Technical Requirements for Cryptographic Components of Hybrid Post-Quantum Public-Key Infrastructure Deliverable 5.1, HAPKIDO Project

João Diogo Duarte, Alessandro Amadori, and Gabriele Spini

TNO, Applied Crypto & Quantum Algorithms, The Hague, The Netherlands

e-mail: {joao.duarte}{alessandro.amadori}{gabriele.spini}@tno.nl

Executive Summary

This document forms deliverable 5.1 for the HAPKIDO (Hybrid quantum-safe Public-Key Infrastructure Development for Organisations) project. The goal of this document is to present the technical requirements that the cryptographic components of hybrid (classical/post-quantum) PKIs need to satisfy.

The document focuses on security requirements and on compatibility requirements; performance requirements are too dependent on applications and use-cases to be discussed in general terms.

Requirements have been identified based on existing guidelines on security, as well as on the current initiatives in terms of formatting cryptographic components for hybrid certificates. For the latter, the source material comes from the latest X.509 version from ITU-T, and from drafts of standards from IETF. We make the remark that the two propose different approaches to include both classical and post-quantum signatures and components in certificates.

Contents

1 Acronyms and Definitions

1.1 Acronyms

Acronym	Full Name
AES	Advanced Encryption Standard
BER	Basic Encoding Rules
CA	Certificate Authority
CCM	Counter with CBC-MAC Mode
CMS	Cryptographic Message Syntax
CRL	Certification Revocation List
CSR	Certificate Signing Request
DER	Distinguished Encoding Rules
GCM	Galois Counter Mode
KEM	Key Encapsulation Mechanism
OID	Object Identifier
SHA	Secure Hash Algorithm

1.2 Definitions

Definition 1.1 (Authenticated Encryption Cipher) *A form of encryption which also ensures the authenticity of the data that is being encrypted.*

Definition 1.2 (Authority) *An entity that is responsible for the issuance of certificates or of revocation lists [X.509, 2019].*

Definition 1.3 (Block Cipher) *A cipher that operates on fixed-length blocks of data bits.*

Definition 1.4 (Certification Authority) *An authority trusted by one or more entities to create and sign public-key certificates. Optionally the certification authority may create the subjects' keys [X.509, 2019].*

Definition 1.5 (Certificate Revocation List) *A signed list indicating a set of public-key certificates that are no longer considered valid by the issuing certification authority (CA) [X.509, 2019].*

Definition 1.6 (Cipher) *An algorithm that encrypt and decrypt information with the use of one or more keys.*

Definition 1.7 (Component Algorithm) *A single basic algorithm which is contained within a composite algorithm [Ounsworth and Pala, 2022].*

Definition 1.8 (Composite Algorithm) *An algorithm which is a sequence of two or more component algorithms [Ounsworth and Pala, 2022].*

Definition 1.9 (Cryptographic Message Syntax) *A syntax that is used to digitally sign, digest, authenticate, or encrypt arbitrary message content [Housley, 2009].*

Definition 1.10 (Digital Signature) *The result of a cryptographic transformation of data that provides data integrity and data origin authentication [FIPS 186, 2013].*

Definition 1.11 (Key Encapsulation Mechanism) *A group of public-key cryptographic algorithms that generate and encapsulate a symmetric key to be transmitted to the receiving parties. The key is then derived by these parties by a process called decapsulation.*

Definition 1.12 (Message Authentication Code) *A cryptographic checksum on data that is designed to reveal if the data has been modified in any way, albeit intentional or accidental [SP 800-38C, 2007].*

Definition 1.13 (Multiple Cryptographic Algorithms Public-key Certificate) *Public-key certificate that includes extensions for an alternative public-key scheme, an alternative digital-signature algorithm and an alternative digital signature [X.509, 2019].*

Definition 1.14 (Object identifier) *An ordered list of integer values which unambiguously identifies an object [X.660, 2011].*

Definition 1.15 (Private-key) *(In a public-key cryptosystem) that key of an entity's key pair which is known only by that entity [X.509, 2019].*

Definition 1.16 (Public-key) *That key of an entity's key pair which is publicly known by others [X.660, 2011].*

Definition 1.17 (Public-key Certificate) *A certificate binds a public-key to an entity, allows for the key's validation and provides undeniability of use of the entity's private key. It is rendered unforgeable by a digital signature with the private key of the CA that issued it [X.509, 2019].*

Definition 1.18 (Public-key Certificate Validation) *The process of ensuring that a public-key certificate was valid at a given time, including possibly the construction and processing of a certification path, and ensuring that all public-key certificates in that path were valid (e.g., were not expired or revoked) at that given time [X.509, 2019].*

Definition 1.19 (Public-key Infrastructure) *The infrastructure supporting the overall management of public-keys, including authentication, integrity and/or non- repudiation services [X.509, 2019].*

Definition 1.20 (Stream Cipher) *A cipher that operates on single bits of data.*

Definition 1.21 (Symmetric Cipher) *A cipher whose encryption and decryption keys are the same. Symmetric ciphers can either be block or stream ciphers.*

Definition 1.22 (Trust Anchor) *An entity that is trusted by a relying party and used for validating public-key certificates [X.509, 2019].*

2 Introduction

2.1 Purpose

At the time of writing, there are no generalised cryptographic requirements for hybrid public-key infrastructures. It is imperative that meaningful requirements are readily available to ensure a baseline level of security and compatibility. Hence, deliverable 5.1 aims to identify appropriate requirements for hybrid PKIs. For an introduction to public-key infrastructures and relevant concepts, please refer to [Amadori et al., 2022].

