Optimizing Certificate Validation in OT Environments

Steffen Fries*, Rainer Falk*, Andreas Guettinger+
Foundational Technologies*; Smart Infrastructure+
Siemens AG, Germany
e-mail: { steffen.fries | rainer.falk | andreas.guettinger }@siemens.com

***Abstract*—User authentication based on digital certificates is becoming more and more common in industrial environments, also known as Operational Technology (OT). Users may be both human users, as well as technical users like devices or applications. Common to all types of certificate-based authentication is that a certificate must be validated before trusting it. This poses a significant effort on the involved devices concerning compute power, memory, and the complexity of the required validation logic. Different approaches already exist to offload certificate validation from specifically constrained end entities. To increase overall efficiency even more, this paper proposes an optimization for infrastructures supporting offloading certificate validation.**

Keywords–cybersecurity; credential; digital certificate; public-key infrastructure; device authentication, industrial security; power system automation.

I. INTRODUCTION

User authentication in Operational Technology (OT), including critical infrastructures is increasingly achieved involving X.509 [1] certificates and corresponding private keys as authentication and authorization credentials. Users in this context may be human users, but also technical users like devices or applications. Examples for critical infrastructures are power system automation systems, spanning from centralized power generation up to increasingly deployed Distributed Energy Resources (DER). Further examples are industrial automation or intelligent traffic systems. Utilized credentials are typically managed by a so-called Public-key Infrastructure (PKI) following well-defined operational processes involving Identity and Access Management (IAM) to ensure proper authorization of certificate issuance.

X.509 certificates are prominently used in several security protocols to support secure communication between different entities. Most commonly, Transport Layer Security (TLS, version 1.2 [2] is still widely used, version 1.3 [3] application is increasing) is applied to protect TCP/IP-based communication, or the complementary Datagram Transport Layer Security (DTLS) [4] for the protection of UDP/IP communication. They employ certificates in the handshake for peer authentication and to negotiate security parameters of the intended communication session. Specifically, TLS is used in different industrial environments to protect domain-specific communication protocols. An example from power system automation is to secure IEC 61850 [5], specified in the IEC 62351 series [6]. A further example from the industrial automation domain is OPC-UA [7], which supports TLS as underlying security protocol.

Even though certificates are issued by Certification Authorities (CA), part of a PKI following procedural security requirements and policies, they need to be validated by a relying party, before accepting the certificates to establish trusted communication, or before accepting signed information received from a sender. Certificate validation can be a time and performance consuming process, as it includes the verification of the certificate itself but also the verification along the certification path up to a common trust anchor (root certificate). This effort becomes amplified when Post-Quantum Cryptography (PQC) is used for X.509 certificates (see also [8]). This is caused by the much larger key sizes for public-keys for some of the post-quantum cryptographic algorithms, which may turn out to be a problem when employed on constrained devices. The certification validation policies may become complex and even operator-specific during the transition phase towards PQC if classical and PQC algorithms are used in combination.

This paper provides insight into typically used certificate validation approaches and proposes a novel approach to keep the effort for the validation of the relying party certificate overall as low as possible. This optimization targets operational infrastructures using constrained devices, which may either not have enough processing power or memory to perform the validation locally or devices in environments, or which do not have access to information from other sources required in the validation procedure. It utilizes available technology but provides an enhancement to also limit the burden on the infrastructure.

The remainder of this industrial research paper is structured as follows. Section II introduces certificates and already known approaches for their validation. Section III introduces an optimization concept of certificate validation services, to allow a relying party to determine trustworthiness in the received certificate more efficiently. Section IV concludes the paper and gives an outlook towards future work.

II. RELATED TECHNOLOGY

This section provides an overview of X.509 certificates, their structure and validation options. Specifically discussed is the potential offloading of certificate validation tasks (partially or completely) from an end entity to a supporting infrastructure or communication peer to save memory, time, and processing power. Moreover, local caching of revocation information is further considered, as it is already applied in today's communication infrastructures.

A. Public-key Certificates – Structure and Validation

As stated before, ITU-T X.509 [1] certificates are used for different purposes like entity authentication, e.g., in the context of key establishment in security protocols like TLS or DTLS, or to provide authenticity and integrity-protection of data, e.g., for firmware or software updates. Figure 1 shows the general concept of a public-key certificate, the binding of the user identity to the corresponding public-key. The user possesses also the corresponding private key, which is kept secret and is used to provide proof-of-possession, which can be verified by the relying party based on the certificate.

