Application Notes

Wire Format

07 October 2024

Table of Contents

This appnote defines the nCore wire format for any data types required in key attestation verification (see Key Attestation Format).

Any other data types are omitted.

The contents are split as follows:

- primitive types
- common types (i.e. those which are not defined as primitives but are used by more than one of the structures below)
- a section each per structure required for key attestation (including descendant types used exclusively in that structure)

1. Primitive type formats

This section defines the wire format for a number of primitive types.

1.1. M_ASCIIString

This represents an arbitrary ASCII string.

Strings always include a final 0 byte, which is included in the length *n* below.

Field	Size	Format	Meaning
n	4 bytes	Little-endian unsigned integer	Number of bytes in string
Content	*n* bytes	ASCII data	The contents of the string
Padding	0-3 bytes	0-valued bytes	Rounds length up to a multiple of 4 bytes total

1.2. M_Bignum

This represents an arbitrary integer.

All bignums are multiples of 4 bytes long. Apart from that, normally the minimal representation is used, except that 0 is always represented by 4 bytes.

Field	Size	Format	Meaning
n	4 bytes	Little-endian unsigned integer	Number of bytes in bignum
Content	*n* bytes	Little-endian integer	The value of the integer

1.3. M_ByteBlock

This represents an arbitrary byte string

Field	Size	Format	Meaning
n	4 bytes	Little-endian unsigned integer	Number of bytes in byteblock
Content	*n* bytes	Binary data	The contents of the byteblock
Padding	0-3 bytes	0-valued bytes	Rounds length up to a multiple of 4 bytes total

1.4. M_FileID

This represents an 11-byte string, usually a filename on a smartcard.

Field	Size	Format	Meaning
Content	11 bytes	Binary data	The contents of the file ID

1.5. M_Hash

This represents a 20-byte string, often the result of a SHA-1 hash or similar.

Field	Size	Format	Meaning
Content	20 bytes	Binary data	The contents of the hash

1.6. M_Hash32

This represents a 32-byte string, often the result of a SHA-256 hash or similar.

Field	Size	Format	Meaning
Content	32 bytes	Binary data	The contents of the hash

1.7. M_Hash64

This represents a 64-byte string, often the result of a SHA-512 hash or similar.

Field	Size	Format	Meaning
Content	64 bytes	Binary data	The contents of the hash

1.8. M_Word

This represents a 32-bit unsigned integer.

Field	Size	Format	Meaning
Value	4 bytes	Little-endian unsigned integer	The value of the integer

2. Common type formats

This section defines the wire format for types used by more than one of the top-level structures defined in other sections.

2.1. M_DSADiscreteLogGroup

This represents a structure with the following fields:

Field	Size	Format
p	variable	See M_Bignum
q	variable	See M_Bignum
g	variable	See M_Bignum

2.2. M_ECName

An enumeration type, represented as a 4-byte little-endian unsigned integer. Possible values include:

Value	Name
2	`ECName_NISTP192`
3	`ECName_NISTP224`
4	`ECName_NISTP256`
5	`ECName_NISTP384`
6	`ECName_NISTP521`
7	`ECName_NISTB163`
8	`ECName_NISTB233`
9	`ECName_NISTB283`
10	`ECName_NISTB409`
11	`ECName_NISTB571`
12	`ECName_NISTK163`
13	`ECName_NISTK233`
14	`ECName_NISTK283`

Value	Name
15	ECName_NISTK409
16	ECName_NISTK571
17	ECName_ANSIB163v1
18	ECName_ANSIB191v1
19	ECName_SECP160r1
22	ECName_SECP256k1
23	ECName_BrainpoolP160r1
24	ECName_BrainpoolP160t1
25	ECName_BrainpoolP192r1
26	ECName_BrainpoolP192t1
27	ECName_BrainpoolP224r1
28	ECName_BrainpoolP224t1
29	ECName_BrainpoolP256r1
30	ECName_BrainpoolP256t1
31	ECName_BrainpoolP320r1
32	ECName_BrainpoolP320t1
33	ECName_BrainpoolP384r1
34	ECName_BrainpoolP384t1
35	ECName_BrainpoolP512r1
36	ECName_BrainpoolP512t1

2.3. M_EllipticCurve

This represents a structure with the following fields:

Field	Size	Format
name	4 bytes	Little-endian unsigned integer. See M_ECName
data	variable	Depends on field `name`. See below.

2.3.1. M_EllipticCurve.data representations

All of the supported values of `name` correspond to an empty (zero-length) `data` field.

