

RX ファミリ

R20AN0548JJ0115

Rev.1.15

2022.03.31

TSIP(Trusted Secure IP)モジュール Firmware Integration Technology
(バイナリ版)

要旨

本資料は、RX ファミリ搭載の TSIP(Trusted Secure IP) および TSIP-Lite を活用するためのソフトウェア・ドライバの使用方法を記します。このソフトウェア・ドライバは TSIP ドライバと呼びます。TSIP ドライバは表 1 にまとめた暗号機能、およびファームウェアアップデートをセキュアに行うための API を持ちます。

表 1 各種暗号アルゴリズム

| | | TSIP-Lite(注 1) | TSIP(注 2) |
|--------------|---------|-------------------------------------|---|
| 公開鍵暗号 | 暗号化/復号 | - | RSAES-PKCS1-v1_5 |
| | 署名生成/検証 | - | RSASSA-PKCS1-v1_5, ECDSA |
| | 鍵生成 | - | RSA(1024/2048 bit), ECC P-192/224/256/384 |
| 共通鍵暗号 | AES | AES(128/256 bit) ECB/CBC/GCM/CCM | AES(128/256 bit) ECB/CBC/GCM/CCM |
| | DES | - | Triple-DES(56/56x2/56x3 bit) ECB/CBC |
| | ARC4 | - | ARC4(2048 bit) |
| ハッシュ | SHA | - | SHA-1, SHA-256 |
| | MD5 | - | MD5 |
| メッセージ認証 | | CMAC(AES), GMAC | CMAC(AES), GMAC, HMAC(SHA) |
| 疑似乱数ビット生成 | | SP 800-90A | SP 800-90A |
| 乱数生成 | | SP 800-22 で検定済み | SP 800-22 で検定済み |
| SSL/TLS 連携機能 | | - | TLS1.2, TLS1.3 準拠 サポートしている cipher suite(TLS1.2): TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 サポートしている cipher suite(TLS1.3) (注 3): TLS_AES_128_GCM_SHA256 TLS_AES_128_CCM_SHA256 |
| 鍵更新機能 | | AES | AES, RSA, DES, ARC4, ECC, HMAC |
| 鍵共有 | | - | ECDH P-256, ECDHE P-512, DH(2048 bit) |
| Key Wrap | | AES(128/256 bit) | AES(128/256 bit) |

- 【注】 1. 対象デバイスは、RX231 グループ、RX23W グループ、RX66T グループ、RX72T グループです。
2. 対象デバイスは、RX65N, RX651 グループ、RX66N グループ、RX671 グループ、RX72M グループ、RX72N グループです。
3. 対象デバイスは RX65N, RX651 グループ、RX66N グループ、RX72M グループ、RX72N グループです。

TSIP ドライバは、Firmware Integration Technology(FIT)モジュールとして提供されます。FIT の概念については以下 URL を参照してください。

<https://www.renesas.com/jp/ja/products/software-tools/software-os-middleware-driver/software-package/fit.html>

動作確認デバイス

RX231 グループ、RX23W グループ、RX65N, RX651 グループ、RX66T グループ、RX671 グループ、RX72M グループ、RX72N グループ、RX72T グループ

TSIP 機能がある製品型名については各 RX マイコンのユーザーズマニュアルを参照してください。

RX ファミ리에搭載される TSIP ドライバの詳細について書かれたアプリケーションノートおよびソースファイルを別途ご用意しています。

また、本アプリケーションノートではサンプルの鍵を使って説明しています。量産等に適用する場合は独自の鍵を生成する必要があり、それらの詳細が書かれたアプリケーションノートを別途ご用意しています。

ルネサスマイコンをご採用/ご採用予定のお客様にご提供させていただいておりますので、お取引のあるルネサスエレクトロニクス営業窓口にお問合せください。

<https://www.renesas.com/contact/>

目次

| | |
|--|----|
| 1. 概要 | 9 |
| 1.1 用語 | 9 |
| 1.2 TSIP 概要 | 11 |
| 1.3 製品構成 | 12 |
| 1.4 開発環境 | 14 |
| 1.5 コードサイズ | 15 |
| 1.6 セクション情報 | 15 |
| 1.7 性能情報(RX231) | 16 |
| 1.8 性能情報(RX23W) | 19 |
| 1.9 性能情報(RX66T) | 22 |
| 1.10 性能情報(RX72T) | 25 |
| 1.11 性能情報(RX65N) | 28 |
| 1.12 性能情報(RX671) | 36 |
| 1.13 性能情報(RX72M) | 44 |
| 1.14 性能情報(RX72N) | 52 |
| 2. API 情報 | 60 |
| 2.1 ハードウェアの要求 | 60 |
| 2.2 ソフトウェアの要求 | 60 |
| 2.3 サポートされているツールチェーン | 61 |
| 2.4 ヘッダファイル | 61 |
| 2.5 整数型 | 61 |
| 2.6 API データ構造 | 61 |
| 2.7 戻り値 | 62 |
| 2.8 FIT モジュールの追加方法 | 63 |
| 3. API 関数 | 64 |
| 3.1 API 一覧 | 64 |
| 3.2 状態遷移図 | 76 |
| 3.3 API 使用時の注意事項 | 77 |
| 3.3.1 各 API の呼び出し方法 | 77 |
| 3.3.2 BSP FIT モジュールに関する注意事項 | 78 |
| 4. API 関数詳細説明(TSIP-Lite/TSIP 共通) | 79 |
| 4.1 R_TSIP_Open | 79 |
| 4.2 R_TSIP_Close | 80 |
| 4.3 R_TSIP_SoftwareReset | 81 |
| 4.4 R_TSIP_GetVersion | 82 |
| 4.5 R_TSIP_GenerateAes128KeyIndex | 83 |
| 4.6 R_TSIP_GenerateAes256KeyIndex | 84 |
| 4.7 R_TSIP_GenerateUpdateKeyRingKeyIndex | 85 |
| 4.8 R_TSIP_UpdateAes128KeyIndex | 86 |
| 4.9 R_TSIP_UpdateAes256KeyIndex | 87 |
| 4.10 R_TSIP_GenerateAes128RandomKeyIndex | 88 |
| 4.11 R_TSIP_GenerateAes256RandomKeyIndex | 89 |
| 4.12 R_TSIP_GenerateRandomNumber | 90 |

| | | |
|------|-------------------------------------|-----|
| 4.13 | R_TSIP_StartUpdateFirmware | 91 |
| 4.14 | R_TSIP_GenerateFirmwareMAC | 92 |
| 4.15 | R_TSIP_VerifyFirmwareMAC | 96 |
| 4.16 | R_TSIP_Aes128EcbEncryptInit | 97 |
| 4.17 | R_TSIP_Aes128EcbEncryptUpdate | 98 |
| 4.18 | R_TSIP_Aes128EcbEncryptFinal | 99 |
| 4.19 | R_TSIP_Aes128EcbDecryptInit | 100 |
| 4.20 | R_TSIP_Aes128EcbDecryptUpdate | 101 |
| 4.21 | R_TSIP_Aes128EcbDecryptFinal | 102 |
| 4.22 | R_TSIP_Aes256EcbEncryptInit | 103 |
| 4.23 | R_TSIP_Aes256EcbEncryptUpdate | 104 |
| 4.24 | R_TSIP_Aes256EcbEncryptFinal | 105 |
| 4.25 | R_TSIP_Aes256EcbDecryptInit | 106 |
| 4.26 | R_TSIP_Aes256EcbDecryptUpdate | 107 |
| 4.27 | R_TSIP_Aes256EcbDecryptFinal | 108 |
| 4.28 | R_TSIP_Aes128CbcEncryptInit | 109 |
| 4.29 | R_TSIP_Aes128CbcEncryptUpdate | 110 |
| 4.30 | R_TSIP_Aes128CbcEncryptFinal | 111 |
| 4.31 | R_TSIP_Aes128CbcDecryptInit | 112 |
| 4.32 | R_TSIP_Aes128CbcDecryptUpdate | 113 |
| 4.33 | R_TSIP_Aes128CbcDecryptFinal | 114 |
| 4.34 | R_TSIP_Aes256CbcEncryptInit | 115 |
| 4.35 | R_TSIP_Aes256CbcEncryptUpdate | 116 |
| 4.36 | R_TSIP_Aes256CbcEncryptFinal | 117 |
| 4.37 | R_TSIP_Aes256CbcDecryptInit | 118 |
| 4.38 | R_TSIP_Aes256CbcDecryptUpdate | 119 |
| 4.39 | R_TSIP_Aes256CbcDecryptFinal | 120 |
| 4.40 | R_TSIP_Aes128GcmEncryptInit | 121 |
| 4.41 | R_TSIP_Aes128GcmEncryptUpdate | 122 |
| 4.42 | R_TSIP_Aes128GcmEncryptFinal | 123 |
| 4.43 | R_TSIP_Aes128GcmDecryptInit | 124 |
| 4.44 | R_TSIP_Aes128GcmDecryptUpdate | 125 |
| 4.45 | R_TSIP_Aes128GcmDecryptFinal | 126 |
| 4.46 | R_TSIP_Aes256GcmEncryptInit | 127 |
| 4.47 | R_TSIP_Aes256GcmEncryptUpdate | 128 |
| 4.48 | R_TSIP_Aes256GcmEncryptFinal | 129 |
| 4.49 | R_TSIP_Aes256GcmDecryptInit | 130 |
| 4.50 | R_TSIP_Aes256GcmDecryptUpdate | 131 |
| 4.51 | R_TSIP_Aes256GcmDecryptFinal | 132 |
| 4.52 | R_TSIP_Aes128CcmEncryptInit | 133 |
| 4.53 | R_TSIP_Aes128CcmEncryptUpdate | 134 |
| 4.54 | R_TSIP_Aes128CcmEncryptFinal | 135 |
| 4.55 | R_TSIP_Aes128CcmDecryptInit | 136 |
| 4.56 | R_TSIP_Aes128CcmDecryptUpdate | 137 |
| 4.57 | R_TSIP_Aes128CcmDecryptFinal | 138 |
| 4.58 | R_TSIP_Aes256CcmEncryptInit | 139 |
| 4.59 | R_TSIP_Aes256CcmEncryptUpdate | 140 |