2.2 Scope

There are two important considerations when discussing scope for these requirements. Firstly, these requirements are meant to be application- and use case-independent. This has the advantage that it allows for a broader audience to make use of these requirements. However, the downside is that most non-functional requirements, such as performance, will not be considered. This is because performance (such as storage and signature time) are very application-dependent. For example, a laptop holding certificates will have completely different and more relaxed performance requirements than a small IoT device.

Secondly, only requirements that solely apply to hybrid PKIs will be considered. This is to avoid repeating requirements that apply to both hybrid and classical PKIs as that is out of scope for this deliverable. For cryptographic requirements of classical PKIs, please refer to [OASIS, 2006] and for classical X.509 certificates, refer to [X.509, 2019]. Note that for the former reference, some standards have been revised and since the page was published in 2006, some standards are out-of-date. Hence, for each standard in [OASIS, 2006], refer to its latest revision.

2.3 Requirements Overview

The two main areas of hybrid PKIs that have been identified for requirements are:

- **Cryptographic**: What requirements need to be satisfied to ensure that cryptographic schemes provide at least 128-bit security against quantum adversaries?
- **Multiple Cryptographic Algorithms Public-key Certificates**: What requirements are necessary to implement hybrid certificates?

The requirements have been elicited with the following structure:

- Requirement itself
- Requirement ID (a unique ID which indexes the requirement)
- Sources
- Further comments

3 Sources and Techniques

3.1 Sources

The following sources were used to elicit the requirements:

* International and national standards, namely:

  - **ANSI**

    ANSI X9.62 Public-Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature [ANSI X9.62, 2005]

  - **FIPS**

    FIPS 180-4 Secure Hash Standard (SHS) [FIPS 180-4, 2015]

    FIPS 197 Advanced encryption standard (AES) [FIPS 197, 2001]

    FIPS 198-1 The Keyed-Hash Message Authentication Code (HMAC) [FIPS 198-1, 2008]

  - **ISO/IEC**

    ISO/IEC 29167-21:2018 Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers [ISO/IEC 29167-21:2018, 2010]

  - **ITU-T**

    ITU-T X.509 Information technology - Open Systems Interconnection - The Directory Public-key and attribute certificate frameworks [X.509, 2019]

  - **NIST**

    NIST 800-107 Rev. 1 Recommendation for Applications Using Approved Hash Algorithms [SP 800-107 Rev. 1, 2012]

    NIST SP 800-185 SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash [SP 800-185, 2016]

    NIST SP 800-38B Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication [SP 800-38B, 2016]

    NIST Selected Algorithms 2022 [Chen et al., 2022]

* Requests for Comments (RFCs) and PKCS standards, namely:

  - **RFCs**

    RFC 2119 Key words for use in RFCs to Indicate Requirement Levels [Bradner, 1997]

    RFC 8446 The Transport Layer Security (TLS) Protocol Version 1.3 [Rescorla, 2018]

    Internet-Draft draft-truskovsky-lamps-pq-hybrid-x509-01 Multiple Public-Key Algorithm X.509 Certificates [Truskovsky et al., 2018a]

    Internet-Draft draft-ounsworth-pq-composite-keys-02 Composite Public and Private Keys For Use In Internet PKI [Ounsworth et al., 2022]

    Internet-Draft draft-ounsworth-pq-composite-sigs-07 Composite Signatures For Use In Internet PKI [Ounsworth and Pala, 2022]

  - **PKCS**

OASIS PKCS 11 TC [OASIS, 2020]

* Journal Publications

  A Fast Quantum Mechanical Algorithm for Database Search [Grover, 1996]

  Algorithms for Quantum Computation: Discrete Logarithms and Factoring [Shor, 1994]

  Transitioning to a Quantum-Resistant Public-Key Infrastructure [Bindel et al., 2017]

3.2 Techniques

The following techniques have been used to construct the requirements:

- Document and literature analysis
- Communication with experts involved in the creation of new Internet-Drafts concerning hybrid PKIs

4 Requirements

4.1 Cryptographic Requirements

In this section, requirements for the cryptographic aspect of hybrid PKIs are presented.

Requirement When selecting a symmetric block cipher, AES with a key length of **at least** 256 bits *must* be used.

Requirement ID C.CSB.01

Sources [Rescorla, 2018] [OASIS, 2020] [ISO/IEC 29167-21:2018, 2010] [FIPS 197, 2001] [Grover, 1996]

Further Comments The reason AES was chosen instead of other equally strong ciphers is due to its status as the de-facto symmetric block cipher standard.

Requirement When selecting a symmetric stream cipher, ChaCha20 with a key length of **at least** 256 bits *must* be used.

Requirement ID C.CSS.01

Sources [Rescorla, 2018] [Grover, 1996]

Further Comments The reason ChaCha20 was chosen instead of other equally strong ciphers is due to its current widespread use and lack of a de-facto stream cipher standard.

Requirement When selecting an authenticated encryption cipher, either AES-GCM or AES-CCM with a key length of **at least** 256 bits or ChaCha20-Poly1305 with a key length of **at least** 256 bits *must* be used.

Requirement ID C.CAC.01

Sources [Rescorla, 2018] [Grover, 1996]

Further Comments -

Requirement When selecting a cryptographic hash, either SHA2 or SHA3 with an output size of **at least** 384 bits *must* be used.

Requirement ID C.CCH.01

Sources [Rescorla, 2018] [FIPS 180-4, 2015] [SP 800-107 Rev. 1, 2012] [ANSI X9.62, 2005] [Grover, 1996]

Further Comments -

Requirement When selecting a message authentication code, either GMAC, CMAC, HMAC or KMAC constructions with *underlying primitives* that achieve **at least** 128 bits of quantum security *must* be used.