2.4. M_KeyType

An enumeration type, represented as a 4-byte little-endian unsigned integer. Possible values include:

Value	Name
1	`KeyType_RSAPublic`
2	`KeyType_RSAPrivate`
3	`KeyType_DSAPublic`
19	`KeyType_DSAPrivate`
39	`KeyType_KCDSAPublic`
40	`KeyType_KCDSAPrivate`
44	`KeyType_ECPublic`
45	`KeyType_ECPrivate`
46	`KeyType_ECDSAPublic`
47	`KeyType_ECDSAPrivate`
65	`KeyType_Ed25519Public`
66	`KeyType_Ed25519Private`

2.5. M_Mech

An enumeration type, represented as a 4-byte little-endian unsigned integer. Possible values include:

Value	Name
170	`Mech_DSAhSHA256`
187	`Mech_ECDSAhSHA512`

3. CipherText format

This section defines the wire format for M_CipherText and its descendant types.

In a key attestation bundle as described in Key attestation bundle construction, the following fields will use this format:

- `modstatesig`
- `kcsig`
- `CertKMaKMCbKNSO`
- `CertKMaKMCaKFIPSbKNSO`
- `CertKREaKRAbKNSO`

3.1. M_CipherText

This represents a structure with the following fields:

Field	Size	Format
mech	4 bytes	Little-endian unsigned integer. See M_Mech
data	variable	Depends on field `mech`. See below.
iv	variable	Depends on field `mech`. See below.

3.1.1. M_CipherText.data representations

This depends on the value of the `mech` field, as follows:

Name of `mech`	Value of `mech`	Format of `data`
Mech_DSAhSHA256	170	M_Mech_DSA_Cipher
Mech_ECDSAhSHA512	187	M_Mech_ECDSA_Cipher

Any supported values of `mech` not present in the table correspond to an empty (zero-length) `data` field.

3.1.2. M_CipherText.iv representations

All of the supported values of `mech` correspond to an empty (zero-length) `iv` field.

3.2. M_Mech_DSA_Cipher

This represents a structure with the following fields:

Field	Size	Format
r	variable	See M_Bignum
s	variable	See M_Bignum

3.3. M_Mech_ECDSA_Cipher

This represents a structure with the following fields:

Field	Size	Format
r	variable	See M_Bignum
s	variable	See M_Bignum

4. KeyData format

This section defines the wire format for M_KeyData and its descendant types.

In a key attestation bundle as described in Key attestation bundle construction, the following fields will use this format:

- `pubkeydata`
- `knsopub`

4.1. M_KeyData

This represents a structure with the following fields:

Field	Size	Format
type	4 bytes	Little-endian unsigned integer. See M_KeyType
data	variable	Depends on field `type`. See below.

4.1.1. M_KeyData.data representations

This depends on the value of the `type` field, as follows:

Name of `type`	Value of `type`	Format of `data`
KeyType_RSAPublic	1	M_KeyType_RSAPublic_Data
KeyType_RSAPrivate	2	M_KeyType_RSAPrivate_Data
KeyType_DSAPublic	3	M_KeyType_DSAPublic_Data
KeyType_DSAPrivate	19	M_KeyType_DSAPrivate_Data
KeyType_KCDSAPublic	39	M_KeyType_KCDSAPublic_Data
KeyType_KCDSAPrivate	40	M_KeyType_KCDSAPrivate_Data
KeyType_ECPublic	44	M_KeyType_ECPublic_Data
KeyType_ECPrivate	45	M_KeyType_ECPrivate_Data
KeyType_ECDSAPublic	46	M_KeyType_ECPublic_Data
KeyType_ECDSAPrivate	47	M_KeyType_ECPrivate_Data
KeyType_Ed25519Public	65	M_KeyType_Random_Data
KeyType_Ed25519Private	66	M_KeyType_Random_Data

Any supported values of `type` not present in the table correspond to an empty (zero-length) `data` field.

4.2. M_ECPoint

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_ECPoint_flags
x	variable	See M_Bignum
y	variable	See M_Bignum

4.3. M_ECPoint_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`Infinity`

4.4. M_KeyType_DSAPrivate_Data

This represents a structure with the following fields:

Field	Size	Format
dlg	variable	See M_DSADiscreteLogGroup
x	variable	See M_Bignum

4.5. M_KeyType_DSAPublic_Data

This represents a structure with the following fields:

Field	Size	Format
dlg	variable	See M_DSADiscreteLogGroup
y	variable	See M_Bignum

4.6. M_KeyType_ECPrivate_Data

This represents a structure with the following fields:

Field	Size	Format
curve	variable	See M_EllipticCurve
d	variable	See M_Bignum

4.7. M_KeyType_ECPublic_Data

This represents a structure with the following fields:

Field	Size	Format
curve	variable	See M_EllipticCurve
Q	variable	See M_ECPoint

4.8. M_KeyType_KCDSAPrivate_Data

This represents a structure with the following fields:

Field	Size	Format
dlg	variable	See M_DSADiscreteLogGroup
y	variable	See M_Bignum
x	variable	See M_Bignum

4.9. M_KeyType_KCDSAPublic_Data

This represents a structure with the following fields:

