INTERNATIONAL STANDARD

ISO/IEC 7816-8

Fifth edition
2021-08

AMENDMENT 1
2023-11

Identification cards — Integrated circuit cards —

Part 8: Commands and mechanisms for security operations

AMENDMENT 1: Interoperability for the interchange of security operations using quantum safe cryptography

Cartes d'identification — Cartes à circuit intégré —

Partie 8: Commandes et mécanismes pour les opérations de sécurité

AMENDEMENT 1: Interopérabilité pour l'échange d'opérations de sécurité utilisant la cryptographie quantique sécurisée

Reference number
ISO/IEC 7816-8:2021/Amd. 1:2023(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and security devices for personal identification*.

A list of all parts in the ISO/IEC 7816 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Identification cards — Integrated circuit cards —

Part 8:
Commands and mechanisms for security operations

AMENDMENT 1: Interoperability for the interchange of security operations using quantum safe cryptography

Normative references

Change ISO/IEC 7816-4 to ISO/IEC 7816-4:2020

Terms and definitions

Add the following new term entries:

3.8
certificate chain
ordered list of certificates that starts with an end-entity certificate, includes one or more certificate authority (CA) certificates, and ends with the end-entity certificate's root CA certificate, where each certificate in the chain is the certificate of the CA that issued the previous certificate

3.9
classic McEliece
code-based quantum safe key encapsulation algorithm retained in the course of the third round of the National Institute of Standards and Technology (NIST) contest

3.10
code-based
cryptosystem based on error correcting code

3.11
common parameter
public value that is used to control the operation of a cryptographic algorithm or that is used by a cryptographic algorithm to compute outputs

3.12
crypto-agility
property that permits changing or upgrading cryptographic algorithms or parameters

[SOURCE: ETSI TR 103 619[14]]

3.13
crystals-dilithium
lattice-based quantum safe signature algorithm as selected by the National Institute of Standards and Technology (NIST) contest for standardization

3.14
crystals-kyber
lattice-based quantum safe key encapsulation algorithm as selected by the National Institute of Standards and Technology (NIST) contest for standardization

3.15
discrete logarithm
computation of logarithms with regards to multiplicative cyclic groups

Note 1 to entry: This problem is considered as a hard mathematical problem when the number is large, and is the basis of some asymmetric cryptography scheme.

3.16
El Gammal
asymmetric encryption/decryption protocol based on the discrete logarithm problem and using a throwable mask

3.17
falcon
lattice-based quantum safe signature algorithm as selected by the National Institute of Standards and Technology (NIST) contest for standardization

3.18
frodoKEM
lattice-based quantum safe key encapsulation algorithm retained in the course of the third round of the National Institute of Standards and Technology (NIST) contest

3.19
hash-based
digital signatures constructed using hash functions

3.20
hybrid certificate
certificate secured using both a regular asymmetric signature algorithm and a quantum safe signature algorithm

3.21
key encapsulation algorithm
class of encryption techniques designed to secure symmetric cryptographic key material for transmission using asymmetric (public-key) algorithms

Note 1 to entry: The term used by National Institute of Standards and Technology (NIST) is “Key Encapsulation Mechanism”.

3.22
lattice-based
cryptosystem based on lattice problems

3.23
Leighton Micali signature
LMS
hash-based quantum safe signature algorithm

Note 1 to entry: This algorithm is defined in Reference [15].

3.24
Merkle tree
data structure where the data is hashed and combined until there is a singular root hash that represents the entire structure

3.25
NTRU
lattice-based quantum safe key encapsulation algorithm retained in the course of the third round of the National Institute of Standards and Technology (NIST) contest

3.26
public key parameters
set of components and characteristics describing the public key

3.27
private key parameters
set of components and characteristics describing the private key

3.28
quantum safe
characteristic of an algorithm or cryptosystem that is secure against both quantum and classical computers

3.29
quantum safe cryptography
cryptographic systems that are secure against both quantum and classical computers

Note 1 to entry: This encompasses any cryptosystem (e.g. symmetric, asymmetric).

3.30
Saber
lattice-based quantum safe key encapsulation algorithm retained in the course of the third round of the National Institute of Standards and Technology (NIST) contest

3.31
sphincs+
hash-based quantum safe signature algorithm as selected by the National Institute of Standards and Technology (NIST) contest for standardization

Note 1 to entry: See Table AMD.1.20.

3.32
XMSS
hash-based quantum safe signature algorithm

Note 1 to entry: The algorithm is defined in Reference [15].

Clause 4

Add the following before BCD:

AlgID Algorithm Identifier

Add the following after CRT:

CRT(2) optimized mode of computation for RSA algorithm exploiting the properties of the Chinese remainder theorem, and making use of a private key under a reduced form

Add the following after GQ2:

HSS hierarchical signature system

Add the following after LDS:

LMS Hash-based Leighton Micali Signature

LM-OTS Leighton Micali One-Time Signature

Add the following after OID:

PKI Public Key Infrastructure

Add the following after PSO:

QSC Quantum Safe Cryptography

Add the following after SEID:

SFM regular mode of computation of RSA algorithm

Add the following after TLV:

XMSS Extended Merkle (Hierarchical) Signature Scheme

Subclause 5.1

Add the following at the end of the first paragraph of subclause 5.1:

This document aims additionally at providing a set of generic QSC features. These generic features cater to a variety of QSC algorithms. This approach seeks to enhance this document with crypto-agility means and hybrid cryptography capabilities.

Subclause 5.2

Add the following subclause heading under subclause 5.2:

5.2.1 General

Subclause 5.2.1

Replace NOTE 2 with the following:

NOTE 2 The private key can be stored in

— an internal EF the reference of which is known before issuing the command, or

— a DO'7F48' as cardholder private key template, or

— a QSC template DO'7F75' featuring private key with private key parameters, or

— a QSC template DO'7F76' featuring private key with private key parameters alongside common parameters.

Replace NOTE 3 with the following:

NOTE 3 The public key can be stored for example in

— a DO'7F49' as cardholder public key template, or

— a QSC template DO'7F75' featuring exclusively public key parameters, or

— a QSC template DO'7F76' featuring public key parameters alongside common parameters, see examples on Table AMD.1.2 to Table AMD.1.7.

Subclause 5.2.1

In the bullet list under NOTE 4, add “or” to the first and second bullet. Replace “.” with “, or” at the end of the fourth bullet; append to the fourth bullet the following:

— a QSC template DO’7F75’ (INS = ‘47’) encapsulating public key and QSC key component data objects from Table AMD.1.1, or

— a QSC template DO’7F76’ (INS = ‘47’) encapsulating public key and QSC common parameter component data objects from Table AMD.1.1.

EXAMPLE 1 The QSC templates are provided in Table AMD.1.2 and Table AMD.1.5 for the hash-based signature algorithm.

EXAMPLE 2 The QSC templates are provided in Table AMD.1.3, Table AMD.1.4, Table AMD.1.6 and Table AMD.1.7 for the lattice-based algorithm.

Subclause 5.2.1

Change the EXAMPLE to EXAMPLE 3

Subclause 5.2.1

Change the sentence right before existing Table 4 as follows:

For the coding of the DO stating information about the private part of the key pair, Table 4 applies except for QSC. For QSC, the private key shall be nested into template DO’7F75’ or DO’7F76’ using the DOs as proposed in Table AMD.1.1.

Subclause 5.2

Add the following new subclauses 5.2.2 to 5.2.6 after 5.2.1:

5.2.2 QSC template data objects

This QSC template is a generic template catering to any type of QSC, e.g. hash-based or lattice-based, and its parameters, features general purposes data objects in its key information part whereas the key specific parameters, e.g. for private or public key or common parameters, are nested in template key value part. QSC template data objects are described in Table AMD.1.1.

Table AMD.1.1 — QSC key components within template DO‘7F75’, DO‘7F76’ or DO‘7F77’ with cardinality indication

Tag	Description	Cardinality	
		at creation	at update
’06’	**OID**, conditional (present at creation and if DO’80’ is absent) [c]	0 or 1	0
’80’	**AlgID**, conditional (present at creation and if first DO’06’ is absent)	0 or 1	0
’81’	**Key type**, conditional (present at creation, absent at update); unique per template	1 or 2	0
’82’	**Key size**, conditional (present at creation, absent at update) [b]	From 0 up to n	0
’8E’	**Identifier of external common parameters**, conditional (optional at creation, absent at update)	0 or 1	0
’8F’	**Identifier of the QSC template**, conditional (optional at creation and at update)	0 or 1	0 or 1
’5C’	**Parameters Organization Descriptor,** optional (may nest tag ’81’, tag ’82’ or tag ‘83’) [d] **‘81’** denotes private key parameters **‘82’** denotes public key parameters **‘83’** denotes common parameters	From 0 up to n	From 0 up to n
’06’	**Application-specific QSC parameters mapping**	0 or 1	0 or 1
’9X’	**Key value** (either private, public key, or common parameters), inferred by Key type (DO’81’), by template tag (DO’7F75’, DO’7F76’, DO’7F77’) and by Parameters Organization Descriptor (DO’5C’) [a]	0 or 1	0 or 1

NOTE 1 Use of DO’83’ to DO’8D’ is RFU, can serve for future indication of further characteristics, e.g. type of memory (transient, persistent);

NOTE 2 If common parameters are merged within private or public key template, the Key Type (DO’81’) can be replicated (e.g. hash-based protocol).

NOTE 3 DO’8F’ is the Private/Public Key identifier to be used by the external world to request a digital signature computation/verification or encryption/decryption from the ICC, e.g. the value of DO’8F’ can be assigned to DO’84’ to identify a private key from within a CRT (control reference template). DO’8F’ can be assigned to DO’83’ value in order to identify the public key from within a CRT.

NOTE 4 Additional private tags that can be used in template ‘7F75’ to nest application specific data, e.g. pre-generated private keys or proprietary parameter(s) involved in key generation or state maintenance, are outside interoperability scope.

NOTE 5 DO’8E’ is no longer useful if common parameters and private/public key are nested in the same template.

[a] Key value component (‘9X’) may be provisioned entirely at once or in several parts at creation and update.

[b] DO’82’ may be replicated for each level of the Merkle Tree with HSS hash-based scheme (see Figures F.6 and F.7).

[c] This DO’06’ describing the algorithm shall be the first DO in the template; only one OID describing the algorithm shall be present. Furthermore, OID describing the algorithm and DO AlgID are mutually exclusive. When occurring right after a Tag List DO’5C’, DO’06’ is meant for QSC parameter mapping and is not mutually exclusive with AlgID; this DO’06’ value is out of scope of this document.

[d] The usage of DO’81’, DO‘82’ and DO‘83’ are described under EXAMPLES 4 in the text below.

For hash-based LMS private key under template DO’7F75’, see Table AMD.1.8; for hash-based LMS merged private key and common parameters under template DO’7F76’, see Table AMD.1.9. For lattice-based Crystals-Dilithium private key under template DO’7F75’, see Table AMD.1.10; for lattice-based Crystals-Dilithium private key and common parameters merged under template DO’7F76’, see Table AMD.1.12. For lattice-based Falcon private key under template DO’7F75’, see Table AMD.1.11; for lattice-based Falcon private key and common parameters merged under template DO’7F76’, see Table AMD.1.13.

The templates DO‘7F77’ nesting only common parameters for hash-based LMS key, lattice-based Crystals-Dilithium, lattice-based Falcon are presented respectively on Table AMD.1.14, Table AMD.1.15 and Table AMD.1.16.

Each QSC template comprises key information data objects and key value data objects with Parameters Organization Descriptor data object. Key information data objects consist of OID, AlgID, Key type, Key size, identifier of external common parameters and identifier of the QSC template data object. Key value data objects are set of DO'9X' inferred by Key type (DO'81'), see Table AMD.1.22 for details. Five different structures can be nested under QSC templates DO'7F75', DO'7F76' or DO'7F77' as follows:

— private key under QSC template DO'7F75';

— public key under QSC template DO'7F75';

— common parameters under QSC template DO'7F77';

— common parameters and private key under QSC template DO'7F76';

— common parameters and public key under QSC template DO'7F76'.