| | | |
|------|-----------------------------------|-----|
| 4.60 | R_TSIP_Aes256CcmEncryptFinal | 141 |
| 4.61 | R_TSIP_Aes256CcmDecryptInit | 142 |
| 4.62 | R_TSIP_Aes256CcmDecryptUpdate | 143 |
| 4.63 | R_TSIP_Aes256CcmDecryptFinal | 144 |
| 4.64 | R_TSIP_Aes128CmacGenerateInit | 145 |
| 4.65 | R_TSIP_Aes128CmacGenerateUpdate | 146 |
| 4.66 | R_TSIP_Aes128CmacGenerateFinal | 147 |
| 4.67 | R_TSIP_Aes256CmacGenerateInit | 148 |
| 4.68 | R_TSIP_Aes256CmacGenerateUpdate | 149 |
| 4.69 | R_TSIP_Aes256CmacGenerateFinal | 150 |
| 4.70 | R_TSIP_Aes128CmacVerifyInit | 151 |
| 4.71 | R_TSIP_Aes128CmacVerifyUpdate | 152 |
| 4.72 | R_TSIP_Aes128CmacVerifyFinal | 153 |
| 4.73 | R_TSIP_Aes256CmacVerifyInit | 154 |
| 4.74 | R_TSIP_Aes256CmacVerifyUpdate | 155 |
| 4.75 | R_TSIP_Aes256CmacVerifyFinal | 156 |
| 4.76 | R_TSIP_Aes128KeyWrap | 157 |
| 4.77 | R_TSIP_Aes256KeyWrap | 158 |
| 4.78 | R_TSIP_Aes128KeyUnwrap | 159 |
| 4.79 | R_TSIP_Aes256KeyUnwrap | 160 |
| 5. | API 関数詳細説明(TSIP 用) | 161 |
| 5.1 | R_TSIP_Sha1Init | 161 |
| 5.2 | R_TSIP_Sha1Update | 162 |
| 5.3 | R_TSIP_Sha1Final | 163 |
| 5.4 | R_TSIP_Sha256Init | 164 |
| 5.5 | R_TSIP_Sha256Update | 165 |
| 5.6 | R_TSIP_Sha256Final | 166 |
| 5.7 | R_TSIP_Md5Init | 167 |
| 5.8 | R_TSIP_Md5Update | 168 |
| 5.9 | R_TSIP_Md5Final | 169 |
| 5.10 | R_TSIP_GetCurrentHashDigestValue | 170 |
| 5.11 | R_TSIP_GenerateTdesKeyIndex | 171 |
| 5.12 | R_TSIP_GenerateTdesRandomKeyIndex | 172 |
| 5.13 | R_TSIP_UpdateTdesKeyIndex | 173 |
| 5.14 | R_TSIP_TdesEcbEncryptInit | 174 |
| 5.15 | R_TSIP_TdesEcbEncryptUpdate | 175 |
| 5.16 | R_TSIP_TdesEcbEncryptFinal | 176 |
| 5.17 | R_TSIP_TdesEcbDecryptInit | 177 |
| 5.18 | R_TSIP_TdesEcbDecryptUpdate | 178 |
| 5.19 | R_TSIP_TdesEcbDecryptFinal | 179 |
| 5.20 | R_TSIP_TdesCbcEncryptInit | 180 |
| 5.21 | R_TSIP_TdesCbcEncryptUpdate | 181 |
| 5.22 | R_TSIP_TdesCbcEncryptFinal | 182 |
| 5.23 | R_TSIP_TdesCbcDecryptInit | 183 |
| 5.24 | R_TSIP_TdesCbcDecryptUpdate | 184 |
| 5.25 | R_TSIP_TdesCbcDecryptFinal | 185 |

| | | |
|------|--|-----|
| 5.26 | R_TSIP_GenerateArc4KeyIndex | 186 |
| 5.27 | R_TSIP_GenerateArc4RandomKeyIndex | 187 |
| 5.28 | R_TSIP_UpdateArc4KeyIndex | 188 |
| 5.29 | R_TSIP_Arc4EncryptInit | 189 |
| 5.30 | R_TSIP_Arc4EncryptUpdate | 190 |
| 5.31 | R_TSIP_Arc4EncryptFinal | 191 |
| 5.32 | R_TSIP_Arc4DecryptInit | 192 |
| 5.33 | R_TSIP_Arc4DecryptUpdate | 193 |
| 5.34 | R_TSIP_Arc4DecryptFinal | 194 |
| 5.35 | R_TSIP_GenerateRsa1024PublicKeyIndex | 195 |
| 5.36 | R_TSIP_GenerateRsa1024PrivateKeyIndex | 197 |
| 5.37 | R_TSIP_GenerateRsa2048PublicKeyIndex | 198 |
| 5.38 | R_TSIP_GenerateRsa2048PrivateKeyIndex | 200 |
| 5.39 | R_TSIP_GenerateRsa1024RandomKeyIndex | 201 |
| 5.40 | R_TSIP_GenerateRsa2048RandomKeyIndex | 202 |
| 5.41 | R_TSIP_UpdateRsa1024PublicKeyIndex | 203 |
| 5.42 | R_TSIP_UpdateRsa1024PrivateKeyIndex | 204 |
| 5.43 | R_TSIP_UpdateRsa2048PublicKeyIndex | 205 |
| 5.44 | R_TSIP_UpdateRsa2048PrivateKeyIndex | 206 |
| 5.45 | R_TSIP_RsaesPkcs1024Encrypt | 207 |
| 5.46 | R_TSIP_RsaesPkcs1024Decrypt | 208 |
| 5.47 | R_TSIP_RsaesPkcs2048Encrypt | 209 |
| 5.48 | R_TSIP_RsaesPkcs2048Decrypt | 210 |
| 5.49 | R_TSIP_RsassaPkcs1024SignatureGenerate | 211 |
| 5.50 | R_TSIP_RsassaPkcs1024SignatureVerification | 213 |
| 5.51 | R_TSIP_RsassaPkcs2048SignatureGenerate | 215 |
| 5.52 | R_TSIP_RsassaPkcs2048SignatureVerification | 217 |
| 5.53 | R_TSIP_Rsa2048DhKeyAgreement | 219 |
| 5.54 | R_TSIP_Sha1HmacGenerateInit | 220 |
| 5.55 | R_TSIP_Sha1HmacGenerateUpdate | 221 |
| 5.56 | R_TSIP_Sha1HmacGenerateFinal | 222 |
| 5.57 | R_TSIP_Sha256HmacGenerateInit | 223 |
| 5.58 | R_TSIP_Sha256HmacGenerateUpdate | 224 |
| 5.59 | R_TSIP_Sha256HmacGenerateFinal | 225 |
| 5.60 | R_TSIP_Sha1HmacVerifyInit | 226 |
| 5.61 | R_TSIP_Sha1HmacVerifyUpdate | 227 |
| 5.62 | R_TSIP_Sha1HmacVerifyFinal | 228 |
| 5.63 | R_TSIP_Sha256HmacVerifyInit | 229 |
| 5.64 | R_TSIP_Sha256HmacVerifyUpdate | 230 |
| 5.65 | R_TSIP_Sha256HmacVerifyFinal | 231 |
| 5.66 | R_TSIP_GenerateTlsRsaPublicKeyIndex | 232 |
| 5.67 | R_TSIP_UpdateTlsRsaPublicKeyIndex | 233 |
| 5.68 | R_TSIP_TlsRootCertificateVerification | 234 |
| 5.69 | R_TSIP_TlsCertificateVerification | 236 |
| 5.70 | R_TSIP_TlsCertificateVerificationExtension | 238 |
| 5.71 | R_TSIP_TlsGeneratePreMasterSecret | 240 |
| 5.72 | R_TSIP_TlsEncryptPreMasterSecretWithRsa2048PublicKey | 241 |

| | | |
|-------|--|-----|
| 5.73 | R_TSIP_TlsGenerateMasterSecret | 242 |
| 5.74 | R_TSIP_TlsGenerateSessionKey..... | 243 |
| 5.75 | R_TSIP_TlsGenerateVerifyData | 245 |
| 5.76 | R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrieves | 246 |
| 5.77 | R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key | 248 |
| 5.78 | R_TSIP_GenerateTlsP256EccKeyIndex | 249 |
| 5.79 | R_TSIP_GenerateTls13P256EccKeyIndex | 250 |
| 5.80 | R_TSIP_Tls13GenerateEcdheSharedSecret..... | 251 |
| 5.81 | R_TSIP_Tls13GenerateHandshakeSecret..... | 252 |
| 5.82 | R_TSIP_Tls13GenerateServerHandshakeTrafficKey..... | 253 |
| 5.83 | R_TSIP_Tls13ServerHandshakeVerification | 254 |
| 5.84 | R_TSIP_Tls13GenerateClientHandshakeTrafficKey | 256 |
| 5.85 | R_TSIP_Tls13GenerateMasterSecret | 257 |
| 5.86 | R_TSIP_Tls13GenerateApplicationTrafficKey..... | 258 |
| 5.87 | R_TSIP_Tls13UpdateApplicationTrafficKey | 260 |
| 5.88 | R_TSIP_Tls13EncryptInit..... | 262 |
| 5.89 | R_TSIP_Tls13EncryptUpdate | 263 |
| 5.90 | R_TSIP_Tls13EncryptFinal..... | 264 |
| 5.91 | R_TSIP_Tls13DecryptInit | 265 |
| 5.92 | R_TSIP_Tls13DecryptUpdate | 266 |
| 5.93 | R_TSIP_Tls13DecryptFinal..... | 267 |
| 5.94 | R_TSIP_Tls13CertificateVerifyGenerate | 268 |
| 5.95 | R_TSIP_Tls13CertificateVerifyVerification | 269 |
| 5.96 | R_TSIP_GenerateEccP192PublicKeyIndex | 270 |
| 5.97 | R_TSIP_GenerateEccP224PublicKeyIndex | 271 |
| 5.98 | R_TSIP_GenerateEccP256PublicKeyIndex | 272 |
| 5.99 | R_TSIP_GenerateEccP384PublicKeyIndex | 273 |
| 5.100 | R_TSIP_GenerateEccP192PrivateKeyIndex | 274 |
| 5.101 | R_TSIP_GenerateEccP224PrivateKeyIndex | 275 |
| 5.102 | R_TSIP_GenerateEccP256PrivateKeyIndex | 276 |
| 5.103 | R_TSIP_GenerateEccP384PrivateKeyIndex | 277 |
| 5.104 | R_TSIP_GenerateEccP192RandomKeyIndex..... | 278 |
| 5.105 | R_TSIP_GenerateEccP224RandomKeyIndex..... | 279 |
| 5.106 | R_TSIP_GenerateEccP256RandomKeyIndex..... | 280 |
| 5.107 | R_TSIP_GenerateEccP384RandomKeyIndex..... | 281 |
| 5.108 | R_TSIP_GenerateSha1HmacKeyIndex..... | 282 |
| 5.109 | R_TSIP_GenerateSha256HmacKeyIndex..... | 283 |
| 5.110 | R_TSIP_UpdateEccP192PublicKeyIndex..... | 284 |
| 5.111 | R_TSIP_UpdateEccP224PublicKeyIndex..... | 285 |
| 5.112 | R_TSIP_UpdateEccP256PublicKeyIndex..... | 286 |
| 5.113 | R_TSIP_UpdateEccP384PublicKeyIndex..... | 287 |
| 5.114 | R_TSIP_UpdateEccP192PrivateKeyIndex..... | 288 |
| 5.115 | R_TSIP_UpdateEccP224PrivateKeyIndex..... | 289 |
| 5.116 | R_TSIP_UpdateEccP256PrivateKeyIndex..... | 290 |
| 5.117 | R_TSIP_UpdateEccP384PrivateKeyIndex..... | 291 |
| 5.118 | R_TSIP_UpdateSha1HmacKeyIndex | 292 |
| 5.119 | R_TSIP_UpdateSha256HmacKeyIndex | 293 |

