
**Information technology — Trusted
Platform Module —**

**Part 4:
Commands**

*Technologies de l'information — Module de plate-forme de confiance —
Partie 4: Commandes*



Reference number
ISO/IEC 11889-4:2009(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2009

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2009

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Table of Contents

1. Scope	1
1.1 Key words	1
1.2 Statement Type	1
2. Normative references	2
3. Abbreviated Terms	3
4. Admin Startup and State	5
4.1 TPM_Init	5
4.2 TPM_Startup	6
4.3 TPM_SaveState	8
5. Admin Testing	10
5.1 TPM_SelfTestFull	10
5.2 TPM_ContinueSelfTest	10
5.3 TPM_GetTestResult	12
6. Admin Opt-in	13
6.1 TPM_SetOwnerInstall	13
6.2 TPM_OwnerSetDisable	13
6.3 TPM_PhysicalEnable	14
6.4 TPM_PhysicalDisable	15
6.5 TPM_PhysicalSetDeactivated	15
6.6 TPM_SetTempDeactivated	16
6.7 TPM_SetOperatorAuth	17
7. Admin Ownership	18
7.1 TPM_TakeOwnership	18
7.2 TPM_OwnerClear	20
7.3 TPM_ForceClear	22
7.4 TPM_DisableOwnerClear	23
7.5 TPM_DisableForceClear	24
7.6 TSC_PhysicalPresence	24
7.7 TSC_ResetEstablishmentBit	26
8. The Capability Commands	28
8.1 TPM_GetCapability	28
8.2 TPM_SetCapability	29
8.3 TPM_GetCapabilityOwner	30
9. Auditing	32
9.1 Audit Generation	32
9.2 Effect of audit failing	33
9.3 TPM_GetAuditDigest	34

9.4	TPM_GetAuditDigestSigned	35
9.5	TPM_SetOrdinalAuditStatus	37
10.	Administrative Functions - Management	38
10.1	TPM_FieldUpgrade	38
10.2	TPM_SetRedirection	40
10.3	TPM_ResetLockValue	41
11.	Storage functions	43
11.1	TPM_Seal	43
11.2	TPM_Unseal	46
11.3	TPM_UnBind	49
11.4	TPM_CreateWrapKey	51
11.5	TPM_LoadKey2	53
11.6	TPM_GetPubKey	56
11.7	TPM_Sealx	57
12.	Migration	60
12.1	TPM_CreateMigrationBlob	60
12.2	TPM_ConvertMigrationBlob	63
12.3	TPM_AuthorizeMigrationKey	64
12.4	TPM_MigrateKey	66
12.5	TPM_CMK_SetRestrictions	67
12.6	TPM_CMK_ApproveMA	69
12.7	TPM_CMK_CreateKey	70
12.8	TPM_CMK_CreateTicket	72
12.9	TPM_CMK_CreateBlob	74
12.10	TPM_CMK_ConvertMigration	77
13.	Maintenance Functions (optional)	80
13.1	TPM_CreateMaintenanceArchive	81
13.2	TPM_LoadMaintenanceArchive	83
13.3	TPM_KillMaintenanceFeature	85
13.4	TPM_LoadManuMaintPub	86
13.5	TPM_ReadManuMaintPub	87
14.	Cryptographic Functions	88
14.1	TPM_SHA1Start	88
14.2	TPM_SHA1Update	89
14.3	TPM_SHA1Complete	89
14.4	TPM_SHA1CompleteExtend	90
14.5	TPM_Sign	91
14.6	TPM_GetRandom	93
14.7	TPM_StirRandom	93
14.8	TPM_CertifyKey	94

14.9	TPM_CertifyKey2	98
15.	Endorsement Key Handling	101
15.1	TPM_CreateEndorsementKeyPair	101
15.2	TPM_CreateRevocableEK	102
15.3	TPM_RevokeTrust	104
15.4	TPM_ReadPubek	105
15.5	TPM_OwnerReadInternalPub	106
16.	Identity Creation and Activation	107
16.1	TPM_MakeIdentity	107
16.2	TPM_ActivateIdentity	110
17.	Integrity Collection and Reporting	113
17.1	TPM_Extend	113
17.2	TPM_PCRRead	114
17.3	TPM_Quote	115
17.4	TPM_PCR_Reset	116
17.5	TPM_Quote2	118
18.	Changing AuthData	120
18.1	TPM_ChangeAuth	120
18.2	TPM_ChangeAuthOwner	122
19.	Authorization Sessions	123
19.1	TPM_OIAP	123
19.1.1	Actions to validate an OIAP session	124
19.2	TPM_OSAP	125
19.2.1	Actions to validate an OSAP session	128
19.3	TPM_DSAP	129
19.4	TPM_SetOwnerPointer	132
20.	Delegation Commands	134
20.1	TPM_Delegate_Manage	134
20.2	TPM_Delegate_CreateKeyDelegation	137
20.3	TPM_Delegate_CreateOwnerDelegation	139
20.4	TPM_Delegate_LoadOwnerDelegation	142
20.5	TPM_Delegate_ReadTable	144
20.6	TPM_Delegate_UpdateVerification	145
20.7	TPM_Delegate_VerifyDelegation	147
21.	Non-volatile Storage	148
21.1	TPM_NV_DefineSpace	149
21.2	TPM_NV_WriteValue	152
21.3	TPM_NV_WriteValueAuth	154
21.4	TPM_NV_ReadValue	156
21.5	TPM_NV_ReadValueAuth	158

22. Session Management	160
22.1 TPM_KeyControlOwner	160
22.2 TPM_SaveContext	162
22.3 TPM_LoadContext	164
23. Eviction	167
23.1 TPM_FlushSpecific	167
24. Timing Ticks	169
24.1 TPM_GetTicks	169
24.2 TPM_TickStampBlob	170
25. Transport Sessions	172
25.1 TPM_EstablishTransport	172
25.2 TPM_ExecuteTransport	175
25.3 TPM_ReleaseTransportSigned	181
26. Monotonic Counter	184
26.1 TPM_CreateCounter	184
26.2 TPM_IncrementCounter	185
26.3 TPM_ReadCounter	186
26.4 TPM_ReleaseCounter	187
26.5 TPM_ReleaseCounterOwner	188
27. DAA commands	190
27.1 TPM_DAA_Join	190
27.2 TPM_DAA_Sign	205
28. Deprecated commands	215
28.1 Key commands	215
28.1.1 TPM_EvictKey	215
28.1.2 TPM_Terminate_Handle	216
28.2 Context management	217
28.2.1 TPM_SaveKeyContext	217
28.2.2 TPM_LoadKeyContext	218
28.2.3 TPM_SaveAuthContext	219
28.2.4 TPM_LoadAuthContext	220
28.3 DIR commands	220
28.3.1 TPM_DirWriteAuth	221
28.3.2 TPM_DirRead	222
28.4 Change Auth	222
28.4.1 TPM_ChangeAuthAsymStart	223
28.4.2 TPM_ChangeAuthAsymFinish	226
28.5 TPM_Reset	228
28.6 TPM_OwnerReadPubek	229
28.7 TPM_DisablePubekRead	230

28.8	TPM_LoadKey	231
29.	Deleted Commands	234
29.1	TPM_GetCapabilitySigned	234
29.2	TPM_GetOrdinalAuditStatus	234
29.3	TPM_CertifySelfTest	235
30.	Bibliography	237

List of Tables

Table 1. TPM_Init Incoming Parameters and Sizes	6
Table 2. TPM_Init Outgoing Parameters and Sizes	6
Table 3. TPM_SaveState Incoming Parameters and Sizes	9
Table 4. TPM_SaveState Outgoing Parameters and Sizes	9
Table 5. TPM_SelfTestFull Incoming Operands and Sizes	10
Table 6. TPM_SelfTestFull Outgoing Operands and Sizes	10
Table 7. TPM_ContinueSelfTest Incoming Operands and Sizes	10
Table 8. TPM_ContinueSelfTest Outgoing Operands and Sizes	11
Table 9. TPM_GetTestResult Incoming Operands and Sizes	12
Table 10. TPM_GetTestResult Outgoing Operands and Sizes	12
Table 11. TPM_SetOwnerInstall Incoming Operands and Sizes	13
Table 12. TPM_SetOwnerInstall Outgoing Operands and Sizes	13
Table 13. TPM_OwnerSetDisable Incoming Operands and Sizes	13
Table 14. TPM_OwnerSetDisable Outgoing Operands and Sizes	14
Table 15. TPM_PhysicalEnable Incoming Operands and Sizes	14
Table 16. TPM_PhysicalEnable Outgoing Operands and Sizes	14
Table 17. TPM_PhysicalDisable Incoming Operands and Sizes	15
Table 18. TPM_PhysicalEnable Outgoing Operands and Sizes	15
Table 19. TPM_PhysicalSetDeactivated Incoming Operands and Sizes	15
Table 20. TPM_PhysicalSetDeactivated Outgoing Operands and Sizes	15
Table 21. TPM_SetTemp Deactivated Incoming Operands and Sizes	16
Table 22. TPM_SetTemp Deactivated Outgoing Operands and Sizes	16
Table 23. TPM_SetOperatorAuth Incoming Operands and Sizes	17
Table 24. TPM_SetOperatorAuth Outgoing Operands and Sizes	17
Table 25. TPM_TakeOwnership Incoming Operands and Sizes	18
Table 26. TPM_TakeOwnership Outgoing Operands and Sizes	19
Table 27. TPM_OwnerClear Incoming Operands and Sizes	20
Table 28. TPM_OwnerClear Outgoing Operands and Sizes	20
Table 29. TPM_ForceClear Incoming Operands and Sizes	22
Table 30. TPM_ForceClear Outgoing Operands and Sizes	23
Table 31. TPM_DisableOwnerClear Incoming Operands and Sizes	23
Table 32. TPM_DisableOwnerClear Outgoing Operands and Sizes	23
Table 33. TPM_DisableForceClear Incoming Operands and Sizes	24
Table 34. TPM_DisableForceClear Outgoing Operands and Sizes	24
Table 35. TSC_PhysicalPresence Incoming Operands and Sizes	25
Table 36. TSC_PhysicalPresence Outgoing Operands and Sizes	25
Table 37. TCG_ResetEstablishmentBit Incoming Operands and Sizes	27
Table 38. TCG_ResetEstablishmentBit Outgoing Operands and Sizes	27
Table 39. TPM_GetCapability Incoming Parameters and Sizes	28

