TRUSTED®
COMPUTING
GROUP

REFERENCE

TCG Guide for TPM Library Specification Changes Revision 1.38 to 1.59

Version 1.4
March 20, 2020

Contact:
admin@trustedcomputinggroup.org

PUBLISHED

DISCLAIMERS, NOTICES, AND LICENSE TERMS

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, DOCUMENT OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this document and to the implementation of this document, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this document or any information herein.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on document licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

CONTENTS

1 SCOPE

This document describes changes in the TCG Trusted Platform Module Library specification, family 2.0 (in the following called “Library specification”) revision 1.59 compared to the prior published revision 1.38.

This document does not describe changes of Library specification revision 1.59 compared to the published revision 1.16.

2 References

[Lib_138] TCG Trusted Platform Module Library, Family 2.0, Revision 1.38
https://trustedcomputinggroup.org/resource/tpm-library-specification/

[Lib_159] TCG Trusted Platform Module Library, Family 2.0, Revision 1.59
https://trustedcomputinggroup.org/resource/tpm-library-specification/

[Err] TCG Errata for TCG Trusted Platform Module Library, Family 2.0, Revision 1.38
https://trustedcomputinggroup.org/resource/errata-for-tpm-library-specification-2-0/

[Alg] TCG Algorithm Registry
https://trustedcomputinggroup.org/resource/tcg-algorithm-registry/

[PTP] TCG PC Client Platform TPM Profile (PTP) Specification
https://trustedcomputinggroup.org/resource/pc-client-platform-tpm-profile-ptp-specification/

3 Introduction

The purpose of this document is to guide TPM, TSS, and other software developers when migrating from Library specification revision 1.38 [Lib_138] to revision 1.59 [Lib_159], and to point out the potential impact on software when switching from a TPM implementing revision 1.38 to a TPM implementing revision 1.59. Further, this document provides references to the relevant clauses in Library specification revision 1.59 that have been modified.

Section 4 describes new commands and features of Library specification revision 1.59. Not all new commands/features might be implemented by a TPM. The commands/features that are actually implemented by a TPM are defined by TCG platform-specific TPM profile specifications, such as the TCG PC Client Platform TPM Profile Specification [PTP].

Section 4.4 describes commands and structures which have been modified in Library specification revision 1.59. The old behavior of revision 1.38 is compared to the new behavior of revision 1.59, and the potential impact on software is outlined for each change.

Section 5.7 describes definitions which have been modified in Library specification revision 1.59 that may impact interoperability.

Section 7 lists selected errata of Library specification revision 1.38 where a recommendation is made to software developers to reduce the potential impact of each erratum. The current list of all known errata is defined in [Err].

4 New Commands/ Features

4.1 TPM2_CertifyX509

4.1.1 Description

A new attestation command TPM2_CertifyX509() was introduced in Library specification revision 1.59 that enables the TPM to create an X.509 certificate for a loaded TPM key. For details, see the following clauses of [Lib_159].

4.1.2 Specification References

- New command TPM2_CertifyX509(), see
  - Part 1, 31.6 X.509 Certificate Signing
  - Part 3, 18.8 TPM2_CertifyX509
- New object attribute *x509sign*, see
  - Part 2, 8.3.3.13 Bit[19] – *x509sign*
- New attribute structure TPMA_X509_KEY_USAGE used in TPM2_CertifyX509(), see
  - Part 2, 8.11 TPMA_X509_KEY_USAGE

4.2 Attached Components

4.2.1 Description

The Attached Components (AC) commands introduced in Library specification revision 1.59 enable the access control mechanisms of a TPM to be used to manage sensitive data for other components that may be attached to the TPM.

The Library specification only defines the AC command APIs, it does not provide reference code for the functionality of the AC commands. The implementation of the AC functionality is vendor-specific.

For details, see the following clauses of [Lib_159].