Usually, common parameter(s) are implicitly known as soon as the cryptographic algorithm is known. Therefore template '7F75' may suffice instead of '7F76'.

The problem of scarcity of context-specific tags arises whenever more than 15 parameters are needed to describe a structure within a QSC template, i.e. with lattice-based Crystals-Dilithium. Therefore, to allow for nesting replicated and ordered primitive tags (bit b6 set to 0) while covering more than 15 parameters the optional Tag List data object DO'5C' (see ISO/IEC 7816-4:2020, 8.4.3) serving as Parameters Organization Descriptor shall be used right after the key information to indicate recursive tag numbering (i.e. cyclic tag numbering) whenever employed within the template. The use of DO'5C' may denote rearrangement of the QSC templates that are mapped explicitly in this document or in ISO/IEC 7816-6 (e.g. it can be useful when some mapped parameters are hidden in a QSC template or when common parameter and public or private key parameters are swapped).

{'5C'-L-(tag1- tag2-…- tagN)} denotes replicated tags in the template as follows: tag numbering starts with tag'90' till tag1; then tag numbering resumes with tag'90' right after tag1 and incremented by one till tag2; then tag numbering resumes with tag'90' right after tag2 and incremented by one till next tag in the Tag List DO'5C' etc. See example of implementation of DO'5C' below.

EXAMPLES 1

{'5C'-'01'-('95')} means tags numbering of a sequence starting from DO'90' is incremented by one and replicated once right after tag '95' resulting in a template as follows:

{'7F75'-L-{key information-{'5C'-01-('95')}-{'90'-L-V}-{'91'-L-V}-{'92'-L-V}-{'93'-L-V}-{'94'-L-V}-{'**95**'-L-V}-{'**90**'-L-V}-{'91'-L-V}-{'92'-L-V}…}}

{'5C'-'02'-('95''93')} means tags numbering of a sequence starting from DO'90' is incremented by one and replicated twice right after tag '95' then tag '93' resulting in a template as follows:

{'7F75'-L-{key information-{'5C'-'02'-('95', '93')}-{'90'-L-V}-{'91'-L-V}-{'92'-L-V}-{'93'-L-V}-{'94'-L-V}-{'**95**'-L-V}-{'**90**'-L-V}-{'91'-L-V}-{'92'-L-V}-{'**93**'-L-V}-{'**90**'-L-V}-{'91'-L-V}-{'92'-L-V}-{'93'-L-V}-{'94'-L-V…}}

DO'5C' may be used even when less than 15 parameters are present in the template to indicate the way such template shall be updated or expanded with further data objects when replication is deemed useful.

For tag numbering that restarts with a tag different from DO'90', data object '5C' shall use the following coding with the context-specific DO'9F2X' as indicator.

{'5C'-L-(tag1-'9F2X_1'-tag2-'9F2X_2'-…-tagN-'9F2X_n')} denotes replicated tags in the template as follows: tag numbering starts with tag'90' till tag1; then tag numbering resumes with tag'9X_1' right after tag1 and incremented by one till tag2; then tag numbering resumes with tag'9X_2' right after tag2 and incremented by one till next tag in the Tag List DO'5C' etc. See example of implementation of DO'5C' below.

EXAMPLES 2

{‘5C’-‘03’-(‘95’-‘9F22’)} means tags numbering of a sequence starting from DO‘92’ is incremented by one and replicated once right after tag ‘95’ resulting in a template as follows:

{‘7F75’-L-{key information-{‘5C’-03-(‘95’-‘9F22’)}-{‘90’-L-V}-{‘91’-L-V}-{‘92’-L-V}-{‘93’-L-V}-{‘94’-L-V}-{‘**95**’-L-V}-{‘**92**’-L-V}-{‘93’-L-V}-{‘94’-L-V}…}}

{‘5C’-‘07’-(‘95’-‘9F23’-‘94’-‘9F26’-‘97’)} means:

— for ‘95’-‘9F23’ : right after tag ‘95’, a new sequence is started with tag numbering from ‘93’ (instead of DO’90’), and incremented by one;

— for ‘94’-‘9F26’: right after tag ‘94’, a new sequence is started with tag numbering from ‘96’ (instead of DO’90’), and incremented by one;

— for ‘97’: right after tag ‘97’, a new sequence is started with tag numbering from ‘90’, and incremented by one.

{‘7F75’-L-{key information-{‘5C’-‘07’-(‘95’-‘9F23’-‘94’-‘9F26’-‘97’)}-{‘90’-L-V}-{‘91’-L-V}-{‘92’-L-V}-{‘93’-L-V}-{‘94’-L-V}-{‘**95**’-L-V}-{‘**93**’-L-V}-{‘**94**’-L-V}-{‘**96**’-L-V}-{‘**97**’-L-V}-{‘90’-L-V}-{‘91’-L-V}-}-{‘92’-L-V}-{‘93’-L-V}-{‘94’-L-V}-{‘95’-L-V}-{‘96’-L-V}-{‘97’-L-V}-{‘98’-L-V}-{‘99’-L-V}…}}

Where there is no tag before the DO’9F2X’ in the DO’5C’, i.e. {‘5C’-L-(‘9F2X’-…)}, it denotes that tag numbering starts with tag’9X’. See example of implementation of DO’5C’ below.

EXAMPLES 3

{‘5C’-‘02’-(‘9F23’)} means that following current DO’5C’, tag numbering starts with tag ‘93’, resulting in a template as follows:

{‘7F75’-L-{key information-{‘5C’-‘02’-(‘9F23’)}-{‘**93**’-L-V}-{‘94’-L-V}-{‘95’-L-V}…}}

In addition to its properties described above, DO’5C’ may serve to organize parameters into dedicated containers within QSC templates, whereby simplifying explicitly template parsing without ambiguity. To this aim, the three data objects DO’81’, DO’82’ and DO’83’ are nested in DO’5C’ to indicate respectively the presence of private key parameters, public key parameters and common parameters followed by a series of tags as described above, see examples below.

EXAMPLES 4

{‘5C’-‘02’-(‘81’-‘92’)} denotes that following DO’5C’, the current QSC template contains private key parameters numbered as a sequence starting from DO‘90’ and incremented by one, and replicated once right after tag ‘92’, resulting in a template as follows:

{‘7F75’-L-{key information-{‘5C’-‘02’(‘81’, ‘92’)}-{‘90’-L-V}-{‘91’-L-V}-{‘**92’**-L-V}-{‘**90’**-L-V}-{‘91’-L-V}-{‘92’-L-V}-{‘93’-L-V}-{‘94’-L-V}…}}

{‘5C’-‘01’-(‘83’)} denotes that following DO’5C’, the current QSC template contains, common parameters numbered as a sequence starting from DO‘90’, resulting in a template as follows:

{‘7F77’-L-{key information-{‘5C’-‘01’-(‘83’)}-{‘90’-L-V}-{‘91’-L-V}-{‘92’-L-V}-{‘90’-L-V}-{‘94’-L-V}…}}

The main benefit of such a tag organization with DO’5C’ is that it makes different arbitrary parameters numbering stay interoperable with each other within the same data object generation.

Table AMD.1.2 to Table AMD.1.16 describe key template structures featuring ‘5C’DO usage.

When using Tag List DO‘5C’ for parameters ordering into containers, the actual mapping of such parameters (i.e. assignment of a given DO’9X’ to a specific QSC component), may depend on the implementation, in which case, the Tag List may be immediately followed with a DO’06’ denoting the implementation authority (e.g. a specification) that determines the mapping. Otherwise, the by-default mapping is the one described in this document, for instance Crystals-Dilithium, Falcon, LMS. Crystals-Dilithium key sizes are given as examples in key templates; actual values may be checked against dilithium v3.1. See Reference [16].

Table AMD.1.2 — QSC template DO'7F75' encapsulating Hash-based LMS public key

Tag	L	Value			Note
0x7F75	Var.				QSC template encapsulating hash-based LMS public key
		Tag	**Length**	**Value**	
		0x80	0x04	0xFE0101XX	Algorithm identifier: Hash-based LMS and 'XX' indicating hash function (see 5.2.3)
		0x81	0x02	0x0012	Key type: ALG_TYPE_LMS_PUBLIC (see 5.2.4)
		0x82	0x02	0x0100	Key size (e.g. SHA-256)
		0x8E	Var.		Identifier of external common parameters
		0x8F	Var.		Identifier of the QSC template (same as private key)
		0x5C	0x01	0x82	Parameters Organization Descriptor see Table AMD.1.1
		0x90	Var.		Public key (root of tree)

NOTE 1 This table describes an example of LMS parameters for public key with leaves (level 2) and root (level 1) as per example from Reference [23] TestCase 2 (p.54). As a general description, in a hash-based hierarchical signature system (HSS) comprising a hierarchy of Merkle trees, L represents the number of stages of Merkle trees. The number of leaves of each Merkle tree valuates to 2^h where h is the height of the Merkle tree, and the leaves represent tuples of signature key (secret and public) on top of those trees. The parameter q is a 32-bit integer (called index) that indicates the leaf of the Merkle tree where the OTS public key can be looked up. Each leaf of the tree is comprised of the value of the public key of an LM-OTS public/private key pair. Each node within the tree has a unique node number, and the leaves have node numbers 2^h, (2^h)+1, (2^h)+2, ..., (2^h)+(2^h)-1. In general, the j-th node at level i has node number 2^i + j. Thus the root node has node number 1 (i.e. "root level = 2^0 = 1"), see Reference [23] for details.

NOTE 2 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.

See hash-based schemes taxonomy in Annex F, Figure F.1.

Table AMD.1.3 — QSC template DO'7F75' encapsulating Lattice-based Crystals-Dilithium public key

Tag	L	Value			Note
0x7F75	Var.				QSC template encapsulating Lattice-based Crystals-Dilithium public key
		Tag	**Length**	**Value**	
		0x80	0x04	see 5.2.3	AlgID
		0x81	0x02	see Table AMD.1.22	keytype "Crystals-Dilithium public key"
		0x82	0x02	0x04A0	keysize (1312 = rho + t1) values: 1312, 1952, 2592
		0x8E	0x01	0x02	Identifier of external common parameters
		0x8F	0x01	0x01	Identifier of the QSC template (same as private = keypair)
		0x5C	0x01	0x82	see Table AMD.1.1
		0x90	0x20	0x.....	*rho(length=32), same for all Crystals-Dilithium NIST levels*
		0x91	0x820500	0x.....	*t1(length=1280)* for NIST levels 2, 3 and 5, values: 1280, 1920, 2560

NOTE 1 See keysize according to NIST security levels in Reference [16].

NOTE 2 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.

Table AMD.1.4 — QSC template DO'7F75' encapsulating Lattice-based Falcon public key

Tag	L	Value			Note
0x7F75	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	0xFE0105XX	AlgID Lattice-based FALCON and 'XX' indicating hash function (see 5.2.3)
		0x81	0x02	see Table AMD.1.22	keytype "Falcon public key"
		0x82	0x02	0x0381	keysize (=897) values = 897,1793
		0x8E	0x01	0x02	Identifier of external common parameters
		0x8F	0x01	0x01	Identifier of the QSC template (same as private = keypair)
		0x5C	0x01	0x82	
		0x90	0x820381	0x……	*h(length=897)* value = 897,1793
NOTE 1 For recommended Falcon parameters see, e.g. public keysize on[17]. NOTE 2 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.					