Table 40. TPM_GetCapability Outgoing Parameters and Sizes	28
Table 41. TPM_SetCapability Incoming Parameters and Sizes	29
Table 42. TPM_SetCapability Outgoing Parameters and Sizes	30
Table 43. TPM_GetCapabilityOwner Incoming Operands and Sizes	30
Table 44. TPM_GetCapabilityOwner Outgoing Operands and Sizes	31
Table 45. TPM_GetAuditDigest Incoming Parameters and Sizes	34
Table 46. TPM_GetAuditDigest Outgoing Parameters and Sizes	34
Table 47. TPM_GetAuditDigestSigned Incoming Parameters and Sizes	35
Table 48. TPM_GetAuditDigestSigned Outgoing Parameters and Sizes	36
Table 49. TPM_SetOrdinalAuditStatus Incoming Parameters and Sizes	37
Table 50. TPM_SetOrdinalAuditStatus Outgoing Parameters and Sizes	37
Table 51. TPM_FieldUpgrade Parameters	38
Table 52. TPM_SetRedirection Incoming Operands and Sizes	40
Table 53. TPM_SetRedirection Outgoing Operands and Sizes	40
Table 54. TPM_ResetLockValue Incoming Operands and Sizes	41
Table 55. TPM_ResetLockValue Outgoing Operands and Sizes	42
Table 56. TPM_Seal Incoming Operands and Sizes	44
Table 57. TPM_Seal Outgoing Operands and Sizes	44
Table 58. TPM_Unseal Incoming Operands and Sizes	46
Table 59. TPM_Unseal Outgoing Operands and Sizes	47
Table 60. TPM_UnBind Incoming Operands and Sizes	49
Table 61. TPM_UnBind Outgoing Operands and Sizes	50
Table 62. TPM_CreateWrapKey Incoming Operands and Sizes	51
Table 63. TPM_CreateWrapKey Outgoing Operands and Sizes	51
Table 64. TPM_WrapKey Incoming Operands and Sizes	54
Table 65. TPM_WrapKey Outgoing Operands and Sizes	54
Table 66. TPM_GetPubKey Incoming Operands and Sizes	56
Table 67. TPM_GetPubKey Outgoing Operands and Sizes	56
Table 68. TPM_Sealx Incoming Operands and Sizes	57
Table 69. TPM_Sealx Outgoing Operands and Sizes	58
Table 70. TPM_CreateMigrationBlob Incoming Operands and Sizes	61
Table 71. TPM_CreateMigrationBlob Outgoing Operands and Sizes	62
Table 72. TPM_ConvertMigrationBlob Incoming Operands and Sizes	63
Table 73. TPM_ConvertMigrationBlob Outgoing Operands and Sizes	64
Table 74. TPM_AuthorizeMigrationKey Incoming Operands and Sizes	65
Table 75. TPM_AuthorizeMigrationKey Outgoing Operands and Sizes	65
Table 76. TPM_MigrateKey Incoming Operands and Sizes	66
Table 77. TPM_MigrateKey Outgoing Operands and Sizes	67
Table 78. TPM_CMK_SetRestrictions Incoming Operands and Sizes	67
Table 79. TPM_CMK_SetRestrictions Outgoing Operands and Sizes	68

Table 80. TPM_CMK_ApproveMA Incoming Operands and Sizes	69
Table 81. TPM_CMK_ApproveMA Outgoing Operands and Sizes	69
Table 82. TPM_CMK_CreateKey Incoming Operands and Sizes	70
Table 83. TPM_CMK_CreateKey Outgoing Operands and Sizes	71
Table 84. TPM_CMK_CreateTicket Incoming Operands and Sizes	73
Table 85. TPM_CMK_CreateTicket Outgoing Operands and Sizes	73
Table 86. TPM_CMK_CreateBlob Incoming Operands and Sizes	74
Table 87. TPM_CMK_CreateBlob Outgoing Operands and Sizes	75
Table 88. TPM_CMK_ConvertMigration Incoming Operands and Sizes	77
Table 89. TPM_CMK_ConvertMigration Outgoing Operands and Sizes	78
Table 90. TPM_CreateMaintenanceArchive Incoming Operands and Sizes	81
Table 91. TPM_CreateMaintenanceArchive Outgoing Operands and Sizes	81
Table 92. TPM_LoadMaintenanceArchive Incoming Operands and Sizes	83
Table 93. TPM_LoadMaintenanceArchive Outgoing Operands and Sizes	83
Table 94. TPM_KillMaintenanceFeature Incoming Operands and Sizes	85
Table 95. TPM_KillMaintenanceFeature Outgoing Operands and Sizes	85
Table 96. TPM_LoadManuMaintPub Incoming Operands and Sizes	86
Table 97. TPM_LoadManuMaintPub Outgoing Operands and Sizes	86
Table 98. TPM_ReadManuMaintPub Incoming Operands and Sizes	87
Table 99. TPM_ReadManuMaintPub Outgoing Operands and Sizes	87
Table 100. TPM_SHA1Start Incoming Operands and Sizes	88
Table 101. TPM_SHA1Start Outgoing Operands and Sizes	88
Table 102. TPM_SHA1Update Incoming Operands and Sizes	89
Table 103. TPM_SHA1Update Outgoing Operands and Sizes	89
Table 104. TPM_SHA1Complete Incoming Operands and Sizes	89
Table 105. TPM_SHA1Complete Outgoing Operands and Sizes	90
Table 106. TPM_SHA1CompleteExtend Incoming Operands and Sizes	90
Table 107. TPM_SHA1CompleteExtend Outgoing Operands and Sizes	90
Table 108. TPM_Sign Incoming Operands and Sizes	91
Table 109. TPM_Sign Outgoing Operands and Sizes	91
Table 110. TPM_GetRandom Incoming Operands and Sizes	93
Table 111. TPM_GetRandom Outgoing Operands and Sizes	93
Table 112. TPM_StirRandom Incoming Operands and Sizes	93
Table 113. TPM_StirRandom Outgoing Operands and Sizes	94
Table 114. TPM_CertifyKey Incoming Operands and Sizes	95
Table 115. TPM_CertifyKey Outgoing Operands and Sizes	95
Table 116. TPM_CertifyKey2 Incoming Operands and Sizes	98
Table 117. TPM_CertifyKey2 Outgoing Operands and Sizes	99
Table 118. TPM_CreateEndorsementKeyPair Incoming Operands and Sizes	101
Table 119. TPM_CreateEndorsementKeyPair Outgoing Operands and Sizes	101

Table 120. TPM_CreateRevocableEK Incoming Operands and Sizes	103
Table 121. TPM_CreateRevocableEK Outgoing Operands and Sizes	103
Table 122. TPM_RevokeTrust Incoming Operands and Sizes	104
Table 123. TPM_RevokeTrust Outgoing Operands and Sizes	104
Table 124. TPM_ReadPubek Incoming Operands and Sizes	105
Table 125. TPM_ReadPubek Outgoing Operands and Sizes	105
Table 126. TPM_OwnerReadInternalPub Incoming Operands and Sizes	106
Table 127. TPM_OwnerReadInternalPub Outgoing Operands and Sizes	106
Table 128. TPM_MakeIdentity Incoming Operands and Sizes	107
Table 129. TPM_MakeIdentity Outgoing Operands and Sizes	108
Table 130. Properties of the new identity	108
Table 131. TPM_ActivateIdentity Incoming Parameters and Sizes	110
Table 132. TPM_ActivateIdentity Outgoing Parameters and Sizes	111
Table 133. TPM_Extend Incoming Operands and Sizes	113
Table 134. TPM_Extend Outgoing Operands and Sizes	113
Table 135. TPM_PCRRead Incoming Operands and Sizes	114
Table 136. TPM_PCRRead Outgoing Operands and Sizes	114
Table 137. TPM_Quote Incoming Operands and Sizes	115
Table 138. TPM_Quote Outgoing Operands and Sizes	115
Table 139. TPM_PCR_Reset Incoming Parameters and Sizes	117
Table 140. TPM_PCR_Reset Outgoing Parameters and Sizes	117
Table 141. TPM_Quote2 Incoming Operands and Sizes	118
Table 142. TPM_Quote2 Outgoing Operands and Sizes	118
Table 143. TPM_ChangeAuth Incoming Operands and Sizes	120
Table 144. TPM_ChangeAuth Outgoing Operands and Sizes	121
Table 145. TPM_ChangeAuthOwner Incoming Operands and Sizes	122
Table 146. TPM_ChangeAuthOwner Outgoing Operands and Sizes	122
Table 147. TPM_OIAP Incoming Operands and Sizes	123
Table 148. TPM-OIAP Outgoing Operands and Sizes	123
Table 149. TPM_OSAP Incoming Operands and Sizes	125
Table 150. TPM_OSAP Outgoing Operands and Sizes	125
Table 151. TPM_DSAP Incoming Operands and Sizes	129
Table 152. TPM_DSAP Outgoing Operands and Sizes	129
Table 153. TPM_SetOwnerPointer Incoming Operands and Sizes	132
Table 154. TPM_SetOwnerPointer Outgoing Operands and Sizes	133
Table 155. TPM_Delegate_Manage Incoming Operands and Sizes	135
Table 156. TPM_Delegate_Manage Outgoing Operands and Sizes	135
Table 157. TPM_Delegate_CreateKeyDelegation Incoming Operands and Sizes	137
Table 158. TPM_Delegate_CreateKeyDelegation Outgoing Operands and Sizes	138
Table 159. TPM_Delegate_CreateOwnerDelegation Incoming Operands and Sizes	140

Table 160. TPM_Delegate_CreateOwnerDelegation Outgoing Operands and Sizes	140
Table 161. TPM_Delegate_LoadOwnerDelegation Incoming Operands and Sizes	142
Table 162. TPM_Delegate_LoadOwnerDelegation Outgoing Operands and Sizes	143
Table 163. TPM_Delegate_ReadTable Incoming Operands and Sizes	144
Table 164. TPM_Delegate_ReadTable Outgoing Operands and Sizes	144
Table 165. TPM_Delegate_UpdateVerification Incoming Operands and Sizes	145
Table 166. TPM_Delegate_UpdateVerification Outgoing Operands and Sizes	146
Table 167. TPM_Delegate_VerifyDelegation Incoming Operands and Sizes	147
Table 168. TPM_Delegate_VerifyDelegation Outgoing Operands and Sizes	147
Table 169. TPM_NV_DefineSpace Incoming Operands and Sizes	149
Table 170. TPM_NV_DefineSpace Outgoing Operands and Sizes	149
Table 171. TPM_NV_WriteValue Incoming Operands and Sizes	152
Table 172. TPM_NV_WriteValue Outgoing Operands and Sizes	152
Table 173. TPM_NV_WriteValueAuth Incoming Operands and Sizes	154
Table 174. TPM_NV_WriteValueAuth Outgoing Operands and Sizes	155
Table 175. TPM_NV_ReadValue Incoming Operands and Sizes	156
Table 176. TPM_NV_ReadValue Outgoing Operands and Sizes	156
Table 177. TPM_NV_ReadValueAuth Incoming Operands and Sizes	158
Table 178. TPM_NV_ReadValueAuth Outgoing Operands and Sizes	158
Table 179. TPM_KeyControlOwner Incoming Parameters and Sizes	161
Table 180. TPM_KeyControlOwner Outgoing Parameters and Sizes	161
Table 181. TPM_SaveContext Incoming Parameters and Sizes	162
Table 182. TPM_SaveContext Outgoing Parameters and Sizes	162
Table 183. TPM_LoadContext Incoming Parameters and Sizes	164
Table 184. TPM_LoadContext Outgoing Parameters and Sizes	165
Table 185. TPM_FlushSpecific Incoming Parameters and Sizes	167
Table 186. TPM_FlushSpecific Outgoing Parameters and Sizes	167
Table 187. TPM_GetTicks Incoming Parameters and Sizes	169
Table 188. TPM_GetTicks Outgoing Parameters and Sizes	169
Table 189. TPM_TickStampBlob Incoming Parameters and Sizes	170
Table 190. TPM_TickStampBlob Outgoing Parameters and Sizes	171
Table 191. TPM_EstablishTransport Incoming Parameters and Sizes	172
Table 192. TPM_EstablishTransport Outgoing Parameters and Sizes	173
Table 193. TPM_ExecuteTransport Incoming Parameters and Sizes	175
Table 194. TPM_ExecuteTransport Outgoing Parameters and Sizes	176
Table 195. TPM_ReleaseTransportSigned Incoming Parameters and Sizes	181
Table 196. TPM_ReleaseTransportSigned Outgoing Parameters and Sizes	182
Table 197. TPM_CreateCounter Incoming Parameters and Sizes	184
Table 198. TPM_CreateCounter Outgoing Parameters and Sizes	184
Table 199. TPM_IncrementCounter Incoming Parameters and Sizes	185