Requirement ID C.CMC.01

Sources [SP 800-38B, 2016] [FIPS 198-1, 2008] [SP 800-107 Rev. 1, 2012] [SP 800-185, 2016]
[Grover, 1996]

Further Comments -

Requirement When selecting a digital signature scheme, refer to national and international regulation for a list of appropriate choices. As a fallback, any of the digital signatures in the NIST Selected Algorithms *may* be used. At the time of writing, these are SPHINCS+, CRYSTALS-DILITHIUM and FALCON [Chen et al., 2022] with security level of at least 3.

Requirement ID C.CDS.01

Sources [Chen et al., 2022] [Shor, 1994]

Further Comments -

Requirement When selecting a key encapsulation mechanism, refer to national and international regulation for a list of appropriate choices. As a fallback, any of KEMs in the NIST Selected Algorithms *may* be used. At the time of writing, this is only CRYSTALS-KYBER [Chen et al., 2022] with security level of at least 3.

Requirement ID C.CKM.01

Sources [Chen et al., 2022] [Shor, 1994]

Further Comments -

4.2 Multiple Cryptographic Algorithms (Hybrid) Public key Certificates Requirements

The X.509 certificate format is defined in two standards: the ITU-T X.509 standard [X.509, 2019] and RFC 5280 of the IETF [Rescorla, 2008]. X509 certificates from IETF (the organisation that handles Internet Drafts and RFCs) and from ITU-T are compatible with each other. The reason for this split is that the ITU-T X.509 certificate extensions are designed to be used in a variety of domains, such as telecommunications and governmental agencies [ITU-T, 2022]. The IETF X.509 certificate extensions further tailor these extension fields for the Internet [Rescorla, 2008].

In terms of hybrid certificates, the two organisations are most likely going to diverge. The ITU-T has already standardised how to construct a multiple cryptographic algorithms (hybrid) public-key certificate in [X.509, 2019] and these constructions are based on an Internet-Draft by [Truskovsky et al., 2018a]. The IETF, at the time of writing, has two Internet-Drafts [Ounsworth et al., 2022, Ounsworth and Pala, 2022] that describe how to combine classical and post-quantum signatures and public/private keys. Note that these two IETF drafts are not in competition, rather, they complement each other as one focuses on keys and the other on signatures. In addition, these are work-in-progress, meaning that their contents and requirements may change.

Due to this divergence, requirements for both the ITU-T and IETF ways of handling hybrid certificates will be presented. This will not explain the process of generating and managing certificates as the focus is on their internal structure. Furthermore, since an RFC/Internet-Draft is essentially a description of a protocol that is intertwined with a series of requirements, many of the requirements in this section are directly quoted from the Internet-Drafts and/or ITU-T X.509 or are paraphrased.

Note that when referring to public-keys, unless specified otherwise, we are referring to the subject's public-key and *not* the certificate issuer's public-key. Furthermore, when referring to signatures, unless specified otherwise, we are referring to the CA signature of the subject's X.509 certificate.

4.2.1 IETF Hybrid Certificates

4.2.1.1 Example of IETF Hybrid Certificate

The guidelines for these kinds of certificates defined by the two Internet-Drafts [Ounsworth et al., 2022, Ounsworth and Pala, 2022] are not mature enough to be able to provide a full example.

The idea of the certificate is to introduce a new Object Identifier which identifies composite signatures and public-keys.

4.2.1.2 Composite Keys in Composite Signatures

Requirement A certificate's composite signature *may* be associated with a subject composite public-key as defined in [Ounsworth et al., 2022].

Requirement ID O.CKS.01

Sources [Ounsworth and Pala, 2022]

Further Comments -

4.2.1.3 Key Usage in Composite Signatures

Requirement The subject composite public-key associated with a CA's composite signature *must* have a signing-type key usage.

Requirement ID O.KUS.01

Sources [Ounsworth and Pala, 2022]

Further Comments -

Requirement In a Certification Authority (CA) certificate that utilises composite public keys, any combination of the values "`digitalSignature`, `nonRepudiation`, `keyCertSign`, `cRLSign`" *may* be present.

Requirement ID O.KUS.02

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on `id-composite-key`, refer to Section 3.1.1 of [Ounsworth and Pala, 2022].

4.2.1.4 Key Usage in Composite Public/Private Keys

Requirement Key usage *must* be used with composite keys, with the requirement that the specified key usage *must* apply to all component keys.

Requirement ID O.KUK.01

Sources [Ounsworth et al., 2022]

Further Comments -

4.2.1.5 CompositeSignatureValue

Requirement The certificate's `CompositeSignatureValue` *must* contain one signature value produced by each component algorithm, and in the same order as in the associated `CompositeParams` object.

Requirement ID O.CSV.01

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on `CompositeSignatureValue` and `CompositeParams`, refer to Section 3.3 and 4.1, respectively, of [Ounsworth and Pala, 2022].

Requirement A certificate's `CompositeSignatureValue` *must* contain the same number of component signatures as the corresponding public and private keys.

Requirement ID O.CSV.02

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on `CompositeSignatureValue`, refer to Section 3.3 of [Ounsworth and Pala, 2022].

4.2.1.6 CompositePublicKey

Requirement A composite key *must* contain at least two component public-keys.

Requirement ID O.CUK.01

Sources [Ounsworth et al., 2022]

Further Comments -

Requirement A `CompositePublicKey` *must not* contain a component public-key which itself describes a composite key.

Requirement ID O.CUK.02

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `CompositePublicKey`, refer to Section 3.2 of [Ounsworth et al., 2022].

Requirement Each component `SubjectPublicKeyInfo` *shall* contain an `AlgorithmIdentifier` OID which identifies the public-key type and parameters for the public-key contained within it.

Requirement ID O.CUK.03

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `SubjectPublicKeyInfo` and `AlgorithmIdentifier`, refer to Section 3.2 of [Ounsworth et al., 2022].