Table AMD.1.5 — QSC template DO'7F76' encapsulating Hash-based LMS public key and common parameters

Tag	L	Value			Note
0x7F76	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	see 5.2.3	AlgID
		0x81	0x02	see Table AMD.1.22	keytype "LMS public key"
		0x82	0x01	0x20	keysize (e.g. SHA256)
		0x8F	0x01	0x01	Identifier of the QSC template (same as private = keypair)
		0x5C[a]	0x01	0x83	Parameters Organization Descriptor; indicates common parameters, see Table AMD.1.1
		0x90	0x10	0xD08F…	*LMS Identifier level1*
		0x91	0x04	0x00000006	*LMS type level 1 (e.g. LM_SHA256_M32_H10)*
		0x92	0x04	0x00000003	*LM-OTS type level 1 (e.g. LMOTS_SHA256_N32_W4)*
		0x5C	0x01	0x82	Parameters Organization Descriptor; indicates public key parameters, see Table AMD.1.1
		0x90	0x20	0x32A5…	*public key (root of the tree)*
		0x5C	0x01	0x83	Parameters Organization Descriptor; indicates common parameters, see Table AMD.1.1
		0x90	0x10	0x215F…	*LMS Identifier level2*
		0x91	0x04	0x00000005	*LMS type level 2*
		0x92	0x04	0x00000004	*LM-OTS type level 2*
NOTE 1 Parameter values borrowed from example in Reference [23] TestCase 2 (p.54) with leaves = level 2 and root = level 1, see Note 1 to Table AMD.1.2. NOTE 2 DO'8E' (identifier of external common parameters) is no longer useful if common parameters and public key are nested in the same template. NOTE 3 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers. [a] Instead of using multiple instances of Tag List DO'5C', one unique instance may be coded as '5C058392829083' for the same purpose.					

Table AMD.1.6 — QSC template DO'7F76' Lattice-based Crystals-Dilithium public key and common parameters

Tag	L	Value			Note
0x7F76	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	see 5.2.3	AlgID
		0x81	0x02	see Table AMD.1.22	keytype "Crystals-Dilithium Public key"
		0x82	0x02	0x0531	keysize (e.g. 1312+17 = rho + t1) keysize may take in account size public + size parameters = sum (size of all 0x9X data elements)
		0x8F	0x01	0x01	Identifier of the QSC template
		0x5C	0x01	0x83	Common parameters [a]
		0x90	0x03	0x7FE001	*q (=8380417)*
		0x91	0x01	0x0E	*d(=14)*
		0x92	0x01	0x3C	*weight of c (=60)*
		0x93	0x03	0x07FE00	*g1(=523776)*
		0x94	0x03	0x03FF00	*g2(=261888)*
		0x95	0x01	0x04	*k(=4)*
		0x96	0x01	0x03	*l(=3)*
		0x97	0x01	0x06	*h(=6)*
		0x98	0x02	0x0145	*b(=325)*
		0x99	0x01	0x50	*w(=80)*
		0x5C	0x01	0x82	Public key parameters
		0x90	0x20	0x.....	*rho(length=32)*
		0x91	0x820500	0x.....	*t1(length=1280)*

NOTE 1 See Reference [16].

NOTE 2 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.

[a] This template '7F76' is not necessary as the common parameter(s) listed from DO'90' to DO'99' are likely to be implicitly known as soon as the algorithm is known. Therefore template '7F75' may suffice.

Table AMD.1.7 — QSC template DO'7F76' encapsulating Lattice-based Falcon public key and common parameters

Tag	L	Value			Note
0x7F76	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	0xFE0105XX	AlgID Lattice-based FALCON and 'XX' indicating hash function (see 5.2.3)
		0x81	0x02	see Table AMD.1.22	keytype "Falcon public key"
		0x82	0x02	0x0381	keysize (=897)
		0x8F	0x01	0x01	Identifier of the QSC template
		0x5C	0x01	0x83	Common parameters [a]
		0x90	0x02	0x3001	*q (=8380417)*
		0x91	0x02	0x0200	*d(=14)*
		0x92	0x04	0x029845D6	*weight of c (=60)*
		0x5C	0x01	0x82	

NOTE 1 See Note 1 to Table AMD.1.4.

NOTE 2 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.

[a] This template '7F76' is not necessary as the common parameter(s) listed from DO'90' to DO'92' are likely to be implicitly known as soon as the algorithm is known. Therefore template '7F75' may suffice.

Table AMD.1.7 *(continued)*

Tag	L	Value			Note
		0x90	0x820381	0x.....	*h(length=897)*
NOTE 1 See Note 1 to Table AMD.1.4.					
NOTE 2 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.					
[a] This template '7F76' is not necessary as the common parameter(s) listed from DO'90' to DO'92' are likely to be implicitly known as soon as the algorithm is known. Therefore template '7F75' may suffice.					

Table AMD.1.8 — QSC template DO'7F75' encapsulating Hash-based LMS private key

Tag	L	Value			Note
0x7F75	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	see 5.2.3	*AlgID*
		0x81	0x02	see Table AMD.1.22	*keytype "LMS private key"*
		0x82	0x20		*keysize (e.g. SHA256)* (may be omitted)
		0x8E	0x01	0x02	*Identifier of external common parameters*
		0x8F	0x01	0x01	*Identifier of the QSC template (same as public = keypair)*
		0x5C	0x01	0x81	see Table AMD.1.1
		0x90	0x20	0x558B8...	*privatekey level 1* (see Reference [23], Appendix A)
		0x91	0x20	0xA1C4...	*privatekey level 2* (see Reference [23], Appendix A)
NOTE 1 With leaves = level 2 and root = level 1, see Note 1 to Table AMD.1.2.					
NOTE 2 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.					

For hash-based scheme example of key generation, see F.2.

Table AMD.1.9 — QSC template DO'7F76' encapsulating Hash-based LMS merged private key and common parameters

Tag	L	Value			Note
0x7F76	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	see 5.2.3	AlgID
		0x81	0x02	see Table AMD.1.22	keytype "LMS private key"
		0x82	0x01	0x20	keysize (e.g. SHA256) (may be omitted)
		0x8F	0x01	0x01	Identifier of the QSC template
		0x5C	0x01	0x83	see Table AMD.1.1
		0x90	0x10	0xD08F...	*LMS Identifier level1*
		0x91	0x04	0x00000006	*LM type level 1 (e.g. LM_SHA256_M32_H10)*
		0x92	0x04	0x00000003	*LM-OTS type level 1 (e.g. LMOTS_SHA256_N32_W4)*
		0x5C	0x01	0x81	see Table AMD.1.1
		0x90	0x20	0x558B8...	*privatekey level 1* (see Reference [23], Appendix A)
		0x5C	0x01	0x83	see Table AMD.1.1
		0x90	0x10	0x215F...	*LMS Identifier level2*
		0x91	0x04	0x00000005	*LMS type level 2*
		0x92	0x04	0x00000004	*LM-OTS type level 2*
		0x5C	0x01	0x81	see Table AMD.1.1
		0x90	0x20	0xA1C4...	*privatekey level 2* (see Reference [23], Appendix A)
NOTE 1 See Note 1 to Table AMD.1.2.					
NOTE 2 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.					

Table AMD.1.10 — QSC template DO'7F75' encapsulating Lattice-based Crystals-Dilithium private key

Tag	L	Value			Note
0x7F75	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	see 5.2.3	AlgID
		0x81	0x02	see Table AMD.1.22	keytype "Crystals-Dilithium private key"
		0x82	0x02	0x09BC	keysize (=2492)
		0x8E	0x01	0x02	Identifier of external common parameters
		0x8F	0x01	0x01	Identifier of the QSC template (same as public = keypair)
		0x5C	0x01	0x81	see Table AMD.1.1
		0x90	0x20	0x.....	*rho(length=32) same for all NIST levels*
		0x91	0x20	0x.....	*K(length=32) same for all NIST levels*
		0x92	0x20	0x.....	*tr(length=32) same for all NIST levels*
		0x93	0x820180	0x.....	*s1(length=3∗128=384)*
		0x94	0x820180	0x.....	*s2(length=3∗128=384)*
		0x95	0x820680	0x.....	*t0(length=1664)* values = 1344,1792,2240,2688
NOTE 1 See Reference [16].					
NOTE 2 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.					

Table AMD.1.11 — QSC template DO'7F75' encapsulating Lattice-based Falcon private key

Tag	L	Value			Note
0x7F75	Var.	**Tag**	**Length**	**Value**	
		0x81	0x02	see Table AMD.1.22	keytype "Falcon private key"
		0x82	0x02	0x1001	keysize (=4097) values = 4097,8193
		0x8E	0x01	0x02	Identifier of external common parameters
		0x8F	0x01	0x01	Identifier of the QSC template (same as public = keypair)
		0x5C	0x01	0x81	see Table AMD.1.1
		0x90	0x820200	0x.....	*f(length=512) values = 512,1024*
		0x91	0x820200	0x.....	*g(length=512) values = 512,1024*
		0x92	0x820200	0x.....	*F(length=512) values = 512,1024*
		0x93	0x820200	0x.....	*G(length=512) values = 512,1024*
NOTE 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.					

Table AMD.1.12 — QSC template DO'7F76' encapsulating Lattice-based Crystals-Dilithium private key and common parameters

Tag	L	Value			Note
0x7F76	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	see 5.2.3	AlgID optional
		0x81	0x02	see Table AMD.1.22	keytype "Crystals-Dilithium private key"
		0x82	0x02	0x09CD	keysize (2492+17)
		0x8F	0x01	0x01	Identifier of the QSC template
		0x5C	0x01	0x83	see Table AMD.1.1 [a]
NOTE 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.					
[a] This template '7F76' is not necessary as the common parameter(s) listed from DO'90' to DO'99' are likely to be implicitly known as soon as the algorithm is known. Therefore template '7F75' may suffice.					

Table AMD.1.12 *(continued)*

Tag	L	Value			Note
		0x90	0x03	0x7FE001	*q (=8380417)*
		0x91	0x01	0x0E	*d(=14)*
		0x92	0x01	0x3C	*weight of c (=60)*
		0x93	0x03	0x07FE00	*g1(=523776)*
		0x94	0x03	0x03FF00	*g2(=261888)*
		0x95	0x01	0x04	*k(=4)*
		0x96	0x01	0x03	*l(=3)*
		0x97	0x01	0x06	*h(=6)*
		0x98	0x02	0x0145	*b(=325)*
		0x99	0x01	0x50	*w(=80)*
		0x5C	0x01	0x81	see Table AMD.1.1
		0x90	0x20	0x.....	*rho(length=32)*
		0x91	0x20	0x.....	*K(length=32)*
		0x92	0x20	0x.....	*tr(lengh=32)*
		0x93	0x820180	0x.....	*s1(length=3∗128=384)*
		0x94	0x820180	0x.....	*s2(length=3∗128=384)*
		0x95	0x820680	0x.....	*t0(length=1664)*

NOTE 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.

[a] This template '7F76' is not necessary as the common parameter(s) listed from DO'90' to DO'99' are likely to be implicitly known as soon as the algorithm is known. Therefore template '7F75' may suffice.

Table AMD.1.13 — QSC template DO'7F76' encapsulating Lattice-based Falcon private key and common parameters

Tag	L	Value			Note
0x7F76	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	0xFE0105XX	AlgID Lattice-based FALCON and 'XX' indicating hash function (see 5.2.3)
		0x81	0x02	see Table AMD.1.22	keytype "Falcon private key"
		0x82	0x02	0x1001	keysize (=4097)
		0x8F	0x01	0x01	Identifier of the QSC template
		0x5C	0x01	0x83	see Table AMD.1.1 [a]
		0x90	0x02	0x3001	*q (=12289)*
		0x91	0x02	0x0200	*n(=512)*
		0x92	0x04	0x029845D6	*bound b(=43533782)*
		0x5C	0x01	0x81	see Table AMD.1.1
		0x90	0x820200	0x.....	*f(length=512)*
		0x91	0x820200	0x.....	*g(length=512)*
		0x92	0x820200	0x.....	*F(length=512)*
		0x93	0x820200	0x.....	*G(length=512)*

NOTE 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.

[a] This template '7F76' is not necessary as the common parameter(s) listed from DO'90' to DO'92' are likely to be implicitly known as soon as the algorithm is known. Therefore template '7F75' may suffice.

Table AMD.1.14 — QSC template DO'7F77' encapsulating Hash-based LMS common parameters

Tag	L	Value			Note
0x7F77	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	see 5.2.3	*AlgID*
		0x81	0x02	See Table AMD.1.22	*keytype "LMS parameters"*
		0x82	0x01	0x20	*keysize (e.g. SHA256)* may be omitted
		0x8F	0x01	0x02	*Identifier of the QSC template*
		0x5C	0x01	0x83	see Table AMD.1.1
		0x90	0x10	0xD08F…	*LMS Identifier level1*
		0x91	0x04	0x00000006	*LM type level 1 (e.g. LM_SHA256_M32_H10)*
		0x92	0x04	0x00000003	*LM-OTS type level 1 (e.g. LMOTS_SHA256_N32_W4)*
		0x5C	0x01	0x83	see Table AMD.1.1
		0x90	0x10	0x215F…	*LMS Identifier level2*
		0x91	0x04	0x00000005	*LMS type level 2*
		0x92	0x04	0x00000004	*LM-OTS type level 2*
NOTE 1 See Note 1 to Table AMD.1.2.					
NOTE 2 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.					