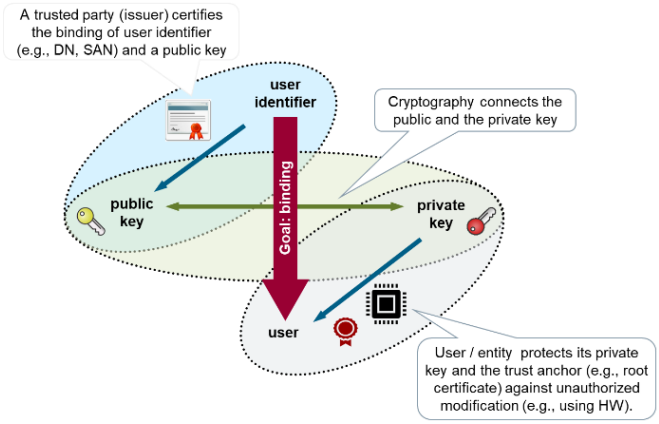

Figure 1. Concept of Binding Public-keys to User Identities [8]

The certificate itself is issued by a trusted third party, a CA of a PKI that digitally signs the certificate during certificate issuing. When the certificate is used by the user to authenticate, the certificate signature is verified by the relying party as part of certificate validation, similar for all certificates in the path up to a trust anchor (sometimes also called root certificate). Also, further attributes included in the certificate are validated.

In addition to public-key certificates, attribute certificates are also defined in X.509, which can be seen as temporary enhancement of public-key certificates. They do not contain public keys but additional attributes that are typically connected to the holder of the public-key certificate [8]. Regarding the certificate validation, which is the focus of this paper, they are handled in a similar way. For simplicity, the paper will therefore concentrate on public-key certificates for the description of the validation optimization, as this is the most broadly used form of X.509 certificates.

```
Certificate ::= SIGNED{TBSCertificate}

TBSCertificate ::= SEQUENCE {
  version                 [0]  Version DEFAULT v1,
  serialNumber                 CertificateSerialNumber,
  signature                    AlgorithmIdentifier{{SupportedAlgorithms}},
  issuer                       Name,
  validity                     Validity,
  subject                      Name,
  subjectPublicKeyInfo         SubjectPublicKeyInfo,
  issuerUniqueIdentifier  [1] IMPLICIT UniqueIdentifier OPTIONAL,
  ...,
  [[2:  -- if present, version shall be v2 or v3
  subjectUniqueIdentifier  [2] IMPLICIT UniqueIdentifier OPTIONAL]],
  [[3:  -- if present, version shall be v2 or v3
  extensions               [3]  Extensions OPTIONAL ]]
  -- If present, version shall be v3]]
} (CONSTRAINED BY { -- shall be DER encoded -- } )
```

Figure 2. Public-key Certificate structure (see [1])

ITU-T X.509 [1] defines the structure and content of public-key certificates, as well as the verification of the components. As shown in Figure 2, the certificate is a structure signed by the CA, containing the `subject` as the name of the entity (user) and the `subjectPublicKeyInfo` structure with further information about cryptographic algorithm and the contained public-key. The signature is created typically using traditional asymmetric cryptographic signature algorithms like Rivest Shamir, Adleman (RSA) or Elliptic Curve Digital Signature Algorithm (ECDSA). Different key sizes are supported by these signature algorithms. As outlined in [8], PQC algorithms are increasingly demanded to address potential threats in the advent of a cryptographically relevant quantum computer. A PQC algorithm considered as replacement is for instance Module-Lattice-Based Digital Signature Algorithm (ML-DSA), formerly known as CRYSTALS-Dilithium [9], which has a much larger key size compared to traditional cryptographic algorithms.

During the certificate validation, several components of the certificate are verified. Depending on an organization's security policy, the minimum set of components of an X.509 certificate to be verified comprises the

- expected identity (typically contained in the subject or subject alternative name),
- validity period,
- signature of issuing certificate authority.

In addition, the certificate revocation state is checked. This information is commonly provided by the issuing CA and indicates if the certificate has been revoked before the validity end has been reached. The revocation information can be fetched from the CA in different ways (see subsections II.B.1) and II.B.2) below). Revocation may be done if the certificate or the corresponding key has been compromised, or the certificate was superseded.

