resp_get0_produced_at(bs);
    struct tm producedat_struct;
    err=ASN1_TIME_to_tm(producedat,&producedat_struct);


    if (!err)
        return NULL;

    outfile=malloc(25);
    sprintf(outfile,"%04d-%2d-%02dT%02d:%02d:%02dZ",producedat_struct.tm_year
       +1900,producedat_struct.tm_mon+1,producedat_struct.tm_mday,
       producedat_struct.tm_hour,producedat_struct.tm_min,producedat_struct.
       tm_sec);

    tiny_resp->producedat= malloc(22);
    temp=tiny_resp->producedat;
    *temp=TAG_ZERO;
    temp++;
    *temp=strlen(outfile)+TEXT_TAG;
    temp++;
    string2ByteArray (outfile,temp);

    //get nonce--------------------------------------------------
    int resp_idx;
    X509_EXTENSION *resp_ext;
    //get idx for nonce
    resp_idx = OCSP_BASICRESP_get_ext_by_NID(bs, NID_id_pkix_OCSP_Nonce, -1);
    resp_ext = OCSP_BASICRESP_get_ext(bs, resp_idx); //get the extension
    //get the octet string
    ASN1_OCTET_STRING *resp_nonce;
    resp_nonce=X509_EXTENSION_get_data(resp_ext);
    uint8_t *nonce_charString =resp_nonce->data;
    // int nonce_len= resp_nonce->length;
    temp=tiny_resp->nonce;
    *temp=one_byte_n_bs;
    temp++;
    *temp=noncesize;
    temp++;
    memcpy(temp, nonce_charString+2, noncesize); //+2 to remove previous
       encoding

    //add response type (this is just an arbitrary value for this scope)
    tiny_resp->responseType=1;

    //we need to allocate 200 bytes mem for our responseData
    uint8_t * responseData=malloc(responseData_size); //remember to free all my
        mallocs
    size_t responseData_len=0;

    // response_type: unsigned int
    uint8_t *walk=responseData+2; //walk is gonna traverse //leave two bytes for
         bytestring encoding

    *walk=tiny_resp->responseType; //we know that's 1 byte representable
    walk++; //increment ptr
    responseData_len++;

    // responderID:   byteString
    memcpy(walk,tiny_resp->responderID,strlen(tiny_resp->responderID));
    walk+=strlen(tiny_resp->responderID); //move len respID bytes
    responseData_len+=strlen(tiny_resp->responderID);


    // producedAt:    time with tag 0
    memcpy(walk,tiny_resp->producedat,strlen(tiny_resp->producedat));
    walk+=strlen(tiny_resp->producedat);
    responseData_len+=strlen(tiny_resp->producedat);


    // nonce:         bytestring
    memcpy(walk,tiny_resp->nonce,sizeof(tiny_resp->nonce));
    walk+=sizeof(tiny_resp->nonce);
    responseData_len+=sizeof(tiny_resp->nonce);

    // certID:        CBOR map
    *walk=CBOR_ARRAY+4; //add array encoding for 4 items
    walk++;
    responseData_len++;
    *walk=tiny_resp->certID.hashAlg;
    walk++;
    responseData_len++;
    memcpy(walk,tiny_resp->certID.issuer_h,sizeof(tiny_resp->certID.issuer_h));
    walk+=sizeof(tiny_resp->certID.issuer_h);
    responseData_len+=sizeof(tiny_resp->certID.issuer_h);
    memcpy(walk,tiny_resp->certID.issuer_kh,sizeof(tiny_resp->certID.issuer_kh))
        ;
    walk+=sizeof(tiny_resp->certID.issuer_kh);
    responseData_len+=sizeof(tiny_resp->certID.issuer_kh);
    memcpy(walk,tiny_resp->certID.sn,sizeof(tiny_resp->certID.sn));
    walk+=sizeof(tiny_resp->certID.sn);
    responseData_len+=sizeof(tiny_resp->certID.sn);

    // cert status:   unsigned int
    *walk=tiny_resp->certStatus;
    walk++;
    responseData_len++;
```