4.2.1.7 CompositePrivateKey

Requirement A `CompositePrivateKey` *must* contain at least two component private keys.

Requirement ID O.CIK.01

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `CompositePrivateKey`, refer to Section 3.3 of [Ounsworth et al., 2022].

Requirement The order of component signature private keys *must* correspond to the order of the component public-keys.

Requirement ID O.CIK.02

Sources [Ounsworth et al., 2022]

Further Comments -

4.2.1.8 Encoding Rules for Composite Signatures

Requirement When an octet string is required, the DER encoding of the composite data structure *shall* be used directly.

Requirement ID O.SER.01

Sources [Ounsworth and Pala, 2022]

Further Comments For an overview of DER encoding, refer to [X.690, 2021].

Requirement When a bit string is required, the octets of the DER encoded composite data structure *shall* be used as the bits of the bit string, with the most significant bit of the first octet becoming the first bit, and so on, ending with the least significant bit of the last octet becoming the last bit of the bit string.

Requirement ID O.SER.02

Sources [Ounsworth and Pala, 2022]

Further Comments For an overview of DER encoding, refer to [X.690, 2021]. In addition, the text is identical as O.KER.02, however, this requirement focuses on composite signatures.

Requirement In the interests of simplicity and avoiding compatibility issues, implementations that parse structures in Requirement O.SER.02 *may* accept both BER and DER.

Requirement ID O.SER.03

Sources [Ounsworth and Pala, 2022]

Further Comments For an overview of BER and DER encodings, refer to [X.690, 2021].

4.2.1.9 Encoding Rules for Composite Keys

Requirement When an octet string is required, the DER encoding of the composite data structure *shall* be used directly.

Requirement ID O.KER.01

Sources [Ounsworth et al., 2022]

Further Comments For an overview of DER encoding, refer to [X.690, 2021].

Requirement When a bit string is required, the octets of the DER encoded composite data structure *shall* be used as the bits of the bit string, with the most significant bit of the first octet becoming the first bit, and so on, ending with the least significant bit of the last octet becoming the last bit of the bit string.

Requirement ID O.KER.02

Sources [Ounsworth et al., 2022]

Further Comments For an overview of DER encoding, refer to [X.690, 2021]. In addition, the text is identical as O.SER.02, however, this requirement focuses on composite keys.

4.2.1.10 id-alg-composite (Generic Composite Signatures)

Requirement The `id-alg-composite` identifier *must* be used in the `sa-CompositeSignature.identifier`.

Requirement ID O.IAC.01

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on `id-alg-composite` and `sa-CompositeSignature.identifier`, refer to Section 4.1 and Section 3.2, respectively, of [Ounsworth and Pala, 2022].

Requirement The number of certificate signatures that have been composited *must* be included in the `id-alg-composite` identifier.

Requirement ID O.IAC.02

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on `id-alg-composite`, refer to Section 4.1 of [Ounsworth and Pala, 2022].

Requirement The certificate signature's `CompositeParams` sequence *must* contain the same component algorithms listed in the same order as in the associated subject's `CompositePublicKey`.

Requirement ID O.IAC.03

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on `CompositeParams` and `CompositePublicKey`, refer to Section 4.1 and Section 5.1, respectively, of [Ounsworth and Pala, 2022].

4.2.1.11 Explicit Composite Signatures Variant (Optional)

The following requirement is optional, but useful when interacting with protocols that use OIDs to identify cryptographic algorithms.

Requirement In the Explicit Composite Signatures variant, the certificate's signature is encoded as defined in Section 3.2 in [Ounsworth and Pala, 2022], however the `sa-CompositeSignature.identifier` *shall* be an OID which is registered to represent a specific combination of component signature algorithms.

Requirement ID O.ECS.01

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on `sa-CompositeSignature`, refer to Section 3.2 of [Ounsworth and Pala, 2022].

4.2.1.12 Explicit Composite Keys Variant (Optional)

The following three requirements are optional, but useful when interacting with protocols that use OIDs to identify cryptographic algorithms.
Note that the these three requirements can still be fulfilled alongside the requirements in Section 4.2.1.11. This is because the Explicit Composite Signatures Variant specifically applies to composite signatures whereas the Explicit Composite Keys Variant applies to composite keys.

Requirement Implementations *must* check that the component `AlgorithmIdentifier` OIDs and parameters match those expected by the definition of the explicit algorithm.

Requirement ID O.ECK.01

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `AlgorithmIdentifier`, refer to Section 3.2 of [Ounsworth et al., 2022].

Requirement In this variant, the public-key is encoded as defined in Section 3 and Section 3.2 of [Ounsworth et al., 2022], however the `PUBLIC-KEY.id` *shall* be an OID which is registered to represent a specific combination of component public-key types.

Requirement ID O.ECK.02

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `PUBLIC-KEY.id`, refer to Section 4.2 of [Ounsworth et al., 2022].

Requirement Implementations *should* first parse a component's `SubjectPublicKeyInfo .algorithm`, and ensure that it matches what is expected for that position in the explicit key, and then proceed to parse the `SubjectPublicKeyInfo.subjectPublicKey`.

Requirement ID O.ECK.03

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `SubjectPublicKeyInfo.algorithm` and `Sub jectPublicKeyInfo.subjectPublicKey`, refer to Section 4.2 of [Ounsworth et al., 2022].

4.2.1.13 Composite Signature Generation Process

Requirement The certificate's signature generation process *must* fail if one of the private keys is a composite with the OID `id-alg-composite`.

Requirement ID O.SGP.01

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on `id-alg-composite`, refer to Section 4.1 of [Ounsworth and Pala, 2022].