| | |
|---|-----|
| 5.120 R_TSIP_EcdsaP192SignatureGenerate | 294 |
| 5.121 R_TSIP_EcdsaP224SignatureGenerate | 295 |
| 5.122 R_TSIP_EcdsaP256SignatureGenerate | 296 |
| 5.123 R_TSIP_EcdsaP384SignatureGenerate | 297 |
| 5.124 R_TSIP_EcdsaP192SignatureVerification | 298 |
| 5.125 R_TSIP_EcdsaP224SignatureVerification | 300 |
| 5.126 R_TSIP_EcdsaP256SignatureVerification | 302 |
| 5.127 R_TSIP_EcdsaP384SignatureVerification | 303 |
| 5.128 R_TSIP_EcdhP256Init | 304 |
| 5.129 R_TSIP_EcdhP256ReadPublicKey | 305 |
| 5.130 R_TSIP_EcdhP256MakePublicKey | 306 |
| 5.131 R_TSIP_EcdhP256CalculateSharedSecretIndex | 308 |
| 5.132 R_TSIP_EcdhP256KeyDerivation | 309 |
| 5.133 R_TSIP_EcdheP512KeyAgreement | 311 |
| 6. コールバック関数 | 312 |
| 6.1 TSIP_GEN_MAC_CB_FUNC_T 型 | 312 |
| 7. 鍵データの運用 | 315 |
| 7.1 AES ユーザ鍵の運用 | 315 |
| 7.1.1 AES ユーザ鍵インストール概要 | 315 |
| 7.1.2 AES ユーザ鍵 encrypted key の作成方法 | 316 |
| 7.2 TDES ユーザ鍵の運用 | 317 |
| 7.2.1 TDES ユーザ鍵インストール概要 | 317 |
| 7.2.2 TDES ユーザ鍵 encrypted key の作成方法 | 319 |
| 7.3 ARC4 ユーザ鍵の運用 | 320 |
| 7.3.1 ARC4 ユーザ鍵インストール概要 | 320 |
| 7.3.2 ARC4 ユーザ鍵 encrypted key の作成方法 | 321 |
| 7.4 HMAC ユーザ鍵の運用 | 322 |
| 7.4.1 HMAC ユーザ鍵インストール概要 | 322 |
| 7.4.2 HMAC ユーザ鍵 encrypted key の作成方法 | 323 |
| 7.5 RSA 公開鍵、秘密鍵の運用 | 324 |
| 7.5.1 RSA 公開鍵、秘密鍵データインストール概要 | 324 |
| 7.5.2 RSA 公開鍵、秘密鍵 encrypted key の作成方法 | 326 |
| 7.6 ECC 公開鍵、秘密鍵の運用 | 328 |
| 7.6.1 ECC 公開鍵、秘密鍵データインストール概要 | 328 |
| 7.6.2 ECC 公開鍵、秘密鍵 encrypted key の作成方法 | 330 |
| 8. 付録 | 333 |
| 8.1 動作確認環境 | 333 |
| 8.2 トラブルシューティング | 334 |
| 9. 参考ドキュメント | 335 |

1. 概要

1.1 用語

本資料中の用語説明をいたします。鍵の用語は各 MCU のユーザーズマニュアル ハードウェア編 TSIP もしくはセキュリティ機能の章にある「鍵インストール概念図」と(図 1-1)合わせてご確認ください。

表 1-1 用語説明

| 用語 | 内容 | 鍵インストール概念図との対応 |
|----------------------------|---|----------------------------|
| ユーザ鍵、user key | AES、DES、ARC4、HMAC の場合、ユーザが設定する共通鍵 RSA、ECC の場合、ユーザが設定する公開鍵、秘密鍵 | Key-1 |
| encrypted key | user key を provisioning key を使って AES128 で暗号化した鍵情報 | eKey-1 |
| 鍵生成情報、key index | user key などの鍵情報を TSIP で使用できるデータに変換したデータ。 user key は key index に変換される。 | Index-1 もしくは Index-2 |
| provisioning key | user key を AES128 で暗号化&MAC 付与するための、ユーザが設定する AES128 共通鍵束 | Key-2 |
| encrypted provisioning key | TSIP で encrypted key を復号し、key index に変換するための鍵情報 provisioning key が DLM サーバでラッピングされた鍵情報 | Index-2 |
| DLM サーバ | Renesas 鍵管理サーバ Device Lifecycle Management サーバの略 provisioning key をラッピングするのに使用する | - |

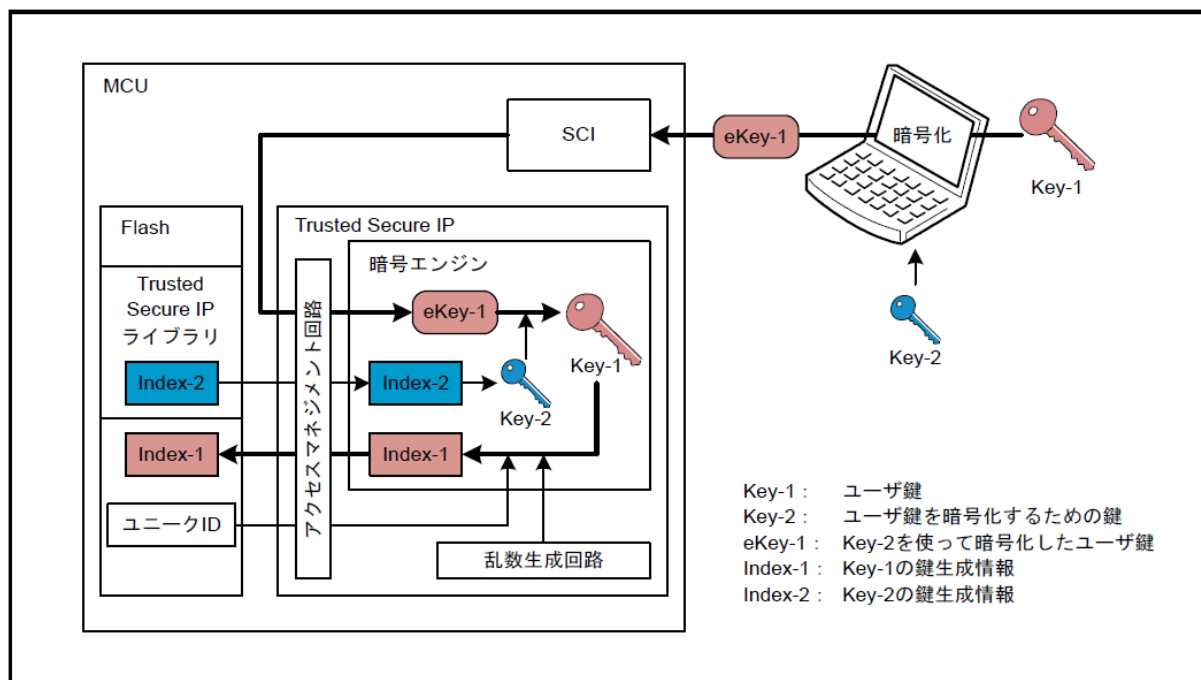


図 1-1 鍵インストール概念図 (RX65N グループ、RX651 グループ ユーザーズマニュアル ハードウェア
編 52. Trusted Secure IP 図 52.4 より抜粋)

1.2 TSIP 概要

RX ファミリ内の Trusted Secure IP (TSIP) ブロックは、不正アクセスを監視することで、MCU 内部に安全な領域を作成します。これにより、TSIP は暗号化エンジンおよび暗号鍵 (user key) を確実に安全に使用することが可能です。TSIP は、TSIP ブロックの外部において、暗号鍵 (user key) を安全で解読不可能な鍵生成情報と呼ばれる形式で扱います。このため信頼できる安全な暗号処理において最も重要な要素である暗号鍵 (user key) を、フラッシュメモリ内に保存することが可能です。

TSIP ブロックには安全領域があり、暗号化エンジン、平文鍵用のストレージおよび Hidden Root Key が格納されています。

TSIP は、TSIP 内部で鍵生成情報から暗号演算に使用する暗号鍵 (user key) を復元します。鍵生成情報は、Unique ID に紐付けられて生成されているため、デバイス固有の値になります。このため、あるデバイスの鍵生成情報を別のデバイスにコピーして使用することができません。アプリケーションから TSIP ハードウェアにアクセスするためには、TSIP ドライバを使用する必要があります。