As stated before, the verification must be done not only for the end entity certificate, but for all certificates in the certificate chain up to the trust anchor, including the verification of their revocation state, which also requires communication with different issuing CAs.

B. Certificate Validation Support Approaches

1) Online Certificate Validation Protocol

CAs typically provide Certificate Revocation Lists (CRLs), containing information about revoked certificates, signed by the CA. These lists may grow and may be difficult to handle, specifically on constrained devices. CRLs are generally distributed by a CRL distribution point to which at least temporary access is necessary. An alternative is the use of the Online Certificate Status Protocol (OCSP, IETF RFC 6960, [10]). It enables clients to query the revocation state of single or set of certificates via an OCSP responder. This lifts the handling of complete CRLs from the clients. OCSP support needs an online connection to the OCSP responder. OCSP responder URL and CRL-DP URL are included in issued certificates.

2) Server Certificate Validation Protocol

Applying OCSP, as shown in the previous subsection, still requires validation of certificate components locally on the

verifying device. A further approach exists, which delegates the certificate validation to a central authority. It is specified as Server Certificate Validation Protocol (SCVP, IETF RFC 5055, [11]) and allows a client to send the certificate in question and a validation policy to the SCVP server, which takes over end entity certificate validation, certificate path construction, and certificate path validation. This increases efficiency on the client side, but still poses load to the server side, specifically in networks with a high number of clients employing certificates more frequently. The approach proposed in Section III kicks in here to optimize server-side processing.

3) Certificate Authorization Validation Lists

The complete opposite way to certificate revocation is the explicit authorization of certificates using so called Certificate Authorization Validation Lists (CertAVL, see ITU-T X.509, [1]). They constitute allow lists, which explicitly provide the information, which certificates are considered trustworthy. This allows to offload revocation handling to the central point creating the CertAVL. X.509 also defines critical extensions to mandate the validation of an CertAVL before accepting it. In contrast to CRLs or OCSP responses, CertAVLs are managed by the system operator, not by the issuing CA.

4) DNS-based Authentication of Named Entities (DANE)

A further approach is known as Domain Name Service-based Authentication of Named Entities, (DANE, IETF RFC 6698, [12]), which is protected by DNSSEC (IETF RFC 9364, [13]). It enables domain administrators to specify the keys or certificates used by TLS servers as DANE TLSA resource record. The DNS administrator for a domain name is typically authorized to specify identifying information about the zone. Supporting DANE, he also makes an authoritative binding between the domain name and a certificate used by a TLS server in that domain. Thus, a TLS client trusts the certificate information received via DNSSEC, after validation of the DNSSEC signature. It avoids certificate validation, as it got the authoritative information from the DNS server.

5) OCSP Stapling

A further approach is known as OCSP stapling, specifically in the context of TLS. Using OSCP stapling a constraint device can request an OCSP response from the remote site and thus avoid separate communication with an OCSP responder. This may be adventurous in situations when the requesting peer has either communication restrictions and may not reach the OCSP responder or if the OCSP communication protocol is not implemented.

For TLSv1.2 [2], this feature is specified in IETF RFC 6066 [14] as a certificate status request (`status_request`) and response extension allowing TLSv1.2 to provide an OSCP response for the server certificate along with its certificate. As this extension only allows to provide a single OCSP response, a further extension is defined in IETF RFC 6961 [15] for multi-stapling, allowing to request (`status_requestv2`) and staple OCSP responses also for intermediate CA certificates contained in the certificate list of the server certificate message. TLSv1.3 [3] provides support for requesting and stapling OCSP responses as described in IETF RFC 6066 for all certificates in the certificate list provided by the client or the server side. As it works in both directions it can accommodate situations in which the server is the constraint device, and the client is more capable peer. An example scenario would be web-based access to communication controllers.

C. Caching of Revocation Information

A common practice to avoid fetching fresh CRLs whenever a certificate is received and validated, is the usage of CRL caching (see also [1]). CRLs contain information about CRL issue time and when the next update will be provided. This allows a local implementation to cache the CRL for the period until the next update. Caching avoids additional communication and decreases the processing time for certificate validation. The downside of caching until the next update is that emergency updates during the validity period of the CRL may not be recognized. Caching of revocation information is also the base for the proposed optimization described in Section III.