Field	Size	Format
dlg	variable	See M_DSADiscreteLogGroup
y	variable	See M_Bignum

4.10. M_KeyType_RSAPrivate_Data

This represents a structure with the following fields:

Field	Size	Format
p	variable	See M_Bignum
q	variable	See M_Bignum
dmp1	variable	See M_Bignum
dmq1	variable	See M_Bignum
iqmp	variable	See M_Bignum
e	variable	See M_Bignum

4.11. M_KeyType_RSAPublic_Data

This represents a structure with the following fields:

Field	Size	Format
e	variable	See M_Bignum
n	variable	See M_Bignum

4.12. M_KeyType_Random_Data

This represents a structure with the following fields:

Field	Size	Format
k	variable	See M_ByteBlock

5. KeyHashEx format

This section defines the wire format for M_KeyHashEx and its descendant types.

In a key attestation bundle as described in Key attestation bundle construction, the following fields will use this format:

- `hkre`
- `hkra`
- `hkfips`
- `hkmc`
- `hkm`

5.1. M_KeyHashEx

This represents a structure with the following fields:

Field	Size	Format
mech	4 bytes	Little-endian unsigned integer. See M_KeyHashMech
data	variable	Depends on field `mech`. See below.

5.1.1. M_KeyHashEx.data representations

This depends on the value of the `mech` field, as follows:

Name of `mech`	Value of `mech`	Format of `data`
KeyHashMech_SHA1Hash	44	M_KeyHashMech_SHA1Hash_KeyHashData
KeyHashMech_SHA256Hash	93	M_KeyHashMech_SHA256Hash_KeyHashData
KeyHashMech_SHA512Hash	95	M_KeyHashMech_SHA512Hash_KeyHashData

Any supported values of `mech` not present in the table correspond to an empty (zero-length) `data` field.

5.2. M_KeyHashMech

An enumeration type, represented as a 4-byte little-endian unsigned integer. Possible values include:

Value	Name
44	`KeyHashMech_SHA1Hash`
93	`KeyHashMech_SHA256Hash`
95	`KeyHashMech_SHA512Hash`

5.3. M_KeyHashMech_SHA1Hash_KeyHashData

This represents a structure with the following fields:

Field	Size	Format
hash	variable	See M_Hash

5.4. M_KeyHashMech_SHA256Hash_KeyHashData

This represents a structure with the following fields:

Field	Size	Format
hash	variable	See M_Hash32

5.5. M_KeyHashMech_SHA512Hash_KeyHashData

This represents a structure with the following fields:

Field	Size	Format
hash	variable	See M_Hash64

6. ModCertMsg format

This section defines the wire format for M_ModCertMsg and its descendant types.

In a key attestation bundle as described in Key attestation bundle construction, the following fields will use this format:

- `modstatemsg`
- `kcmsg`

6.1. M_ModCertMsg

This represents a structure with the following fields:

Field	Size	Format
type	4 bytes	Little-endian unsigned integer. See M_ModCertType
data	variable	Depends on field `type`. See below.

6.1.1. M_ModCertMsg.data representations

This depends on the value of the `type` field, as follows:

Name of `type`	Value of `type`	Format of `data`
ModCertType_KeyGen	2	M_ModCertType_KeyGen_ModCertData
ModCertType_StateCert	4	M_ModCertType_StateCert_ModCertData

Any supported values of `type` not present in the table correspond to an empty (zero-length) `data` field.

6.2. M_DSAGenerationHash

This represents a structure with the following fields:

Field	Size	Format
hash	4 bytes	Little-endian unsigned integer. See M_Mech

6.3. M_KeyGenParams

This represents a structure with the following fields:

Field	Size	Format
type	4 bytes	Little-endian unsigned integer. See M_KeyType
params	variable	Depends on field `type`. See below.

6.3.1. M_KeyGenParams.params representations

This depends on the value of the `type` field, as follows:

Name of `type`	Value of `type`	Format of `params`
KeyType_RSAPrivate	2	M_KeyType_RSAPrivate_GenParams
KeyType_DSAPrivate	19	M_KeyType_DSAPrivate_GenParams
KeyType_KCDSAPrivate	40	M_KeyType_KCDSAPrivate_GenParams
KeyType_ECPrivate	45	M_KeyType_ECPrivate_GenParams
KeyType_ECDSAPrivate	47	M_KeyType_ECPrivate_GenParams

Any supported values of `type` not present in the table correspond to an empty (zero-length) `params` field.

6.4. M_KeyHashAttrib

This represents a structure with the following fields:

Field	Size	Format
hk	variable	See M_Hash
mech_i	4 bytes	Little-endian unsigned integer. See M_Mech
mech_c	4 bytes	Little-endian unsigned integer. See M_Mech

6.4.1. M_vec_KeyHashAttrib

This represents an array of M_KeyHashAttrib objects.