Requirement The certificate's composite signature *must* produce, and include in the output, a signature value for every component key in and include in the output, a signature value for every component key in the corresponding `CompositePublicKey` and in the same order.

Requirement ID O.SGP.02

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on `CompositePublicKey`, refer to Section 5.1 of [Ounsworth and Pala, 2022].

Requirement For security when using a generic composite signature algorithm as defined in Section 4.1 of [Ounsworth and Pala, 2022], the list of component signature algorithms which *may* be carried in a `CompositeParams` object, *should* be included in the signed message M to prevent an attacker from substituting a weaker algorithm which is compatible with the same public-key.

Requirement ID O.SGP.03

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on `CompositeParams`, refer to Section 4.1 of [Ounsworth and Pala, 2022].

4.2.1.14 Composite Signature Verification Process

Requirement In the absence of an application profile specifying otherwise, compliant applications *must* output "Valid signature" (true) if and only if all of the certificate's component signatures were successfully validated, and "Invalid signature" (false) otherwise.

Requirement ID O.SVP.01

Sources [Ounsworth and Pala, 2022]

Further Comments Essentially, this demands that all signatures in the composite field need to be checked unless a policy specifies otherwise.

Requirement The certificate's signature verification procedure *must* fail if any of the public-keys or algorithm identifiers are composite with the OID `id-alg-composite`.

Requirement ID O.SVP.02

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on `id-alg-composite`, refer to Section 4.1 of [Ounsworth and Pala, 2022].

Requirement In the absence of a policy mechanism for specifying acceptable subsets of algorithms that can be easily updated to reflect new cryptanalytic breakthroughs, clients *must* perform signature verifications in the AND mode defined in Section 5 [Ounsworth and Pala, 2022].

Requirement ID O.SVP.03

Sources [Ounsworth and Pala, 2022]

Further Comments -

4.2.1.15 Asymmetric Key Packages (CMS) for Keys

Requirement When encoding composite private keys, the `privateKeyAlgorithm` in the `OneAsymmetricKey` *shall* be set to `id-composite-key` or to an OID corresponding to an explicit composite key.

Requirement ID O.AKP.01

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `privateKeyAlgorithm`, `OneAsymmetricKey` and `id-composite-key`, refer to Section 5.2, Section 3.3 and Section 4.1, respectively, of [Ounsworth et al., 2022].

Requirement The parameters of the `privateKeyAlgorithm` *shall* be a sequence of `AlgorithmIdentifier` objects, each of which are encoded according to the rules defined for each of the different keys in the composite private key.

Requirement ID O.AKP.02

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `privateKeyAlgorithm` and `AlgorithmIdentifier`, refer to Section 5.2, and Section 3.3 respectively, of [Ounsworth et al., 2022].

Requirement The value of the `privateKey` field in the `OneAsymmetricKey` *shall* be set to the DER encoding of the sequence of private key values that make up the composite key.

Requirement ID O.AKP.03

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `privateKey` and `OneAsymmetricKey`, refer to Section 5.2 and Section 3.3, respectively, of [Ounsworth et al., 2022]. Furthermore, for an overview of DER encoding, refer to [X.690, 2021].

Requirement The number and order of elements in the sequence of private key values *shall* be the same as identified in the sequence of parameters in the `privateKeyAlgorithm`.

Requirement ID O.AKP.04

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `privateKeyAlgorithm`, refer to Section 5.2 of [Ounsworth et al., 2022].

Requirement The value of the `publicKey` (if present) *shall* be set to the DER encoding of the corresponding `CompositePublicKey`.

Requirement ID O.AKP.05

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `privateKey`, `CompositePublicKey` and `privateKeyAlgorithm`, refer to Section 5.1, Section 3.2 and Section 5.2, respectively, of [Ounsworth et al., 2022]. Furthermore, for an overview of DER encoding, refer to [X.690, 2021].

Requirement If the `publicKey` is present, the number and order of component keys *must* be the same as identified in the sequence of parameters in the `privateKeyAlgorithm`.

Requirement ID O.AKP.06

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `publicKey` and `privateKeyAlgorithm`, refer to Section 5.2 of [Ounsworth et al., 2022].

4.2.1.16 Subset Signature Generation in OR Modes

Note that a subset signature is a signature that uses a subset of component algorithms [Ounsworth and Pala, 2022].

Requirement The certificate signer (for example, the CA) *must* produce a signature value with each of their component private keys.

Requirement ID O.SSO.01

Sources [Ounsworth and Pala, 2022]

Further Comments -

Requirement In an effort to keep compliant implementations simple and secure, implementations claiming to be compliant with this draft *must not* generate subset signatures in OR-mode and *must* reject during verification any subset signatures that they encounter.

Requirement ID O.SSO.02

Sources [Ounsworth and Pala, 2022]

Further Comments For an explanation on the OR mode, refer to Section 7.2 in [Ounsworth and Pala, 2022].

4.2.1.17 Key Mismatch in Explicit Composite

Requirement Implementations *must* check that the component `AlgorithmIdentifier` OIDs and parameters match those expected by the definition of the explicit algorithm.

Requirement ID O.KMS.01

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `AlgorithmIdentifier`, refer to Section 3.2 of [Ounsworth et al., 2022].

Requirement Implementations *should* first parse a component's `SubjectPublicKeyInfo.algorithm`, and ensure that it matches what is expected for that position in the explicit key, and then proceed to parse the `SubjectPublicKeyInfo.subjectPublicKey`.

Requirement ID O.KMS.02

Sources [Ounsworth et al., 2022]

Further Comments For an explanation on `SubjectPublicKeyInfo.algorithm` and `SubjectPublicKeyInfo.subjectPublicKey`, refer to Section 4.2 of [Ounsworth et al., 2022].