4.2.2 Specification References

- New commands TPM2_AC_GetCapability(), TPM2_AC_Send(), TPM2_Policy_AC_SendSelect(), see
  - Part 1, 42 Attached Components
  - Part 3, 32.2 TPM2_AC_GetCapability
  - Part 3, 32.3 TPM2_AC_Send
  - Part 3, 32.4 TPM2_Policy_AC_SendSelect
- New interface type TPMI_RH_AC used in TPM2_AC_GetCapability() and TPM2_AC_Send(), see
  - Part 2, 9.25 TPMI_RH_AC
- Attached Component Structures, see
  - Part 2, 16.1 TPM_AT
  - Part 2, 16.2 TPM_AE
  - Part 2, 16.3 TPMS_AC_OUTPUT
  - Part 2, 16.4 TPML_AC_CAPABILITIES
- New handle type TPM_HT_AC, see
  - Part 2, 7.2 TPM_HT (Handle Types)
- New handle value constants, see
  - Part 2, 7.5 TPM_HC (Handle Value Constants)

4.3 TPM2_MAC, TPM2_MAC_Start

4.3.1 Description

Library specification revision 1.59 introduces two generic MAC (Message Authentication Code) commands, which enable the TPM to perform a block cipher MAC on externally supplied data: TPM2_MAC() and TPM2_MAC_Start(). Both commands may perform either HMAC or a block cipher MAC (such as CMAC). The benefit of these generic

MAC commands is that no additional command is needed to support block cipher MACs as the signing scheme is selected by the *inScheme* input parameter.

In revision 1.38, the only MAC, which was supported at the TPM interface, was HMAC using TPM2_HMAC() and TPM2_HMAC_Start().

In revision 1.59, the generic MAC commands may be used instead of TPM2_HMAC() and TPM2_HMAC_Start() as they have the same command code.

The difference between TPM2_HMAC()/TPM2_HMAC_Start() and TPM2_MAC()/TPM2_MAC_Start() is the interface type of the parameter selecting the algorithm (*hashAlg* and *inScheme*).

- For TPM2_MAC() and TPM2_MAC_Start(), a new interface type TPMI_ALG_MAC_SCHEME has been added to allow ALG_CMAC_VALUE to be selected in addition to the hash algorithms. (At time of writing, CMAC is defined as the only supported block cipher mode in the TCG Algorithm Registry [Alg].) If CMAC is not supported by the TPM, then TPM2_MAC() and TPM2_MAC_Start() will return TPM_RC_SYMMETRIC.
- TPM2_HMAC()/TPM2_HMAC_Start() only allow selection of a hash algorithm.

In order to allow a symmetric (SYMCIPHER) signing key with CMAC as *mode* to be created/loaded, the interface type TPMI_ALG_SYM_MODE, which defines the allowed block-cipher modes, has been augmented to include ALG_CMAC_VALUE as *mode*. If CMAC is not supported by the TPM but selected as symmetric mode in the public structure, then the following commands will return TPM_RC_MODE: TPM2_Create(), TPM2_CreatePrimary(), TPM2_CreateLoaded(), TPM2_Load(), TPM2_LoadExternal(), TPM2_Import().

4.3.2 Related Change

In Library specification revision 1.38, the interface type TPMI_ALG_SYM_MODE was also used for the *mode* parameter of TPM2_EncryptDecrypt() and TPM2_EncryptDecrypt2(). TPM2_EncryptDecrypt()/TPM2_EncryptDecrypt2() may only perform symmetric encryption or decryption, but not signing. Therefore in revision 1.59, the interface type of *mode* has been changed for these commands: see section 5.3 TPMI_ALG_CIPHER_MODE – TPM2_EncryptDecrypt/2 (of this document).

4.3.3 Specification References

- New commands TPM2_MAC() and TPM2_MAC_Start(), see
  - Part 1, 11.4.2 Symmetric Block Cipher MAC Algorithms
  - Part 3, 15.6 TPM2_MAC
  - Part 3, 17.3 TPM2_MAC_Start
- New interface type TPMI_ALG_MAC_SCHEME used in TPM2_MAC() and TPM2_MAC_Start(), see
  - Part 2, 9.36 TPMI_ALG_MAC_SCHEME
- Extended interface type TPMI_ALG_SYM_MODE to include ALG_CMAC_VALUE, see
  - Part 2, 9.31 TPMI_ALG_SYM_MODE

4.4 Authenticated Countdown Timer

4.4.1 Description

A new command TPM2_ACT_SetTimeout() was introduced in Library specification revision 1.59 that enables the TPM to manage multiple authenticated countdown timers. For details, see the following clauses of [Lib_159].