Table 200. TPM_IncrementCounter Outgoing Parameters and Sizes	186
Table 201. TPM_ReadCounter Incoming Parameters and Sizes	186
Table 202. TPM_ReadCounter Outgoing Parameters and Sizes	187
Table 203. TPM_ReleaseCounter Incoming Parameters and Sizes	187
Table 204. TPM_ReleaseCounter Outgoing Parameters and Sizes	188
Table 205. TPM_ReleaseCounterOwner Incoming Parameters and Sizes	188
Table 206. TPM_ReleaseCounter Owner Outgoing Parameters and Sizes	189
Table 207. TPM_DAA_Join Incoming Parameters and Sizes	190
Table 208. TPM_DAA_Join Outgoing Operands and Sizes	190
Table 209. Input, Output and Saved Data Associated with Processing	191
Table 210. TPM_DAA_Sign Incoming Operands and Sizes	205
Table 211. TPM_DAA_Sign Outgoing Operands and Sizes	205
Table 212. Input, Output and Saved Data Associated with Processing	206
Table 213. TPM_EvictKey Incoming Operands and Sizes	215
Table 214. TPM_EvictKey Outgoing Operands and Sizes	215
Table 215. TPM_Terminate_Handle Incoming Operands and Sizes	216
Table 216. TPM_Terminate_Handle Outgoing Operands and Sizes	216
Table 217. TPM_SaveKeyContext Incoming Operands and Sizes	217
Table 218. TPM_SaveKeyContext Outgoing Operands and Sizes	217
Table 219. TPM_LoadKeyContext Incoming Operands and Sizes	218
Table 220. TPM_LoadKeyContext Outgoing Operands and Sizes	218
Table 221. TPM_SaveAuthContext Incoming Operands and Sizes	219
Table 222. TPM_SaveAuthContext Outgoing Operands and Sizes	219
Table 223. TPM_LoadAuthContext Incoming Operands and Sizes	220
Table 224. TPM_LoadAuthContext Outgoing Operands and Sizes	220
Table 225. TPM_DirWriteAuth Incoming Operands and Sizes	221
Table 226. TPM_DirWriteAuth Outgoing Operands and Sizes	221
Table 227. TPM_DirRead Incoming Operands and Sizes	222
Table 228. TPM_DirRead Outgoing Operands and Sizes	222
Table 229. TPM_ChangeAuthAsymStart Incoming Operands and Sizes	223
Table 230. TPM_ChangeAuthAsymStart Outgoing Operands and Sizes	224
Table 231. Field Descriptions for certifyInfo parameter	225
Table 232. TPM_ChangeAuthAsymFinish Incoming Operands and Sizes	226
Table 233. TPM_ChangeAuthAsymFinish Outgoing Operands and Sizes	227
Table 234. TPM_Reset Incoming Parameters and Sizes	228
Table 235. TPM_Reset Outgoing Parameters and Sizes	228
Table 236. TPM_OwnerReadPubek Incoming Operands and Sizes	229
Table 237. TPM_OwnerReadPubek Outgoing Operands and Sizes	229
Table 238. TPM_DisablePubekRead Incoming Operands and Sizes	230
Table 239. TPM_DisablePubekRead Outgoing Operands and Sizes	230

Table 240. TPM_LoadKey Incoming Operands and Sizes	232
Table 241. TPM_LoadKey Outgoing Operands and Sizes	232
Table 242. TPM_GetOrdinalAuditStatus Incoming Operands and Sizes	234
Table 243. TPM_GetOrdinalAuditStatus Outgoing Operands and Sizes	234
Table 244. TPM_CertifySelfTest Incoming Operands and Sizes	235
Table 245. TPM_CertifySelfTest Outgoing Operands and Sizes	235

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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 11889-4 was prepared by the Trusted Computing Group (TCG) and was adopted, under the PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

ISO/IEC 11889 consists of the following parts, under the general title *Information technology — Trusted Platform Module*:

- *Part 1: Overview*
- *Part 2: Design principles*
- *Part 3: Structures*
- *Part 4: Commands*

Introduction

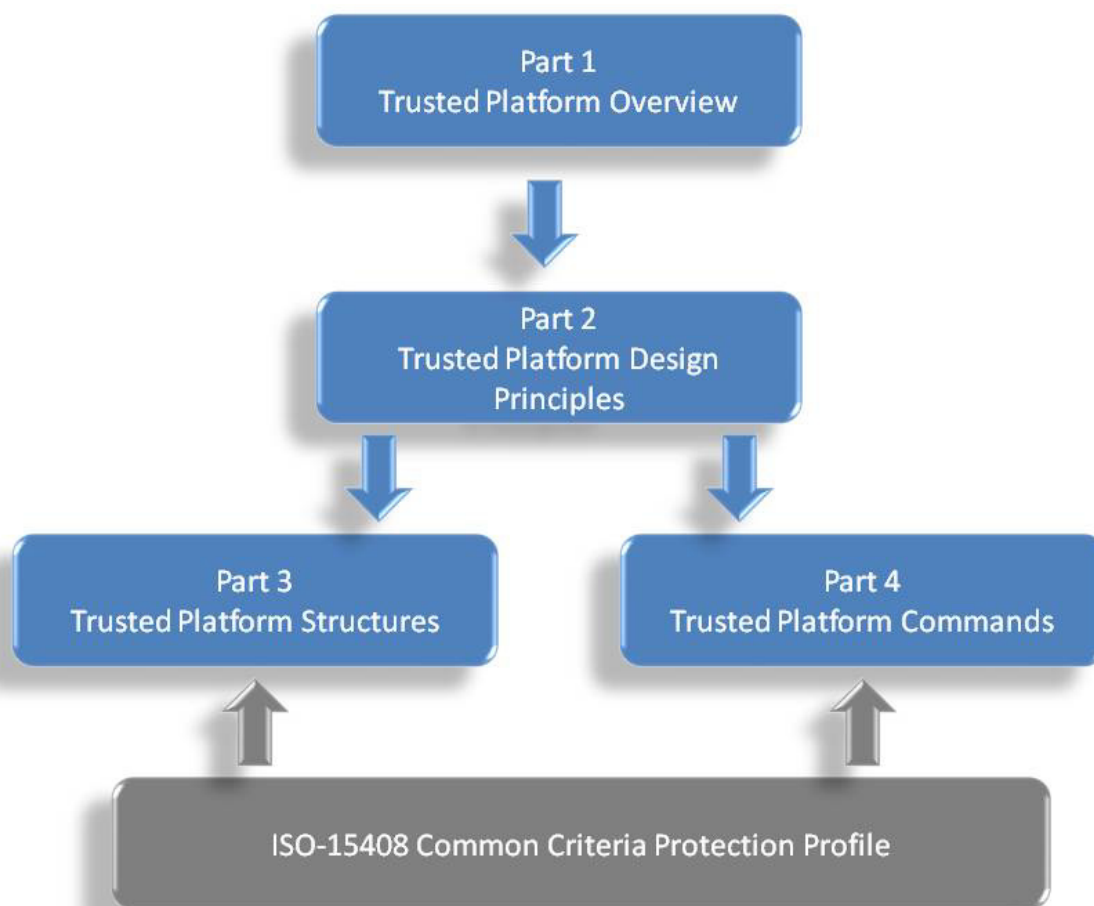


Figure 1. TPM Main Specification Roadmap

Start of informative comment

ISO/IEC 11889 is from the Trusted Computing Group (TCG) Trusted Platform Module (TPM) specification 1.2 version 103. The part numbers for ISO/IEC 11889 and the TCG specification do not match. The reason is the inclusion of the Overview document that is not a member of the TCG part numbering. The mapping between the two is as follows:

ISO Reference	TCG Reference
Part 1 Overview	Not published
Part 2 Design Principles	Part 1 Design Principles
Part 3 Structures	Part 2 Structures
Part 4 Commands	Part 3 Commands

End of informative comment

Information technology — Trusted Platform Module —

Part 4: Commands

1. Scope

ISO/IEC 11889 defines the Trusted Platform Module (TPM), a device that enables trust in computing platforms in general. ISO/IEC 11889 is broken into parts to make the role of each document clear. Any version of the standard requires all parts to be a complete standard.

A TPM designer **MUST** be aware that for a complete definition of all requirements necessary to build a TPM, the designer **MUST** use the appropriate platform specific specification to understand all of the TPM requirements.

Part 4 defines the commands that allow software to communicate with and use the TPM. It defines the command format as both the TPM and calling software must agree on the exact command format, as many of the commands require cryptographic authorization and the format of the authorization must be standardized.

1.1 Key words

The key words “**MUST**,” “**MUST NOT**,” “**REQUIRED**,” “**SHALL**,” “**SHALL NOT**,” “**SHOULD**,” “**SHOULD NOT**,” “**RECOMMENDED**,” “**MAY**,” and “**OPTIONAL**” in this document’s normative statements are to be interpreted as described in RFC-2119, *Key words for use in RFCs to Indicate Requirement Levels*.

1.2 Statement Type

Please note a very important distinction between different sections of text throughout this document. You will encounter two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, you can consider it of the kind normative statements.

For example:

Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the standard the user must read the standard. (This use of **MUST** does not require any action).

End of informative comment

This is the first paragraph of one or more paragraphs (and/or sections) containing the text of the kind normative statements ...