Requirement Similar checks that were referred to in Requirement O.KMS.02 *must* be done when handling the corresponding private key.

Requirement ID O.KMS.03

Sources [Ounsworth et al., 2022]

Further Comments -

4.2.1.18 Reuse of Keys in a Composite Public-Key

Requirement The components of a composite key *should not* also appear in single-key certificates.

Requirement ID O.RPK.01

Sources [Ounsworth et al., 2022]

Further Comments -

Requirement It is *recommended* that component keys in a composite key are not to be re-used in other contexts.

Requirement ID O.RPK.02

Sources [Ounsworth et al., 2022]

Further Comments -

4.2.1.19 Checking for Compromised Key Reuse

Requirement When checking for compromised keys, the `CompositePublicKey` structure *must* be unpacked and the individual component keys *must* be compared to the CRL.

Requirement ID O.CCK.01 item[Sources] [Ounsworth et al., 2022]

Further Comments For an explanation on `CompositePublicKey`, refer to Section 3.2 of [Ounsworth et al., 2022]. According to, [Ounsworth et al., 2022], "for the purposes of key reuse checks, the composite public key structures need to be unpacked so that primitive keys are being compared. For example if the composite key RSA1, PQ1 is revoked for key compromise, then the keys RSA1 and PQ1 need to be individually considered revoked".

4.2.2 ITU-T X.509 Hybrid Certificates

4.2.2.1 Example of ITU-T Hybrid Certificate

This example from the penultimate slide in [Truskovsky et al., 2018b]

```
X.509 Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 4097 (0x1001)
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: C=US, ST=CA, O=Test, CN=ECDSA-HSS-Hybrid-Test..
    Validity:
        Not Before: Dec 12 09:00:00 2021 GMT
        Not After : Jan 12 08:59:59 2022 GMT
    Subject Public Key Info:
        Public Key Algorithm: id-ecPublicKey
        ...
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        ...
        Alt Signature Algorithm:
            hss-with-SHA512
        Subject Alt Public Key Info:
            Public Key:
                00:00:00:...
            Winternitz Value: 8 (0x00000004)
            Tree Height: 25 (0x00000009)
            Leighton-Micali Hierarchical Signature System
        Alt Signature Value:
            Signature:
                30:82:0a:74:...
  Signature Algorithm: ecdsa-with-SHA256
    30:45:02:21:...
```

4.2.2.2 Extensions to X.509

Requirement The issuer *may* mark the extension containing data concerning the post-quantum key and algorithm as critical.

Requirement ID X.ETX.01

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments -

4.2.2.3 CSR Attributes

Requirement An entity creating a CSR *must* include `SubjectAltPublicKeyInfoAttr` and `AltSignatureAlgorithmAttr` and `AltSignatureValueAttr` or none of them.

Requirement ID X.CSR.01

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `SubjectAltPublicKeyInfoAttr`, `AltSignatureAlgorithmAttr` and `AltSignatureValueAttr`, refer to Section 4 of [Truskovsky et al., 2018a].

4.2.2.4 Handling Hybrid Certificates

Requirement If an issuer certificate or root of trust has a post-quantum public-key, but a subject certificate issued by the issuer certificate or root of trust does not contain a post-quantum signature then the verifier *should* reject the subject certificate.

Requirement ID X.HHC.01

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments -

Requirement If a CA is issuing a subject certificate and the issuer certificate or root of trust contains a post-quantum public-key, then the CA *should* add a post-quantum signature to the subject certificate.

Requirement ID X.HHC.01

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments -

4.2.2.5 Creating Multiple Public-Key Algorithm CSRs

Requirement The `PreCertificationRequestInfo` object *must* contain the `SubjectAltPublicKeyInfoAttr` attribute carrying the post-quantum public-key and algorithm for the CSR being created.

Requirement ID X.CMC.01

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `PreCertificationRequestInfo` and `SubjectAltPublicKeyInfoAttr`, refer to Section 3 and 2.2.1, respectively, of [Truskovsky et al., 2018a]

Requirement The `PreCertificationRequestInfo` object *must* contain the `AltSignatureAlgorithmAttr` attribute.

Requirement ID X.CMC.02

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `PreCertificationRequestInfo` and `AltSignatureAlgorithmAttr`, refer to Section 3 and 2.2.2, respectively, of [Truskovsky et al., 2018a]

Requirement After the post-quantum signature is calculated, the post-quantum signature *must* be added as an `AltSignatureValueAttr` attribute to create the `CertificationRequestInfo` object.

Requirement ID X.CMC.03

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `AltSignatureValueAttr` and `CertificationRequestInfo`, refer to Section 2.2.3 and 3 of [Truskovsky et al., 2018a]

Requirement If the system issues a single public-key algorithm CSR, then that CSR *must* NOT contain any of `PreCertificationRequestInfo` and `AltSignatureAlgorithmAttr` and `PublicKeyInfoAttr`.

Requirement ID X.CMC.04

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments -

4.2.2.6 Creating Multiple Public-Key Algorithm Certificates

Requirement A multiple public-key algorithm certificate *may* contain the `SubjectAltPublicKeyInfoExt` extension.

Requirement ID X.CMA.01

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `SubjectAltPublicKeyInfoExt`, refer to Section 2.3.1 of [Truskovsky et al., 2018a]

Requirement If the certificate's subject has a post-quantum public-key which they wish to bind to their identity, then the public-key and algorithm *must* be placed in the `SubjectAltPublicKeyInfoExt` extension. If there is no post-quantum algorithm, then this extension will not be added to the certificate.