6.5. M_KeyType_DSAPrivate_GenParams

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_KeyType_DSAPrivate_GenParams_flags
lenbits	4 bytes	Little-endian unsigned integer. See M_Word
dlg	variable	Empty, or M_DSADiscreteLogGroup if `dlg_present` (0x00000001) is set in flags)
qhash	variable	Empty, or M_DSAGenerationHash if `qhash_present` (0x00000004) is set in flags)

6.6. M_KeyType_DSAPrivate_GenParams_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`dlg_present`
0x00000002	`Strict`
0x00000004	`qhash_present`

6.7. M_KeyType_ECPrivate_GenParams

This represents a structure with the following fields:

Field	Size	Format
curve	variable	See M_EllipticCurve

6.8. M_KeyType_KCDSAPrivate_GenParams

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_KeyType_KCDSAPrivate_GenParams_flags

Field	Size	Format
plen	4 bytes	Little-endian unsigned integer. See M_Word
qlen	4 bytes	Little-endian unsigned integer. See M_Word
dlg	variable	Empty, or M_DSADiscreteLogGroup if `dlg_present` (0x00000001) is set in flags)

6.9. M_KeyType_KCDSAPrivate_GenParams_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`dlg_present`

6.10. M_KeyType_RSAPrivate_GenParams

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_KeyType_RSAPrivate_GenParams_flags
lenbits	4 bytes	Little-endian unsigned integer. See M_Word
given_e	variable	Empty, or M_Bignum if `given_e_present` (0x00000001) is set in flags)
nchecks	variable	Empty, or M_Word if `nchecks_present` (0x00000002) is set in flags)

6.11. M_KeyType_RSAPrivate_GenParams_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`given_e_present`
0x00000002	`nchecks_present`

Value	Name
0x00000004	`UseStrongPrimes`

6.12. M_ModCertType

An enumeration type, represented as a 4-byte little-endian unsigned integer. Possible values include:

Value	Name
2	`ModCertType_KeyGen`
4	`ModCertType_StateCert`

6.13. M_ModCertType_KeyGen_ModCertData

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_ModCertType_KeyGen_ModCertData_flags
genparams	variable	See M_KeyGenParams
acl	variable	See M_ACL
hka	variable	See M_Hash
hkaex	variable	Empty, or M_KeyHashEx if `hkaex_present` (0x00000002) is set in flags)

6.14. M_ModCertType_KeyGen_ModCertData_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`Public`
0x00000002	`hkaex_present`

6.15. M_ModCertType_StateCert_ModCertData

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_ModCertType_StateCert_ModCertData_flags
state	variable	See M_ModuleAttribList

6.16. M_ModCertType_StateCert_ModCertData_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. No flags are currently defined for this field.

6.17. M_ModKeyInfoEx

This represents a structure with the following fields:

Field	Size	Format
v	4 bytes	Little-endian unsigned integer. See M_Word
hk	variable	See M_KeyHashEx
type	4 bytes	Little-endian unsigned integer. See M_KeyType
mech_i	4 bytes	Little-endian unsigned integer. See M_Mech
mech_c	4 bytes	Little-endian unsigned integer. See M_Mech

6.17.1. M_vec_ModKeyInfoEx

This represents an array of M_ModKeyInfoEx objects.

6.18. M_ModuleAttrib

This represents a structure with the following fields:

Field	Size	Format
tag	4 bytes	Little-endian unsigned integer. See M_ModuleAttribTag
value	variable	Depends on field `tag`. See below.

6.18.1. M_ModuleAttrib.value representations

This depends on the value of the `tag` field, as follows:

Name of `tag`	Value of `tag`	Format of `value`
ModuleAttribTag_ESN	2	M_ModuleAttribTag_ESN_Value
ModuleAttribTag_KML	3	M_ModuleAttribTag_KML_Value
ModuleAttribTag_KNSO	5	M_ModuleAttribTag_KNSO_Value
ModuleAttribTag_KMList	6	M_ModuleAttribTag_KMList_Value
ModuleAttribTag_KLF2	13	M_ModuleAttribTag_KLF2_Value
ModuleAttribTag_KMLEx	19	M_ModuleAttribTag_KMLEx_Value
ModuleAttribTag_KNSOEx	20	M_ModuleAttribTag_KNSOEx_Value
ModuleAttribTag_ModKeyInfoEx	21	M_ModuleAttribTag_ModKeyInfoEx_Value
ModuleAttribTag_KLF2Ex	22	M_ModuleAttribTag_KMLEx_Value

Any supported values of `tag` not present in the table correspond to an empty (zero-length) `value` field.

6.18.2. M_vec_ModuleAttrib

This represents an array of M_ModuleAttrib objects.