Table AMD.1.15 — QSC template DO'7F77' encapsulating Lattice-based Crystals-Dilithium common parameters

Tag	L	Value			Note
0x7F77	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	0xFE0105XX	AlgID Lattice-based FALCON and 'XX' indicating hash function (see 5.2.3)
		0x81	0x02	see Table AMD.1.22	keytype "Crystals-Dilithium parameters"
		0x82	Var.		keysize
		0x8F	0x01	0x02	Identifier of the QSC template
		0x5C	0x01	0x83	see Table AMD.1.1
		0x90	0x03	0x7FE001	*q (=8380417) same for all NIST security levels*
		0x91	0x01	0x0E	*d(=14) same for all NIST security levels*
		0x92	0x01	0x3C	*weight of c (=60) same for all NIST security levels*
		0x93	0x03	0x07FE00	*g1(=523776) same for all NIST security levels*
		0x94	0x03	0x03FF00	*g2(=261888) same for all NIST security levels*
		0x95	0x01	0x04	*k(=4) values = 3,4,5,6*
		0x96	0x01	0x03	*l(=3) values = 2,3,4,5*
		0x97	0x01	0x06	*h(=6) values = 7,6,5,3*
		0x98	0x02	0x0145	*b(=325) values = 375,325,275,175*
		0x99	0x01	0x50	*w(=80) values = 64,80,96,120*
NOTE 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.					

Table AMD.1.16 — QSC template DO'7F77' encapsulating Lattice-based Falcon common parameters

Tag	L	Value			Note
0x7F77	Var.	**Tag**	**Length**	**Value**	
		0x80	0x04	0xFE0105XX	AlgID Lattice-based FALCON and 'XX' indicating hash function (see 5.2.3)
		0x81	0x02	see Table AMD.1.22	keytype "Falcon parameters"
		0x82	Var.		keysize
		0x8F	0x01	0x02	Identifier of the QSC template
		0x5C	0x01	0x83	see Table AMD.1.1
		0x90	0x02	0x3001	*q (=12289)*
		0x91	0x02	0x0200	*n(=512) values = 512,1024*
		0x92	0x04	0x029845D6	*bound b(=43533782)* values = 43533782, 87067565
NOTE 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.					

5.2.3 QSC algorithm identifier

DO'80' under DO'7F75', DO'7F76' or DO'7F77' is a QSC algorithm identifier DO with variable length value field; it encodes as follows:

— the first byte codes the category of algorithm identifier that is either legacy Algorithm identifier, or QSC Algorithm identifier as described from Table AMD.1.18 to Table AMD.1.21. This first byte ensures backward compatibility and extensibility, i.e. this byte allows for legacy support of existing templates '7F49' according to ISO/IEC 7816-8:2021, GlobalPlatform, EN 419 212[19], CEN TS 15480[20], BSI TR 03110[21], IAS ECC[22];

— the second byte codes the function, see Table AMD.1.18;

— the third byte codes the algorithm of "Compute Digital Signature", "Verify Digital Signature/Certificate" or "Encipher/Decipher" if the corresponding bit is set in the second byte (see Table AMD.1.19 and Table AMD.1.20); the fourth byte codes the algorithm of the hash used for "Compute Digital Signature" and "Verify Digital Signature/Certificate" if the corresponding bit is set in the second byte. It indicates the hashing function used to compute the digital signature input and its encoding is different for each value set for byte 2, see Table AMD.1.21;

Usually, common parameter(s) are implicitly known as soon as the cryptographic algorithm is known. Therefore template '7F75' may suffice instead of '7F76'.

When used within the QSC templates DO'7F75', DO'7F76' or DO'7F77', different DO'80' codings are distinguishable.

Table AMD.1.17 — DO'80' first byte value coding

b8	b7	b6	b5	b4	b3	b2	b1	Value
X	X	X	X	X	X	X	X	if present, value coded in the three subsequent bytes
1	1	1	1	1	1	1	1	Legacy AlgID (valuate to 'FF') as per EN 419212
1	1	1	1	1	1	1	0	AlgID (valuate to 'FE') according to this table
NOTE Any other value is RFU.								

Table AMD.1.18 — DO'80' second byte value coding

b8	b7	b6	b5	b4	b3	b2	b1	Value
-	-	-	-	-	-	X	X	Operation
-	-	-	-	-	-	0	0	No information given
-	-	-	-	-	-	0	1	Compute Digital Signature / Verify Digital Signature / Verify Digital Certificate
-	-	-	-	-	-	1	0	Encipher/decipher

NOTE Any other value is RFU.

Table AMD.1.19 — DO'80' third byte value coding for compute digital signature or verify digital signature or verify digital certificate algorithm

b8	b7	b6	b5	b4	b3	b2	b1	Value [a]
1	X	X	X	X	X	X	X	Proprietary encoding of algorithm reference
0	0	0	0	0	0	0	0	No information given
0	0	0	0	0	0	0	1	Hash-based LMS
0	0	0	0	0	0	1	0	Hash-based XMSS
0	0	0	0	0	0	1	1	Hash-based SPHINCS+
0	0	0	0	0	1	0	0	Lattice-based Crystals-Dilithium
0	0	0	0	0	1	0	1	Lattice-based Falcon

NOTE Any other value is RFU.

[a] The names of algorithms specified in this table may be deviated from the final NIST specification.

Table AMD.1.20 — DO'80' third byte value coding for encipher/decipher algorithm

b8	b7	b6	b5	b4	b3	b2	b1	Value [b]
1	X	X	X	X	X	X	X	Proprietary encoding of algorithm reference
0	0	0	0	0	0	0	0	No information given
0	0	0	0	0	0	0	1	Crystals-Kyber
0	0	0	0	0	0	1	0	NTRU [a]
0	0	0	0	0	0	1	1	Saber [a]
0	0	0	0	0	1	0	0	FrodoKEM [a]
0	0	0	0	0	1	0	1	Classic McEliece [a]

NOTE Any other value is RFU.

[a] The algorithm identifier provided in this section denotes the version of Classic McEliece, FrodoKEM, Ntru and Saber submitted in the course of the third round of NIST contest. If another version of these algorithms is employed, OID (DO'06') from Table AMD.1.1 shall be used instead of algorithm identifier (DO'80').

[b] The names of algorithms specified in this table may be deviated from the final NIST specification.

Table AMD.1.21 — DO'80' fourth byte value coding for hash of compute/verify digital signature or verify certificate

b8	b7	b6	b5	b4	b3	b2	b1	Value
1	X	X	X	X	X	X	X	Proprietary encoding of algorithm reference
0	0	0	0	0	0	0	0	No information given
0	0	0	0	0	0	0	1	SHA-224
0	0	0	0	0	0	1	0	SHA-256

NOTE Any other value is RFU.

[a] 256-bit size is used.

Table AMD.1.21 *(continued)*

b8	b7	b6	b5	b4	b3	b2	b1	Value
0	0	0	0	0	0	1	1	SHA-384
0	0	0	0	0	1	0	0	SHA-512
0	0	0	0	0	1	0	1	SHA3-224
0	0	0	0	0	1	1	0	SHA3-256
0	0	0	0	0	1	1	1	SHA3-384
0	0	0	0	1	0	0	0	SHA3-512
0	0	0	0	1	0	0	1	SHAKE256[a]
NOTE Any other value is RFU.								
[a] 256-bit size is used.								

5.2.4 QSC key type data object

A Key Type DO'81' is a two-byte length value field within template DO'7F75', DO'7F76' or DO'7F77' that informs about the QSC algorithm and private and public key structures (i.e. parameters and their organization in the template).

NOTE Key type proves useful, e.g. when reserving memory space for the keys that will be provisioned later. DO'80' informs about the crypto suite, whereas key type DO'81' infers to key structure. There can be multiple key types for the same AlgID, e.g. RSA private key may have two representations: CRT and SFM usable with the same algorithm.

Table AMD.1.22 describes Key type values.

Table AMD.1.22 — DO'81' value coding

Most significant byte	Least significant byte	Value
0x00	0x01..0xFF	legacy key type (existing key types)
0x01	0x01..0xFF	ISO key type
0x02..0xEF	0x01..0xFF	RFU
0xF0..0xFF	0x01..0xFF	proprietary
NOTE 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.		

For the sake of interoperability, extensibility and backward compatibility, the assignment of value to QSC algorithms (according to coding from Table AMD.1.22) may be started by incrementing the values of existing key type which may result in the following examples:

— ALG_TYPE_CRYSTALS_DILITHIUM_PUBLIC 0x0116 (dec. 22)

— ALG_TYPE_CRYSTALS_DILITHIUM_PRIVATE 0x0117 (dec. 23)

— ALG_TYPE_CRYSTALS_DILITHIUM_PARAMETERS 0x0118 (dec. 24)

— ALG_TYPE_FALCON_PUBLIC 0x0119 (dec. 25)

— ALG_TYPE_FALCON_PRIVATE 0x011A (dec. 26)

— ALG_TYPE_FALCON_PARAMETERS 0x011B (dec. 27)

— ALG_TYPE_LMS_PUBLIC 0x011C (dec. 28)

— ALG_TYPE_LMS_PRIVATE 0x011D (dec. 29)

— ALG_TYPE_LMS_PARAMETERS 0x011E (dec. 30)

— ALG_TYPE_XMSS_PUBLIC 0x011F (dec. 31)

— ALG_TYPE_XMSS_PRIVATE 0x0120 (dec. 32)

— ALG_TYPE_XMSS_PARAMETERS 0x0121 (dec. 33)

NOTE 1 The key type allocation example defined above starts at decimal value 22 to avoid collision with legacy values belonging to https://docs.oracle.com/en/java/javacard/3.1/jc_api_srvc/api_classic/constant-values.html

NOTE 2 0xABCD denotes the hexadecimal number 'ABCD' for easier reference from implementers.

5.2.5 QSC key size data object

DO'82' is of variable length to host Key Size in bits under template DO'7F75', DO'7F76' or DO'7F77'. DO'82' shall be present when the cryptographic key is present in the template. DO'82' denotes the size of entire key components.

5.2.6 QSC identifier of external common parameters data object

DO'8E' under QSC (private/public key) template 'DO'7F75' is used for identifying to the corresponding common parameters template DO'7F77' as described in Figure AMD.1.1. The value field of the identifier of external common parameters DO'8E' under a QSC template DO'7F75' is equal to the value of the common parameters identifier DO'8F' under its corresponding common parameters template DO'7F77'.

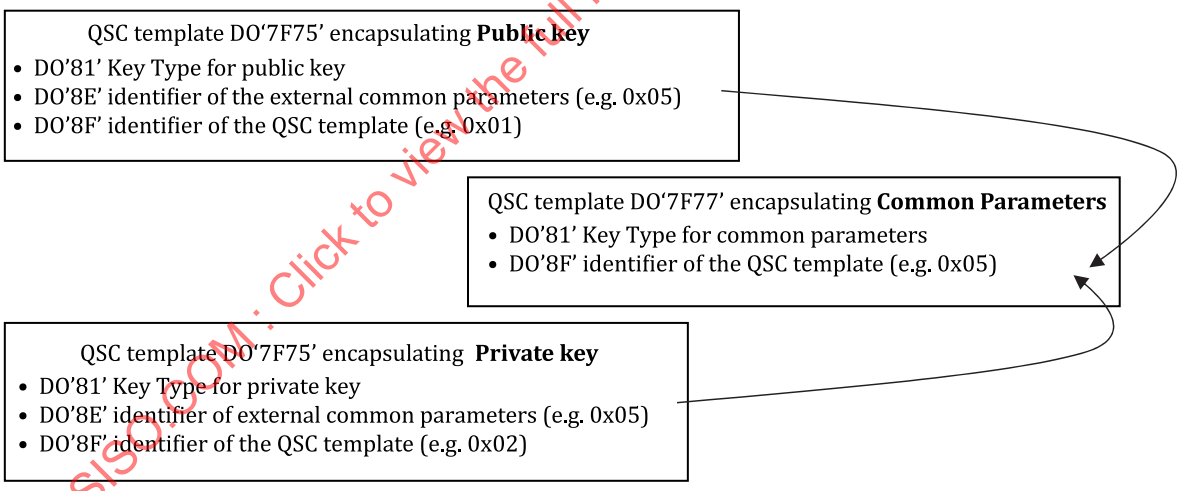


Figure AMD.1.1 — Identifier of external common parameters DO'8E' in QSC template

A QSC (private/public key) template DO'7F75' and its corresponding common parameters template DO'7F77' may be in the same DF.