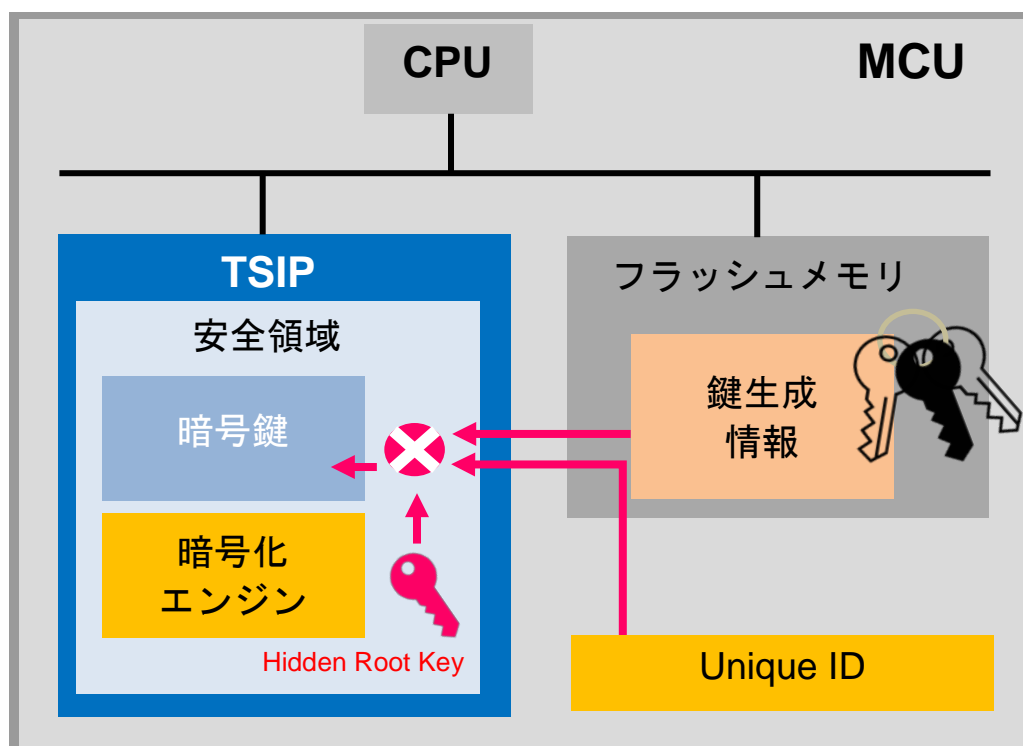


図 1-2 TSIP 搭載 MCU

1.3 製品構成

本製品は、以下の表 1-2 のファイルが含まれます。

表 1-2 製品構成

| ファイル/ディレクトリ(太字)名 | | 内容 |
|--------------------------------------|--------------------------------------|---|
| r20an0548jj0115-rx-tsip-security.pdf | | TSIP ドライバアプリケーションノート(日本語) |
| r20an0548ej0115-rx-tsip-security.pdf | | TSIP ドライバアプリケーションノート(英語) |
| reference_documents | | FIT モジュールを各種統合開発環境で使用方法等を記したドキュメントを格納するフォルダ |
| ja | | FIT モジュールを各種統合開発環境で使用方法等を記したドキュメントを格納するフォルダ(日本語) |
| | r01an1826jj0110-rx.pdf | CS+に組み込む方法(日本語) |
| | r01an1723ju0121-rx.pdf | e2studio に組み込む方法(日本語) |
| | r20an0451js0140-e2studio-sc.pdf | スマート・コンフィグレータ ユーザーガイド(日本語) |
| | r01an5792jj0101-rx-tsip.pdf | AES 暗号プロジェクト アプリケーションノート(日本語) |
| | r01an5880jj0101-rx-tsip.pdf | TLS 連携機能プロジェクト アプリケーションノート(日本語) |
| | en | FIT モジュールを各種統合開発環境で使用方法等を記したドキュメントを格納するフォルダ(英語) |
| | r01an1826ej0110-rx.pdf | CS+に組み込む方法(英語) |
| | r01an1723eu0121-rx.pdf | e2studio に組み込む方法(英語) |
| | r20an0451es0140-e2studio-sc.pdf | スマート・コンフィグレータ ユーザーガイド(英語) |
| | r01an5792ej0101-rx-tsip.pdf | AES 暗号プロジェクト アプリケーションノート(英語) |
| | r01an5880ej0101-rx-tsip.pdf | TLS 連携機能プロジェクト アプリケーションノート(英語) |
| FITModules | | FIT モジュールフォルダ |
| | r_tsip_rx_v1.15.l.zip | TSIP ドライバ FIT Module |
| | r_tsip_rx_v1.15.l.xml | TSIP ドライバ FIT Module e2 studio FIT プラグイン用 XML ファイル |
| | r_tsip_rx_v1.15.l_extend.mdf | TSIP ドライバ FIT Module スマート・コンフィグレータ用コンフィグレーション設定ファイル |
| FITDemos | | デモプロジェクトフォルダ |
| | rx231_rsk_tsip_sample | 鍵書き込み方法、鍵更新方法を示す RX231 用プロジェクト |
| | rx65n_2mb_rsk_tsip_sample | 鍵書き込み方法、鍵更新方法を示す RX65N 用プロジェクト |
| | rx66t_rsk_tsip_sample | 鍵書き込み方法、鍵更新方法を示す RX66T 用プロジェクト |
| | rx671_rsk_tsip_sample | 鍵書き込み方法、鍵更新方法を示す RX671 用プロジェクト |
| | rx72m_rsk_tsip_sample | 鍵書き込み方法、鍵更新方法を示す RX72M 用プロジェクト |
| | rx72n_rsk_tsip_sample | 鍵書き込み方法、鍵更新方法を示す RX72N 用プロジェクト |
| | rx72t_rsk_tsip_sample | 鍵書き込み方法、鍵更新方法を示す RX72T 用プロジェクト |
| | rx65n_2mb_rsk_tsip_aes_sample | RX65N 用 AES 暗号プロジェクト |
| | rx72n_ek_tsip_aes_sample | RX72N 用 AES 暗号プロジェクト |

| | | |
|-------------|--|-----------------------|
| | rx_tsip_freertos_mbedtls_sample | TLS 連携機能プロジェクト |
| tool | | |
| | Renesas Secure Flash Programmer.exe | 鍵とユーザプログラムに対し暗号化するツール |

1.4 開発環境

TSIP ドライバは以下の開発環境を用いて開発しました。ユーザアプリケーション開発時は以下のバージョン、またはより新しいものをご使用ください。

(1)統合開発環境

「8.1 動作確認環境」の項目「統合開発環境」を参照してください。

(2)C コンパイラ

「8.1 動作確認環境」の項目「C コンパイラ」を参照してください。

(3)エミュレータデバッガ

E1/E20/E2 Lite

(4)評価ボード

「8.1 動作確認環境」の項目「使用ボード」を参照してください。

いずれも、暗号機能付きの特別版の製品です。

製品型名をよくご確認の上、ご購入ください。

評価およびデモプロジェクト作成は、e² studio と CC-RX の組合せで実施しました。

プロジェクト変換機能で e² studio から CS+への変換が可能ですが、コンパイルエラー等問題が発生する場合はお問い合わせください。

1.5 コードサイズ

本モジュールの ROM サイズ、RAM サイズ、最大使用スタックサイズを下表に示します。

下表の値は下記条件で確認しています。

モジュールリビジョン: r_tsip_rx rev1.15

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00

(統合開発環境のデフォルト設定に"-lang = c99"オプションを追加)

GCC for Renesas RX 8.3.0.202104

(統合開発環境のデフォルト設定に"-std=gnu99"オプションを追加)

IAR C/C++ Compiler for Renesas RX version 4.20.01

(統合開発環境のデフォルト設定)

| ROM、RAM およびスタックのコードサイズ | | | | |
|------------------------|------|------------------|-------------|--------------|
| デバイス | 分類 | 使用メモリ | | |
| | | Renesas Compiler | GCC | IAR Compiler |
| TSIP-Lite | ROM | 54,675 バイト | 55,310 バイト | 54,150 バイト |
| | RAM | 796 バイト | 796 バイト | 796 バイト |
| | スタック | 184 バイト | - | 164 バイト |
| TSIP | ROM | 306,103 バイト | 312,602 バイト | 298,896 バイト |
| | RAM | 7,432 バイト | 7,432 バイト | 7,432 バイト |
| | スタック | 888 バイト | - | 856 バイト |

1.6 セクション情報

TSIP ドライバはデフォルトセクションを使用します。

1.7 性能情報(RX231)

以下に RX231 の TSIP-Lite ドライバの性能情報を示します。性能はコアクロックである ICLK のサイクル単位での計測になります。TSIP-Lite の動作クロック PCLKB は ICLK : PCLKB = 2 : 1 の設定をしています。

最適化レベル 2 で実施しています。

表 1-3 共通 API の性能

| API | 性能 (単位 : サイクル) |
|--------------------------------------|----------------|
| R_TSIP_Open | 7,359,176 |
| R_TSIP_Close | 444 |
| R_TSIP_GetVersion | 32 |
| R_TSIP_GenerateAes128KeyIndex | 3,990 |
| R_TSIP_GenerateAes256KeyIndex | 4,342 |
| R_TSIP_GenerateAes128RandomKeyIndex | 2,228 |
| R_TSIP_GenerateAes256RandomKeyIndex | 3,062 |
| R_TSIP_GenerateRandomNumber | 936 |
| R_TSIP_GenerateUpdateKeyRingKeyIndex | 4,328 |
| R_TSIP_UpdateAes128KeyIndex | 3,528 |
| R_TSIP_UpdateAes256KeyIndex | 3,888 |

表 1-4 Firmware 検証の性能

| API | 性能 (単位 : サイクル) | | |
|--------------------------|----------------|----------|----------|
| | 2K バイト処理 | 4K バイト処理 | 6K バイト処理 |
| R_TSIP_VerifyFirmwareMAC | 12,014 | 23,276 | 34,538 |