4.4.2 Specification References

- New command TPM2_ACT_SetTimeout(), see
  - Part 1, 43 Authenticated Countdown Timer (ACT)
  - Part 3, 33.2 TPM2_ACT_SetTimeout
- New attribute structure TPMA_ACT used in TPM2_GetCapability(), see
  - Part 2, 8.12 TPMA_ACT

- New interface type TPMI_RH_HIERARCHY_POLICY used in TPM2_SetPrimaryPolicy(), see
    - Part 2, 9.16 TPMI_RH_HIERARCHY_POLICY
    - Section 5.7 TPMI_RH_HIERARCHY_POLICY – TPM2_SetPrimaryPolicy (of this document)
- New interface type TPMI_RH_ACT used in TPM2_ACT_SetTimeout(), see
    - Part 2, 9.26 TPMI_RH_ACT
- New property structure TPMS_ACT_DATA used in TPM2_GetCapability(), see
    - Part 2, 10.8.5 TPMS_ACT_DATA
- New TPM capability TPM_CAP_ACT and list TPML_ACT_DATA used in TPM2_GetCapability(), see
    - Part 2, 10.9.13 TPML_ACT_DATA
    - Part 2, 10.10.1 TPMU_CAPABILITIES
    - Part 3, 30.2 TPM2_GetCapability

5 Modified Commands and Structures

5.1 TPM2_NV_Certify

5.1.1 Description

The purpose of this change is to enable TPM2_NV_Certify() to certify the digest of the contents of an NV Index. If the data size of the NV Index to be certified is very large (e.g. 1024 bytes), a single response of TPM2_NV_Certify() as specified in Library specification revision 1.38 may be larger than software can receive/process. This change avoids software having to certify the data content in smaller chunks using multiple TPM2_NV_Certify() commands.

5.1.2 Old Behavior

In Library specification revision 1.38, the command TPM2_NV_Certify() certified the contents of an NV Index or a portion of an NV Index.

If the input parameters *size* and *offset* were both zero, this command would sign the attestation structure with the NV content set to Empty Buffer. In practice, this had no particular value.

5.1.3 New Behavior

In Library specification revision 1.59, this combination of *size* and *offset* both set to zero certifies the digest of the contents of an NV Index.

A new structure tag (TPM_ST_ATTEST_NV_DIGEST), and a new attestation structure (TPMS_NV_DIGEST_CERTIFY_INFO) have been added.

If either *size* or *offset* are non-zero, then TPM2_NV_Certify() certifies the contents of an NV Index or a portion of an NV Index. This is the same behavior as in revision 1.38. In this case,

- *certifyInfo*.*type* in the response will be set to TPM_ST_ATTEST_NV, and
- *certifyInfo.attested* will contain a TPMS_NV_CERTIFY_INFO.

If *size* and *offset* are both zero, then TPM2_NV_Certify() will certify the digest of the contents of an NV Index. This is a different behavior as in revision 1.38. In this case,

- *certifyInfo*.*type* in the response will be set to TPM_ST_ATTEST_NV_DIGEST, and
- *certifyInfo.attested* will contain a TPMS_NV_DIGEST_CERTIFY_INFO.

The digest in the TPMS_NV_DIGEST_CERTIFY_INFO is created using the digest of the selected signing scheme.

Affected commands: TPM2_NV_Certify()

5.1.4 Impact on Software

This change should have no impact on software using the TPM as the ability to certify zero bytes of an NV Index data is useless. It is almost certain that this functionality was not used in practice.

5.1.5 Specification References

- Extended command functionality of TPM2_NV_Certify(), see
  - Part 3, 31.16 TPM2_NV_Certify
- New structure tag TPM_ST_ATTEST_NV_DIGEST, see
  - Part 2, 6.9 TPM_ST (Structure Tags)
- Extended TPMI_ST_ATTEST to include TPM_ST_ATTEST_NV_DIGEST, see
  - Part 2, 10.12.10 TPMI_ST_ATTEST
- New attestation structure TPMS_NV_DIGEST_CERTIFY_INFO used by TPM2_NV_Certify(), see
  - Part 2, 10.12.9 TPMS_NV_DIGEST_CERTIFY_INFO
- Extended TPMU_ATTEST to include TPMS_NV_DIGEST_CERTIFY_INFO, see
  - Part 2, 10.12.11 TPMU_ATTEST

5.2 TPM2B_PRIVATE_KEY_RSA – TPM2_Create, TPM2_Load

5.2.1 Description

The purpose of this change is faster command operation of TPM2_Load() in the case of an RSA key.