Subclause 5.3.1, Table 8

Change Data field description as follows:

Either input DO(s) (see Table 7) or DO'73', or both, and optionally an extended header list describing the output (primitive or constructed) (see Table 6).

Subclause 5.3.4, Table 12

Update existing Table 12 as follows:

Change Response data field as follows:

Absent (hash-code stored in the card), or hash-code, or data objects for intermediate hash-code (e.g. 'A0' - L -{{'90'- L - <intermediate hash-code followed by a bit counter>} {'80' – L - <data to hash (not limited to one block)>}})[c]

Add the following footnote in Table 12:

[c] P1-P2 = '90 80' or '90 A0' with QSC shall return data objects for intermediate hash-code when current SE has HT (CRT) nesting DO'9F01' (see paragraph below Table 12).

Add footnote [c] to the Command data field row in Table 12.

Add the following after Table 12:

For QSC, prior to PSO HASH command (INS = '2A'), a MSE SET command with CRT (HT) nesting DO'9F01' denotes that the ICC shall perform an intermediate hash-code computation, and shall return the output in a structured response data field, i.e. PSO HASH with {'A0' – L –{{'90'- L - <intermediate hash-code followed by a bit counter>} {'80' - L- <data to hash (not limited to one block)> }}}.

DO'9F01' shall feature a value '00'; any other value is RFU.

Alternatively, an MSE RESTORE establishing a security environment nesting a CRT (HT) that contains a DO'9F01' has the same effect.

If ICC does not support QSC intermediate hash-code computation with involvement of IFD for hash completion, status bytes SW1-SW2 ='62 88' may be returned on MSE SET command.

Subclause 5.3.7

Change the sentence before the list as follows:

The following three cases shall be distinguished:

Add a third bullet to the bullet list in the 4th paragraph as follows:

— for QSC certificate verification see 5.3.12.

Add the following subclauses 5.3.10 to 5.3.14 after subclause 5.3.9

5.3.10 "QSC COMPUTE DIGITAL SIGNATURE operation"

For QSC signature schemes, the following options are considered for the hash computation of the message to be signed regardless of the QSC algorithm:

— mode 1: partly on-card (parameters) then remaining off-card (message) hash-code processing then signature;

— mode 2: partly on-card (parameters) then partial off-card (message) hash-code then partial on-card (message) then signature;

— mode 3: entirely off-card with replacement of the message by its hash that is pre-computed outside ICC, then resulting pre-hash submitted to signature by ICC;

— mode 4: entirely off-card with digital signature input hashed outside ICC after parameters delivery by ICC, then signature by ICC;

— mode 5: partly off-card, then remaining on-card, then signature;

— mode 6: entire on-card hash then signature.

Descriptions are provided in the annexes with examples as follows:

— with hash-based scheme:

  — for mode 1, see workflow in F.4, Figure F.9;

  — for mode 2, see workflow in F.4, Figure F.10;

  — for mode 3, see workflow in F.4, Figure F.11;

  — for mode 4, see workflow in F.4, Figure F.12;

— with Crystals-Dilithium (lattice-based scheme):

  — entire off-card hash then signature (mode 4), see workflow in F.5, Figure F.13;

  — partly hash off-card then partly hash on-card then signature (mode 5), see workflow in F.5, Figure F.14;

  — entire on-card hash then signature (mode 6), see workflow in F.5, Figure F.15.

For hash-based scheme description, see Winternitz-OTS signature details in Figure F.4 and for signature verification, see Figure F.5 in F.4.

Table AMD.1.23 shows cryptographic components for QSC signature and encryption. This table only lists information pertaining to algorithms that are selected candidates following the third round of the NIST contest. LMS and XMSS do not belong to this category. Accordingly, the signature result in response APDU is a multi-part data out that may be conveyed either as a byte string or as a structure. Fixed length components allow for byte string without separator, but for structured format, encapsulation in a data object is necessary.

As PSO command-response data field with INS='2A' cannot convey or return structured format i.e. set or sequence of data objects, the PSO command-response pair with INS='2B' shall be used for QSC with all PSO operations except for PSO HASH and VERIFY CERTIFICATE operations. Annex F describes examples relying on PSO operations with INS='2A' only to highlight the limitation that may occur from using PSO with INS='2A' with respect to command-response data field content.

Table AMD.1.23 — Algorithm proposals out of NIST 3rd contest round — Cryptographic material

Algorithm	Signature/ Decryption	Public key	Private key	Signature/ Encrypted data
CRYSTALS-DILITHIUM	S	(rho, t1) fixed size	(rho, K, tr, s1, s2, t0) fixed size	(z, h, c) fixed size
Falcon	S	h fixed size	(f,g,F) fixed size	(r,s) (r) fixed size (s) variable size (maximum size can be deduced)

Table AMD.1.23 *(continued)*

Algorithm	Signature/ Decryption	Public key	Private key	Signature/ Encrypted data
CRYSTALS-KYBER	D	(t,rho) fixed size	(s,t,rho,H(t||rho),z) fixed size	(c1,c2,k) fixed size
NTRU	D	h fixed size	(f,fp,hq,s) fixed size	(c,k) fixed size
Saber	D	(seed, b) fixed size	(z, H(seed||b), seed, b, s) fixed size	(c, b', h) fixed size
Classic McEliece	D	T fixed size	(gamma, s) fixed size	(c0,c1, k) fixed size

For QSC signature computation, PSO Compute digital signature (INS = '2B', P1 = '02', P2 = '00') shall feature:

— command data field with data to be signed within DO'80' (for plain value not encoded in BER-TLV);

— response data field with signature nested within DO'73' containing a DO'80' denoting the signature format either with

  — value '00' for byte string, followed with DO'81' nesting the byte string value, or

  — value '01' for structured (BER-TLV), in which case the structured components may be ordered as per NIST reference publication when available (e.g. in accordance with Reference [15] for LMS or XMSS), and followed with data object(s) with sequenced tags starting from DO'81' and incrementing tag number to represent all the signature components.

Examples of usage: {'73'-L-({'80'-'01'-'00'}-{'81'-L-(byte string)})}, {'73'-L-({'80'-'01'-'01'}-{'81'-L-(Value)}-{'82'-L-(Value)}-{'83'-L-(Value)})}

See further examples of DO'73' usage provided in Table AMD.1.B.1, Table AMD.1.B.2, Table AMD.1.B.3 Table AMD.1.B.4, Table AMD.1.B.5, or Table AMD.1.B.6.

For information only, Annex F describes QSC signature using PSO with INS = '2A'.

NOTE The same approach can apply to regular signature featuring several components, e.g. ECDSA with two components.

For examples of signature cryptographic components, see Table AMD.1.23.

5.3.11 "QSC VERIFY DIGITAL SIGNATURE operation"

For QSC signature verification, PSO Verify digital signature (INS = '2B', P1 = '05', P2 = '00') shall feature:

— command data field with a message nested within DO'80' (for plain value not encoded in BER-TLV) as per Table 7, alongside the signature to be verified that is nested within a DO'73' containing a DO'80' denoting the signature format either with

  — value '00' for byte string, followed with DO'81' nesting the byte string value, or

  — value '01' for structured (BER-TLV), in which case the structured components may be ordered as per NIST reference publication when available (e.g. in accordance with Reference [15] for LMS or XMSS), and followed with data object(s) with sequenced tags starting from DO'81' and incrementing tag number to represent all the signature components.

See examples of DO'73' usage provided in Table AMD.1.B.1, Table AMD.1.B.2, Table AMD.1.B.3, Table AMD.1.B.4, Table AMD.1.B.5, or Table AMD.1.B.6.

NOTE The same approach can apply to regular signature featuring several components, e.g. ECDSA with two components.

For examples of signature cryptographic components, see Table AMD.1.23.

5.3.12 "QSC usage of VERIFY CERTIFICATE operation"

The IFD shall indicate in the course of key selection if a hybrid certificate (see B.6.3) will be sent, so that the ICC may inform them in case it does not support it.

To denote that a hybrid certificate will be sent to ICC, i.e. by subsequent PSO Verify Certificate, an MSE Set command shall use the context-specific tag '96' under CRTs (DST) as follows:

— DO'96' is one-byte length and valuates to '01' to denote hybrid certificate variant 1 (e.g. with Crystals-Dilithium or Falcon or hash-based signature alongside regular signature), see B.6.3;

— DO'96' is one-byte length and valuates to '02' to denote hybrid certificate variant 2 (e.g. with Crystals-Dilithium or Falcon or hash-based signature alongside regular signature), see B.6.3;

— DO'96' is one-byte length and valuates to '00' to denote non-hybrid certificate, e.g. (regular certificate with RSA or ECC based signature).

Alternatively, an MSE RESTORE establishing a security environment nesting a CRT (DST) that contains a DO'96' has the same effect.

DO'96' fits as well under certificate template DO'7F21' to indicate the current certificate is a hybrid one. In case ICC does not support hybrid certificate it may return to the MSE Set command prior to PSO – VERIFY CERTIFICATE command status bytes with SW1-SW2 set to '62 88' and meant for 'operation or data format not supported'.

For examples of signature cryptographic components, see Table AMD.1.23.

For verification of QSC hybrid certificate and full QSC certificate, only self-descriptive certificates shall be verified by ICC, i.e. PSO VERIFY CERTIFICATE with INS= '2A' and P1P2='00BE'. See hybrid certificate and full-QSC certificate examples in B.6.

5.3.13 "QSC ENCIPHER operation"

For QSC Encryption, PSO Encipher (INS = '2B', P1 = '07', P2 = '00') shall feature:

— command data field with data to be encrypted nested within DO'80' (for plain value not encoded in BER-TLV);

— response data field with encrypted data nested within DO'73' containing a DO'80' denoting the encryption format either with

  — value '00' for byte string, followed with DO'81' nesting the byte string value, or

  — value '01' for structured (BER-TLV), in which case the structured components may be ordered as per NIST reference publication when available (e.g. in accordance with Reference [15] for LMS or XMSS), and followed with data object(s) with sequenced tags starting from DO'81' and incrementing tag number to represent all the encryption components.

See examples of DO'73' usage provided in Table AMD.1.B.1, Table AMD.1.B.2, Table AMD.1.B.3, Table AMD.1.B.4, Table AMD.1.B.5, or Table AMD.1.B.6.

NOTE The same approach can apply to regular encryption featuring several components, e.g. El Gammal with two components.

For examples of encryption cryptographic components, see Table AMD.1.23.

5.3.14 "QSC DECIPHER operation"

For QSC Decryption, PSO Decipher (INS = '2B', P1 = '08', P2 = '00') shall feature:

— command data field with data to be decrypted nested within DO'73' containing a DO'80' denoting the encryption format either with

  — value '00' for byte string, followed with DO'81' nesting the byte string value, or

  — value '01' for structured (BER-TLV), in which case the structured components may be ordered as per NIST reference publication when available (e.g. in accordance with Reference [15] for LMS or XMSS), and followed with data object(s) with sequenced tags starting from DO'81' and incrementing tag number to represent all the encryption components.

— response data field with deciphered data nested within DO'80' (for plain value not encoded in BER-TLV).

See examples of DO'73' usage provided in Table AMD.1.B.1, Table AMD.1.B.2, Table AMD.1.B.3 Table AMD.1.B.4, Table AMD.1.B.5, or Table AMD.1.B.6.