```c

    //add byte string encoding so that you sign the length of the response as well
    walk=responseData; //go to head
    *walk=one_byte_n_bs;
    walk++;
    *walk=responseData_len;

    //Sign responseData

    rkey = load_key(respkeyfile, FORMAT_UNDEF, 0, passin, NULL,
                    "responder private key");
    if (rkey == NULL) printf("Failed to load rkey!!\n");

    /* Create the Message Digest Context */
    if(!(mdctx = EVP_MD_CTX_create())) printf("Failed to create Message Digest Context\n");
    /* Initialise the DigestSign operation - SHA-256 has been selected as the message digest function in this example */
    if(1 != EVP_DigestSignInit(mdctx, NULL, EVP_sha256(), NULL, rkey)) printf("Failed to initialise signing op\n");
    /* Call update with the message */
    if(1 != EVP_DigestSignUpdate(mdctx, responseData, responseData_len+2)) printf("Signing Failed!\n"); //+2 to include cbor bytestring encoding
    /* Finalise the DigestSign operation */
    /* First call EVP_DigestSignFinal with a NULL sig parameter to obtain the length of the
     * signature. Length is returned in slen */
    size_t slen=0;
    if(1 != EVP_DigestSignFinal(mdctx, NULL, &slen)) printf("failed to get signatureLen\n");
    // printf("Signature Length:%ld\n",slen); //72 bytes because DER encoding adds 8 bytes
    //mbedtls also deals with DER encoded signatures
    /* Allocate memory for the signature based on size in slen */
    if(!(sig = malloc(slen+2))) printf("Failed to allocate mem for sig\n"); //+2 for bytestring encoding
    /* Obtain the signature */
    uint8_t *sig_noenc=sig;
    sig_noenc+=2; //leave room for encoding
    if(1 != EVP_DigestSignFinal(mdctx, sig_noenc, &slen)) printf("Failed to obtain signature value\n");
    sig_noenc=sig;
    *sig_noenc=one_byte_n_bs; //byte string header
    sig_noenc++;
    *sig_noenc=slen;




    size_t tinyresp_total_len=2+2+responseData_len+2+slen+1;

    uint8_t *signed_tinyResponse=malloc(tinyresp_total_len); //total_len[2]-
        cborBytestring_header[2]-responseData[responseData_len]-
        cborBytestring_header[2]-signature[slen]+sigAlg[1]
    walk=signed_tinyResponse;
    uint16_t *walk_16;
    walk_16=(uint16_t*)signed_tinyResponse;
    *walk_16=tinyresp_total_len;
    walk+=2;
    memcpy(walk,responseData,responseData_len+2);
    walk=walk+2+responseData_len;
    memcpy(walk,sig,slen+2);
    walk=walk+2+slen;
    *walk=3; //some label for the sigalg



    return signed_tinyResponse; //bytestring

}
```

2 Example of the lightweight profiled OCSP Response

Figure 1 shows an example of a lightweight profiled OCSP Response, the figure contains the CBOR encodings of each header in the lightweight profile of the OCSP response proposed by the research in chapter 4. `Tiny Response including ECDSA-p256 Signature` contains the complete response structure after the concatenation of all the listed headers in the figure. `issuerKeyHash`, `issuerHash` and `hashAlg` and `Serial` are encoded as a CBOR array and named certID.

