TRUSTED® COMPUTING GROUP

ERRATA

Errata for TCG Trusted Platform Module Library

Family “2.0”
Level 00 Revision 01.83
January 25, 2024

Version 3
August 11, 2026

Contact: admin@trustedcomputinggroup.org

Published

DISCLAIMERS, NOTICES, AND LICENSE TERMS

CHANGE HISTORY

VERSION	DATE	DESCRIPTION
2	June 10, 2025	Added • 2.2 CryptHmacSign [code] • 3.1.1 TPM2_NV_Certify with signHandle = NULL [specification text, code] • 3.1.2 TPM2_Quote with signHandle = NULL [specification text, code]
3	May 29, 2025	Added • 2.3 Sequence object • 2.4 TPM2_ActivateCredential label • 2.5 RSA-OEAP decode • 2.6 ObjectFlush

CONTENTS

1 Introduction

This document describes errata and clarifications for the TCG Trusted Platform Module Library Family “2.0” Level 00 Revision 01.83 as published. The information in this document is likely – but not certain – to be incorporated into a future version of the specification. Suggested fixes proposed in this document may be modified before being published in a later TCG Specification. Therefore, the contents of this document are not normative and only become normative when included in an updated version of the published specification. Note that since the errata in this document are non-normative, the patent licensing rights granted by Section 16.4 of the Bylaws do not apply.

The heading of each errata in section 2 indicates whether the errata affects the specification text or the reference code implementation. This is indicated by the word “specification text” or “code” in square brackets (“[]”).

2 Errata

2.1 TPM_SPEC Date Constants [specification text, code]

Table 6 in Part 2, 6.1 TPM_SPEC (Specification Version Values) should be replaced with:

Table 6 — Definition of (UINT32) TPM_SPEC Constants <>

Name	Value	Comments
TPM_SPEC_FAMILY	0x322E3000	ASCII “2.0” with null terminator
TPM_SPEC_LEVEL	00	the level number for the specification
TPM_SPEC_VERSION	183	the version number of the spec (001.83 * 100)
TPM_SPEC_YEAR	2026	the year of the version
TPM_SPEC_DAY_OF_YEAR	32	the day of the year (February 1)

That is, the spec date fields TPM_SPEC_YEAR and TPM_SPEC_DAY_OF_YEAR should be set to the date of the most recent Errata document that revised the behavior of the TPM.

NOTE The date in Table 6 reflects an internal publication date for Errata v3.

2.2 CryptHmacSign [code]

The function CryptHmacSign() in the reference code in Part 4, 7.151 does not check that the TPMT_SIGNATURE provided in the *signature* parameter has the expected *sigAlg* selector value. To fix the issue, the function will check that *signature*'s *sigAlg* is TPM_ALG_HMAC or otherwise the TPM will return TPM_RC_SCHEME.

2.3 Sequence object [code]

The reference code for Part 3 does not check if certain handle variables reference a sequence object. The affected cases are:

- *activateHandle* in TPM2_ActivateCredential(),
- *objectHandle* in TPM2_Certify() and TPM2_CertifyCreation(), and
- *sendObject* in TPM2_AC_Send().

To fix the issue, the commands will check that the object referenced by these handles is not a (hash, MAC, or Event) sequence object or otherwise the TPM will return TPM_RC_TYPE or TPM_RC_SEQUENCE.

A minimal patch that adds these checks in the reference code is:

```
--- a/TPMCmd/tpm/src/command/AttachedComponent/AC_Send.c
+++ b/TPMCmd/tpm/src/command/AttachedComponent/AC_Send.c
@@ -52,6 +52,10 @@ TPM2_AC_Send(AC_Send_In*  in,  // IN: input parameter list
     // permanent handle.
     else if(HandleGetType(in->authHandle) != TPM_HT_PERMANENT)
         return TPM_RCS_HANDLE + RC_AC_Send_authHandle;
+
+    // Cannot send a sequence object to an attached component
+    if(ObjectIsSequence(object))
+        return TPM_RCS_TYPE + RC_AC_Send_sendObject;
     // Make sure that the object to be duplicated has the right attributes
     if(IS_ATTRIBUTE(
            object->publicArea.objectAttributes, TPMA_OBJECT, encryptedDuplication)

--- a/TPMCmd/tpm/src/command/Attestation/Certify.c
+++ b/TPMCmd/tpm/src/command/Attestation/Certify.c
@@ -28,6 +28,10 @@ TPM2_Certify(Certify_In*  in,  // IN: input parameter list
     if(!CryptSelectSignScheme(signObject, &in->inScheme))
```



```
          return TPM_RCS_SCHEME + RC_Certify_inScheme;

+     // Cannot certify a sequence object
+     if(ObjectIsSequence(certifiedObject))
+         return TPM_RCS_TYPE + RC_Certify_objectHandle;
+
      // Command Output
      // Filling in attest information
      // Common fields

--- a/TPMCmd/tpm/src/command/Attestation/CertifyCreation.c
+++ b/TPMCmd/tpm/src/command/Attestation/CertifyCreation.c
@@ -26,6 +26,10 @@ TPM2_CertifyCreation(CertifyCreation_In*  in,  // IN: input parameter list
      OBJECT*          certified  = HandleToObject(in->objectHandle);
      OBJECT*          signObject = HandleToObject(in->signHandle);
      // Input Validation
+     // Cannot certify creation of a sequence object
+     if(ObjectIsSequence(certified))
+         return TPM_RCS_TYPE + RC_CertifyCreation_objectHandle;
+
      if(!IsSigningObject(signObject))
          return TPM_RCS_KEY + RC_CertifyCreation_signHandle;
      if(!CryptSelectSignScheme(signObject, &in->inScheme))

--- a/TPMCmd/tpm/src/command/Object/ActivateCredential.c
+++ b/TPMCmd/tpm/src/command/Object/ActivateCredential.c
@@ -43,6 +43,10 @@ TPM2_ActivateCredential(ActivateCredential_In*  in,  // IN: input parameter list
         || !IS_ATTRIBUTE(object->publicArea.objectAttributes, TPMA_OBJECT, restricted))
          return TPM_RCS_TYPE + RC_ActivateCredential_keyHandle;

+     // Cannot activate credential of sequence object
+     if(ObjectIsSequence(activateObject))
+         return TPM_RCS_TYPE + RC_ActivateCredential_activateHandle;
+
      // Command output

      // Decrypt input credential data via asymmetric decryption.  A
--
```