NOTE The same approach can apply to regular decryption featuring several components, e.g. El Gammal with two components.

For examples of encryption cryptographic components, see Table AMD.1.23.

Annex A

Add the following new subclause A.8 after A.7:

A.8 Sequence of commands for hybrid self-descriptive card verifiable certificate verification

As any card verifiable certificate, verification of hybrid card verifiable certificate requires two steps: selection of the verification key and verification of the certificate together with the import of the public key. However, because of the very nature of hybrid certificate, first, the verification can require two keys: one regular asymmetric public key, and another one which is a quantum safe asymmetric public key, and secondly, upon successful verification, not only one public key, but two public keys (regular and quantum safe) may be imported in the ICC.

Generally, the top key(s) of a certificate chain available in the ICC is (are) the public key(s) of the root CA. The two steps shall be performed repeatedly until the correct terminal public key(s) is (are) available in the ICC. The ICC verifies this certificate with the public key(s) set in the previous step. If the verification fails, the ICC responds with appropriate status bytes. When successful, the ICC stores the public key(s) contained in the certificate and allows a further import of certificates along the certificate chain. This process is called multi-stage verification.

The role of the certificate holder should be in conformity with its signer. The selection of root CA public key(s) can always be accepted to verify a chained certificate. Any other key(s) that was imported with a certificate may be applied only if the role of its certificate matches the purpose of verification of the next incoming certificate.

The MSE command selects the key(s) to use for verification of the certificate to be received in the consecutive PSO command as shown in Table AMD.1.A.19.

Table AMD.1.A.19 — Example for hybrid certificate verification

Command	Operation	P1-P2	Command data field	Response data field
mse	set	'81 P2'	{'B6' – L – <Digital signature template> {'83' – L – <Issuer identification number#1>} {'96' '01' '01'} } {'B6' – L – <Digital signature template> {'83' – L – <Issuer identification number#2>} } Issuer identification number#1 and Issuer identification number#2 identify the public key(s) to use for the verification of the hybrid certificate. These identifiers shall match the CAR(s) present in the hybrid certificate (if present) The DO'96' indicates to the ICC that a hybrid certificate variant 1 will be sent (see B.6.3).	Absent
pso	Verify certificate	'00BE'	{'7F21' – L – <cardholder certificate>}	Absent

For verification of hybrid certificate with the command PSO – VERIFY CERTIFICATE, it is recommended that the whole certificate (including the template '7F21') is sent in the command data field.

Annex B, subclause B.2

Change the first paragraph in B.2 as follows:

Table B.1 shows data objects relevant for card-verifiable certificates. These data objects are either borrowed from ISO/IEC 7816-6, which supersedes their definition in Table B.1, or newly defined for QSC purposes.

Annex B, subclause B.2

Replace Table B.1 with the following new table:

Table B.1 — Interindustry data objects relevant for card-verifiable certificates (non-exhaustive list)

Tag	Data element	Reg [a]	Hybrid [b]	Full QSC [c]
'06'	Object identifier	X	X	X
'42'	Issuer identification number	X	X	X
'5F20'	Cardholder name	X	X	X
'5F24'	Certificate expiration date	X	X	X
'5F25'	Certificate effective date	X	X	X
'5F29'	Certificate profile indicator	X	X	X
'5F37'	Static internal authentication (signature of a certificate, produced by the issuer)	X	X	
'5F49'	Cardholder public key	X	X	
'5F4C'	Certificate holder authorization	X	X	X
'5F4E'	Certificate content	X	N/A[d]	N/A[d]
'65'	Cardholder related data (e.g. certificate extensions)	X	X	X
'7F21'	Cardholder certificate	X	X	X
'7F49'	Public key template	X	X	
'7F4C'	Certificate holder authorization template (CHAT)	X	X	X
'7F4E'	Certificate content template	X	X	X
'7F3D'	Template nesting QSC digital signature objects [f]		X	X
'7F75'	Public key template, see 5.2		X	X
'7F76'	Common parameters and public key template, see 5.2		X	X
'96'	Descriptor of the certificate type (hybrid or not), see 5.3.7[e]		X	

NOTE 1 See examples provided in Table AMD.1.B.1, Table AMD.1.B.2, Table AMD.1.B.3 Table AMD.1.B.4, Table AMD.1.B.5, or Table AMD.1.B.6.

NOTE 2 See examples of signature cryptographic components in Table AMD.1.23.

[a] Applicable to regular CVC.

[b] Applicable to Hybrid certificate with Regular and QSC.

[c] Applicable to full QSC certificate.

[d] only self-descriptive certificates are considered.

[e] optional for non-hybrid certificate.

[f] DO'7F3D' (Table B.1) may feature a DO'73' containing a DO'80' denoting the signature format either with

— value '00' for byte string, followed with DO'81' nesting the byte string value, or

— value '01' for structured (BER-TLV), in which case the structured components may be ordered as per NIST reference publication when available (e.g. in accordance with Reference [15] for LMS or XMSS), and followed with data object(s) with sequenced tags starting from DO'81' and incrementing tag number to represent all the signature components.

Annex B, subclause B.5.10

Add the following new subclause B.6 after B.5.10:

B.6 "QSC-based self-descriptive certificates"

B.6.1 General

The QSC algorithms selected through the NIST contest[13] are newly standardized; their maturity may ramp up over a transition period. Issuers risk management policy will determine the relevance and pace of QSC adoption for certificates.

However, in order to cope with this situation, the best approach would be to combine QSC algorithms with regular asymmetric ones to

— put in place risk mitigation of the quantum computer threat, and

— still rely on field-proven maturity of regular asymmetric cryptography algorithms.

Accordingly, for the transition period and until QSC is widely deployed, this risk management approach leverages the use of hybrid certificates combining at the same time (1) a QSC signature and (2) a regular signature.

Two options are considered as follows:

— pure QSC based certificate;

— hybrid certificate combining regular asymmetric cryptography and QSC.

NOTE The procedure whereby an ICC validates a certificate is out of the scope of the current document and is implementation-specific.

B.6.2 Pure QSC based certificate

A card verifiable certificate may only make use of QSC algorithm, in which case it is called a pure QSC based certificate. In that case, it may use data objects as defined in Table B.1.

B.6.3 Hybrid certificate featuring regular cryptography and QSC

This option allows for two variants as follows:

— Variant 1: This hybrid certificate is issued using two cryptographic materials (public key and signature). Such a certificate comprises two certificate content templates ('7F4E'), i.e. one for regular asymmetric cryptography and one for QSC materials, and is signed twice. The DO '96' as well as the two certificate content templates ('7F4E') are signed respectively with regular asymmetric cryptography (RSA or ECC) and with QSC so that to obtain two signatures. Existing PKI is updated so that each CA contains two types of cryptographic materials, regular, i.e. RSA or ECC and QSC. See Figure AMD.1.B.1 and related CVC structure in Table B.1.

— The structure of such a certificate shall be the following:

  — It shall be structured within the template '7F21'.

  — The first DO within the template '7F21' shall be the DO '96' catering to indicate hybrid certificate variant 1 ('96' = '01').

  — The second DO within the template '7F21' shall be the certificate content template ('7F4E') containing regular cryptographic material.

  — The third DO within the template '7F21' shall be the certificate content template ('7F4E') containing QSC material.

  — The fourth DO ('5F37') within the template '7F21' shall contain the cryptographic signature computed using regular cryptography over the three first DOs ('96', '7F4E' and '7F4E') in the same order as the one used when structuring the certificate.

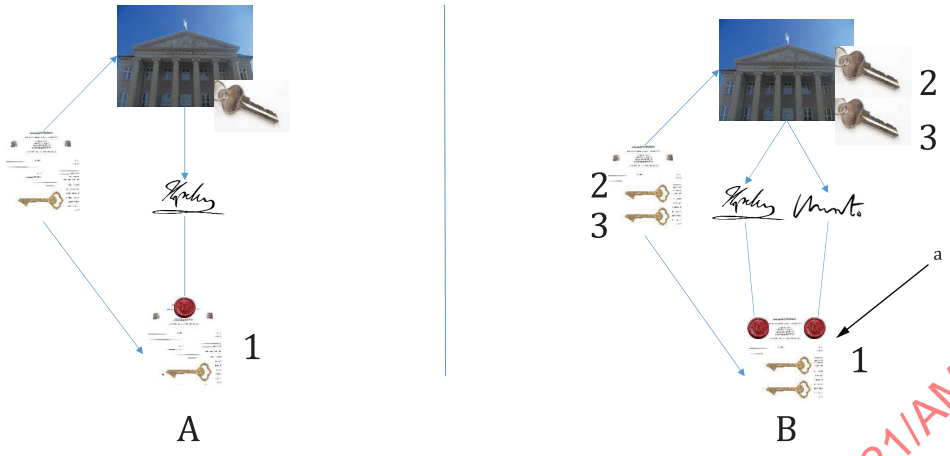
  — The fifth DO within the template '7F21' shall be the template '7F3D' containing the cryptographic signature computed using QSC. This signature shall be computed:

    — over the three first DOs ('96', '7F4E' and '7F4E') in the same order as the one used when structuring the certificate;

  - in accordance with the information provided in the third DO (certificate content template - '7F4E') that contains QSC material;

- Variant 2: After issuance, the regular certificate is sealed using QSC by another authority. An existing certificate already issued is enhanced with additional QSC signature that applies onto the certificate content template ('7F4E') and the DO containing the signature of the regular certificate ('5F37'). Thus, the result is a hybrid certificate containing two cryptographic signatures: the signature of the regular certificate (computed with regular asymmetric cryptography, e.g. RSA or ECC), as well as the seal (relying on QSC) applied on the regular certificate. The existing PKI is unchanged. See Figure AMD.1.B.2 and related CVC structure in Table B.1.

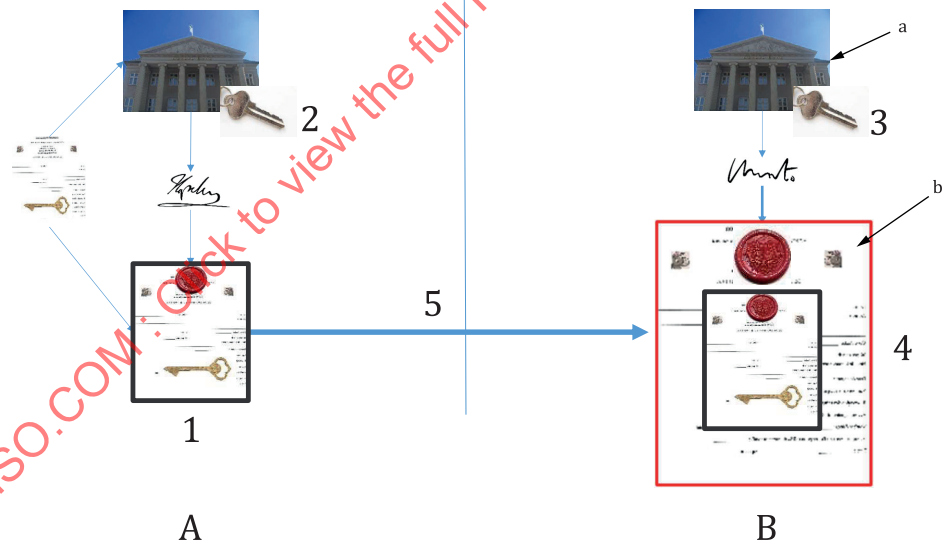
- The structure of such a certificate shall be the following:
  - It shall be structured within the template '7F21'.
  - The first DO within the template '7F21' shall be the DO '96' catering to indicate hybrid certificate variant 2 ('96' = '02').
  - The second DO within the template '7F21' shall be the certificate content template ('7F4E') of the regular certificate.
  - The third DO within the template '7F21' shall be the DO containing the cryptographic signature of the regular certificate ('5F37').
  - The fourth DO within the template '7F21' shall be the certificate content templates ('7F4E') containing QSC material.
  - The fifth DO within the template '7F21' shall be the template '7F3D' containing the cryptographic signature computed using QSC. This signature shall be computed as follows:
    - over the first DO ('96') and the regular certificate data objects which are
      - the certificate content template (second DO – '7F4E'), and
      - the DO containing the signature (third DO – '5F37')
      - in the same order as the one used when structuring the certificate;
    - in accordance with the information provided in the fourth DO (certificate content template – '7F4E') that contains QSC material;



Key

A regular approach
B hybrid approach
1 certificate
2 regular cryptography
3 QSC
a For security, the whole content shall be signed twice: with regular cryptography and with QSC.