III. Proposed Optimization of Certificate Validation Service

As discussed in Section II, different approaches are already available to offload certificate validation from a client. Not all of them are equally suited for OT networks. For instance, DNS is not always available, which limits the possibility to utilize the DANE approach outlined in Section II.B.4) For OT networks, specifically the use of allow lists as in Section II.B.3) or the complete offload of certificate validation as in Section II.B.2) becomes more interesting. While the use of SCVP optimizes the client-side operation, the handling of the SCVP response server can be optimized, too. This is the focus of the novel approach in this section.

As specified in IETF RFC 5055 [11], an SCVP client sends a request containing the certificate to be validated including specific verifications to be done, like the construction and validation of the certification path, key usages, etc. The validation result will be provided to the requesting client, which in turn only needs to verify the server's signed response. To optimize the SCVP server handling, it is proposed that the result of a certificate validation or certificate chain validation is provided on an SCVP Response Collector (SRC in Figure 3). This information can be used to reduce the response time for client queries for the same certificate or certificate chain, as it is no longer necessary to perform all validations separately. The SRC can be realized via different mechanisms, like:

1. publishing the result of the validation of the certificate and/or the certificate chain in a public directory (e.g., LDAP, HTTP, FTP, ...),
2. publishing the result of the validation of the certificate and/or the certificate chain using in hash chain-based ledger technology (e.g., Ethereum, Hyperledger).

Note that the choice of realization of the SRC specifically for a chosen ledger technology may have an influence on the validation effort. This counts for both the infrastructure for the SRC, but also on the client side for the interaction to query and process a SCVP response.

In addition, the security policy of the organization may need to consider that caching of validation results provides an optimization but also requires further consideration in an organization's security policy. An example storage duration of validation results to ensure it matches freshness requirements.

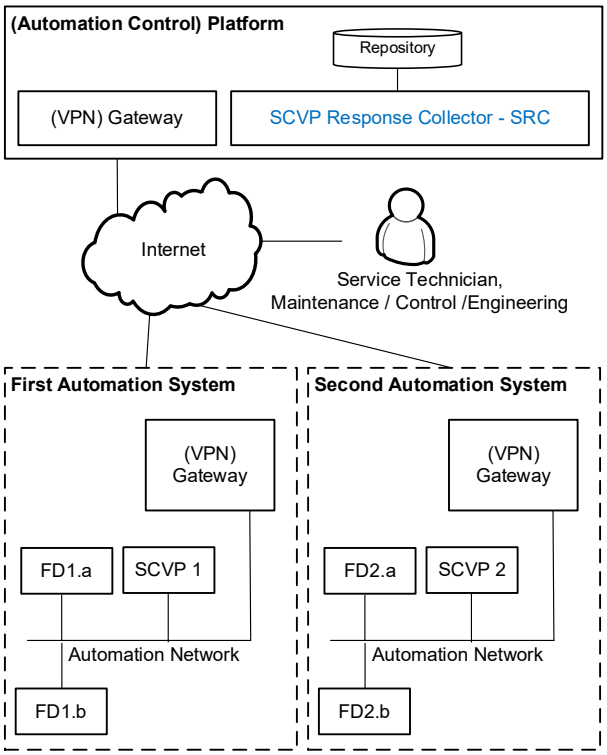

Figure 3. Example Setup for SRC operation

Figure 3 shows two automation environments (e.g., production environment, substation, etc., described as First and Second Automation System), in which SCVP server (SCVP 1 and 2) exist, which work locally as proxy for certificate validation and certificate chain validation. In both automation environments, local communication and communication with outside the domains (e.g., for remote maintenance, access to information in other networks) takes place. As soon as a Field Device (FD) of the automation environment 1 makes a certificate validation request, this is processed by the local SCVP 1 server. If a client (e.g., FD1.a) is allowed to process cached responses, SCVP 1 can first ask the SCVP Response Collector (SRC), which may be public or part of the control center, whether a corresponding validation already exists. If not, it carries out the validation and makes the response available to the field device FD. At the same time, it publishes the result of its validation in a repository as a signed data structure (signed with the private key of the SCVP 1) and thus makes the information available to the SRC for subsequent requests.

The connection to a repository can be realized either via LDAP or via ledger technology. The described interaction and abstract message flow is shown in Figure 4.