2.4 TPM2_ActivateCredential label

TPM2_ActivateCredential() may support the label "IDENTITY2" in addition to "IDENTITY" as defined in Part 1, B.10.4 and C.6.4 to decrypt the credential blob to identify that a TPM firmware version has fixed the errata documented in section 2.3 Sequence object. The use of “IDENTITY2” is not intended to be a general change to TPM2_ActivateCredential and is not expected to be added as an alternative in future versions of the TPM Specification.

2.5 RSA-OAEP decode [code]

The reference code uses a timing-dependent implementation for RSA-OAEP decryption. RSA private key operations (including padding) should be provided by an implementation's crypto library. Cryptographic libraries are expected to take special care to avoid timing side-channels like this. A minimal patch that significantly reduces (but does not totally eliminate) secret-dependent execution time in OAEP decryption in the reference code is:

```
--- a/TPMCmd/tpm/src/crypt/CryptRsa.c
+++ b/TPMCmd/tpm/src/crypt/CryptRsa.c
@@ -381,8 +381,7 @@ static TPM_RC OaepDecode(
      TPM_RC retVal = TPM_RC_SUCCESS;

      // Strange size (anything smaller can't be an OAEP padded block)
-     // Also check for no leading 0
-     if((padded->size < (unsigned)((2 * hLen) + 2)) || (padded->buffer[0] != 0))
+     if(padded->size < (unsigned)((2 * hLen) + 2))
          ERROR_EXIT(TPM_RC_VALUE);
      // Use the hash size to determine what to put through MGF1 in order
      // to recover the seedMask
@@ -414,7 +413,11 @@ static TPM_RC OaepDecode(
```



```
      if((CryptHashBlock(hashAlg, label->size, (BYTE*)label->buffer, hLen, seedMask))
         != hLen)
          FAIL(FATAL_ERROR_INTERNAL);
 -    if(memcmp(seedMask, mask, hLen) != 0)
 +    if(!MemoryEqual(seedMask, mask, hLen))
 +        ERROR_EXIT(TPM_RC_VALUE);
 +
 +    // Check for the leading 0x00 byte.
 +    if(padded->buffer[0] != 0)
          ERROR_EXIT(TPM_RC_VALUE);

      // find the start of the data
 --
```

2.6 Object Flush [code]

The reference code only marks the occupied flag in an OBJECT structure during ObjectFlush. The implementation should entirely clear the structure to ensure no potentially sensitive information remains. A patch for the reference code is below.

```
--- a/TPMCmd/tpm/src/subsystem/Object.c
+++ b/TPMCmd/tpm/src/subsystem/Object.c
@@ -16,6 +16,7 @@
 // Note: This could be converted to a macro.
 void ObjectFlush(OBJECT* object)
 {
+    MemorySet(object, 0, sizeof(*object));
     object->attributes.occupied = CLEAR;
 }
```


3 Clarifications

3.1 Digest computation when signHandle is NULL

3.1.1 TPM2_NV_Certify with signHandle = NULL [specification text, code]

If TPM2_NV_Certify is used to certify the hash digest of the NV Index data content (i.e., *size* and *offset* are both zero) and *signHandle* is TPM_RH_NULL, then this command can either:

- Option 1: If the signing scheme or the scheme hash algorithm is not TPM_ALG_NULL, compute *nvDigest* using the hash algorithm of the selected scheme. Otherwise, return TPM_RC_SCHEME.
- Option 2: Set *nvDigest* in the TPMS_DIGEST_CERTIFY_INFO structure to Empty Buffer.

The reference code sets *nvDigest* to Empty Buffer (option 2), while the description in Part 3, section 18.1 Introduction (of Attestation Commands) indicates that all of the actions of the command are performed and only the signature is absent (option 1).

A future TPM Library specification version may only allow option 1.

3.1.2 TPM2_Quote with signHandle = NULL [specification text, code]

According to Part 3, section 18.4 (NOTE 2 and 3) TPM2_Quote allows two different behaviors if *signHandle* set to TPM_RH_NULL:

- Option 1: If the signing scheme or the scheme hash algorithm is not TPM_ALG_NULL, compute pcr*Digest* using the hash algorithm of the selected scheme (and return the TPMS_ATTEST structure). Otherwise, return TPM_RC_SCHEME.
- Option 2: Optionally, always return TPM_RC_SCHEME.

The reference code implements option 2.

A future TPM Library specification version may only allow option 1.