To understand the standard the user **MUST** read the standard. (This use of **MUST** indicates a keyword usage and requires an action).

2. Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies

- ISO 8825-1|ITU-T X.690**: Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
- ISO/IEC 10118-3**, Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions, Clause 9, SHA-1
- ISO/IEC 18033-3**, Information technology — Security techniques — Encryption algorithms — Part 3, Block ciphers, Clause 5.1 AES
- IEEE P1363**, Institute of Electrical and Electronics Engineers: Standard Specifications For Public-Key Cryptography
- IETF RFC 2104**, Internet Engineering Task Force Request for Comments 2104: HMAC: Keyed-Hashing for Message Authentication
- IETF RFC 2119**, Internet Engineering Task Force Request for Comments 2119: Key words for use in RFCs to Indicate Requirement Levels
- PKCS #1 Version 2.1**, RSA Cryptography Standard. This document is superseded by P1363, except for section 7.2 that defines the V1.5 RSA signature scheme in use by the TPM.

3. Abbreviated Terms

Abbreviation	Description
AACP	Asymmetric Authorization Change Protocol
ADCP	Authorization Data Change Protocol
ADIP	Authorization Data Insertion Protocol
AIK	Attestation Identity Key
AMC	Audit Monotonic Counter
APIP	Time-Phased Implementation Plan
AuthData	Authentication Data or Authorization Data, depending on the context
BCD	Binary Coded Decimal
BIOS	Basic Input/Output System
CA	Certification of Authority
CDI	Controlled Data Item
CMK	Cerifiable/Certified Migratable Keys
CRT	Chinese Remainder Theorem
CRTM	Core Root of Trust Measurement
CTR	Counter-mode encryption
DAA	Direct Autonomous Attestation
DIR	Data Integrity Register
DOS	Disk Operating System
DSA	Digital Signature Algorithm
DSAP	Delegate-Specific Authorization Protocol
ECB	Electronic Codebook Mode
EK	Endorsement Key
ET	ExecuteTransport or Entity Type
FIPS	Federal Information Processing Standard
GPIO	General Purpose I/O
HMAC	Hash Message Authentication Code
HW	Hardware Interface
IB	Internal Base
I/O	Input/Output
IV	Initialization Vector
KH	Key Handle
LEAP	Lightweight Extensible Authentication Protocol for wireless computer networks
LK	Loaded Key
LOM	Limited Operation Mode
LPC	Low Pin Count
LSB	Least Significant Byte
MA	Migration Authority/Authorization
MIDL	Microsoft Interface Definition Language
MSA	Migration Selection Authority
MSB	Most Significant Byte
NV	Non-volatile
NVRAM	Non-Volatile Random Access Memory

Abbreviation	Description
OAEP	Optimal Asymmetric Encryption Padding
OEM	Original Equipment Manufacturer
OIAP	Object-Independent Authorization Protocol
OID	Object Identifier
OSAP	Object-Specific Authorization Protocol
PCR	Platform Configuration Register
PI	Personal Information
PII	Personally Identifiable Information
POST	Power On Self Test
PRIVEK	Private Endorsement Key
PRNG	Pseudo Random Number Generator
PSS	Probabilistic Signature Scheme
PUBEK	Public Endorsement Key
RNG	Random Number Generator
RSA	Algorithm for public-key cryptography. The letters R, S, and A represent the initials of the first public describers of the algorithm.
RTM	Release to Manufacturing/Ready to Market
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
SHA	Secure Hash Algorithm
SRK	Storage Root Key
STF	Self Test Failed
TA	Time Authority
TBB	Threading Building Blocks
TCG	Trusted Computing Group
TCV	Tick Count Value
TIR	Tick Increment Rate
TIS	TPM Interface Specification
TNC	Trusted Network Connect
TOE	Target of Evaluation
TOS	Trusted Operating System
TPCA	Trusted Platform Computing Alliance
TPM	Trusted Platform Module
TPME	Trusted Platform Module Entity
TSC	Tick Stamp Counter
TSC_	TPM Software Connection, when used as a command prefix
TSN	Tick Session Name
TSR	Tick Stamp Reset
TSRB	TickStampReset for blob
TSS	TCG Software Stack
TTP	Trusted Third Party/Time-Triggered Protocol
TS	Tick Stamp
UTC	Universal Time Clock
VPN	Virtual Private Network

4. Admin Startup and State

Start of informative comment:

This section is the commands that start a TPM.

End of informative comment.

4.1 TPM_Init

Start of informative comment:

TPM_Init is a physical method of initializing a TPM. There is no TPM_Init ordinal as this is a platform message sent on the platform internals to the TPM. On a PC this command arrives at the TPM via the LPC bus and informs the TPM that the platform is performing a boot process.

TPM_Init puts the TPM into a state where it waits for the command TPM_Startup, which specifies the type of initialization that is required.

End of informative comment.

Definition

`TPM_Init()`;

Operation of the TPM. This is not a command that any software can execute. It is inherent in the design of the TPM and the platform that the TPM resides on.

Parameters

None

Description

1. The TPM_Init signal indicates to the TPM that platform initialization is taking place. The TPM SHALL set the TPM into a state such that the only legal command to receive after the TPM_Init is the TPM_Startup command. The TPM_Startup will further indicate to the TPM how to handle and initialize the TPM resources.
2. The platform design MUST be that the TPM is not the only component undergoing initialization. If the TPM_Init signal forces the TPM to perform initialization then the platform MUST ensure that ALL components of the platform receive an initialization signal. This is to prevent an attacker from causing the TPM to initialize to a state where various masquerades are allowable. For instance, on a PC causing the TPM to initialize and expect measurements in PCR0 but the remainder of the platform does not initialize.
3. The design of the TPM MUST be such that the ONLY mechanism that signals TPM_Init also signals initialization to the other platform components.

Actions

1. The TPM sets TPM_STANY_FLAGS -> postInitialise to TRUE.

4.2 TPM_Startup

Start of informative comment:

TPM_Startup is always preceded by TPM_Init, which is the physical indication (a system-wide reset) that TPM initialization is necessary.

There are many events on a platform that can cause a reset and the response to these events can require different operations to occur on the TPM. The mere reset indication does not contain sufficient information to inform the TPM as to what type of reset is occurring. Additional information known by the platform initialization code needs transmitting to the TPM. The TPM_Startup command provides the mechanism to transmit the information.

The TPM can startup in three different modes:

A "clear" start where all variables go back to their default or non-volatile set state

A "save" start where the TPM recovers appropriate information and restores various values based on a prior TPM_SaveState. This recovery requires an invocation of TPM_Init to be successful.

A failing "save" start must shut down the TPM. The CRTM cannot leave the TPM in a state where an untrusted upper software layer could issue a "clear" and then extend PCRs and thus mimic the CRTM.

A "deactivated" start where the TPM turns itself off and requires another TPM_Init before the TPM will execute in a fully operational state.

End of informative comment.

Table 1. TPM_Init Incoming Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal TPM_ORD_Startup
4	2	2S	2	TPM_STARTUP_TYPE	startupType	Type of startup that is occurring

Table 2. TPM_Init Outgoing Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_Startup

Description

TPM_Startup MUST be generated by a trusted entity (the RTM or the TPM, for example).

1. If the TPM is in failure mode
 - a. TPM_STANY_FLAGS -> postInitialize is still set to FALSE
 - b. The TPM returns TPM_FAILEDSELFTEST

Actions

1. If TPM_STANY_FLAGS -> postInitialise is FALSE,
 - a. Then the TPM MUST return TPM_INVALID_POSTINIT, and exit this capability

2. If stType = TPM_ST_CLEAR
 - a. Ensure that sessions associated with resources TPM_RT_CONTEXT, TPM_RT_AUTH, TPM_RT_DAA_TPM, and TPM_RT_TRANS are invalidated
 - b. Reset TPM_STCLEAR_DATA -> PCR[] values to each correct default value
 - i. pcrReset is FALSE, set to 0x00..00
 - ii. pcrReset is TRUE, set to 0xFF..FF
 - c. Set the following TPM_STCLEAR_FLAGS to their default state
 - i. PhysicalPresence
 - ii. PhysicalPresenceLock
 - iii. disableForceClear
 - d. The TPM MAY initialize auditDigest to all zeros
 - i. If not initialized to all zeros, the TPM SHALL ensure that auditDigest contains a valid value.
 - ii. If initialization fails, the TPM SHALL set auditDigest to all zeros and SHALL set the internal TPM state so that the TPM returns TPM_FAILEDSELFTEST to all subsequent commands.
 - e. The TPM SHALL set TPM_STCLEAR_FLAGS -> deactivated to the same state as TPM_PERMANENT_FLAGS -> deactivated
 - f. The TPM MUST set the TPM_STANY_DATA fields to:
 - i. TPM_STANY_DATA->contextNonceSession is set to all zeros
 - ii. TPM_STANY_DATA->contextCount is set to 0
 - iii. TPM_STANY_DATA->contextList is set to 0
 - g. The TPM MUST set TPM_STCLEAR_DATA fields to:
 - i. Invalidate contextNonceKey
 - ii. countID to zero
 - iii. ownerReference to TPM_KH_OWNER
 - h. The TPM MUST set the following TPM_STCLEAR_FLAGS to
 - i. bGlobalLock to FALSE
 - i. Determine which keys should remain in the TPM
 - i. For each key that has a valid preserved value in the TPM
 1. if parentPCRStatus is TRUE then call TPM_FlushSpecific(keyHandle)
 2. if isVolatile is TRUE then call TPM_FlushSpecific(keyHandle)
 - ii. Keys under control of the OwnerEvict flag MUST stay resident in the TPM
3. If stType = TPM_ST_STATE
 - a. If the TPM has no state to restore, the TPM MUST set the internal state such that it returns TPM_FAILEDSELFTEST to all subsequent commands.
 - b. The TPM MAY determine for each session type (authorization, transport, DAA, ...) to release or maintain the session information. The TPM reports how it manages sessions in the TPM_GetCapability command.

- c. The TPM SHALL take all necessary actions to ensure that all PCRs contain valid preserved values. If the TPM is unable to successfully complete these actions, it SHALL enter the TPM failure mode.
 - i. For resettable PCR the TPM MUST set the value of TPM_STCLEAR_DATA -> PCR[] to the resettable PCR default value. The TPM MUST NOT restore a resettable PCR to a preserved value
 - d. The TPM MAY initialize auditDigest to all zeros.
 - i. Otherwise, the TPM SHALL take all actions necessary to ensure that auditDigest contains a valid value. If the TPM is unable to successfully complete these actions, the TPM SHALL initialize auditDigest to all zeros and SHALL set the internal state such that the TPM returns TPM_FAILEDSELFTEST to all subsequent commands.
 - e. The TPM MUST restore the following flags to their preserved states:
 - i. All values in TPM_STCLEAR_FLAGS
 - ii. All values in TPM_STCLEAR_DATA
 - f. The TPM MUST restore all keys that have a valid preserved value.
 - g. The TPM resumes normal operation. If the TPM is unable to resume normal operation, it SHALL enter the TPM failure mode.
4. If stType = TPM_ST_DEACTIVATED
- a. Invalidate sessions
 - i. Ensure that all resources associated with saved and active sessions are invalidated
 - b. Set the TPM_STCLEAR_FLAGS to their default state.
 - c. Set TPM_STCLEAR_FLAGS -> deactivated to TRUE
5. The TPM MUST ensure that state associated with TPM_SaveState is invalidated
6. The TPM MUST set TPM_STANY_FLAGS -> postInitialise to FALSE

4.3 TPM_SaveState

Start of informative comment:

This warns a TPM to save some state information.