6.19. M_ModuleAttribList

This represents a structure with the following fields:

Field	Size	Format
n_attribs	4 bytes	Little-endian unsigned integer.
attribs	variable	*n_attribs* copies of `M_ModuleAttrib`. See M_vec_ModuleAttrib

6.20. M_ModuleAttribTag

An enumeration type, represented as a 4-byte little-endian unsigned integer. Possible values include:

Value	Name
2	ModuleAttribTag_ESN
3	ModuleAttribTag_KML
5	ModuleAttribTag_KNSO
6	ModuleAttribTag_KMList
13	ModuleAttribTag_KLF2
19	ModuleAttribTag_KMLEx
20	ModuleAttribTag_KNSOEx
21	ModuleAttribTag_ModKeyInfoEx
22	ModuleAttribTag_KLF2Ex

6.21. M_ModuleAttribTag_ESN_Value

This represents a structure with the following fields:

Field	Size	Format
esn	variable	See M_ASCIIString

6.22. M_ModuleAttribTag_KLF2_Value

This represents a structure with the following fields:

Field	Size	Format
hklf2	variable	See M_Hash
klf2pub	variable	See M_KeyData
mech_i	4 bytes	Little-endian unsigned integer. See M_Mech

6.23. M_ModuleAttribTag_KMLEx_Value

This represents a structure with the following fields:

Field	Size	Format
hk	variable	See M_KeyHashEx

Field	Size	Format
pubkey	variable	See M_KeyData
mech_i	4 bytes	Little-endian unsigned integer. See M_Mech

6.24. M_ModuleAttribTag_KML_Value

This represents a structure with the following fields:

Field	Size	Format
hkml	variable	See M_Hash
kmlpub	variable	See M_KeyData
mech_i	4 bytes	Little-endian unsigned integer. See M_Mech

6.25. M_ModuleAttribTag_KMList_Value

This represents a structure with the following fields:

Field	Size	Format
n_hkms	4 bytes	Little-endian unsigned integer.
hkms	variable	*n_hkms* copies of `M_KeyHashAttrib`. See M_vec_KeyHashAttrib

6.26. M_ModuleAttribTag_KNSOEx_Value

This represents a structure with the following fields:

Field	Size	Format
hknso	variable	See M_KeyHashEx
publicperms	variable	See M_NSOPerms

6.27. M_ModuleAttribTag_KNSO_Value

This represents a structure with the following fields:

Field	Size	Format
hknso	variable	See M_Hash
publicperms	variable	See M_NSOPerms

6.28. M_ModuleAttribTag_ModKeyInfoEx_Value

This represents a structure with the following fields:

Field	Size	Format
n_kms	4 bytes	Little-endian unsigned integer.
kms	variable	*n_kms* copies of `M_ModKeyInfoEx`. See M_vec_ModKeyInfoEx

6.29. M_NSOPerms

This represents a structure with the following fields:

Field	Size	Format
ops	4 bytes	Little-endian unsigned integer. See M_NSOPerms_ops

6.30. M_NSOPerms_ops

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`LoadLogicalToken`
0x00000002	`ReadFile`
0x00000004	`WriteShare`
0x00000008	`WriteFile`
0x00000010	`EraseShare`
0x00000020	`EraseFile`
0x00000040	`FormatToken`
0x00000080	`SetKM`
0x00000100	`RemoveKM`

Value	Name
0x00000200	`GenerateLogToken`
0x00000400	`ChangeSharePIN`
0x00000800	`OriginateKey`
0x00001000	`NVMemAlloc`
0x00002000	`NVMemFree`
0x00004000	`GetRTC`
0x00008000	`SetRTC`
0x00010000	`DebugSEEWorld`
0x00020000	`SendShare`
0x00040000	`ForeignTokenOpen`

7. ACL format

This section defines the wire format for M_ACL and its descendant types.

In a key attestation bundle as described in Key attestation bundle construction, the following fields will use this format:

- `kcmsg.data.acl`

7.1. M_ACL

This represents a structure with the following fields:

Field	Size	Format
n_groups	4 bytes	Little-endian unsigned integer.
groups	variable	*n_groups* copies of `M_PermissionGroup`. See M_vec_PermissionGroup

7.2. M_Act

An enumeration type, represented as a 4-byte little-endian unsigned integer. Possible values include:

Value	Name
1	`Act_OpPermissions`
2	`Act_MakeBlob`
3	`Act_MakeArchiveBlob`
5	`Act_DeriveKey`
47	`Act_DeriveKeyEx`

7.3. M_Act_DeriveKeyEx_Details

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_Act_DeriveKeyEx_Details_flags

Field	Size	Format
role	4 bytes	Little-endian unsigned integer. See M_DeriveRole
mech	4 bytes	Little-endian unsigned integer. See M_DeriveMech
n_otherkeys	4 bytes	Little-endian unsigned integer.
otherkeys	variable	*n_otherkeys* copies of `M_KeyRoleIDEx`. See M_vec_KeyRoleIDEx
params	variable	Empty, or M_DKMechParams if `params_present` (0x00000001) is set in flags)