```
responderID:58 5c 2f 43 3d 53 45 2f 53 54 3d 53 74 6f 63 6b 68 6f 6c 6d 2f 4f 3d
45 44 48 4f 43 2d 4f 43 53 50 2f 4f 55 3d 45 44 48 4f 43 2d 4f 43 53 50 2f 43 4e
3d 44 65 6d 70 2d 4f 43 53 50 5f 45 43 2f 65 6d 61 69 6c 41 64 64 72 65 73 73 3d
45 44 48 4f 43 2d 4f 43 53 50 40 69 6f 2e 63 6f 6d
Serial:42 10 02
issuerKeyHash:54 a1 76 fa 31 49 b9 e1 c9 f6 40 08 6e 4a 65 0e 30 dc 31 45 62
issuerHash:54 90 c2 48 eb 88 1a ad 4c 41 e5 f8 a8 62 cc cd 1f c2 46 ca 7b
hashAlg:1
certStatus:1 (Good)
producedAt:c0 74 32 30 32 33 2d 20 31 2d 30 33 54 30 30 3a 33 31 3a 35 36 5a
nonce:58 20 8a f6 f4 30 eb e1 8d 34 18 40 17 a9 a1 1b f5 11 c8 df f8 f8 34 73 0b
96 c1 b7 c8 db ca 2f c3 b6
responseData:58 c7 01 58 5c 2f 43 3d 53 45 2f 53 54 3d 53 74 6f 63 6b 68 6f 6c
6d 2f 4f 3d 45 44 48 4f 43 2d 4f 43 53 50 2f 4f 55 3d 45 44 48 4f 43 2d 4f 43 53
50 2f 43 4e 3d 44 65 6d 70 2d 4f 43 53 50 5f 45 43 2f 65 6d 61 69 6c 41 64 64 72
65 73 73 3d 45 44 48 4f 43 2d 4f 43 53 50 40 69 6f 2e 63 6f 6d c0 74 32 30 32 33
2d 20 31 2d 30 33 54 30 30 3a 33 31 3a 35 36 5a 58 20 8a f6 f4 30 eb e1 8d 34 18
40 17 a9 a1 1b f5 11 c8 df f8 f8 34 73 0b 96 c1 b7 c8 db ca 2f c3 b6 84 01 54 90
c2 48 eb 88 1a ad 4c 41 e5 f8 a8 62 cc cd 1f c2 46 ca 7b 54 a1 76 fa 31 49 b9 e1
c9 f6 40 08 6e 4a 65 0e 30 dc 31 45 62 42 10 02 01
SignatureVal:58 47 30 45 02 20 62 83 0b 07 04 75 49 c3 2d 4c ec 18 84 a7 a4 47
16 3b 71 d9 77 48 8a 58 4a eb 4d 7a aa 61 73 52 02 21 00 a2 15 ae 12 b3 c0 d1 15
bf 06 a9 d1 c6 05 90 6c bd b0 b0 22 1b c8 14 6d b4 24 af 4a e5 4e b3 d6
Full concatenation of all the previous gives the complete bytestring:
Lightweight Profiled OCSP Response:58 c7 01 58 5c 2f 43 3d 53 45 2f 53 54 3d 53
74 6f 63 6b 68 6f 6c 6d 2f 4f 3d 45 44 48 4f 43 2d 4f 43 53 50 2f 4f 55 3d 45 44
48 4f 43 2d 4f 43 53 50 2f 43 4e 3d 44 65 6d 70 2d 4f 43 53 50 5f 45 43 2f 65 6d
61 69 6c 41 64 64 72 65 73 73 3d 45 44 48 4f 43 2d 4f 43 53 50 40 69 6f 2e 63 6f
6d c0 74 32 30 32 33 2d 20 31 2d 30 33 54 30 30 3a 33 31 3a 35 36 5a 58 20 8a f6
f4 30 eb e1 8d 34 18 40 17 a9 a1 1b f5 11 c8 df f8 f8 34 73 0b 96 c1 b7 c8 db ca
2f c3 b6 84 01 54 90 c2 48 eb 88 1a ad 4c 41 e5 f8 a8 62 cc cd 1f c2 46 ca 7b 54
a1 76 fa 31 49 b9 e1 c9 f6 40 08 6e 4a 65 0e 30 dc 31 45 62 42 10 02 01 58 47 30
45 02 20 62 83 0b 07 04 75 49 c3 2d 4c ec 18 84 a7 a4 47 16 3b 71 d9 77 48 8a 58
4a eb 4d 7a aa 61 73 52 02 21 00 a2 15 ae 12 b3 c0 d1 15 bf 06 a9 d1 c6 05 90 6c
bd b0 b0 22 1b c8 14 6d b4 24 af 4a e5 4e b3 d6 03
```

Figure 1: An example of an OCSP response with the Lightweight profile

Appendix B

1 Function implementation for parsing `staple-request` and performing an lightweight profiled OCSP request using the functions implemented by the research

Listing 1 will show the function implemented for converting the OCSP response to the lightweight profile proposed by the research, according to the design presented in chapter 5.

Listing 1: Function for processing `staple-request`, performing an lightweight profiled OCSP request and constructing `staple-response` in EAD_2