If the relevant shielded storage is non-volatile, this command need have no effect.

If the relevant shielded storage is volatile and the TPM alone is unable to detect the loss of external power in time to move data to non-volatile memory, this command should be presented before the TPM enters a low or no power state.

End of informative comment.

Table 3. TPM_SaveState Incoming Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SaveState.

Table 4. TPM_SaveState Outgoing Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SaveState.

Description

1. Preserved values **MUST** be non-volatile.
2. If data is never stored in a volatile medium, that data **MAY** be used as preserved data. In such cases, no explicit action may be required to preserve that data.
3. If an explicit action is required to preserve data, it **MUST** be possible for the TPM to determine whether preserved data is valid.
4. If a parameter mirrored by any preserved value is altered, all preserved values **MUST** be declared invalid.
5. The TPM **MAY** declare all preserved values invalid in response to any command other than TPM_Init.

Actions

1. Store TPM_STCLEAR_DATA -> PCR contents except for
 - a. If the PCR attribute pcrReset is TRUE
 - b. Any platform identified debug PCR
2. The auditDigest **MUST** be handled according to the audit requirements as reported by TPM_GetCapability.
 - a. If the ordinalAuditStatus is TRUE for the TPM_SaveState ordinal and the auditDigest is being stored in the saved state, the saved auditDigest **MUST** include the TPM_SaveState input parameters and **MUST NOT** include the output parameters.
3. All values in TPM_STCLEAR_DATA **MUST** be preserved.
4. All values in TPM_STCLEAR_FLAGS **MUST** be preserved.
5. The contents of any key that is currently loaded **SHOULD** be preserved if the key's parentPCRStatus indicator is FALSE and its isVolatile indicator is FALSE.
6. The contents of any key that has TPM_KEY_CONTROL_OWNER_EVICT set **MUST** be preserved
7. The contents of any key that is currently loaded **MAY** be preserved.
8. The contents of sessions (authorization, transport, DAA, etc.) **MAY** be preserved as reported by TPM_GetCapability.

5. Admin Testing

5.1 TPM_SelfTestFull

Start of informative comment:

TPM_SelfTestFull tests all of the TPM capabilities.

Unlike TPM_ContinueSelfTest, which may optionally return immediately and then perform the tests, TPM_SelfTestFull always performs the tests and then returns success or failure.

End of informative comment.

Table 5. TPM_SelfTestFull Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SelfTestFull

Table 6. TPM_SelfTestFull Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SelfTestFull

Actions

1. TPM_SelfTestFull SHALL cause a TPM to perform self-test of each TPM internal function.
 - a. If the self-test succeeds, return TPM_SUCCESS.
 - b. If the self-test fails, return TPM_FAILEDSELFTEST.
2. Failure of any test results in overall failure, and the TPM goes into failure mode.
3. If the TPM has not executed the action of TPM_ContinueSelfTest, the TPM
 - a. MAY perform the full self-test.
 - b. MAY return TPM_NEEDS_SELFTEST.

5.2 TPM_ContinueSelfTest

Start of informative comment:

TPM_ContinueSelfTest informs the TPM that it should complete the self-test of all TPM functions.

The TPM may return success immediately and then perform the self-test, or it may perform the self-test and then return success or failure.

End of informative comment.

Table 7. TPM_ContinueSelfTest Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_ContinueSelfTest

Table 8. TPM_ContinueSelfTest Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_ContinueSelfTest

Description

1. Prior to executing the actions of TPM_ContinueSelfTest, if the TPM receives a command C1 that uses an untested TPM function, the TPM MUST take one of these actions:
 - a. The TPM MAY return TPM_NEEDS_SELFTEST
 - i. This indicates that the TPM has not tested the internal resources required to execute C1.
 - ii. The TPM does not execute C1.
 - iii. The caller MUST issue TPM_ContinueSelfTest before re-issuing the command C1.
 1. If the TPM permits TPM_SelfTestFull prior to completing the actions of TPM_ContinueSelfTest, the caller MAY issue TPM_SelfTestFull rather than TPM_ContinueSelfTest.
 - b. The TPM MAY return TPM_DOING_SELFTEST
 - i. This indicates that the TPM is doing the actions of TPM_ContinueSelfTest implicitly, as if the TPM_ContinueSelfTest command had been issued.
 - ii. The TPM does not execute C1.
 - iii. The caller MUST wait for the actions of TPM_ContinueSelfTest to complete before reissuing the command C1.
 - c. The TPM MAY return TPM_SUCCESS or an error code associated with C1.
 - i. This indicates that the TPM has completed the actions of TPM_ContinueSelfTest and has completed the command C1.
 - ii. The error code MAY be TPM_FAILEDSELFTEST.

Actions

1. If TPM_PERMANENT_FLAGS -> FIPS is TRUE or TPM_PERMANENT_FLAGS -> TPMpost is TRUE
 - a. The TPM MUST run all self-tests
2. Else
 - a. The TPM MUST complete all self-tests that are outstanding
 - i. Instead of completing all outstanding self-tests the TPM MAY run all self-tests
3. The TPM either
 - a. MAY immediately return TPM_SUCCESS
 - i. When TPM_ContinueSelfTest finishes execution, it MUST NOT respond to the caller with a return code.
 - b. MAY complete the self-test and then return TPM_SUCCESS or TPM_FAILEDSELFTEST.

5.3 TPM_GetTestResult

Start of informative comment:

TPM_GetTestResult provides manufacturer specific information regarding the results of the self-test. This command will work when the TPM is in self-test failure mode. The reason for allowing this command to operate in the failure mode is to allow TPM manufacturers to obtain diagnostic information.

End of informative comment.

Table 9. TPM_GetTestResult Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_GetTestResult

Table 10. TPM_GetTestResult Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_GetTestResult
4	4	3S	4	UINT32	outDataSize	The size of the outData area
5	<>	4S	<>	BYTE[]	outData	The outData this is manufacturer specific

Description

This command will work when the TPM is in self test failure mode or limited operation mode.

Actions

1. The TPM SHALL respond to this command with a manufacturer specific block of information that describes the result of the latest self-test
2. The information MUST NOT contain any data that uniquely identifies an individual TPM.

6. Admin Opt-in

6.1 TPM_SetOwnerInstall

Start of informative comment:

When enabled but without an owner this command sets the PERMANENT flag that allows or disallows the ability to insert an owner.

End of informative comment.

Table 11. TPM_SetOwnerInstall Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SetOwnerInstall
4	1	2S	1	BOOL	state	State to which ownership flag is to be set.

Table 12. TPM_SetOwnerInstall Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SetOwnerInstall

Action

1. If the TPM has a current owner, this command immediately returns with TPM_SUCCESS.
2. The TPM validates the assertion of physical presence. The TPM then sets the value of TPM_PERMANENT_FLAGS -> ownership to the value in state.

6.2 TPM_OwnerSetDisable

Start of informative comment:

The TPM owner sets the PERMANENT disable flag

End of informative comment.

Table 13. TPM_OwnerSetDisable Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_OwnerSetDisable
4	1	2S	1	BOOL	disableState	Value for disable state – enable if TRUE
5	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for owner authentication.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
6	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
7	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
8	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authentication. HMAC key: ownerAuth.

Table 14. TPM_OwnerSetDisable Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_OwnerSetDisable
4	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
5	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
6	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: ownerAuth.

Action

1. The TPM SHALL authenticate the command as coming from the TPM Owner. If unsuccessful, the TPM SHALL return TPM_AUTHFAIL.
2. The TPM SHALL set the TPM_PERMANENT_FLAGS -> disable flag to the value in the disableState parameter.

6.3 TPM_PhysicalEnable**Start of informative comment:**

Sets the PERMANENT disable flag to FALSE using physical presence as authorization.

End of informative comment.**Table 15. TPM_PhysicalEnable Incoming Operands and Sizes**

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_PhysicalEnable

Table 16. TPM_PhysicalEnable Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_PhysicalEnable

Action

1. Validate that physical presence is being asserted, if not return TPM_BAD_PRESENCE
2. The TPM SHALL set the TPM_PERMANENT_FLAGS.disable value to FALSE.

6.4 TPM_PhysicalDisable

Start of informative comment:

Sets the PERMANENT disable flag to TRUE using physical presence as authorization

End of informative comment.

Table 17. TPM_PhysicalDisable Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_PhysicalDisable

Table 18. TPM_PhysicalEnable Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_PhysicalDisable

Action

1. Validate that physical presence is being asserted, if not return TPM_BAD_PRESENCE
2. The TPM SHALL set the TPM_PERMANENT_FLAGS.disable value to TRUE.

6.5 TPM_PhysicalSetDeactivated

Start of informative comment:

Enables the TPM using physical presence as authorization.

This command is not available when the TPM is disabled.

End of informative comment.

Table 19. TPM_PhysicalSetDeactivated Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_PhysicalSetDeactivated
4	1	2S	1	BOOL	state	State to which deactivated flag is to be set.

Table 20. TPM_PhysicalSetDeactivated Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_PhysicalSetDeactivated

Action

1. Validate that physical presence is being asserted, if not return TPM_BAD_PRESENCE
2. The TPM SHALL set the TPM_PERMANENT_FLAGS.deactivated flag to the value in the state parameter.

6.6 TPM_SetTempDeactivated

Start of informative comment:

This command allows the operator of the platform to deactivate the TPM until the next boot of the platform.

This command requires operator authentication. The operator can provide the authentication by either the assertion of physical presence or presenting the operator AuthData value.

End of informative comment.

Table 21. TPM_SetTemp Deactivated Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SetTempDeactivated
4	4		4	TPM_AUTHHANDLE	authHandle	Auth handle for operation validation. Session type MUST be OIAP
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
5	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
6	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
7	20			TPM_AUTHDATA	operatorAuth	HMAC key: operatorAuth

Table 22. TPM_SetTemp Deactivated Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SetTempDeactivated
4	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
5	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
6	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: operatorAuth.

Action

1. If tag = TPM_TAG_REQ_AUTH1_COMMAND
 - a. If TPM_PERMANENT_FLAGS -> operator is FALSE return TPM_NOOPERATOR
 - b. Validate command and parameters using operatorAuth, on error return TPM_AUTHFAIL
2. Else
 - a. If physical presence is not asserted the TPM MUST return TPM_BAD_PRESENCE
3. The TPM SHALL set the TPM_STCLEAR_FLAGS.deactivated flag to the value TRUE.