5.2.2 Old Behavior

In Library specification revision 1.38, when an RSA key was created with TPM2_Create(), the private structure (TPM2B_PRIVATE) returned by TPM2_Create() contained only one CRT prime (P). When an RSA key was loaded with TPM2_Load(), TPM2_Load() had to recalculate the remaining four CRT primes (Q, dP, dQ, QInv) from the prime P and the public modulus N, which consumed time.

The limitation of one prime was caused by the RSA private key structure (TPM2B_PRIVATE_KEY_RSA), whose maximum buffer size was limited to MAX_RSA_KEY_BYTES/2 (128 bytes for RSA 2k).

5.2.3 New Behavior

In Library specification revision 1.59, to optimize the performance of the TPM2_Load() command, the private structure (TPM2B_PRIVATE) output by TPM2_Create() may contain all five CRT primes (P, Q, dP, dQ, QInv) for RSA keys that have *fixedTPM* parents. This saves the recalculation step when loading the RSA key.

To maintain compatibility, RSA keys that do not have *fixedTPM* parents (implying that the public and private structure can be loaded directly in a TPM that is different from the TPM that generated the RSA key) are returned in the same (one-prime) format by TPM2_Create() as in revision 1.38. The private structure returned by TPM2_Duplicate() always contains the RSA key in the one-prime format.

The maximum buffer size of the RSA private key structure (TPM2B_PRIVATE_KEY_RSA) has been increased to RSA_PRIVATE_SIZE (640 bytes for RSA 2k).

Affected commands: TPM2_Create(), TPM2_Load()

5.2.4 Impact on Software

Software using the TPM must increase the buffer size for the RSA private key structure (TPM2B_PRIVATE_KEY_RSA) in the same way as the reference code. Since the new (five-primes) format is only applied in TPM2_Create() to RSA keys that have *fixedTPM* parents, this change is backwards compatible. If allowed by their attribute setting, RSA keys can be created on a TPM implementing revision 1.59, and be loaded on a TPM implementing an older revision, and the other way around.

5.2.5 Specification References

- Modified RSA private key structure TPM2B_PRIVATE_KEY_RSA, see
  - Part 2, 11.2.4.7 TPM2B_PRIVATE_KEY_RSA

5.3 TPMI_ALG_CIPHER_MODE – TPM2_EncryptDecrypt/2

5.3.1 Description

This change is necessary to enable the changes to the MAC commands described in section 4.3 TPM2_MAC, TPM2_MAC_Start (of this document).

5.3.2 Old Behavior

In Library specification revision 1.38, the interface type of the *mode* parameter of TPM2_EncryptDecrypt() and TPM2_EncryptDecrypt2() was defined as TPMI_ALG_SYM_MODE, which allowed selection of a symmetric encryption/decryption mode.

5.3.3 New Behavior

In Library specification revision 1.59, the interface type of the *mode* parameter of TPM2_EncryptDecrypt() and TPM2_EncryptDecrypt2() has been changed to TPMI_ALG_CIPHER_MODE. The new interface type TPMI_ALG_CIPHER_MODE equals the previous definition of TPMI_ALG_SYM_MODE in revision 1.38 and is therefore backwards compatible.

Affected commands: TPM2_EncryptDecrypt(), TPM2_EncryptDecrypt2()

5.3.4 Impact on Software

Software using the TPM must change the interface type of the *mode* parameter in the same way as the reference code. As TPMI_ALG_CIPHER_MODE is identical to the previous definition of TPMI_ALG_SYM_MODE in revision 1.38, there is no impact on compatibility.