7.4. M_Act_DeriveKeyEx_Details_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`params_present`

7.5. M_Act_DeriveKey_Details

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_Act_DeriveKey_Details_flags
role	4 bytes	Little-endian unsigned integer. See M_DeriveRole
mech	4 bytes	Little-endian unsigned integer. See M_DeriveMech
n_otherkeys	4 bytes	Little-endian unsigned integer.
otherkeys	variable	*n_otherkeys* copies of `M_KeyRoleID`. See M_vec_KeyRoleID
params	variable	Empty, or M_DKMechParams if `params_present` (0x00000001) is set in flags)

7.6. M_Act_DeriveKey_Details_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`params_present`

7.7. M_Act_MakeArchiveBlob_Details

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_Act_MakeArchiveBlob_Details_flags
mech	4 bytes	Little-endian unsigned integer. See M_Mech
kahash	variable	Empty, or M_Hash if `kahash_present` (0x00000001) is set in flags)
blobfile	variable	Empty, or M_MakeBlobFilePerms if `blobfile_present` (0x00000002) is set in flags)

7.8. M_Act_MakeArchiveBlob_Details_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`kahash_present`
0x00000002	`blobfile_present`

7.9. M_Act_MakeBlob_Details

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_Act_MakeBlob_Details_flags
kmhash	variable	Empty, or M_Hash if `kmhash_present` (0x00000004) is set in flags)
kthash	variable	Empty, or M_Hash if `kthash_present` (0x00000008) is set in flags)

Field	Size	Format
ktparams	variable	Empty, or M_TokenParams if `ktparams_present` (0x00000010) is set in flags)
blobfile	variable	Empty, or M_MakeBlobFilePerms if `blobfile_present` (0x00000040) is set in flags)

7.10. M_Act_MakeBlob_Details_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`AllowKmOnly`
0x00000002	`AllowNonKm0`
0x00000004	`kmhash_present`
0x00000008	`kthash_present`
0x00000010	`ktparams_present`
0x00000020	`AllowNullKmToken`
0x00000040	`blobfile_present`

7.11. M_Act_OpPermissions_Details

This represents a structure with the following fields:

Field	Size	Format
perms	4 bytes	Little-endian unsigned integer. See M_Act_OpPermissions_Details_perms

7.12. M_Act_OpPermissions_Details_perms

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`DuplicateHandle`

Value	Name
0x00000002	`UseAsCertificate`
0x00000004	`ExportAsPlain`
0x00000008	`GetAppData`
0x00000010	`SetAppData`
0x00000020	`ReduceACL`
0x00000040	`ExpandACL`
0x00000080	`Encrypt`
0x00000100	`Decrypt`
0x00000200	`Verify`
0x00000400	`UseAsBlobKey`
0x00000800	`UseAsKM`
0x00001000	`Sign`
0x00002000	`GetACL`
0x00004000	`UseAsLoaderKey`
0x00008000	`SignModuleCert`

7.13. M_Action

This represents a structure with the following fields:

Field	Size	Format
type	4 bytes	Little-endian unsigned integer. See M_Act
details	variable	Depends on field `type`. See below.

7.13.1. M_Action.details representations

This depends on the value of the `type` field, as follows:

Name of `type`	Value of `type`	Format of `details`
Act_OpPermissions	1	M_Act_OpPermissions_Details
Act_MakeBlob	2	M_Act_MakeBlob_Details

Name of `type`	Value of `type`	Format of `details`
Act_MakeArchiveBlob	3	M_Act_MakeArchiveBlob_Details
Act_DeriveKey	5	M_Act_DeriveKey_Details
Act_DeriveKeyEx	47	M_Act_DeriveKeyEx_Details

Any supported values of `type` not present in the table correspond to an empty (zero-length) `details` field.

7.13.2. M_vec_Action

This represents an array of M_Action objects.

7.14. M_DKMechParams

This represents a structure with the following fields:

Field	Size	Format
mech	4 bytes	Little-endian unsigned integer. See M_DeriveMech
params	variable	Depends on field `mech`. See below.

7.14.1. M_DKMechParams.params representations

All of the supported values of `mech` correspond to an empty (zero-length) `params` field.

7.15. M_DeriveMech

An enumeration type, represented as a 4-byte little-endian unsigned integer. Possible values include:

Value	Name
29	`DeriveMech_PublicFromPrivate`

7.16. M_DeriveRole

An enumeration type, represented as a 4-byte little-endian unsigned integer. Possible values include:

Value	Name
1	`DeriveRole_BaseKey`

7.17. M_FileDeviceFlags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`NVMem`
0x00000002	`PhysToken`
0x00000004	`SoftToken`

7.18. M_KeyHashAndMech

This represents a structure with the following fields:

Field	Size	Format
hash	variable	See M_Hash
mech	4 bytes	Little-endian unsigned integer. See M_Mech

7.19. M_KeyHashExAndMech

This represents a structure with the following fields:

Field	Size	Format
hash	variable	See M_KeyHashEx
mech	4 bytes	Little-endian unsigned integer. See M_Mech

7.20. M_KeyRoleID

This represents a structure with the following fields:

Field	Size	Format
role	4 bytes	Little-endian unsigned integer. See M_DeriveRole

Field	Size	Format
hash	variable	See M_Hash

7.20.1. M_vec_KeyRoleID

This represents an array of M_KeyRoleID objects.