6.7 TPM_SetOperatorAuth

Start of informative comment:

This command allows the setting of the operator AuthData value.

There is no confidentiality applied to the operator authentication as the value is sent under the assumption of being local to the platform. If there is a concern regarding the path between the TPM and the keyboard then unless the keyboard is using encryption and a secure channel an attacker can read the values.

End of informative comment.

Table 23. TPM_SetOperatorAuth Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SetOperatorAuth
4	20	2S	20	TPM_SECRET	operatorAuth	The operator AuthData

Table 24. TPM_SetOperatorAuth Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_SetOperatorAuth

Action

1. If physical presence is not asserted the TPM MUST return TPM_BAD_PRESENCE
2. The TPM SHALL set the TPM_PERMANENT_DATA -> operatorAuth
3. The TPM SHALL set TPM_PERMANENT_FLAGS -> operator to TRUE

7. Admin Ownership

7.1 TPM_TakeOwnership

Start of informative comment:

This command inserts the TPM Ownership value into the TPM.

End of informative comment.

Table 25. TPM_TakeOwnership Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_TakeOwnership
4	2	2S	2	TPM_PROTOCOL_ID	protocolID	The ownership protocol in use.
5	4	3S	4	UINT32	encOwnerAuthSize	The size of the encOwnerAuth field
6	<>	4S	<>	BYTE[]	encOwnerAuth	The owner AuthData encrypted with PUBEK
7	4	5S	4	UINT32	encSrAuthSize	The size of the encSrAuth field
8	<>	6S	<>	BYTE[]	encSrAuth	The SRK AuthData encrypted with PUBEK
9	<>	7S	<>	TPM_KEY	srkParams	Structure containing all parameters of new SRK. pubKey.keyLength & encSize are both 0. This structure MAY be TPM_KEY12.
10	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for this command
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
11	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
12	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
13	20			TPM_AUTHDATA	ownerAuth	Authorization session digest for input params. HMAC key: the new ownerAuth value. See actions for validation operations

Table 26. TPM_TakeOwnership Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_TakeOwnership
4	<>	3S	<>	TPM_KEY	srkPub	Structure containing all parameters of new SRK. srkPub.encData is set to 0. This structure MAY be TPM_KEY12.
5	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
6	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
7	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: the new ownerAuth value

Description

The type of the output srkPub MUST be the same as the type of the input srkParams, either both TPM_KEY or both TPM_KEY12.

Actions

1. If TPM_PERMANENT_DATA -> ownerAuth is valid return TPM_OWNER_SET
2. If TPM_PERMANENT_FLAGS -> ownership is FALSE return TPM_INSTALL_DISABLED
3. If TPM_PERMANENT_DATA -> endorsementKey is invalid return TPM_NO_ENDORSEMENT
4. Verify that authHandle is of type OIAP on error return TPM_AUTHFAIL
5. If protocolID is not TPM_PID_OWNER, the TPM MAY return TPM_BAD_PARAMETER
6. Create A1 a TPM_SECRET by decrypting encOwnerAuth using PRIVEK as the key
 - a. This requires that A1 was encrypted using the PUBEK
 - b. Validate that A1 is a length of 20 bytes, on error return TPM_BAD_KEY_PROPERTY
7. Validate the command and parameters using A1 and ownerAuth, on error return TPM_AUTHFAIL
8. Validate srkParams
 - a. If srkParams -> keyUsage is not TPM_KEY_STORAGE return TPM_INVALID_KEYUSAGE
 - b. If srkParams -> migratable is TRUE return TPM_INVALID_KEYUSAGE
 - c. If srkParams -> algorithmParms -> algorithmID is NOT TPM_ALG_RSA return TPM_BAD_KEY_PROPERTY
 - d. If srkParams -> algorithmParms -> encScheme is NOT TPM_ES_RSAESOAEP_SHA1_MGF1 return TPM_BAD_KEY_PROPERTY
 - e. If srkParams -> algorithmParms -> sigScheme is NOT TPM_SS_NONE return TPM_BAD_KEY_PROPERTY
 - f. If srkParams -> algorithmParms -> parms -> keyLength MUST be greater than or equal to 2048, on error return TPM_BAD_KEY_PROPERTY
 - g. If TPM_PERMANENT_FLAGS -> FIPS is TRUE
 - i. If srkParams -> authDataUsage specifies TPM_AUTH_NEVER return TPM_NOTFIPS
9. Generate K1 according to the srkParams, on error return TPM_BAD_KEY_PROPERTY
 - a. This includes copying PCRInfo from srkParams to K1

10. Create A2 a TPM_SECRET by decrypting encSrkJAuth using the PRIVEK
 - a. This requires A2 to be encrypted using the PUBEK
 - b. Validate that A2 is a length of 20 bytes, on error return TPM_BAD_KEY_PROPERTY
 - c. Store A2 in K1 -> usageAuth
11. Store K1 in TPM_PERMANENT_DATA -> srk
12. Store A1 in TPM_PERMANENT_DATA -> ownerAuth
13. Create TPM_PERMANENT_DATA -> contextKey according to the rules for the algorithm in use by the TPM to save context blobs
14. Create TPM_PERMANENT_DATA -> delegateKey according to the rules for the algorithm in use by the TPM to save delegate blobs
15. Create TPM_PERMANENT_DATA -> tpmProof by using the TPM RNG
16. Export TPM_PERMANENT_DATA -> srk as srkPub
17. Set TPM_PERMANENT_FLAGS -> readPubek to FALSE
18. Calculate resAuth using the newly established TPM_PERMANENT_DATA -> ownerAuth

7.2 TPM_OwnerClear

Start of informative comment:

The TPM_OwnerClear command performs the clear operation under Owner authentication. This command is available until the Owner executes the TPM_DisableOwnerClear, at which time any further invocation of this command returns TPM_CLEAR_DISABLED.

End of informative comment.

Table 27. TPM_OwnerClear Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_OwnerClear
4	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for owner authentication.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
5	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
6	1	4H1	1	BOOL	continueAuthSession	Ignored
7	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authentication. HMAC key: ownerAuth.

Table 28. TPM_OwnerClear Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_OwnerClear
4	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
5	1	4H1	1	BOOL	continueAuthSession	Fixed value FALSE
6	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: old ownerAuth.

Actions

1. Verify that the TPM Owner authorizes the command and all of the input, on error return TPM_AUTHFAIL.
2. If TPM_PERMANENT_FLAGS -> disableOwnerClear is TRUE then return TPM_CLEAR_DISABLED.
3. Unload all loaded keys.
 - a. If TPM_PERMANENT_FLAGS -> FIPS is TRUE, the memory locations containing secret or private keys MUST be set to all zeros.
4. The TPM MUST NOT modify the following TPM_PERMANENT_DATA items
 - a. endorsementKey
 - b. revMajor
 - c. revMinor
 - d. manuMaintPub
 - e. auditMonotonicCounter
 - f. monotonicCounter
 - g. pcrAttrib
 - h. rngState
 - i. EKReset
 - j. maxNVBufSize
 - k. lastFamilyID
 - l. tpmDAASeed
 - m. authDIR[0]
 - n. daaProof
 - o. daaBlobKey
5. The TPM MUST invalidate the following TPM_PERMANENT_DATA items and any internal resources associated with these items
 - a. ownerAuth
 - b. srk
 - c. delegateKey
 - d. delegateTable
 - e. contextKey
 - f. tpmProof
 - g. operatorAuth
6. The TPM MUST reset to manufacturing defaults the following TPM_PERMANENT_DATA items
 - a. noOwnerNVWrite MUST be set to 0
 - b. ordinalAuditStatus
 - c. restrictDelegate
7. The TPM MUST invalidate or reset all fields of TPM_STANY_DATA
 - a. Nonces SHALL be reset
 - b. Lists (e.g. contextList) SHALL be invalidated

8. The TPM MUST invalidate or reset all fields of TPM_STCLEAR_DATA except the PCR's
 - a. Nonces SHALL be reset
 - b. Lists (e.g. contextList) SHALL be invalidated
 - c. deferredPhysicalPresence MUST be set to 0
9. The TPM MUST set the following TPM_PERMANENT_FLAGS to their default values
 - a. disable
 - b. deactivated
 - c. readPubek
 - d. disableOwnerClear
 - e. disableFullIDALogicInfo
10. The TPM MUST set the following TPM_PERMANENT_FLAGS
 - a. ownership to TRUE
 - b. operator to FALSE
 - c. maintenanceDone to FALSE
 - d. allowMaintenance to TRUE
11. The TPM releases all TPM_PERMANENT_DATA -> monotonicCounter settings
 - a. This includes invalidating all currently allocated counters. The result will be no currently allocated counters and the new owner will need to allocate counters. The actual count value will continue to increase.
12. The TPM MUST deallocate all defined NV storage areas where
 - a. TPM_NV_PER_OWNERWRITE is TRUE if nvIndex does not have the "D" bit set
 - b. TPM_NV_PER_OWNERREAD is TRUE if nvIndex does not have the "D" bit set
 - c. The TPM MUST NOT deallocate any other currently defined NV storage areas.
13. The TPM MUST invalidate all familyTable entries
14. The TPM MUST terminate all sessions, active or saved.

7.3 TPM_ForceClear

Start of informative comment:

The TPM_ForceClear command performs the Clear operation under physical access. This command is available until the execution of the TPM_DisableForceClear, at which time any further invocation of this command returns TPM_CLEAR_DISABLED.

End of informative comment.

Table 29. TPM_ForceClear Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_ForceClear

Table 30. TPM_ForceClear Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_ForceClear

Actions

1. The TPM SHALL check for the assertion of physical presence, if not present return TPM_BAD_PRESENCE
2. If TPM_STCLEAR_FLAGS -> disableForceClear is TRUE return TPM_CLEAR_DISABLED
3. The TPM SHALL execute the actions of TPM_OwnerClear (except for the TPM_Owner-Authentication check)

7.4 TPM_DisableOwnerClear

Start of informative comment:

The TPM_DisableOwnerClear command disables the ability to execute the TPM_OwnerClear command permanently. Once invoked the only method of clearing the TPM will require physical access to the TPM.

After the execution of TPM_ForceClear, ownerClear is re-enabled and must be explicitly disabled again by the new TPM Owner.

End of informative comment.

Table 31. TPM_DisableOwnerClear Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_DisableOwnerClear
4	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for owner authentication.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
5	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
6	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
7	20			TPM_AUTHDATA	ownerAuth	The authorization session digest for inputs and owner authentication. HMAC key: ownerAuth.

Table 32. TPM_DisableOwnerClear Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_DisableOwnerClear
4	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
5	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
6	20			TPM_AUTHDATA	resAuth	The authorization session digest for the returned parameters. HMAC key: ownerAuth.