Figure AMD.1.B.1 — Hybrid certificate of unique issuance approach (Variant 1)



Key

A issuance with regular approach
B update in hybrid approach in post issuance
1 certificate
2 regular cryptography
3 QSC
4 hybrid certificate
5 add QSC signature to extend life time of certificates
a Either same or different authority in charge of security.

[b] No public key in the certificate.

Figure AMD.1.B.2 — Hybrid certificate of post-issuance approach (Variant 2)

B.6.4 Examples of certificates featuring QSC

This subclause provides some examples of certificates featuring QSC in accordance with B.6.3.

Table AMD.1.B.1 describes hybrid certificate, variant 1 where the QSC material only contains the public key.

Table AMD.1.B.1 — Example of hybrid certificate — Variant 1

'7F21'									
	Var.								
		'96'	**'01'**	**'01'**					**Descriptor of the certificate type (hybrid variant 1)**
		'7F4E'							**Certificate content template – REGULAR CRYPTOGRAPHIC MATERIAL**
			Var.						
				'5F29'	Var.	-			Certificate Profile Identifier
				'42'	Var.	-			Certificate Authority Reference
				'7F49'	Var.	-			Public key
				'5F20'	Var.	-			Certificate Holder Reference
				'7F4C'	Var.	-			Certificate Holder Authorization Template (CHAT)
				'5F25'	Var.	-			Certificate Effective Date
				'5F24'	Var.	-			Certificate Expiration Date
				'65'	Var.	-			Cardholder related data (e.g. certificate extensions)
				'XY'	Var.	-			Other DO
		'7F4E'							**Certificate content template – QSC MATERIAL**
			Var.						
				'5F29'	Var.	-			Certificate Profile Identifier
				'42'	Var.	-			Certificate Authority Reference
				'7F75'	Var.	-			Public key
				'5F20'	Var.	-			Certificate Holder Reference
				'7F4C'	Var.	-			Certificate Holder Authorization Template (CHAT)
				'5F25'	Var.	-			Certificate Effective Date
				'5F24'	Var.	-			Certificate Expiration Date
				'65'	Var.	-			Cardholder related data (e.g. certificate extensions)
				'XY'	Var.	-			Other DO
		'5F37'	Var.	-					**Digital signature – COMPUTED USING REGULAR CRYPTOGRAPHIC MATERIAL**
		'7F3D'	Var.						**Template nesting QSC digital signature objects – COMPUTED USING QSC MATERIAL**
				'73'	Var.				
						'80'	01'	01'	Structured components ordered as per NIST reference publication
						'81'	Var.	-	Signature component 1
						'8n'	Var.	-	Signature component n

Table AMD.1.B.2 describes hybrid certificate – variant 1 where the QSC material contains the public key and common parameters.

Table AMD.1.B.2 — Example of hybrid certificate — Variant 1

'7F21'									
	Var.								
		'96'	**'01'**	**'01'**					**Descriptor of the certificate type (hybrid variant 1)**
		'7F4E'							**Certificate content template – REGULAR CRYPTOGRAPHIC MATERIAL**
			Var.						
				'5F29'	Var.	-			Certificate Profile Identifier
				'42'	Var.	-			Certificate Authority Reference
				'7F49'	Var.	-			Public key
				'5F20'	Var.	-			Certificate Holder Reference
				'7F4C'	Var.	-			Certificate Holder Authorization Template (CHAT)
				'5F25'	Var.	-			Certificate Effective Date
				'5F24'	Var.	-			Certificate Expiration Date
				'65'	Var.	-			Cardholder related data (e.g. certificate extensions)
				'XY'	Var.	-			Other DO
		'7F4E'							**Certificate content template – QSC MATERIAL**
			Var.						
				'5F29'	Var.	-			Certificate Profile Identifier
				'42'	Var.	-			Certificate Authority Reference
				'7F76'	Var.	-			Public key + common parameters
				'5F20'	Var.	-			Certificate Holder Reference
				'7F4C'	Var.	-			Certificate Holder Authorization Template (CHAT)
				'5F25'	Var.	-			Certificate Effective Date
				'5F24'	Var.	-			Certificate Expiration Date
				'65'	Var.	-			Cardholder related data (e.g. certificate extensions)
				'XY'	Var.	-			Other DO
		'5F37'	**Var.**	-					**Digital signature – COMPUTED USING REGULAR CRYPTOGRAPHIC MATERIAL**
		'7F3D'	**Var.**						**Template nesting QSC digital signature objects – COMPUTED USING QSC MATERIAL**
				73'	Var.				
						80'	01'	01'	Structured components ordered as per NIST reference publication
						81'	Var.	-	Signature component 1
						'8n'	Var.	-	Signature component n

Table AMD.1.B.3 describes hybrid certificate – variant 2 where the QSC material only contains the public key.

Table AMD.1.B.3 — Example of hybrid certificate — Variant 2

'7F21'										
	Var.									
		'96'	**'01'**	**'02'**					**Descriptor of the certificate type (hybrid variant 2)**	
		'7F4E'							**Certificate content template – REGULAR CRYPTOGRAPHIC MATERIAL**	**Regular Certificate**
			Var.							
				'5F29'	Var.	-			Certificate Profile Identifier	
				'42'	Var.	-			Certificate Authority Reference	
				'7F49'	Var.	-			Public key	
				'5F20'	Var.	-			Certificate Holder Reference	
				'7F4C'	Var.	-			Certificate Holder Authorization Template (CHAT)	
				'5F25'	Var.	-			Certificate Effective Date	
				'5F24'	Var.	-			Certificate Expiration Date	
				'65'	Var.	-			Cardholder related data (e.g. certificate extensions)	
				'XY'	Var.	-			Other DO	
		'5F37'	**Var.**	-					**Digital signature – COMPUTED USING REGULAR CRYPTOGRAPHIC MATERIAL**	
		'7F4E'							**Certificate content template – QSC MATERIAL**	**Information to seal the regular certificate**
			Var.							
				'5F29'	Var.	-			Certificate Profile Identifier	
				'42'	Var.	-			Certificate Authority Reference	
				'7F75'	Var.	-			Public key	
				'5F20'	Var.	-			Certificate Holder Reference	
				'7F4C'	Var.	-			Certificate Holder Authorization Template (CHAT)	
				'5F25'	Var.	-			Certificate Effective Date	
				'5F24'	Var.	-			Certificate Expiration Date	
				'65'	Var.	-			Cardholder related data (e.g. certificate extensions)	
				'XY'	Var.	-			Other DO	
		'7F3D'	**Var.**						**Template nesting QSC digital signature objects – COMPUTED USING QSC MATERIAL**	
				73'	Var.					
						80'	01'	01'	Structured components ordered as per NIST reference publication	
						81'	Var.	-	Signature component 1	
						'8n'	Var.	-	Signature component n	

Table AMD.1.B.4 describes hybrid certificate – variant 2 where the QSC material contains the public key and common parameters.

Table AMD.1.B.4 — Example of hybrid certificate — Variant 2

'7F21'										
	Var.									
		'96'	**'01'**	**'02'**					**Descriptor of the certificate type (hybrid variant 2)**	
		'7F4E'							**Certificate content template – REGULAR CRYPTOGRAPHIC MATERIAL**	**Regular Certificate**
			Var.							
				'5F29'	Var.	-			Certificate Profile Identifier	
				'42'	Var.	-			Certificate Authority Reference	
				'7F49'	Var.	-			Public key	
				'5F20'	Var.	-			Certificate Holder Reference	
				'7F4C'	Var.	-			Certificate Holder Authorization Template (CHAT)	
				'5F25'	Var.	-			Certificate Effective Date	
				'5F24'	Var.	-			Certificate Expiration Date	
				'65'	Var.	-			Cardholder related data (e.g. certificate extensions)	
				'XY'	Var.	-			Other DO	
		'5F37'	**Var.**	-					**Digital signature – COMPUTED USING REGULAR CRYPTOGRAPHIC MATERIAL**	
		'7F4E'							**Certificate content template – QSC MATERIAL**	**Information to seal the regular certificate**
			Var.							
				'5F29'	Var.	-			Certificate Profile Identifier	
				'42'	Var.	-			Certificate Authority Reference	
				'7F76'	Var.				Public key + common parameters	
				'5F20'	Var.	-			Certificate Holder Reference	
				'7F4C'	Var.	-			Certificate Holder Authorization Template (CHAT)	
				'5F25'	Var.	-			Certificate Effective Date	
				'5F24'	Var.	-			Certificate Expiration Date	
				'65'	Var.	-			Cardholder related data (e.g. certificate extensions)	
				'XY'	Var.	-			Other DO	
		'7F3D'	**Var.**						**Template nesting QSC digital signature objects – COMPUTED USING QSC MATERIAL**	
				73'	Var.					
						80'	01'	01'	Structured components ordered as per NIST reference publication	
						81'	Var.	-	Signature component 1	
						'8n'	Var.	-	Signature component n	

Table AMD.1.B.5 describes pure QSC based certificate where the QSC material only contains the public key.

Table AMD.1.B.5 — Example of pure QSC based certificate

'7F21'									
	Var.								
		'7F4E'							**Certificate content template**
			Var.						
				'5F29'	Var.	-			Certificate Profile Identifier
				'42'	Var.	-			Certificate Authority Reference
				'7F75'	Var.	-			Public key
				'5F20'	Var.	-			Certificate Holder Reference
				'7F4C'	Var.	-			Certificate Holder Authorization Template (CHAT)
				'5F25'	Var.	-			Certificate Effective Date
				'5F24'	Var.	-			Certificate Expiration Date
				'65'	Var.	-			Cardholder related data (e.g. certificate extensions)
				'XY'	Var.	-			Other DO
		'7F3D'	**Var.**						**Template nesting QSC digital signature objects**
				73'	Var.				
						80'	01'	01'	Structured components ordered as per NIST reference publication
						81'	Var.	-	Signature component 1
						'8n'	Var.	-	Signature component n

Table AMD.1.B.6 describes pure QSC based certificate where the QSC material contains the public key and common parameters.

Table AMD.1.B.6 — Example of pure QSC based certificate

'7F21'									
	Var.								
		'7F4E'							**Certificate content template**
			Var.						
				'5F29'	Var.	-			Certificate Profile Identifier
				'42'	Var.	-			Certificate Authority Reference
				'7F76'	Var.	-			Public key + common parameters
				'5F20'	Var.	-			Certificate Holder Reference
				'7F4C'	Var.	-			Certificate Holder Authorization Template (CHAT)
				'5F25'	Var.	-			Certificate Effective Date
				'5F24'	Var.	-			Certificate Expiration Date
				'65'	Var.	-			Cardholder related data (e.g. certificate extensions)
				'XY'	Var.	-			Other DO
		'7F3D'	**Var.**						**Template nesting QSC digital signature objects**
				73'	Var.				
						80'	'01'	'01'	Structured components ordered as per NIST reference publication
						'81'	Var.	-	Signature component 1
						'8n'	Var.	-	Signature component n

Annex C, subclause C.2.2

Add the following new subclause C.3 after C.2.2:

C.3 QSC key import

For the purpose of import on ICC of QSC private key or public key or common parameters or combination thereof, the IMPORT CARD SECRET command (P1-P2 = '31 00') may be used with either template '7F75' or '7F76' or '7F77' in command data field (see ISO/IEC 7816-9:2017, 6.1.1).

PUT DATA command (INS = 'DA') may be used too with special function (P1-P2 = '00 FF') and data field containing either template '7F75' or '7F76' or '7F77' and possibly further data objects.