7.21. M_KeyRoleIDEx

This represents a structure with the following fields:

Field	Size	Format
role	4 bytes	Little-endian unsigned integer. See M_DeriveRole
hash	variable	See M_KeyHashEx

7.21.1. M_vec_KeyRoleIDEx

This represents an array of M_KeyRoleIDEx objects.

7.22. M_MakeBlobFilePerms

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_MakeBlobFilePerms_flags
devs	variable	Empty, or M_FileDeviceFlags if `devs_present` (0x00000001) is set in flags)
aclhash	variable	Empty, or M_Hash if `aclhash_present` (0x00000002) is set in flags)

7.23. M_MakeBlobFilePerms_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`devs_present`
0x00000002	`aclhash_present`

7.24. M_NVMemRange

This represents a structure with the following fields:

Field	Size	Format
first	4 bytes	Little-endian unsigned integer. See M_Word
last	4 bytes	Little-endian unsigned integer. See M_Word

7.25. M_PermissionGroup

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_PermissionGroup_flags
n_limits	4 bytes	Little-endian unsigned integer.
limits	variable	*n_limits* copies of `M_UseLimit`. See M_vec_UseLimit
n_actions	4 bytes	Little-endian unsigned integer.
actions	variable	*n_actions* copies of `M_Action`. See M_vec_Action
certifier	variable	Empty, or M_Hash if `certifier_present` (0x00000001) is set in flags)
certmech	variable	Empty, or M_KeyHashAndMech if `certmech_present` (0x00000004) is set in flags)
moduleserial	variable	Empty, or M_ASCIIString if `moduleserial_present` (0x00000008) is set in flags)
certmechex	variable	Empty, or M_KeyHashExAndMech if `certmechex_present` (0x00000040) is set in flags)

7.25.1. M_vec_PermissionGroup

This represents an array of M_PermissionGroup objects.

7.26. M_PermissionGroup_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`certifier_present`
0x00000002	`FreshCerts`
0x00000004	`certmech_present`
0x00000008	`moduleserial_present`
0x00000010	`NSOCertified`
0x00000020	`LogKeyUsage`
0x00000040	`certmechex_present`

7.27. M_TokenParams

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_TokenParams_flags
sharesneeded	4 bytes	Little-endian unsigned integer. See M_Word
sharestotal	4 bytes	Little-endian unsigned integer. See M_Word
timelimit	4 bytes	Little-endian unsigned integer. See M_Word

7.28. M_TokenParams_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. Individual bit values are:

Value	Name
0x00000001	`AllTokensRemovable`
0x00000002	`AllButOneRemovable`
0x00000004	`AllowSoftSlots`

7.29. M_UseLim

An enumeration type, represented as a 4-byte little-endian unsigned integer. Possible values include:

Value	Name
1	`UseLim_Global`
3	`UseLim_Time`
4	`UseLim_NonVolatile`
6	`UseLim_Auth`

7.30. M_UseLim_Auth_Details

This represents a structure with the following fields:

Field	Size	Format
id	variable	See M_Hash
max	4 bytes	Little-endian unsigned integer. See M_Word

7.31. M_UseLim_Global_Details

This represents a structure with the following fields:

Field	Size	Format
id	variable	See M_Hash
max	4 bytes	Little-endian unsigned integer. See M_Word

7.32. M_UseLim_NonVolatile_Details

This represents a structure with the following fields:

Field	Size	Format
flags	4 bytes	Little-endian unsigned integer. See M_UseLim_NonVolatile_Details_flags
file	variable	See M_FileID

Field	Size	Format
range	variable	See M_NVMemRange
maxlo	4 bytes	Little-endian unsigned integer. See M_Word
maxhi	4 bytes	Little-endian unsigned integer. See M_Word
prefetch	4 bytes	Little-endian unsigned integer. See M_Word

7.33. M_UseLim_NonVolatile_Details_flags

A bitmap type, represented as a 4-byte little-endian unsigned integer. No flags are currently defined for this field.

7.34. M_UseLim_Time_Details

This represents a structure with the following fields:

Field	Size	Format
seconds	4 bytes	Little-endian unsigned integer. See M_Word

7.35. M_UseLimit

This represents a structure with the following fields:

Field	Size	Format
type	4 bytes	Little-endian unsigned integer. See M_UseLim
details	variable	Depends on field `type`. See below.