Actions

1. The TPM verifies that the authHandle properly authorizes the owner.
2. The TPM sets the TPM_PERMANENT_FLAGS -> disableOwnerClear flag to TRUE.
3. When this flag is TRUE the only mechanism that can clear the TPM is the TPM_ForceClear command. The TPM_ForceClear command requires physical access to the TPM to execute.

7.5 TPM_DisableForceClear

Start of informative comment:

The TPM_DisableForceClear command disables the execution of the TPM_ForceClear command until the next startup cycle. Once this command is executed, the TPM_ForceClear is disabled until another startup cycle is run.

End of informative comment.

Table 33. TPM_DisableForceClear Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_DisableForceClear

Table 34. TPM_DisableForceClear Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_DisableForceClear

Actions

1. The TPM sets the TPM_STCLEAR_FLAGS.disableForceClear flag in the TPM that disables the execution of the TPM_ForceClear command.

7.6 TSC_PhysicalPresence

Start of informative comment:

Some TPM operations require the indication of a human's physical presence at the platform. The presence of the human either provides another indication of platform ownership or a mechanism to ensure that the execution of the command is not the result of a remote software process.

This command allows a process on the platform to indicate the assertion of physical presence. As this command is executable by software there must be protections against the improper invocation of this command.

The physicalPresenceHwEnable and physicalPresenceCmdEnable indicate the ability for either SW or HW to indicate physical presence. These flags can be reset until the physicalPresenceLifetimeLock is set. The platform manufacturer should set these flags to indicate the capabilities of the platform the TPM is bound to.

The command provides two sets of functionality. The first is to enable, permanently, either the HW or the SW ability to assert physical presence. The second is to allow SW, if enabled, to assert physical presence.

End of informative comment.

Table 35. TSC_PhysicalPresence Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TSC_ORD_PhysicalPresence.
4	2	2S	2	TPM_PHYSICAL_PRESENCE	physicalPresence	The state to set the TPM's Physical Presence flags.

Table 36. TSC_PhysicalPresence Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TSC_ORD_PhysicalPresence.

Actions

1. For documentation ease, the bits break into two categories. The first is the lifetime settings and the second is the assertion settings.
 - a. Define A1 to be the lifetime settings: TPM_PHYSICAL_PRESENCE_LIFETIME_LOCK, TPM_PHYSICAL_PRESENCE_HW_ENABLE, TPM_PHYSICAL_PRESENCE_CMD_ENABLE, TPM_PHYSICAL_PRESENCE_HW_DISABLE, and TPM_PHYSICAL_PRESENCE_CMD_DISABLE
 - b. Define A2 to be the assertion settings: TPM_PHYSICAL_PRESENCE_LOCK, TPM_PHYSICAL_PRESENCE_PRESENT, and TPM_PHYSICAL_PRESENCE_NOTPRESENT

Lifetime lock settings

2. If any A1 setting is present
 - a. If TPM_PERMANENT_FLAGS -> physicalPresenceLifetimeLock is TRUE, return TPM_BAD_PARAMETER
 - b. If any A2 setting is present return TPM_BAD_PARAMETER
 - c. If both physicalPresence -> TPM_PHYSICAL_PRESENCE_HW_ENABLE and physicalPresence -> TPM_PHYSICAL_PRESENCE_HW_DISABLE are TRUE, return TPM_BAD_PARAMETER.
 - d. If both physicalPresence -> TPM_PHYSICAL_PRESENCE_CMD_ENABLE and physicalPresence -> TPM_PHYSICAL_PRESENCE_CMD_DISABLE are TRUE, return TPM_BAD_PARAMETER.
 - e. If physicalPresence -> TPM_PHYSICAL_PRESENCE_HW_ENABLE is TRUE Set TPM_PERMANENT_FLAGS -> physicalPresenceHwEnable to TRUE
 - f. If physicalPresence -> TPM_PHYSICAL_PRESENCE_HW_DISABLE is TRUE Set TPM_PERMANENT_FLAGS -> physicalPresenceHwEnable to FALSE
 - g. If physicalPresence -> TPM_PHYSICAL_PRESENCE_CMD_ENABLE is TRUE, Set TPM_PERMANENT_FLAGS -> physicalPresenceCmdEnable to TRUE.
 - h. If physicalPresence -> TPM_PHYSICAL_PRESENCE_CMD_DISABLE is TRUE, Set TPM_PERMANENT_FLAGS -> physicalPresenceCmdEnable to FALSE.
 - i. If physicalPresence -> TPM_PHYSICAL_PRESENCE_LIFETIME_LOCK is TRUE
 - i. Set TPM_PERMANENT_FLAGS -> physicalPresenceLifetimeLock to TRUE
 - j. Return TPM_SUCCESS

SW physical presence assertion

3. If any A2 setting is present
 - a. If any A1 setting is present return TPM_BAD_PARAMETER
 - i. This check here just for consistency, the prior checks would have already ensured that this was ok
 - b. If TPM_PERMANENT_FLAGS -> physicalPresenceCMDEnable is FALSE, return TPM_BAD_PARAMETER
 - c. If both physicalPresence -> TPM_PHYSICAL_PRESENCE_LOCK and physicalPresence -> TPM_PHYSICAL_PRESENCE_PRESENT are TRUE, return TPM_BAD_PARAMETER
 - d. If both physicalPresence -> TPM_PHYSICAL_PRESENCE_PRESENT and physicalPresence -> TPM_PHYSICAL_PRESENCE_NOTPRESENT are TRUE, return TPM_BAD_PARAMETER
 - e. If TPM_STCLEAR_FLAGS -> physicalPresenceLock is TRUE, return TPM_BAD_PARAMETER
 - f. If physicalPresence -> TPM_PHYSICAL_PRESENCE_LOCK is TRUE
 - i. Set TPM_STCLEAR_FLAGS -> physicalPresence to FALSE
 - ii. Set TPM_STCLEAR_FLAGS -> physicalPresenceLock to TRUE
 - iii. Return TPM_SUCCESS
 - g. If physicalPresence -> TPM_PHYSICAL_PRESENCE_PRESENT is TRUE
 - i. Set TPM_STCLEAR_FLAGS -> physicalPresence to TRUE
 - h. If physicalPresence -> TPM_PHYSICAL_PRESENCE_NOTPRESENT is TRUE
 - i. Set TPM_STCLEAR_FLAGS -> physicalPresence to FALSE
 - i. Return TPM_SUCCESS
4. Else // There were no A1 or A2 parameters set
 - a. Return TPM_BAD_PARAMETER

7.7 TSC_ResetEstablishmentBit**Start of informative comment:**

The PC TPM Interface Specification (TIS) specifies setting tpmEstablished to TRUE upon execution of the HASH_START sequence. The setting implies the creation of a Trusted Operating System on the platform. Platforms will use the value of tpmEstablished to determine if operations necessary to maintain the security perimeter are necessary.

The tpmEstablished bit provides a non-volatile, secure reporting that a HASH_START was previously run on the platform. When a platform makes use of the tpmEstablished bit, the platform can reset tpmEstablished as the operation is no longer necessary.

For example, a platform could use tpmEstablished to ensure that, if HASH_START had ever been, executed the platform could use the value to invoke special processing. Once the processing is complete the platform will wish to reset tpmEstablished to avoid invoking the special process again.

The TPM_PERMANENT_FLAGS -> tpmEstablished bit described in the TPM specifications uses positive logic. The TPM_ACCESS register uses negative logic, so that TRUE is reflected as a 0.

End of informative comment.

Table 37. TCG_ResetEstablishmentBit Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TSC_ORD_ResetEstablishmentBit

Table 38. TCG_ResetEstablishmentBit Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TSC_ORD_ResetEstablishmentBit

Actions

1. Validate the assertion of locality 3 or locality 4
2. Set TPM_PERMANENT_FLAGS -> tpmEstablished to FALSE
3. Return TPM_SUCCESS

8. The Capability Commands

Start of informative comment:

The TPM has numerous capabilities that a remote entity may wish to know about. These items include support of algorithms, key sizes, protocols and vendor-specific additions. The TPM_GetCapability command allows the TPM to report back to the requestor what type of TPM it is dealing with.

The request for information requires the requestor to specify which piece of information that is required. The request does not allow the “merging” of multiple requests and returns only a single piece of information.

In failure mode, the TPM returns a limited set of information that includes the TPM manufacturer and version.

In version 1.2 with the deletion of TPM_GetCapabilitySigned the way to obtain a signed listing of the capabilities is to create a transport session, perform TPM_GetCapability commands to list the information and then close the transport session using TPM_ReleaseTransportSigned.

End of informative comment.

4. The standard information provided in TPM_GetCapability MUST NOT provide unique information
5. The TPM has no control of information placed into areas on the TPM like the NV store that is reported by the TPM. Configuration information for these areas could conceivably be unique

8.1 TPM_GetCapability

Start of informative comment:

This command returns current information regarding the TPM.

The limitation on what can be returned in failure mode restricts the information a manufacturer may return when capArea indicates TPM_CAP_MFR.

End of informative comment.

Table 39. TPM_GetCapability Incoming Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_GetCapability
4	4	2S	4	TPM_CAPABILITY_AREA	capArea	Partition of capabilities to be interrogated
5	4	3S	4	UINT32	subCapSize	Size of subCap parameter
6	<>	4S	<>	BYTE[]	subCap	Further definition of information

Table 40. TPM_GetCapability Outgoing Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_GetCapability
4	4	3S	4	UINT32	respSize	The length of the returned capability response
5	<>	4S	<>	BYTE[]	resp	The capability response

Actions

1. The TPM validates the capArea and subCap indicators. If the information is available, the TPM creates the response field and fills in the actual information.
2. The structure document contains the list of caparea and subCap values
3. If the TPM is in failure mode or limited operation mode, the TPM MUST return
 - a. TPM_CAP_VERSION
 - b. TPM_CAP_VERSION_VAL
 - c. TPM_CAP_MFR
 - d. TPM_CAP_PROPERTY -> TPM_CAP_PROP_MANUFACTURER
 - e. TPM_CAP_PROPERTY -> TPM_CAP_PROP_DURATION
 - f. TPM_CAP_PROPERTY -> TPM_CAP_PROP_TIS_TIMEOUT
 - g. The TPM MAY return any other capability.

8.2 TPM_SetCapability

Start of informative comment:

This command sets values in the TPM.

A setValue that is inconsistent with the capArea and subCap is considered a bad parameter.

End of informative comment.

Table 41. TPM_SetCapability Incoming Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	ordinal: TPM_ORD_SetCapability
4	4	2S	4	TPM_CAPABILITY_AREA	capArea	Partition of capabilities to be set
5	4	3S	4	UINT32	subCapSize	Size of subCap parameter
6	<>	4S	<>	BYTE[]	subCap	Further definition of information
7	4	5S	4	UINT32	setValueSize	The size of the value to set
8	<>	6S	<>	BYTE[]	setValue	The value to set
9	4			TPM_AUTHHANDLE	authHandle	The authorization session handle used for owner authentication.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
10	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
11	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization session handle
12	20			TPM_AUTHDATA	ownerAuth	Authorization. HMAC key: owner.usageAuth.