表 1-5 AES の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128EcbEncryptInit | 1,318 | 1,318 | 1,318 |
| R_TSIP_Aes128EcbEncryptUpdate | 612 | 792 | 964 |
| R_TSIP_Aes128EcbEncryptFinal | 566 | 566 | 566 |
| R_TSIP_Aes128EcbDecryptInit | 1,320 | 1,320 | 1,320 |
| R_TSIP_Aes128EcbDecryptUpdate | 722 | 892 | 1,074 |
| R_TSIP_Aes128EcbDecryptFinal | 578 | 578 | 578 |
| R_TSIP_Aes256EcbEncryptInit | 1,628 | 1,630 | 1,630 |
| R_TSIP_Aes256EcbEncryptUpdate | 652 | 892 | 1,134 |
| R_TSIP_Aes256EcbEncryptFinal | 554 | 554 | 554 |
| R_TSIP_Aes256EcbDecryptInit | 1,636 | 1,638 | 1,638 |
| R_TSIP_Aes256EcbDecryptUpdate | 794 | 1,034 | 1,276 |
| R_TSIP_Aes256EcbDecryptFinal | 568 | 568 | 568 |
| R_TSIP_Aes128CbcEncryptInit | 1,380 | 1,380 | 1,380 |
| R_TSIP_Aes128CbcEncryptUpdate | 682 | 862 | 1,034 |
| R_TSIP_Aes128CbcEncryptFinal | 590 | 590 | 590 |
| R_TSIP_Aes128CbcDecryptInit | 1,388 | 1,390 | 1,390 |
| R_TSIP_Aes128CbcDecryptUpdate | 790 | 960 | 1,142 |
| R_TSIP_Aes128CbcDecryptFinal | 600 | 600 | 600 |
| R_TSIP_Aes256CbcEncryptInit | 1,698 | 1,698 | 1,698 |
| R_TSIP_Aes256CbcEncryptUpdate | 720 | 960 | 1,202 |
| R_TSIP_Aes256CbcEncryptFinal | 578 | 578 | 578 |
| R_TSIP_Aes256CbcDecryptInit | 1,704 | 1,706 | 1,706 |
| R_TSIP_Aes256CbcDecryptUpdate | 870 | 1,110 | 1,352 |
| R_TSIP_Aes256CbcDecryptFinal | 588 | 588 | 588 |

表 1-6 AES-GCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128GcmEncryptInit | 5,460 | 5,460 | 5,460 |
| R_TSIP_Aes128GcmEncryptUpdate | 2,832 | 3,332 | 3,832 |
| R_TSIP_Aes128GcmEncryptFinal | 1,296 | 1,296 | 1,296 |
| R_TSIP_Aes128GcmDecryptInit | 5,454 | 5,456 | 5,456 |
| R_TSIP_Aes128GcmDecryptUpdate | 2,420 | 2,518 | 2,616 |
| R_TSIP_Aes128GcmDecryptFinal | 2,084 | 2,084 | 2,084 |
| R_TSIP_Aes256GcmEncryptInit | 6,154 | 6,156 | 6,156 |
| R_TSIP_Aes256GcmEncryptUpdate | 2,938 | 3,476 | 4,014 |
| R_TSIP_Aes256GcmEncryptFinal | 1,328 | 1,328 | 1,328 |
| R_TSIP_Aes256GcmDecryptInit | 6,146 | 6,148 | 6,148 |
| R_TSIP_Aes256GcmDecryptUpdate | 2,514 | 2,632 | 2,760 |
| R_TSIP_Aes256GcmDecryptFinal | 2,114 | 2,114 | 2,114 |

GCM の性能は、ivec を 1024bit、追加認証データを 720bit、認証タグを 128bit に固定して計測しました。

表 1-7 AES-CCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CcmEncryptInit | 2,602 | 2,602 | 2,602 |
| R_TSIP_Aes128CcmEncryptUpdate | 1,522 | 1,690 | 1,868 |
| R_TSIP_Aes128CcmEncryptFinal | 1,174 | 1,174 | 1,174 |
| R_TSIP_Aes128CcmDecryptInit | 2,414 | 2,416 | 2,416 |
| R_TSIP_Aes128CcmDecryptUpdate | 1,420 | 1,588 | 1,766 |
| R_TSIP_Aes128CcmDecryptFinal | 1,936 | 1,936 | 1,936 |
| R_TSIP_Aes256CcmEncryptInit | 2,986 | 2,986 | 2,986 |
| R_TSIP_Aes256CcmEncryptUpdate | 1,728 | 1,976 | 2,214 |
| R_TSIP_Aes256CcmEncryptFinal | 1,208 | 1,208 | 1,208 |
| R_TSIP_Aes256CcmDecryptInit | 2,984 | 2,984 | 2,984 |
| R_TSIP_Aes256CcmDecryptUpdate | 1,626 | 1,874 | 2,112 |
| R_TSIP_Aes256CcmDecryptFinal | 1,968 | 1,968 | 1,968 |

CCM の性能は、ノンスを 104bit、追加認証データを 880bit、MAC を 128bit に固定して計測しました。

表 1-8 AES-CMAC の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CmacGenerateInit | 914 | 914 | 914 |
| R_TSIP_Aes128CmacGenerateUpdate | 812 | 900 | 988 |
| R_TSIP_Aes128CmacGenerateFinal | 1,074 | 1,074 | 1,074 |
| R_TSIP_Aes128CmacVerifyInit | 910 | 914 | 914 |
| R_TSIP_Aes128CmacVerifyUpdate | 806 | 890 | 978 |
| R_TSIP_Aes128CmacVerifyFinal | 1,786 | 1,786 | 1,786 |
| R_TSIP_Aes256CmacGenerateInit | 1,216 | 1,222 | 1,222 |
| R_TSIP_Aes256CmacGenerateUpdate | 876 | 1,000 | 1,128 |
| R_TSIP_Aes256CmacGenerateFinal | 1,152 | 1,152 | 1,152 |
| R_TSIP_Aes256CmacVerifyInit | 1,216 | 1,220 | 1,220 |
| R_TSIP_Aes256CmacVerifyUpdate | 870 | 998 | 1,116 |
| R_TSIP_Aes256CmacVerifyFinal | 1,852 | 1,852 | 1,852 |

表 1-9 AES Key Wrap の性能

| API | 性能 (単位 : サイクル) | |
|------------------------|----------------|----------------|
| | ラップ対象鍵 AES-128 | ラップ対象鍵 AES-256 |
| R_TSIP_Aes128KeyWrap | 9,532 | 15,260 |
| R_TSIP_Aes256KeyWrap | 10,314 | 16,522 |
| R_TSIP_Aes128KeyUnwrap | 11,946 | 17,710 |
| R_TSIP_Aes256KeyUnwrap | 12,724 | 18,968 |

1.8 性能情報(RX23W)

以下に RX23W の TSIP-Lite ドライバの性能情報を示します。性能はコアクロックである ICLK のサイクル単位での計測になります。TSIP-Lite の動作クロック PCLKB は ICLK : PCLKB = 2 : 1 の設定をしています。

最適化レベル 2 で実施しています。

表 1-10 共通 API の性能

| API | 性能 (単位 : サイクル) |
|--------------------------------------|----------------|
| R_TSIP_Open | 7,384,208 |
| R_TSIP_Close | 666 |
| R_TSIP_GetVersion | 40 |
| R_TSIP_GenerateAes128KeyIndex | 4,332 |
| R_TSIP_GenerateAes256KeyIndex | 4,672 |
| R_TSIP_GenerateAes128RandomKeyIndex | 2,414 |
| R_TSIP_GenerateAes256RandomKeyIndex | 3,310 |
| R_TSIP_GenerateRandomNumber | 1,042 |
| R_TSIP_GenerateUpdateKeyRingKeyIndex | 4,694 |
| R_TSIP_UpdaeteAes128KeyIndex | 3,814 |
| R_TSIP_UpdaeteAes256KeyIndex | 4,190 |

表 1-11 Firmware 検証の性能

| API | 性能 (単位 : サイクル) | | |
|--------------------------|----------------|----------|----------|
| | 2K バイト処理 | 4K バイト処理 | 6K バイト処理 |
| R_TSIP_VerifyFirmwareMAC | 12,070 | 23,342 | 34,600 |

表 1-12 AES の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128EcbEncryptInit | 1,500 | 1,502 | 1,502 |
| R_TSIP_Aes128EcbEncryptUpdate | 734 | 912 | 1,096 |
| R_TSIP_Aes128EcbEncryptFinal | 658 | 658 | 658 |
| R_TSIP_Aes128EcbDecryptInit | 1,510 | 1,510 | 1,510 |
| R_TSIP_Aes128EcbDecryptUpdate | 856 | 1,038 | 1,222 |
| R_TSIP_Aes128EcbDecryptFinal | 680 | 680 | 680 |
| R_TSIP_Aes256EcbEncryptInit | 1,824 | 1,826 | 1,826 |
| R_TSIP_Aes256EcbEncryptUpdate | 770 | 1,006 | 1,260 |
| R_TSIP_Aes256EcbEncryptFinal | 662 | 662 | 662 |
| R_TSIP_Aes256EcbDecryptInit | 1,838 | 1,842 | 1,842 |
| R_TSIP_Aes256EcbDecryptUpdate | 916 | 1,166 | 1,406 |
| R_TSIP_Aes256EcbDecryptFinal | 680 | 680 | 680 |
| R_TSIP_Aes128CbcEncryptInit | 1,592 | 1,594 | 1,594 |
| R_TSIP_Aes128CbcEncryptUpdate | 838 | 1,016 | 1,200 |
| R_TSIP_Aes128CbcEncryptFinal | 694 | 694 | 694 |
| R_TSIP_Aes128CbcDecryptInit | 1,604 | 1,606 | 1,606 |
| R_TSIP_Aes128CbcDecryptUpdate | 946 | 1,128 | 1,312 |
| R_TSIP_Aes128CbcDecryptFinal | 708 | 708 | 708 |
| R_TSIP_Aes256CbcEncryptInit | 1,916 | 1,916 | 1,916 |
| R_TSIP_Aes256CbcEncryptUpdate | 870 | 1,106 | 1,360 |
| R_TSIP_Aes256CbcEncryptFinal | 700 | 700 | 700 |
| R_TSIP_Aes256CbcDecryptInit | 1,932 | 1,936 | 1,936 |
| R_TSIP_Aes256CbcDecryptUpdate | 1,014 | 1,264 | 1,504 |
| R_TSIP_Aes256CbcDecryptFinal | 712 | 712 | 712 |