The described approach addresses the design goal to benefit the delegation of certificate validation on (constrained) clients to a more powerful centralized service and to optimize the backend service operation. A locally cached certificate validation result supports the availability of the automation system even if the central validation service should temporarily not be available.

IV. Conclusion and Future Work

This paper provides an introduction to the use of certificates and certificate validation in OT. Focus is placed on an optimization to offload efforts for certificate validation, including the validation of the certification path and revocation state of involved certificates to device external services. The proposed solution simplifies the implementation on constrained devices, e.g., by avoiding additional communication protocol stacks to be supported, and it enhances availability of the automation system if the functionality of a central CRL or OCSP responder in the infrastructure is temporarily not available or connectivity to these remote peers cannot be guaranteed from the OT environment.

The novel approach proposed to optimize the operation of a certificate validation infrastructure in an automaton environment utilizes SCVP as a standardized protocol and combines it with caching of certificate validation information. With this approach an OT system, like an automation system, is enhanced with a (local) caching functionality for certificate validation information. As caching directly relates to the freshness of validation information, the caching time is a parameter to be considered in an organization's security policy. The caching time will typically be determined based on a risk-based approach and may vary between installations.

At the time being, only the concept has been developed. The next consequent step is a practical evaluation of the proposed solution regarding the impact to the overall system, based on implementation to proof its efficiency and effectiveness. Specific points for the evaluation besides the provisioning of cached certificate validation information may comprise the analysis of the

- performance impact on the client site when using cached validation response instead of local calculation. This may be considered specifically with different certificate path length,
- impact on code size for the client side as communication protocol stacks for selected protocols may be omitted,
- impact on the infrastructure site, e.g., depending on the chosen approach for the publishing of validation results as outlined in Section III.