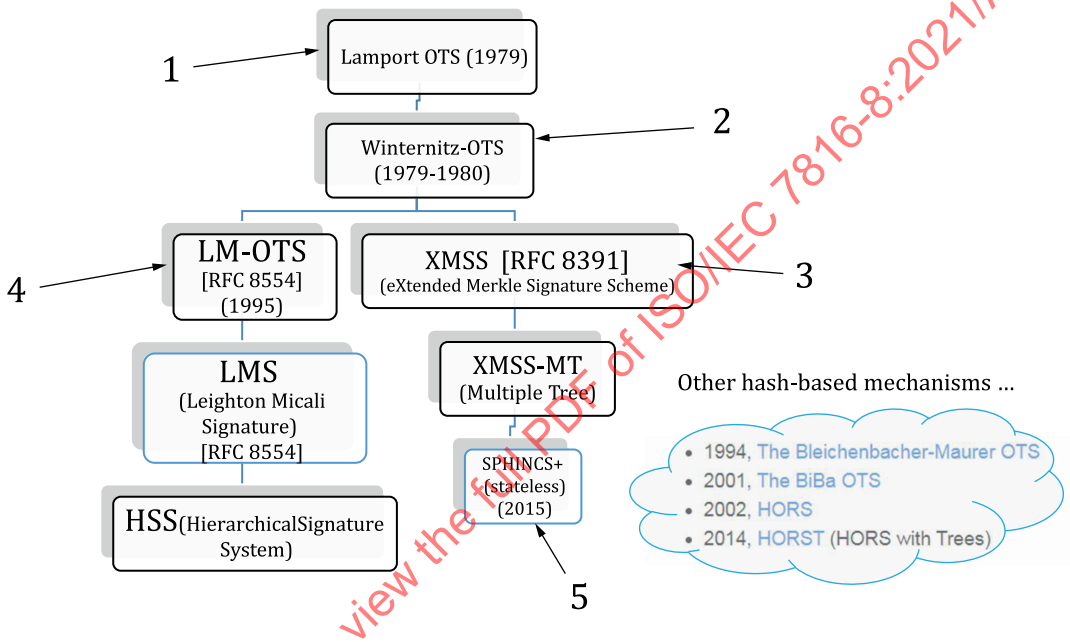
Add the following new Annex F after Annex E:

Annex F
(informative)

QSC Miscellaneous

F.1 Overview of (stateful) hash-based signature

Figure F.1 presents an informative tentative genealogy of hash-based schemes and some common features related to these schemes are described. Further description is provided in F.2 and F.3.



Key

1 The characteristics of these signature systems are small private and public keys and fast signature generation and verification, but large signatures and moderately slow key generation [in comparison with RSA and Elliptic Curve Digital Signature Algorithm (ECDSA)]. This signature only relied on the security of the one-way function.

Private key (512 values) = two data sets with 256 random 256-bit numbers.

Public key (512 hashes) = hash of each of the random numbers.

2 Private and public key is smaller than with Lamport OTS. The signature scheme introduces further parameters (i.e. w parameter).

Private key (32 values) = 32 256-bit random numbers.

Public key (32 values) = each of the private key values is hashed 256 times.

3 Several signatures with the use of one or multiple Merkle Tree.

4 LM-OTS signature is parameterized, i.e. key ID, leaf ID, random, w, checksum.

5 SPHINCS+ 256 has a public key size of 1kB, a private key size of 1kB, and a signature of 41 kB.

It uses WOTS and HORST for hash-based trees.

Figure F.1 — Tentative genealogy for QSC hash-based schemes

Merkle Tree, when employed with hash-based signature scheme, allows for multiple signatures, e.g. from 32 to several thousands. The generation of private key(s) shall preferably be performed dynamically within ICC, but some constraints arise from this scheme:

— memory footprint likely to exceed common ICC capability;

— numerous numbers of hash calls are required which impacts time/performance;

— after each signature, the state shall be maintained to initiate the upcoming signature.

Optimization is necessary but out of scope of this document.

Interoperability parameters are needed for some aspects of hash-based scheme:

— key type: to infer the parameters' structure (DO'9X' series);

— key identifier: for interchange purposes (internal reference, e.g. MSE SET);

— identifier of external common parameters: to look up the common parameters;

— identifier of each level: for key location and navigation in HSS multiple tree;

— LM type for the level (according to NIST Reference [15] and Reference [23]): as the height of the tree 'H';

— LM-OTS type for the level (according to NIST Reference [15] and Reference [23]): as the Winternitz coefficient 'w'.

NOTE 'w' in LMS corresponds to logarithm of w in XMSS.

See QSC templates for representation and coding of those parameters.

F.2 "Winternitz-OTS key generation scheme

According to NIST SP 800-208[15], in the Winternitz signature scheme (hash-based), the message to be signed is hashed to create a digest; the digest is encoded as a base b number; and then each digit of the digest is signed using a hash chain.

A hash chain is created by first randomly generating a secret value, sk_i, which is the private key. The size of i should generally correspond to the targeted strength of the scheme.

The public key is then created by applying the hash function, H, to the secret $b - 1$ times, $H^{b-1}(i)$. The secret key generation processing is outside the interoperability scope. Figure F.2 describes a Winternitz-OTS key generation.

The format of the LM-OTS private key is an internal matter to the implementation (out of scope).

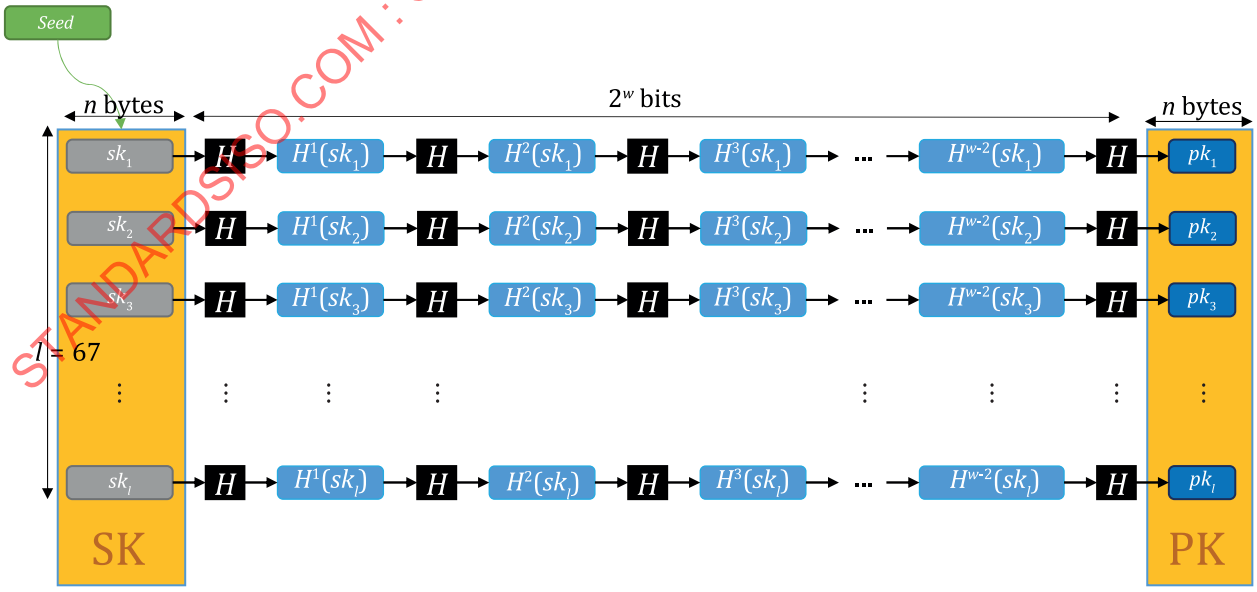


Figure F.2 — Winternitz-OTS, key generation

F.3 "Winternitz-OTS signature and verification mechanism

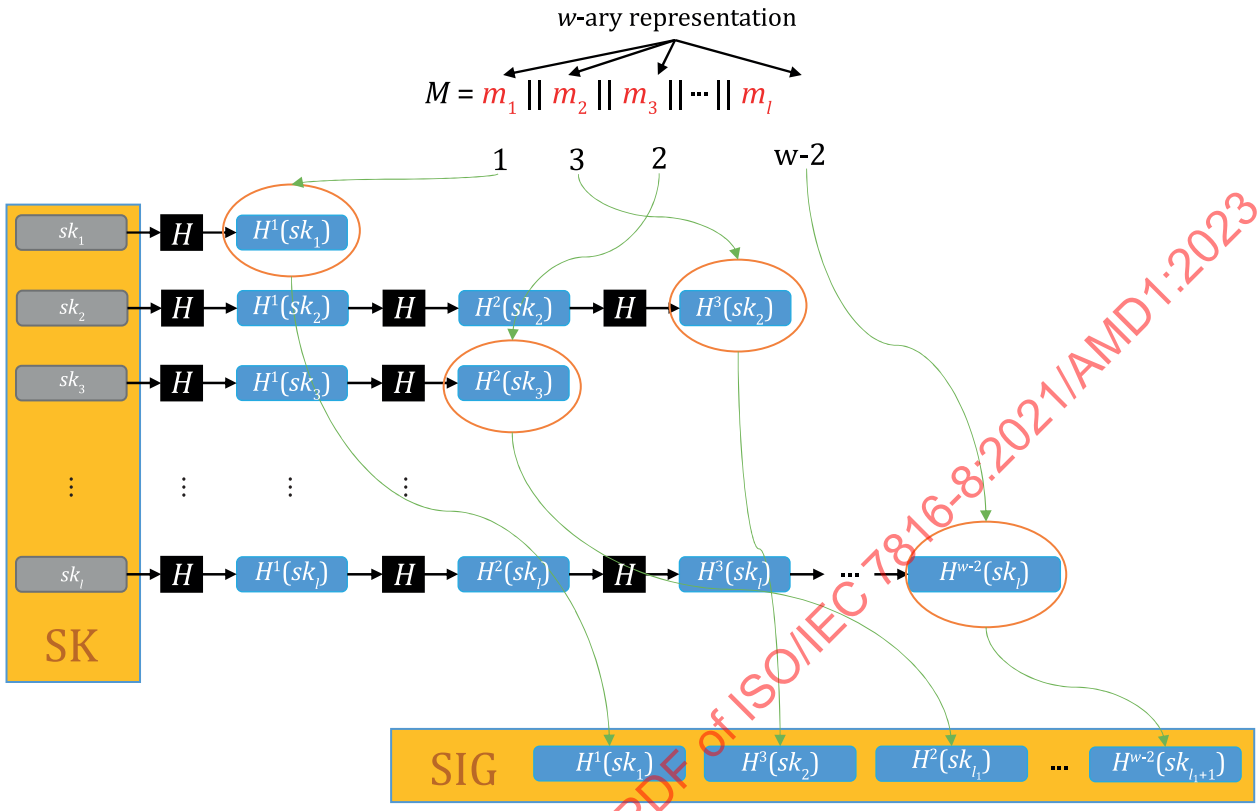


Figure F.3 — Winternitz-OTS signature principle

In the Winternitz signature scheme, the message to be signed is hashed to create a digest; the digest is encoded as a base *b* number and then each digit of the digest is signed using a hash chain. A checksum is needed because an attacker can freely advance any of the Winternitz chains. The checksum prevents an attacker who could find a hash that has every digit larger than the valid hash could replace it and adjust the Winternitz chains. See in Figure F.3 an example of a signature for a 32-bit message digest using *b* = 16. The digest is written as eight hexadecimal digits, and a separate hash chain is used to sign each digit with each hash chain having its own private key.

The parameter w determines the length of the Winternitz chains computed as a part of the OTS signature (which involve 2^w - 1 invocations of the hash function).

b = 16 means that each Digest element valuates to a digit is the interval {0..15} or from '00' to '0F'.

b = 16 means as well that the Digest elements are strings over a nibble (4-bit elements).

For an n-digit message digest, the checksum is computed as $\sum_{k=0}^{n-1}(b-1-N_k)$

An attacker could not, however, generate a signature for a message digest with a kth digit of 0 as this would require finding some value y such that H(*y*)=*sk*, which would not be feasible as long as H is preimage-resistant, see a sample Winternitz signature[15] in Figure F.4 showing the digest with its checksum.