表 1-13 AES-GCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128GcmEncryptInit | 6,198 | 6,200 | 6,200 |
| R_TSIP_Aes128GcmEncryptUpdate | 3,328 | 3,888 | 4,448 |
| R_TSIP_Aes128GcmEncryptFinal | 1,480 | 1,480 | 1,480 |
| R_TSIP_Aes128GcmDecryptInit | 6,196 | 6,196 | 6,196 |
| R_TSIP_Aes128GcmDecryptUpdate | 2,874 | 2,980 | 3,086 |
| R_TSIP_Aes128GcmDecryptFinal | 2,338 | 2,338 | 2,338 |
| R_TSIP_Aes256GcmEncryptInit | 6,928 | 6,930 | 6,930 |
| R_TSIP_Aes256GcmEncryptUpdate | 3,456 | 4,058 | 4,660 |
| R_TSIP_Aes256GcmEncryptFinal | 1,520 | 1,520 | 1,520 |
| R_TSIP_Aes256GcmDecryptInit | 6,928 | 6,930 | 6,930 |
| R_TSIP_Aes256GcmDecryptUpdate | 2,932 | 3,052 | 3,186 |
| R_TSIP_Aes256GcmDecryptFinal | 2,376 | 2,376 | 2,376 |

GCM の性能は、ivec を 1024bit、追加認証データを 720bit、認証タグを 128bit に固定して計測しました。

表 1-14 AES-CCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CcmEncryptInit | 3,036 | 3,036 | 3,036 |
| R_TSIP_Aes128CcmEncryptUpdate | 1,780 | 1,956 | 2,132 |
| R_TSIP_Aes128CcmEncryptFinal | 1,434 | 1,434 | 1,434 |
| R_TSIP_Aes128CcmDecryptInit | 2,730 | 2,732 | 2,732 |
| R_TSIP_Aes128CcmDecryptUpdate | 1,644 | 1,820 | 1,996 |
| R_TSIP_Aes128CcmDecryptFinal | 2,234 | 2,234 | 2,234 |
| R_TSIP_Aes256CcmEncryptInit | 3,292 | 3,292 | 3,292 |
| R_TSIP_Aes256CcmEncryptUpdate | 2,008 | 2,240 | 2,486 |
| R_TSIP_Aes256CcmEncryptFinal | 1,480 | 1,480 | 1,480 |
| R_TSIP_Aes256CcmDecryptInit | 3,300 | 3,300 | 3,300 |
| R_TSIP_Aes256CcmDecryptUpdate | 1,856 | 2,102 | 2,348 |
| R_TSIP_Aes256CcmDecryptFinal | 2,278 | 2,278 | 2,278 |

CCM の性能は、ノンスを 104bit、追加認証データを 880bit、MAC を 128bit に固定して計測しました。

表 1-15 AES-CMAC の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CmacGenerateInit | 1,016 | 1,018 | 1,018 |
| R_TSIP_Aes128CmacGenerateUpdate | 942 | 1,040 | 1,122 |
| R_TSIP_Aes128CmacGenerateFinal | 1,260 | 1,260 | 1,260 |
| R_TSIP_Aes128CmacVerifyInit | 1,016 | 1,016 | 1,016 |
| R_TSIP_Aes128CmacVerifyUpdate | 950 | 1,048 | 1,130 |
| R_TSIP_Aes128CmacVerifyFinal | 2,018 | 2,018 | 2,018 |
| R_TSIP_Aes256CmacGenerateInit | 1,336 | 1,338 | 1,338 |
| R_TSIP_Aes256CmacGenerateUpdate | 1,026 | 1,158 | 1,268 |
| R_TSIP_Aes256CmacGenerateFinal | 1,336 | 1,336 | 1,336 |
| R_TSIP_Aes256CmacVerifyInit | 1,336 | 1,336 | 1,336 |
| R_TSIP_Aes256CmacVerifyUpdate | 1,024 | 1,156 | 1,266 |
| R_TSIP_Aes256CmacVerifyFinal | 2,092 | 2,092 | 2,092 |

表 1-16 AES Key Wrap の性能

| API | 性能 (単位 : サイクル) | |
|------------------------|----------------|----------------|
| | ラップ対象鍵 AES-128 | ラップ対象鍵 AES-256 |
| R_TSIP_Aes128KeyWrap | 10,452 | 16,734 |
| R_TSIP_Aes256KeyWrap | 11,148 | 17,764 |
| R_TSIP_Aes128KeyUnwrap | 13,214 | 19,486 |
| R_TSIP_Aes256KeyUnwrap | 13,994 | 20,602 |

1.9 性能情報(RX66T)

以下に RX66T の TSIP-Lite ドライバの性能情報を示します。性能はコアクロックである ICLK のサイクル単位での計測になります。TSIP-Lite の動作クロック PCLKB は ICLK : PCLKB = 2 : 1 の設定をしています。

最適化レベル 2 で実施しています。

表 1-17 共通 API の性能

| API | 性能 (単位 : サイクル) |
|--------------------------------------|----------------|
| R_TSIP_Open | 7,353,148 |
| R_TSIP_Close | 286 |
| R_TSIP_GetVersion | 22 |
| R_TSIP_GenerateAes128KeyIndex | 3,890 |
| R_TSIP_GenerateAes256KeyIndex | 4,234 |
| R_TSIP_GenerateAes128RandomKeyIndex | 2,192 |
| R_TSIP_GenerateAes256RandomKeyIndex | 2,984 |
| R_TSIP_GenerateRandomNumber | 906 |
| R_TSIP_GenerateUpdateKeyRingKeyIndex | 4,238 |
| R_TSIP_UpdateAes128KeyIndex | 3,460 |
| R_TSIP_UpdateAes256KeyIndex | 3,798 |

表 1-18 Firmware 検証の性能

| API | 性能 (単位 : サイクル) | | |
|--------------------------|----------------|----------|----------|
| | 2K バイト処理 | 4K バイト処理 | 6K バイト処理 |
| R_TSIP_VerifyFirmwareMAC | 11,942 | 23,204 | 34,468 |

表 1-19 AES の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128EcbEncryptInit | 1,284 | 1,278 | 1,278 |
| R_TSIP_Aes128EcbEncryptUpdate | 560 | 742 | 918 |
| R_TSIP_Aes128EcbEncryptFinal | 514 | 510 | 510 |
| R_TSIP_Aes128EcbDecryptInit | 1,284 | 1,286 | 1,286 |
| R_TSIP_Aes128EcbDecryptUpdate | 672 | 854 | 1,030 |
| R_TSIP_Aes128EcbDecryptFinal | 524 | 524 | 524 |
| R_TSIP_Aes256EcbEncryptInit | 1,594 | 1,592 | 1,594 |
| R_TSIP_Aes256EcbEncryptUpdate | 608 | 850 | 1,090 |
| R_TSIP_Aes256EcbEncryptFinal | 520 | 518 | 520 |
| R_TSIP_Aes256EcbDecryptInit | 1,602 | 1,602 | 1,604 |
| R_TSIP_Aes256EcbDecryptUpdate | 748 | 990 | 1,230 |
| R_TSIP_Aes256EcbDecryptFinal | 528 | 526 | 528 |
| R_TSIP_Aes128CbcEncryptInit | 1,336 | 1,334 | 1,334 |
| R_TSIP_Aes128CbcEncryptUpdate | 616 | 800 | 976 |
| R_TSIP_Aes128CbcEncryptFinal | 536 | 536 | 536 |
| R_TSIP_Aes128CbcDecryptInit | 1,342 | 1,344 | 1,342 |
| R_TSIP_Aes128CbcDecryptUpdate | 728 | 910 | 1,086 |
| R_TSIP_Aes128CbcDecryptFinal | 548 | 548 | 548 |
| R_TSIP_Aes256CbcEncryptInit | 1,652 | 1,652 | 1,652 |
| R_TSIP_Aes256CbcEncryptUpdate | 670 | 912 | 1,152 |
| R_TSIP_Aes256CbcEncryptFinal | 538 | 538 | 536 |
| R_TSIP_Aes256CbcDecryptInit | 1,660 | 1,662 | 1,662 |
| R_TSIP_Aes256CbcDecryptUpdate | 814 | 1,060 | 1,300 |
| R_TSIP_Aes256CbcDecryptFinal | 548 | 548 | 548 |

表 1-20 AES-GCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128GcmEncryptInit | 5,120 | 5,116 | 5,116 |
| R_TSIP_Aes128GcmEncryptUpdate | 2,596 | 3,084 | 3,576 |
| R_TSIP_Aes128GcmEncryptFinal | 1,238 | 1,236 | 1,234 |
| R_TSIP_Aes128GcmDecryptInit | 5,120 | 5,126 | 5,126 |
| R_TSIP_Aes128GcmDecryptUpdate | 2,194 | 2,286 | 2,374 |
| R_TSIP_Aes128GcmDecryptFinal | 2,020 | 2,018 | 2,018 |
| R_TSIP_Aes256GcmEncryptInit | 5,830 | 5,830 | 5,830 |
| R_TSIP_Aes256GcmEncryptUpdate | 2,702 | 3,222 | 3,744 |
| R_TSIP_Aes256GcmEncryptFinal | 1,286 | 1,288 | 1,288 |
| R_TSIP_Aes256GcmDecryptInit | 5,830 | 5,830 | 5,832 |
| R_TSIP_Aes256GcmDecryptUpdate | 2,294 | 2,414 | 2,534 |
| R_TSIP_Aes256GcmDecryptFinal | 2,062 | 2,060 | 2,060 |

GCM の性能は、ivec を 1024bit、追加認証データを 720bit、認証タグを 128bit に固定して計測しました。

表 1-21 AES-CCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CcmEncryptInit | 2,452 | 2,450 | 2,450 |
| R_TSIP_Aes128CcmEncryptUpdate | 1,452 | 1,628 | 1,804 |
| R_TSIP_Aes128CcmEncryptFinal | 1,138 | 1,138 | 1,138 |
| R_TSIP_Aes128CcmDecryptInit | 2,238 | 2,240 | 2,240 |
| R_TSIP_Aes128CcmDecryptUpdate | 1,348 | 1,524 | 1,700 |
| R_TSIP_Aes128CcmDecryptFinal | 1,874 | 1,874 | 1,872 |
| R_TSIP_Aes256CcmEncryptInit | 2,818 | 2,816 | 2,816 |
| R_TSIP_Aes256CcmEncryptUpdate | 1,656 | 1,906 | 2,146 |
| R_TSIP_Aes256CcmEncryptFinal | 1,174 | 1,174 | 1,174 |
| R_TSIP_Aes256CcmDecryptInit | 2,812 | 2,810 | 2,810 |
| R_TSIP_Aes256CcmDecryptUpdate | 1,558 | 1,806 | 2,046 |
| R_TSIP_Aes256CcmDecryptFinal | 1,922 | 1,916 | 1,916 |