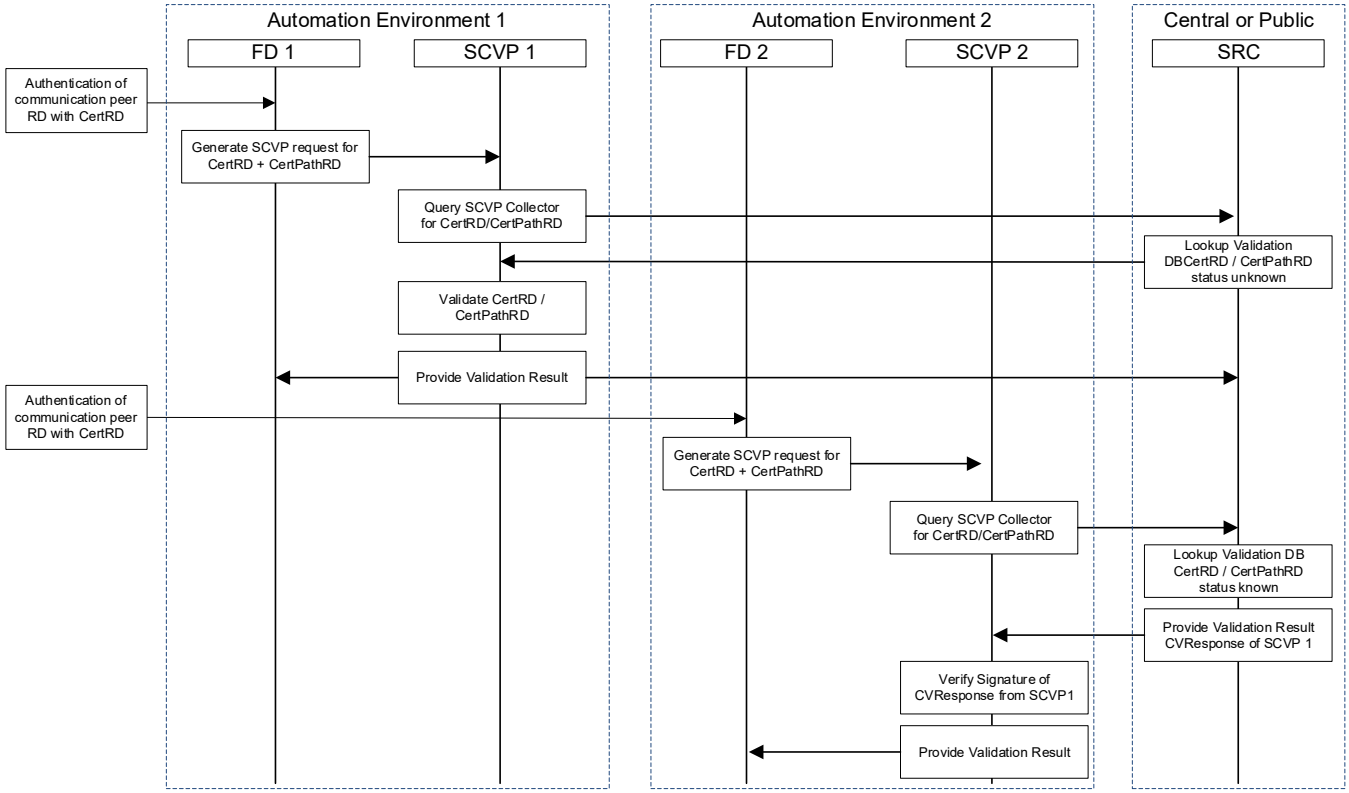

Figure 4. Example Call Flow

REFERENCES

[1] ITU-T X.509 ISO/IEC 9594-8:2020, „ITU-T X.509 Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks", 2019, [Online]. Available from: https://www.itu.int/rec/T-REC-X.509-201910-I/en, [retrieved: August 2025]

[2] T. Dierks and E. Rescorla, IETF RFC 5246, "Transport Layer Security (TLS) Protocol v1.2", August 2008, [Online]. Available from https://tools.ietf.org/html/rfc5246, [retrieved: August 2025]

[3] E. Rescorla, IETF RFC 8446, "Transport Layer Security (TLS) Protocol v1.3", August 2018, [Online]. Available from https://tools.ietf.org/html/rfc8446, [retrieved: August 2025]

[4] E. Rescorla, H. Tschofenig, and N. Modadugu, IETF RFC 9147, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", April 2022, [Online]. Available from https://datatracker.ietf.org/doc/html/rfc9147, [retrieved: August 2025]

[5] IEC 61850-x, "Power systems management and associated information exchange", [Online]. Available from: https://webstore.iec.ch/en/publication/6028, [retrieved: August 2025]

[6] IEC 62351-x, "Power systems management and associated information exchange – Data and communication security", [Online]. Available from: https://webstore.iec.ch/en/publication/6912, [retrieved: August 2025]

[7] OPC-UA, "Open Platform Communications Unified Architecture", [Online]. Available from https://reference.opcfoundation.org/, [retrieved: August 2025]

[8] S. Fries and R. Falk, "Supporting Cryptographic Algorithm Agility with Attribute Certificates", IARIA International Journal of Advances in Security, 2025 vol 17 nr. 1&2, pg. 92-98. [Online], Available from https://www.iariajournals.org/security/sec_v17_n12_2024_paged.pdf, [retrieved: August 2025]

[9] L. Ducas et al., "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme", 2017, [Online]. Available from https://eprint.iacr.org/2017/633.pdf, [retrieved: August 2025]

[10] S. Santesson et al., IETF RFC 6960, "X.509 PKI Online Certificate Status Protocol - OCSP", June 2013, [Online]. Available from https://datatracker.ietf.org/doc/html/rfc6960, [retrieved: August 2025]

[11] T. Freeman, R. Housley, A. Malpani, D. Cooper, and W. Polk, IETF RFC 5055, "Server-Based Certificate Validation Protocol (SCVP)", December 2007, [Online]. Available from https://datatracker.ietf.org/doc/html/rfc5055, [retrieved: August 2025]

[12] P. Hoffman and J. Schlyter, IETF RFC 6698, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", August 2012, [Online]. Available from https://datatracker.ietf.org/doc/html/rfc6698, [retrieved: August 2025]

[13] P. Hoffmann, IETF RFC 9364, "DNS Security Extensions (DNSSEC)", February 2023, [Online]. Available from https://datatracker.ietf.org/doc/html/rfc9364, [retrieved August 2025]

[14] D. Eastlake 3rd, IETF RFC 6066, "Transport Layer Security (TLS) Extensions: Extension Definitions", January 2011, [Online]. Available from https://datatracker.ietf.org/doc/html/rfc6066, [retrieved: August 2025]

[15] Y. Petterson, IETF RFC 6961, "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension", June 2013, [Online]. Available from https://datatracker.ietf.org/doc/html/rfc6961, [retrieved: August 2025]