7.35.1. M_UseLimit.details representations

This depends on the value of the `type` field, as follows:

Name of `type`	Value of `type`	Format of `details`
UseLim_Global	1	M_UseLim_Global_Details
UseLim_Time	3	M_UseLim_Time_Details
UseLim_NonVolatile	4	M_UseLim_NonVolatile_Details

Name of `type`	Value of `type`	Format of `details`
UseLim_Auth	6	M_UseLim_Auth_Details

Any supported values of `type` not present in the table correspond to an empty (zero-length) `details` field.

7.35.2. M_vec_UseLimit

This represents an array of M_UseLimit objects.

8. Example

The code below is a basic example of how the wire format of a byte string can be unmarshaled into a Python object. The byte string used is the `modstatesig` field of the first example provided in the Key attestation format example.

```
class M_CipherText:
    def __init__(self, mech, data, iv):
        self.mech = mech
        self.data = data
        self.iv = iv

        known_mechs = {170: "DSAhSHA256", 187: "ECDSAhSHA512"}
        self.mech_name = known_mechs.get(mech, "Undefined")

    def __str__(self):
        description = ""
        description += f"CipherText.mech= {self.mech_name}\n"
        description += f"          .data.r= {self.data.r}\n"
        description += f"               .s= {self.data.s}"
        if self.iv:
            description += f"\n          .iv= {self.iv}"
        return description

class M_Mech_DSA_Cipher:
    def __init__(self, r, s):
        self.r = r
        self.s = s

class M_Mech_ECDSA_Cipher:
    def __init__(self, r, s):
        self.r = r
        self.s = s

class M_Bignum(int):
    def __new__(cls, content):
        return super().__new__(cls, int.from_bytes(content, byteorder='little'))

    def __str__(self):
        return hex(self)

def unmarshal(bytes_, t):
    if t == M_Bignum:
        length = int.from_bytes(bytes_[:4], byteorder='little')
        return t(bytes_[4:4+length]), bytes_[4+length:]
    elif t == M_Mech_DSA_Cipher:
        r, bytes_ = unmarshal(bytes_, M_Bignum)
        s, bytes_ = unmarshal(bytes_, M_Bignum)
        return t(r, s), bytes_
    elif t == M_Mech_ECDSA_Cipher:
        r, bytes_ = unmarshal(bytes_, M_Bignum)
        s, bytes_ = unmarshal(bytes_, M_Bignum)
        return t(r, s), bytes_
    elif t == M_CipherText:
        mech = int.from_bytes(bytes_[:4], byteorder='little')
        bytes_ = bytes_[4:]
        iv = None # none of the defined variants populate iv
        if mech == 170:
            data, bytes_ = unmarshal(bytes_, M_Mech_DSA_Cipher)
        elif mech == 187:
            data, bytes_ = unmarshal(bytes_, M_Mech_ECDSA_Cipher)
        else:
            print(f"Mech {mech} not yet implemented")
```

```
            data = None
        return t(mech, data, iv), bytes_
    else:
        print(f"Type {t.__name__} not yet implemented")
        return None, bytes_

sig = base64.urlsafe_b64decode
("uwAAAEQAAABIF582_Xr16pI3UgQnbJ6BVC_qkiHGmIqnidVnXo2eFxSl5iKKTRCaIbHzB9LADNKwNygbldB7Idbn6BUP4bpayQEAAEQAAACv8Z4
kRqTNrxXdiZTyyBDWRFVY9Tm0Qq2Pm6EHbSgZyCxjH_XpKVZHxlTVnNW75DE7PT_vMKgDgU5joCLtuzrUigAAAA==")
print(unmarshal(sig, M_CipherText)[0])
print("---")
print(nfkm.unmarshal(sig, nfkm.CipherText)[0])
```

```
CipherText.mech= ECDSAhSHA512
          .data.r=
0x1c95abae10f15e8e7d6217bd0951b2837b0d20cc0d207f3b1219a104d8a22e6a514179e8d5e67d589a78a98c62192ea2f54819e6c270452
3792eaf57afd369f1748
                .s=
0x8ad43abbed22a0634e8103a830ef3f3d3b31e4bbd59cd554c6475629e9f51f632cc819286d07a19b8fad42b439f5585544d610c8f29489d
d15afcda446249ef1af
---
CipherText.mech= ECDSAhSHA512
          .data.r=
0x1c95abae10f15e8e7d6217bd0951b2837b0d20cc0d207f3b1219a104d8a22e6a514179e8d5e67d589a78a98c62192ea2f54819e6c270452
3792eaf57afd369f1748
                .s=
0x8ad43abbed22a0634e8103a830ef3f3d3b31e4bbd59cd554c6475629e9f51f632cc819286d07a19b8fad42b439f5585544d610c8f29489d
d15afcda446249ef1af
```