CCM の性能は、ノンスを 104bit、追加認証データを 880bit、MAC を 128bit に固定して計測しました。

表 1-22 AES-CMAC の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CmacGenerateInit | 876 | 876 | 876 |
| R_TSIP_Aes128CmacGenerateUpdate | 720 | 810 | 898 |
| R_TSIP_Aes128CmacGenerateFinal | 1,024 | 1,022 | 1,022 |
| R_TSIP_Aes128CmacVerifyInit | 878 | 878 | 878 |
| R_TSIP_Aes128CmacVerifyUpdate | 720 | 808 | 896 |
| R_TSIP_Aes128CmacVerifyFinal | 1,714 | 1,714 | 1,714 |
| R_TSIP_Aes256CmacGenerateInit | 1,182 | 1,180 | 1,180 |
| R_TSIP_Aes256CmacGenerateUpdate | 790 | 916 | 1,036 |
| R_TSIP_Aes256CmacGenerateFinal | 1,100 | 1,100 | 1,098 |
| R_TSIP_Aes256CmacVerifyInit | 1,182 | 1,184 | 1,184 |
| R_TSIP_Aes256CmacVerifyUpdate | 792 | 918 | 1,038 |
| R_TSIP_Aes256CmacVerifyFinal | 1,790 | 1,790 | 1,788 |

表 1-23 AES Key Wrap の性能

| API | 性能 (単位 : サイクル) | |
|------------------------|----------------|----------------|
| | ラップ対象鍵 AES-128 | ラップ対象鍵 AES-256 |
| R_TSIP_Aes128KeyWrap | 9,350 | 14,996 |
| R_TSIP_Aes256KeyWrap | 10,040 | 16,072 |
| R_TSIP_Aes128KeyUnwrap | 11,678 | 17,364 |
| R_TSIP_Aes256KeyUnwrap | 12,410 | 18,482 |

1.10 性能情報(RX72T)

以下に RX72T の TSIP-Lite ドライバの性能情報を示します。性能はコアクロックである ICLK のサイクル単位での計測になります。TSIP-Lite の動作クロック PCLKB は ICLK : PCLKB = 2 : 1 の設定をしています。

最適化レベル 2 で実施しています。

表 1-24 共通 API の性能

| API | 性能 (単位 : サイクル) |
|--------------------------------------|----------------|
| R_TSIP_Open | 7,354,918 |
| R_TSIP_Close | 286 |
| R_TSIP_GetVersion | 20 |
| R_TSIP_GenerateAes128KeyIndex | 3,892 |
| R_TSIP_GenerateAes256KeyIndex | 4,234 |
| R_TSIP_GenerateAes128RandomKeyIndex | 2,174 |
| R_TSIP_GenerateAes256RandomKeyIndex | 2,978 |
| R_TSIP_GenerateRandomNumber | 896 |
| R_TSIP_GenerateUpdateKeyRingKeyIndex | 4,242 |
| R_TSIP_UpdateAes128KeyIndex | 3,460 |
| R_TSIP_UpdateAes256KeyIndex | 3,808 |

表 1-25 Firmware 検証の性能

| API | 性能 (単位 : サイクル) | | |
|--------------------------|----------------|----------|----------|
| | 2K バイト処理 | 4K バイト処理 | 6K バイト処理 |
| R_TSIP_VerifyFirmwareMAC | 11,938 | 23,200 | 34,464 |

表 1-26 AES の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128EcbEncryptInit | 1,284 | 1,280 | 1,280 |
| R_TSIP_Aes128EcbEncryptUpdate | 560 | 738 | 914 |
| R_TSIP_Aes128EcbEncryptFinal | 510 | 506 | 506 |
| R_TSIP_Aes128EcbDecryptInit | 1,288 | 1,286 | 1,286 |
| R_TSIP_Aes128EcbDecryptUpdate | 668 | 848 | 1,024 |
| R_TSIP_Aes128EcbDecryptFinal | 518 | 518 | 518 |
| R_TSIP_Aes256EcbEncryptInit | 1,594 | 1,592 | 1,590 |
| R_TSIP_Aes256EcbEncryptUpdate | 606 | 852 | 1,092 |
| R_TSIP_Aes256EcbEncryptFinal | 512 | 512 | 512 |
| R_TSIP_Aes256EcbDecryptInit | 1,602 | 1,604 | 1,602 |
| R_TSIP_Aes256EcbDecryptUpdate | 746 | 990 | 1,230 |
| R_TSIP_Aes256EcbDecryptFinal | 522 | 524 | 524 |
| R_TSIP_Aes128CbcEncryptInit | 1,338 | 1,336 | 1,336 |
| R_TSIP_Aes128CbcEncryptUpdate | 614 | 796 | 972 |
| R_TSIP_Aes128CbcEncryptFinal | 528 | 528 | 528 |
| R_TSIP_Aes128CbcDecryptInit | 1,344 | 1,344 | 1,344 |
| R_TSIP_Aes128CbcDecryptUpdate | 724 | 906 | 1,082 |
| R_TSIP_Aes128CbcDecryptFinal | 542 | 542 | 542 |
| R_TSIP_Aes256CbcEncryptInit | 1,650 | 1,650 | 1,648 |
| R_TSIP_Aes256CbcEncryptUpdate | 666 | 910 | 1,150 |
| R_TSIP_Aes256CbcEncryptFinal | 532 | 532 | 532 |
| R_TSIP_Aes256CbcDecryptInit | 1,660 | 1,660 | 1,660 |
| R_TSIP_Aes256CbcDecryptUpdate | 810 | 1,056 | 1,296 |
| R_TSIP_Aes256CbcDecryptFinal | 538 | 538 | 538 |

表 1-27 AES-GCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128GcmEncryptInit | 5,116 | 5,114 | 5,114 |
| R_TSIP_Aes128GcmEncryptUpdate | 2,576 | 3,064 | 3,552 |
| R_TSIP_Aes128GcmEncryptFinal | 1,240 | 1,238 | 1,238 |
| R_TSIP_Aes128GcmDecryptInit | 5,114 | 5,120 | 5,118 |
| R_TSIP_Aes128GcmDecryptUpdate | 2,196 | 2,284 | 2,372 |
| R_TSIP_Aes128GcmDecryptFinal | 2,014 | 2,012 | 2,012 |
| R_TSIP_Aes256GcmEncryptInit | 5,820 | 5,822 | 5,822 |
| R_TSIP_Aes256GcmEncryptUpdate | 2,698 | 3,218 | 3,740 |
| R_TSIP_Aes256GcmEncryptFinal | 1,278 | 1,278 | 1,278 |
| R_TSIP_Aes256GcmDecryptInit | 5,834 | 5,834 | 5,834 |
| R_TSIP_Aes256GcmDecryptUpdate | 2,292 | 2,412 | 2,532 |
| R_TSIP_Aes256GcmDecryptFinal | 2,058 | 2,056 | 2,056 |

GCM の性能は、ivec を 1024bit、追加認証データを 720bit、認証タグを 128bit に固定して計測しました。

表 1-28 AES-CCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CcmEncryptInit | 2,448 | 2,448 | 2,448 |
| R_TSIP_Aes128CcmEncryptUpdate | 1,452 | 1,628 | 1,804 |
| R_TSIP_Aes128CcmEncryptFinal | 1,136 | 1,136 | 1,136 |
| R_TSIP_Aes128CcmDecryptInit | 2,234 | 2,232 | 2,232 |
| R_TSIP_Aes128CcmDecryptUpdate | 1,348 | 1,524 | 1,700 |
| R_TSIP_Aes128CcmDecryptFinal | 1,874 | 1,874 | 1,872 |
| R_TSIP_Aes256CcmEncryptInit | 2,814 | 2,810 | 2,810 |
| R_TSIP_Aes256CcmEncryptUpdate | 1,658 | 1,904 | 2,144 |
| R_TSIP_Aes256CcmEncryptFinal | 1,176 | 1,174 | 1,174 |
| R_TSIP_Aes256CcmDecryptInit | 2,802 | 2,800 | 2,800 |
| R_TSIP_Aes256CcmDecryptUpdate | 1,560 | 1,808 | 2,048 |
| R_TSIP_Aes256CcmDecryptFinal | 1,918 | 1,916 | 1,916 |

CCM の性能は、ノンスを 104bit、追加認証データを 880bit、MAC を 128bit に固定して計測しました。

表 1-29 AES-CMAC の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CmacGenerateInit | 870 | 872 | 872 |
| R_TSIP_Aes128CmacGenerateUpdate | 716 | 806 | 896 |
| R_TSIP_Aes128CmacGenerateFinal | 1,022 | 1,022 | 1,022 |
| R_TSIP_Aes128CmacVerifyInit | 870 | 870 | 870 |
| R_TSIP_Aes128CmacVerifyUpdate | 716 | 806 | 896 |
| R_TSIP_Aes128CmacVerifyFinal | 1,718 | 1,716 | 1,716 |
| R_TSIP_Aes256CmacGenerateInit | 1,186 | 1,182 | 1,182 |
| R_TSIP_Aes256CmacGenerateUpdate | 792 | 918 | 1,038 |
| R_TSIP_Aes256CmacGenerateFinal | 1,098 | 1,094 | 1,094 |
| R_TSIP_Aes256CmacVerifyInit | 1,184 | 1,184 | 1,184 |
| R_TSIP_Aes256CmacVerifyUpdate | 792 | 918 | 1,038 |
| R_TSIP_Aes256CmacVerifyFinal | 1,786 | 1,786 | 1,786 |