Requirement ID X.CMA.02

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `SubjectAltPublicKeyInfoExt`, refer to Section 2.3.1 of [Truskovsky et al., 2018a]

Requirement If a CA is issuing a certificate with a post-quantum signature, the extensions field of the `PreTBSCertificate` *must* contain the `AltSignatureAlgorithmExt` extension.

Requirement ID X.CMA.03

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `PreTBSCertificate` and `AltSignatureAlgorithmExt`, refer to Section 4 and 2.3.2, respectively, of [Truskovsky et al., 2018a]

Requirement The post-quantum signature of the `PreTBSCertificate` *must* be calculated using the post-quantum private key of the Issuer, which is the private key associated with the public-key found in the Issuer's `SubjectAltPublicKeyInfoExt` extension.

Requirement ID X.CMA.04

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `PreTBSCertificate` and `SubjectAltPublicKeyInfoExt`, refer to Section 4 and 2.3.1, respectively of [Truskovsky et al., 2018a]

Requirement After the post-quantum signature is calculated, the post-quantum signature *must* be added as an `AltSignatureValueExt` extension to the extensions list of the `PreTBSCertificate`, resulting in the `TBSCertificate`.

Requirement ID X.CMA.05

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `AltSignatureValueExt` and `PreTBSCertificate`, refer to Section 2.3.3 and 4 respectively of [Truskovsky et al., 2018a]

Requirement If the upgraded CA's policy allows it to process single public-key algorithm CSRs and issue single public-key algorithm certificates, and the certificate issuer's certificate has a post-quantum public-key, and the CA receives a single-algorithm CSR, the CA *should* still include properly calculated `AltSignatureValueExt` and `AltSignatureAlgorithmExt` extensions in the certificate.

Requirement ID X.CMA.06

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `AltSignatureValueExt` and `AltSignatureAlgorithmExt`, refer to Section 2.3.3 and 2.3.2, respectively, of [Truskovsky et al., 2018a]

4.2.2.7 Verifying Multiple Public-Key Algorithm Certificates

Requirement A certificate that contains an `AltSignatureValueExt` extension but does not contain an `AltSignatureAlgorithmExt` extension cannot be verified under the post-quantum public-key algorithm and so *should* be rejected as being malformed.

Requirement ID X.VMC.01

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `AltSignatureValueExt` and `AltSignatureAlgorithmExt`, refer to Section 2.3.3 and 2.3.2, respectively, of [Truskovsky et al., 2018a]

Requirement A certificate that contains an `AltSignatureAlgorithmExt` extension but does not contain an `AltSignatureValueExt` extension *should* be rejected.

Requirement ID X.VMC.02

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `AltSignatureAlgorithmExt` and `AltSignatureValueExt`, refer to Section 2.3.2 and 2.3.3 of [Truskovsky et al., 2018a]

4.2.2.8 Creating Multiple Public-Key Algorithm Certificate Revocation Lists

Requirement If the CRL issuer's certificate has a `SubjectAltPublicKeyInfoExt` extension, the CRL *should* be created with a post-quantum signature.

Requirement ID X.CMR.01

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `SubjectAltPublicKeyInfoExt`, refer to Section 2.3.1 of [Truskovsky et al., 2018a]

Requirement The extensions field of the `PreTBSCertList` *must* contain the `AltSignatureAlgorithmExt` extension.

Requirement ID X.CMR.02

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `PreTBSCertList` and `AltSignatureAlgorithmExt`, refer to Section 4 and 2.3.2, respectively, of [Truskovsky et al., 2018a]

Requirement The post-quantum signature of the `PreTBSCertList` *must* be calculated using the post-quantum private key of the CRL issuer.

Requirement ID X.CMR.03

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `PreTBSCertList`, refer to Section 4 of [Truskovsky et al., 2018a]

Requirement After the post-quantum signature is calculated, the post-quantum signature *must* be added as an `AltSignatureValueExt` extension to the extensions list of the `PreTBSCertList`.

Requirement ID X.CMR.04

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `AltSignatureValueExt` and `PreTBSCertList`, refer to Section 2.3.3 and 4, respectively, of [Truskovsky et al., 2018a]

Requirement If the CRL issuer's certificate supports a post-quantum key, the CRL *should* be created with a post-quantum signature.

Requirement ID X.CMR.05

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments -

4.2.2.9 Verifying Multiple Public-Key Algorithm Certificate Revocation Lists

Requirement To validate multiple public-key algorithm CRLs, entities that support hybrid X.509 certificates *should* additionally verify the post-quantum signatures along the path as described in Section 4.2 of [Truskovsky et al., 2018a].

Requirement ID X.VRC.01

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments -

Requirement To verify multiple public-key algorithm CRLs, step (g) of the CRL Processing algorithm in Section 5.2 of [Truskovsky et al., 2018a] *must* be modified to instead verify dual signatures on the complete CRL.

Requirement ID X.VRC.02

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments -

Requirement To validate multiple public-key algorithm CRLs, entities that support hybrid X.509 certificates *should* additionally verify the post-quantum signatures.

Requirement ID X.VRC.03

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments -

Requirement When a party that is compliant with hybrid X.509 certificate processes a non-multiple public-key algorithm CRL and encounters a multiple public-key algorithm certificate in the list of revoked certificates, it *should not* treat that certificate as revoked.

Requirement ID X.VRC.03

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments -

Requirement If a multiple public-key algorithm certificate is revoked, whether because the classical key is compromised, the post-quantum key is compromised or other reason, both the classical and post-quantum keys *should* be considered revoked.

Requirement ID X.VRC.04

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments -

4.2.2.10 Post-Quantum Security Considerations

Requirement A CA *should* add a post-quantum signature to any certificate it issues if the issuing certificate contains a `SubjectAltPublicKeyInfoExt` extension.