5.3.5 Specification References

- New interface type TPMI_ALG_CIPHER_MODE, see
  - Part 2, 9.37 TPMI_ALG_CIPHER_MODE
- Modified commands TPM2_EncryptDecrypt(), TPM2_EncryptDecrypt2(), see
  - Part 3, 15.2 TPM2_EncryptDecrypt
  - Part 3, 15.3 TPM2_EncryptDecrypt2

5.4 TPMS_SCHEME_XOR

5.4.1 Description

The definition of TPMS_SCHEME_XOR has been changed to be consistent with TPMS_SCHEME_HASH. TPMS_SCHEME_HASH does not allow a TPM_ALG_NULL for the hash selection. The same restriction has been applied to TPMS_SCHEME_XOR. With this change, neither a KeyedHash object with the *sign* attribute SET nor a KeyedHash object with the *decrypt* attribute SET is allowed to have TPM_ALG_NULL as its hash algorithm.

5.4.2 Old Behavior

In Library specification revision 1.38, an XOR object can have a NULL hash algorithm. This means that the object can have a zero-length key.

5.4.3 New Behavior

As an XOR object with a zero-length key is not useful, Library specification revision 1.59 requires an XOR object to have a valid hash scheme which is not NULL, otherwise it returns TPM_RC_HASH.

Affected commands: All commands that have a public structure as input: TPM2_Create(), TPM2_CreatePrimary(), TPM2_CreateLoaded(), TPM2_Load(), TPM2_LoadExternal(), TPM2_Import().

5.4.4 Impact on Software

This change should have no impact on software using the TPM as the ability to create an XOR object with a zero-length key is useless. It is almost certain that this functionality was not used in practice.

5.4.5 Specification References

- Updated definition of TPMS_SCHEME_XOR, see
  - Part 2, 11.1.21 TPMS_SCHEME_XOR

5.5 TPMI_ALG_KEYEDHASH_SCHEME

5.5.1 Description

This structure has been renamed to better describe its functionality. In Library specification revision 1.59, the interface type TPMI_ALG_HASH_SCHEME has been renamed to TPMI_ALG_KEYEDHASH_SCHEME. Otherwise, the scheme's functionality is unchanged.

TPMI_ALG_HASH_SCHEME / TPMI_ALG_KEYEDHASH_SCHEME is used to marshal the scheme of a KEYEDHASH objects. This change has no impact on software.

5.5.2 Specification References

- Definition of TPMI_ALG_KEYEDHASH_SCHEME, see
  - Part 2, 11.1.19 TPMI_ALG_KEYEDHASH_SCHEME

5.6 MAX_SYM_DATA

5.6.1 Description

This definition has been changed to clarify that this value is implementation dependent.

Library specification revision 1.38 defined that MAX_SYM_DATA shall be 128.

In revision 1.59, it has been clarified that MAX_SYM_DATA is implementation dependent, but for interoperability it should be 128. Therefore, it should have no impact on software.

5.6.2 Specification References

- Updated definition of TPMU_SENSITIVE_CREATE, see
  - Part 2, 11.1.13 TPMU_SENSITIVE_CREATE
  - Part 4, A.2 TpmProfile.h

5.7 TPMI_RH_HIERARCHY_POLICY – TPM2_SetPrimaryPolicy

5.7.1 Description

This change is necessary to enable the *authPolicy* for an ACT (see section 4.4 Authenticated Countdown Timer) to be changed by TPM2_SetPrimaryPolicy().

5.7.2 Old Behavior

In Library specification revision 1.38, the interface type of the authorization handle (*authHandle*) of TPM2_SetPrimaryPolicy() was defined as TPMI_RH_HIERARCHY_AUTH, which allowed selection of the following hierarchy handles: TPM_RH_LOCKOUT, TPM_RH_ENDORSEMENT, TPM_RH_OWNER or TPM_RH_PLATFORM.

5.7.3 New Behavior

In Library specification revision 1.59, the interface type of the authorization handle (*authHandle)* of TPM2_SetPrimaryPolicy() has been changed to TPMI_RH_HIERARCHY_POLICY. The new interface type allows selection of the same hierarchy handles as TPMI_RH_HIERARCHY_AUTH, plus the selection of the ACT handles (TPM_RH_ACT_0 – TPM_RH_ACT_F).