```c
//Application can implement this function to process EAD_1 and act on the protocol state
enum err process_ead_1(struct edhoc_responder_context *c, uint8_t *ead_1, uint32_t *ead_1_len, uint8_t *g_x, uint32_t g_x_len )
{
  printf("processing ead 1!\n");
  PRINT_ARRAY("msg1 ead_1", ead_1, *ead_1_len);
  uint8_t *walk=ead_1;
  bool _nonce;


  //We won't implement a generic parser at this point
  if(*walk==STAPLE_REQUEST_LABEL)
  {
    uint8_t *tinyOCSP_response_staple= (uint8_t*)malloc(AD_DEFAULT_SIZE+64);
    walk++; //pointing at ead_value head now
    //create pointers for responder ID list and optional nonce include
    uint8_t *responderIdList, *nonce_option;
    printf("Received tinyOCSP stapling request.\n");
    //parse staple request in EAD Value
    TRY(parse_stapleRequest_ead_1_value(&walk, &responderIdList, &nonce_option)); //update pointers here
    if (nonce_option!=NULL)
      {
        printf("{stapleRequestLabel:-2,ResponderIdList:NULL,Nonce:True}\n");
        _nonce=true;
      }
    else
      printf("{stapleRequestLabel:-2,ResponderIdList:NULL}\n");


    if (_nonce)
    {
      //create the request appending g_x as nonce
      PRINT_ARRAY("Performing tinyOCSP request with (g_x) as nonce:",g_x,g_x_len
         );
                //here we call our generate ocsp request function which request
                   the lightweight profiled ocsp response
      if(generate_ocsp_request(&tinyOCSP_response_staple, g_x, g_x_len))
        printf("Received tinyOCSP response, can now construct staple for EAD_2\n
           ");
      else
        return error_message_sent;
    }


    //Encoding the staple and constructing staple-response in EAD 2
    uint8_t *size_ptr=tinyOCSP_response_staple;
    uint32_t tinyOCSP_response_staple_len=size_ptr[0]+((size_ptr[1]&0xf0)*4096)
       +((size_ptr[1]&0x0f)*256)-2; //remove length
    printf("response staple len= %d\n",tinyOCSP_response_staple_len);
    //add length of ead_value byteString
    //add staple request label 0x21
    size_ptr=(uint8_t*)&tinyOCSP_response_staple_len;
    uint32_t ead_len_encoding=size_ptr[0]+((size_ptr[1]&0xf0)*4096)+((size_ptr
       [1]&0x0f)*256);
    size_ptr=(uint8_t*)&ead_len_encoding;
    uint8_t *temp=c->ead_2.ptr;
    memcpy(c->ead_2.ptr+7, tinyOCSP_response_staple+2,
       tinyOCSP_response_staple_len); //3 for EAD len encoding and 2 for label
       and ead value encoding
    *(temp+6) =size_ptr[0];
    *(temp+5) =size_ptr[1];
    *(temp+4) =0x59;
    *(temp+3) =0x21;
    uint16_t ead_2_len=tinyOCSP_response_staple_len+4;
    size_ptr=(uint8_t*)&ead_2_len;
    *(temp+2) =size_ptr[0];
    *(temp+1) =size_ptr[1];
    *temp = 0x59;
    uint16_t ead_2_len_enc=ead_2_len+3;
    //[ead_len_encoding(3)][label(1)][staple_len_encoding(3)[staple]
    // memcpy(c->ead_2.ptr+4,&0x59,1);
    // memcpy(c->ead_2.ptr+3,&0x21,1);
    //last memcpy to add total size of ead
    free(tinyOCSP_response_staple);
    c->ead_2.len=ead_2_len_enc;
    PRINT_ARRAY("EAD_2",c->ead_2.ptr,c->ead_2.len);


  }

  return ok;
}
```