Table 42. TPM_SetCapability Outgoing Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation.
		2S	4	TPM_COMMAND_CODE	ordinal	ordinal: TPM_ORD_SetCapability
4	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
5	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
6	20			TPM_AUTHDATA	resAuth	Authorization HMAC key:owner.usageAuth.

Actions

1. If tag = TPM_TAG_RQU_AUTH1_COMMAND, validate the command and parameters using ownerAuth, return TPM_AUTHFAIL on error
2. The TPM validates the capArea and subCap indicators, including the ability to set value based on any set restrictions
3. If the capArea and subCap indicators conform with one of the entries in the structure TPM_CAPABILITY_AREA (Values for TPM_SetCapability)
 - a. The TPM sets the relevant flag/data to the value of setValue parameter.
4. Else
 - a. Return the error code TPM_BAD_PARAMETER.

8.3 TPM_GetCapabilityOwner

Start of informative comment:

TPM_GetCapabilityOwner enables the TPM Owner to retrieve all the non-volatile flags and the volatile flags in a single operation.

The flags summarize many operational aspects of the TPM. The information represented by some flags is private to the TPM Owner. So, for simplicity, proof of ownership of the TPM must be presented to retrieve the set of flags. When necessary, the flags that are not private to the Owner can be deduced by Users via other (more specific) means.

The normal TPM authentication mechanisms are sufficient to prove the integrity of the response. No additional integrity check is required.

End of informative comment.

Table 43. TPM_GetCapabilityOwner Incoming Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	1S	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_GetCapabilityOwner
4	4			TPM_AUTHHANDLE	authHandle	The authorization handle used for Owner authorization.
		2H1	20	TPM_NONCE	authLastNonceEven	Even nonce previously generated by TPM to cover inputs
5	20	3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
6	1	4H1	1	BOOL	continueAuthSession	The continue use flag for the authorization handle
7	20			TPM_AUTHDATA	ownerAuth	The authorization digest for inputs and owner authorization. HMAC key: OwnerAuth.

Table 44. TPM_GetCapabilityOwner Outgoing Operands and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_AUTH1_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4	1S	4	TPM_RESULT	returnCode	The return code of the operation. See section 4.3.
		2S	4	TPM_COMMAND_CODE	ordinal	Ordinal: TPM_ORD_GetCapabilityOwner
4	4	3S	4	TPM_VERSION	version	A properly filled out version structure.
5	4	4S	4	UINT32	non_volatile_flags	The current state of the non-volatile flags.
6	4	5S	4	UINT32	volatile_flags	The current state of the volatile flags.
7	20	2H1	20	TPM_NONCE	nonceEven	Even nonce newly generated by TPM to cover outputs
		3H1	20	TPM_NONCE	nonceOdd	Nonce generated by system associated with authHandle
8	1	4H1	1	BOOL	continueAuthSession	Continue use flag, TRUE if handle is still active
9	20			TPM_AUTHDATA	resAuth	The authorization digest for the returned parameters. HMAC key: OwnerAuth.

Description

For $31 \geq N \geq 0$

1. Bit-N of the TPM_PERMANENT_FLAGS structure is the Nth bit after the opening bracket in the definition of TPM_PERMANENT_FLAGS in the version of ISO/IEC 11889 indicated by the parameter “version”. The bit immediately after the opening bracket is the 0th bit.
2. Bit-N of the TPM_STCLEAR_FLAGS structure is the Nth bit after the opening bracket in the definition of TPM_STCLEAR_FLAGS in the version of ISO/IEC 11889 indicated by the parameter “version”. The bit immediately after the opening bracket is the 0th bit.
3. Bit-N of non_volatile_flags corresponds to the Nth bit in TPM_PERMANENT_FLAGS, and the lsb of non_volatile_flags corresponds to bit0 of TPM_PERMANENT_FLAGS
4. Bit-N of volatile_flags corresponds to the Nth bit in TPM_STCLEAR_FLAGS, and the lsb of volatile_flags corresponds to bit0 of TPM_STCLEAR_FLAGS

Actions

1. The TPM validates that the TPM Owner authorizes the command.
2. The TPM creates the parameter non_volatile_flags by setting each bit to the same state as the corresponding bit in TPM_PERMANENT_FLAGS. Bits in non_volatile_flags for which there is no corresponding bit in TPM_PERMANENT_FLAGS are set to zero.
3. The TPM creates the parameter volatile_flags by setting each bit to the same state as the corresponding bit in TPM_STCLEAR_FLAGS. Bits in volatile_flags for which there is no corresponding bit in TPM_STCLEAR_FLAGS are set to zero.
4. The TPM generates the parameter “version”.
5. The TPM returns non_volatile_flags, volatile_flags and version to the caller.

9. Auditing

9.1 Audit Generation

Start of informative comment:

The TPM generates an audit event in response to the TPM executing a function that has the audit flag set to TRUE for that function.

The TPM maintains an extended value for all audited operations.

Input audit generation occurs before the listed actions and output audit generation occurs after the listed actions.

End of informative comment.

Description

1. The TPM extends the audit digest whenever the ordinalAuditStatus is TRUE for the ordinal about to be executed. The only exception is if the ordinal about to be executed is TPM_SetOrdinalAuditStatus. In that case, output parameter auditing is performed if the ordinalAuditStatus resulting from command execution is TRUE.
2. If the command is malformed
 - a. If the ordinal is unknown, unimplemented, or cannot be determined, no auditing is performed.
 - b. If the ordinal is known and audited, but the “above the line” parameters are malformed and the input parameter digest cannot be determined, use an input digest of all zeros.
 - i. Use an output digest of the return code and ordinal.
 - c. If the ordinal is known and audited, the “above the line” parameters are determined, but the “below the line” parameters are malformed, use an input digest of the “above the line” parameters.
 - i. Use an output digest of the return code and ordinal.
 - d. Malformed in this context means that, when breaking up a command into its parameters, there are too few or too many bytes in the command stream.
 - e. Breaking up a command in this context means only the parsing required to extract the parameters.
 - i. E.g., for parameter set comprising a UINT32 size and a BYTE[] array, the BYTE[] array should not be further parsed.
 - f. If the ordinal returns an error because the TPM is deactivated, disabled, or has no owner, auditing is performed.
 - g. If the ordinal returns an error because the input tag is invalid for the command, auditing is performed.

Actions

The TPM will execute the ordinal and perform auditing in the following manner

1. Map V1 to TPM_STANY_DATA
2. Map P1 to TPM_PERMANENT_DATA
3. If V1 -> auditDigest is all zeros
 - a. Increment P1 -> auditMonotonicCounter by 1

4. Create A1 a TPM_AUDIT_EVENT_IN structure
 - a. Set A1 -> inputParms to the digest of the input parameters from the command
 - i. Digest value according to the HMAC digest rules of the "above the line" parameters (i.e. the first HMAC digest calculation).
 - b. Set A1 -> auditCount to P1 -> auditMonotonicCounter
 - c. Set V1 -> auditDigest to SHA-1 (V1 -> auditDigest || A1)
5. Execute command
 - a. Execution implies the performance of the listed actions for the ordinal.
6. Create A2 a TPM_AUDIT_EVENT_OUT structure
 - a. Set A2 -> outputParms to the digest of the output parameters from the command
 - i. Digest value according to the HMAC digest rules of the "above the line" parameters (i.e. the first HMAC digest calculation).
 - b. Set A2 -> auditCount to P1 -> auditMonotonicCounter
 - c. Set V1 -> auditDigest to SHA-1 (V1 -> auditDigest || A2)

9.2 Effect of audit failing

Start of informative comment:

The TPM audit process could have an internal error when attempting to audit a command.

With one return parameter, The TPM is unable to return both the audit failure and the command success or failure results. To indicate the audit failure, the TPM will return one of two error codes: TPM_AUDITFAIL_SUCCESSFUL (if the command completed successfully) or TPM_AUDITFAIL_UNSUCCESSFUL (if the command completed unsuccessfully).

This new functionality changes the 1.1 TPM functionality when this condition occurs.

End of informative comment.

1. When, in performing the audit process, the TPM has an internal failure (unable to write, SHA-1 failure etc.) the TPM MUST set the internal TPM state such that the TPM returns the TPM_FAILEDSELFTEST error on subsequent attempts to execute a command
2. The return code for the command uses the following rules
 - a. Command result success, Audit success -> return TPM_SUCCESS
 - b. Command result failure, Audit success -> return command result failure
 - c. Command result success, Audit failure -> return TPM_AUDITFAIL_SUCCESSFUL
 - d. Command result failure, Audit failure -> return TPM_AUDITFAIL_UNSUCCESSFUL
3. If the TPM is permanently nonrecoverable after an audit failure, then the TPM MUST always return TPM_FAILEDSELFTEST for every command other than TPM_GetTestResult. This state must persist regardless of power cycling, the execution of TPM_Init or any other actions.

9.3 TPM_GetAuditDigest

Start of informative comment:

This returns the current audit digest. The external audit log has the responsibility to track the parameters that constitute the audit digest.

This value may be unique to an individual TPM. The value however will be changing at a rate set by the TPM Owner. Those attempting to use this value may find it changing without their knowledge. This value represents a very poor source of tracking uniqueness.

End of informative comment.

Table 45. TPM_GetAuditDigest Incoming Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4			TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_GetAuditDigest
4	4			UINT32	startOrdinal	The starting ordinal for the list of audited ordinals

Table 46. TPM_GetAuditDigest Outgoing Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	Tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4			TPM_RESULT	returnCode	The return code of the operation.
5	10			TPM_COUNTER_VALUE	counterValue	The current value of the audit monotonic counter
4	20			TPM_DIGEST	auditDigest	Log of all audited events
5	1			BOOL	more	TRUE if the output does not contain a full list of audited ordinals
5	4			UINT32	ordSize	Size of the ordinal list in bytes
6	<>			UINT32[]	ordList	List of ordinals that are audited.

Description

1. This command is never audited.

Actions

1. The TPM sets auditDigest to TPM_STANY_DATA -> auditDigest
2. The TPM sets counterValue to TPM_PERMANENT_DATA -> auditMonotonicCounter
3. The TPM creates an ordered list of audited ordinals. The list starts at startOrdinal listing each ordinal that is audited.
 - a. If startOrdinal is 0 then the first ordinal that could be audited would be TPM_OIAP (ordinal 0x0000000A)
 - b. The next ordinal would be TPM_OSAP (ordinal 0x0000000B)
4. If the ordered list does not fit in the output buffer the TPM sets more to TRUE
5. Return TPM_STANY_DATA -> auditDigest as auditDigest