Affected commands: TPM2_SetPrimaryPolicy

5.7.4 Impact on Software

Software using the TPM ACT feature must change the interface type of the authorization handle (*authHandle)* in the same way as the reference code. The change does not impact backward compatibility.

5.7.5 Specification References

- New interface type TPMI_RH_HIERARCHY_POLICY, see
  - Part 2, 9.16 TPMI_RH_HIERARCHY_POLICY
- Modified command TPM2_SetPrimaryPolicy(), see
  - Part 3, 24.3 TPM2_SetPrimaryPolicy

6 Modified Definitions

6.1 Label – RSA OAEP, KDFa

6.1.1 Description

The definition of label has been changed to allow software to use a hash digest as label.

Library specification revision 1.38 defines the label used in RSA_OAEP encryption/decryption and in KDFa() as a NULL-terminated string, which is a sequence of non-zero values followed by a value containing zero.

Library specification revision 1.59 changes the definition of label used in RSA_OEAP encryption/decryption to an octet string with a final octet that is zero.

Revision 1.59 changes the definition of label used in KDFa() to an octet string. In this case, the final octet is not required to be zero. The Library specification clarifies the use of the separation indicator for KDFa():

1) If *Label* is not present, a zero octet is added.
2) If *Label* is present and the last octet is not zero, a zero octet is added.
3) If *Label* is present and the last octet is zero, no zero octet is added.

This change affects the label provided in TPM2_RSA_Encrypt(), TPM2_RSA_Decrypt(), and TPM2_CreateLoaded() if a Derived Object is created.

Affected commands: TPM2_RSA_Encrypt(), TPM2_RSA_Decrypt(), TPM2_CreateLoaded()

6.1.2 Impact on Software

If software needs to be interoperable with revision 1.38 (and older) TPM implementations, software must use a *Label* that is a NULL-terminated string. Otherwise, a message encrypted with a TPM implementing an older revision might not be decrypted with a TPM implementing revision 1.59, and the other way around. Also Derived Objects derived from TPMs implementing different Library specification revisions might not be compatible.

6.1.3 Specification References

- Updated definition of label, see
  - Part 1, 11.4.10.2 KDFa()
  - Part 1, B.4 RSAES_OAEP
  - Part 3, 14.2 TPM2_RSA_Encrypt and 14.3 TPM2_RSA_Decrypt
- Corresponding Errata, see
  - Errata, 2.11 Separation Indicator 0x00 in KDFa
  - Errata, 2.29 Label in TPM2_RSA_Encrypt/Decrypt and TPM2_CreateLoaded

7 Selected Errata

7.1 Introduction

This section points out some errata where a recommendation is made to software developers to reduce the potential impact of an erratum. For all known errata, see [Err]. The errata version that is implemented by a TPM is reported in the TPM properties TPM_PT_DAY_OF_YEAR and TPM_PT_YEAR.

7.2 TPM2_DictionaryAttackParameters

7.2.1 Description

An erratum of Library specification revision 1.38 corrected that TPM2_DictionaryAttackParameters() no longer resets *failedTries*.

It is recommended that software reads the current value of *failedTries* before setting new DA parameters in order to avoid accidental lockout. Software should use TPM2_DictionaryAttackLockReset() to reset *failedTries*.

7.2.2 Specification References

- Errata, 2.5 TPM2_DictionaryAttackParameters - *failedTries*
- Part 3, 25.3 TPM2_DictionaryAttackParameters

7.3 ECC Point Padding

7.3.1 Description

An erratum of Library specification revision 1.38 corrected that ECC points loaded with TPM2_LoadExternal() need to be padded with leading zero octets to the size of the curve.

It is recommended that software pads all ECC points input to the TPM.

7.3.2 Specification References

- Errata, 2.30 TPM2_LoadExternal – ECC Point Padding
- Part 1, C.8 ECC Point Padding

7.4 Data Object Size

7.4.1 Description

An erratum of Library specification revision 1.38 corrected that the size of a Data object is allowed to be up to MAX_SYM_DATA.

If a data object needs to be loaded/imported in a TPM implementing revision 1.38, it is recommended that the size of a data object is not larger than the size of the digest produced by the *nameAlg* of the object.

7.4.2 Specification References

- Errata, 2.31 Max Size Check of Data Object