Requirement ID X.PQS.01

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on `SubjectAltPublicKeyInfoExt`, refer to Section 2.3.1 of [Truskovsky et al., 2018a]

Requirement When verifying certificates or CRLs, an application *should* reject certificates or CRLs if they don't contain a post-quantum signature, but the issuer certificate does contain a `SubjectAltPublicKeyInfoExt` or the trust anchor has a post-quantum public-key.

Requirement ID X.PQS.02

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments For an explanation on SubjectAltPublicKeyInfoExt, refer to Section 2.3.1 of [Truskovsky et al., 2018a]

Requirement If the trust anchor is not a certificate, the post-quantum signature *should* be added if the trust anchor has an associated post-quantum public-key.

Requirement ID X.PQS.03

Sources [Truskovsky et al., 2018a] [X.509, 2019]

Further Comments -

References

[Amadori et al., 2022] Amadori, A., Duarte, J. a. D., and Spini, G. (2022). Literature Overview of Public-Key Infrastructures, with Focus on Quantum-Safe Variants Deliverable 4.1, HAPKIDO Project. Report, TNO, Den Haag, Netherlands.

[ANSI X9.62, 2005] ANSI X9.62 (2005). Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA). Standard, ANSI.

[Bindel et al., 2017] Bindel, N., Herath, U., McKague, M., and Stebila, D. (2017). Transitioning to a quantum-resistant public key infrastructure.

[Bradner, 1997] Bradner, S. O. (1997). Key words for use in RFCs to Indicate Requirement Levels. RFC 2119.

[Chen et al., 2022] Chen, L., Moody, D., and Liu, Y.-K. (2022). Post-Quantum Cryptography Selected Algorithms 2022. `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`. [Accessed: 15/09/2022].

[FIPS 180-4, 2015] FIPS 180-4 (2015). Secure Hash Standard (SHS). Standard, NIST, Gaithersburg, MD.

[FIPS 186, 2013] FIPS 186 (2013). Digital Signature Standard (DSS). Standard, NIST, Gaithersburg, MD.

[FIPS 197, 2001] FIPS 197 (2001). Advanced encryption standard (AES). Standard, NIST, Gaithersburg, MD.

[FIPS 198-1, 2008] FIPS 198-1 (2008). The Keyed-Hash Message Authentication Code (HMAC). Standard, NIST, Gaithersburg, MD.

[Grover, 1996] Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *STOC*, pages 212–219. ACM.

[Housley, 2009] Housley, R. (2009). Cryptographic Message Syntax (CMS). RFC 5652.

[ISO/IEC 29167-21:2018, 2010] ISO/IEC 29167-21:2018 (2010). Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers. Standard, International Organization for Standardization, Geneva, CH.

[ITU-T, 2022] ITU-T (2022). First ITU-T X.509 Day. `https://www.itu.int/en/ITU-T/Workshops-and-Seminars/2022/0509/Pages/default.aspx`. [Accessed: 24/10/2022].

[OASIS, 2006] OASIS (2006). PKI Technical Standards. `http://www.oasis-pki.org/resources/techstandards/`. [Accessed: 15/09/2022].

[OASIS, 2020] OASIS (2020). OASIS PKCS 11 TC. `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=pkcs11`. [Accessed on 23-05-2022].

[Ounsworth and Pala, 2022] Ounsworth, M. and Pala, M. (2022). Composite Signatures For Use In Internet PKI. Internet-Draft draft-ounsworth-pq-composite-sigs-07, Internet Engineering Task Force. Work in Progress.

[Ounsworth et al., 2022] Ounsworth, M., Pala, M., and Klaußner, J. (2022). Composite Public and Private Keys For Use In Internet PKI. Internet-Draft draft-ounsworth-pq-composite-keys-02, Internet Engineering Task Force. Work in Progress.

[Rescorla, 2008] Rescorla, E. (2008). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280.

[Rescorla, 2018] Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446.

[Shor, 1994] Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS*, pages 124–134. IEEE Computer Society.

[SP 800-107 Rev. 1, 2012] SP 800-107 Rev. 1 (2012). Recommendation for Applications Using Approved Hash Algorithms. Special publication, NIST, Gaithersburg, MD.

[SP 800-185, 2016] SP 800-185 (2016). SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash. Special publication, NIST, Gaithersburg, MD.

[SP 800-38B, 2016] SP 800-38B (2016). Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication. Special publication, NIST, Gaithersburg, MD.

[SP 800-38C, 2007] SP 800-38C (2007). Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality. Special publication, NIST, Gaithersburg, MD.

[Truskovsky et al., 2018a] Truskovsky, A., Geest, D. V., Fluhrer, S., Kampanakis, P., Ounsworth, M., and Mister, S. (2018a). Multiple Public-Key Algorithm X.509 Certificates. Internet-Draft draft-truskovsky-lamps-pq-hybrid-x509-01, Internet Engineering Task Force. Work in Progress.

[Truskovsky et al., 2018b] Truskovsky, A., Geest, D. V., Fluhrer, S., Kampanakis, P., Ounsworth, M., and Mister, S. (2018b). Multiple Public-Key Algorithm X.509 Certificates. `https://datatracker.ietf.org/meeting/101/materials/slides-101-lamps-multiple-public-key-algorithm-x509-certificates-01.pdf`. [Accessed: 01/12/2022].

[X.509, 2019] X.509 (2019). Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks. Standard, ITU-T, Geneva, Switzerland.

[X.660, 2011] X.660 (2011). X.660 : Information technology - Procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree. Recommendation, ITU-T, Geneva, Switzerland.

[X.690, 2021] X.690 (2021). Information Technology - Abstract Syntax Notation One (ASN.1) & ASN.1 encoding rules. Standard, ITU-T, Geneva, Switzerland.