表 1-30 AES Key Wrap の性能

| API | 性能 (単位 : サイクル) | |
|------------------------|----------------|----------------|
| | ラップ対象鍵 AES-128 | ラップ対象鍵 AES-256 |
| R_TSIP_Aes128KeyWrap | 9,352 | 15,000 |
| R_TSIP_Aes256KeyWrap | 10,046 | 16,078 |
| R_TSIP_Aes128KeyUnwrap | 11,668 | 17,352 |
| R_TSIP_Aes256KeyUnwrap | 12,398 | 18,470 |

1.11 性能情報(RX65N)

以下に RX65N の TSIP ドライバの性能情報を示します。性能はコアクロックである ICLK のサイクル単位での計測になります。TSIP の動作クロック PCLKB は ICLK : PCLKB = 2 : 1 の設定をしています。

最適化レベル 2 で実施しています。

表 1-31 共通 API の性能

| API | 性能 (単位 : サイクル) |
|--------------------------------------|----------------|
| R_TSIP_Open | 5,681,760 |
| R_TSIP_Close | 476 |
| R_TSIP_GetVersion | 34 |
| R_TSIP_GenerateAes128KeyIndex | 2,626 |
| R_TSIP_GenerateAes256KeyIndex | 2,740 |
| R_TSIP_GenerateAes128RandomKeyIndex | 1,460 |
| R_TSIP_GenerateAes256RandomKeyIndex | 2,014 |
| R_TSIP_GenerateRandomNumber | 642 |
| R_TSIP_GenerateUpdateKeyRingKeyIndex | 2,768 |
| R_TSIP_UpdateAes128KeyIndex | 2,234 |
| R_TSIP_UpdateAes256KeyIndex | 2,350 |

表 1-32 Firmware 検証の性能

| API | 性能 (単位 : サイクル) | | |
|--------------------------|----------------|-----------|-----------|
| | 8K バイト処理 | 16K バイト処理 | 24K バイト処理 |
| R_TSIP_VerifyFirmwareMAC | 20,018 | 39,474 | 58,930 |

表 1-33 AES の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128EcbEncryptInit | 1,590 | 1,592 | 1,590 |
| R_TSIP_Aes128EcbEncryptUpdate | 518 | 662 | 840 |
| R_TSIP_Aes128EcbEncryptFinal | 450 | 450 | 450 |
| R_TSIP_Aes128EcbDecryptInit | 1,606 | 1,606 | 1,606 |
| R_TSIP_Aes128EcbDecryptUpdate | 574 | 716 | 896 |
| R_TSIP_Aes128EcbDecryptFinal | 464 | 464 | 464 |
| R_TSIP_Aes256EcbEncryptInit | 1,748 | 1,750 | 1,750 |
| R_TSIP_Aes256EcbEncryptUpdate | 530 | 680 | 860 |
| R_TSIP_Aes256EcbEncryptFinal | 436 | 436 | 436 |
| R_TSIP_Aes256EcbDecryptInit | 1,762 | 1,762 | 1,760 |
| R_TSIP_Aes256EcbDecryptUpdate | 610 | 748 | 930 |
| R_TSIP_Aes256EcbDecryptFinal | 454 | 454 | 454 |
| R_TSIP_Aes128CbcEncryptInit | 1,672 | 1,672 | 1,672 |
| R_TSIP_Aes128CbcEncryptUpdate | 600 | 744 | 926 |
| R_TSIP_Aes128CbcEncryptFinal | 474 | 474 | 474 |
| R_TSIP_Aes128CbcDecryptInit | 1,688 | 1,688 | 1,688 |
| R_TSIP_Aes128CbcDecryptUpdate | 656 | 798 | 978 |
| R_TSIP_Aes128CbcDecryptFinal | 490 | 490 | 490 |
| R_TSIP_Aes256CbcEncryptInit | 1,832 | 1,832 | 1,832 |
| R_TSIP_Aes256CbcEncryptUpdate | 618 | 770 | 950 |
| R_TSIP_Aes256CbcEncryptFinal | 472 | 472 | 472 |
| R_TSIP_Aes256CbcDecryptInit | 1,856 | 1,856 | 1,856 |
| R_TSIP_Aes256CbcDecryptUpdate | 686 | 826 | 1,004 |
| R_TSIP_Aes256CbcDecryptFinal | 482 | 482 | 482 |

表 1-34 AES-GCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128GcmEncryptInit | 5,468 | 5,468 | 5,468 |
| R_TSIP_Aes128GcmEncryptUpdate | 2,074 | 2,176 | 2,266 |
| R_TSIP_Aes128GcmEncryptFinal | 1,304 | 1,304 | 1,304 |
| R_TSIP_Aes128GcmDecryptInit | 5,326 | 5,326 | 5,326 |
| R_TSIP_Aes128GcmDecryptUpdate | 2,056 | 2,150 | 2,238 |
| R_TSIP_Aes128GcmDecryptFinal | 2,204 | 2,204 | 2,206 |
| R_TSIP_Aes256GcmEncryptInit | 5,370 | 5,372 | 5,374 |
| R_TSIP_Aes256GcmEncryptUpdate | 2,100 | 2,216 | 2,304 |
| R_TSIP_Aes256GcmEncryptFinal | 1,090 | 1,090 | 1,090 |
| R_TSIP_Aes256GcmDecryptInit | 5,404 | 5,404 | 5,404 |
| R_TSIP_Aes256GcmDecryptUpdate | 2,096 | 2,200 | 2,286 |
| R_TSIP_Aes256GcmDecryptFinal | 1,974 | 1,972 | 1,972 |

GCM の性能は、ivec を 1024bit、追加認証データを 720bit、認証タグを 128bit に固定して計測しました。

表 1-35 AES-CCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CcmEncryptInit | 2,978 | 2,978 | 2,978 |
| R_TSIP_Aes128CcmEncryptUpdate | 1,108 | 1,208 | 1,296 |
| R_TSIP_Aes128CcmEncryptFinal | 922 | 922 | 922 |
| R_TSIP_Aes128CcmDecryptInit | 3,112 | 3,114 | 3,116 |
| R_TSIP_Aes128CcmDecryptUpdate | 1,048 | 1,134 | 1,222 |
| R_TSIP_Aes128CcmDecryptFinal | 1,916 | 1,914 | 1,916 |
| R_TSIP_Aes256CcmEncryptInit | 2,372 | 2,372 | 2,372 |
| R_TSIP_Aes256CcmEncryptUpdate | 1,166 | 1,266 | 1,354 |
| R_TSIP_Aes256CcmEncryptFinal | 974 | 974 | 974 |
| R_TSIP_Aes256CcmDecryptInit | 2,348 | 2,348 | 2,348 |
| R_TSIP_Aes256CcmDecryptUpdate | 1,078 | 1,168 | 1,256 |
| R_TSIP_Aes256CcmDecryptFinal | 2,026 | 2,026 | 2,028 |

CCM の性能は、ノンスを 104bit、追加認証データを 880bit、MAC を 128bit に固定して計測しました。

表 1-36 AES-CMAC の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CmacGenerateInit | 1,142 | 1,142 | 1,142 |
| R_TSIP_Aes128CmacGenerateUpdate | 662 | 706 | 750 |
| R_TSIP_Aes128CmacGenerateFinal | 792 | 792 | 790 |
| R_TSIP_Aes128CmacVerifyInit | 1,144 | 1,144 | 1,144 |
| R_TSIP_Aes128CmacVerifyUpdate | 664 | 708 | 754 |
| R_TSIP_Aes128CmacVerifyFinal | 1,668 | 1,666 | 1,666 |
| R_TSIP_Aes256CmacGenerateInit | 1,280 | 1,280 | 1,280 |
| R_TSIP_Aes256CmacGenerateUpdate | 690 | 734 | 780 |
| R_TSIP_Aes256CmacGenerateFinal | 820 | 820 | 820 |
| R_TSIP_Aes256CmacVerifyInit | 1,278 | 1,276 | 1,276 |
| R_TSIP_Aes256CmacVerifyUpdate | 700 | 746 | 790 |
| R_TSIP_Aes256CmacVerifyFinal | 1,690 | 1,690 | 1,690 |

表 1-37 AES Key Wrap の性能

| API | 性能 (単位 : サイクル) | |
|------------------------|----------------|----------------|
| | ラップ対象鍵 AES-128 | ラップ対象鍵 AES-256 |
| R_TSIP_Aes128KeyWrap | 8,232 | 12,934 |
| R_TSIP_Aes256KeyWrap | 8,384 | 13,084 |
| R_TSIP_Aes128KeyUnwrap | 9,266 | 13,924 |
| R_TSIP_Aes256KeyUnwrap | 9,418 | 14,078 |

表 1-38 共通 API(TDES ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|-----------------------------------|----------------|
| R_TSIP_GenerateTdesKeyIndex | 2,744 |
| R_TSIP_GenerateTdesRandomKeyIndex | 2,032 |
| R_TSIP_UpdateTdesKeyIndex | 2,366 |

表 1-39 TDES の性能

| API | 性能 (単位 : サイクル) | | |
|-----------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_TdesEcbEncryptInit | 1,048 | 1,048 | 1,048 |
| R_TSIP_TdesEcbEncryptUpdate | 550 | 796 | 1,036 |
| R_TSIP_TdesEcbEncryptFinal | 432 | 432 | 432 |
| R_TSIP_TdesEcbDecryptInit | 1,060 | 1,062 | 1,062 |
| R_TSIP_TdesEcbDecryptUpdate | 582 | 828 | 1,068 |
| R_TSIP_TdesEcbDecryptFinal | 444 | 444 | 444 |
| R_TSIP_TdesCbcEncryptInit | 1,128 | 1,126 | 1,126 |
| R_TSIP_TdesCbcEncryptUpdate | 628 | 874 | 1,114 |
| R_TSIP_TdesCbcEncryptFinal | 460 | 460 | 460 |
| R_TSIP_TdesCbcDecryptInit | 1,132 | 1,132 | 1,134 |
| R_TSIP_TdesCbcDecryptUpdate | 656 | 902 | 1,144 |
| R_TSIP_TdesCbcDecryptFinal | 482 | 482 | 482 |

表 1-40 共通 API(RSA ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|--|----------------|
| R_TSIP_GenerateRsa1024PublicKeyIndex | 37,536 |
| R_TSIP_GenerateRsa1024PrivateKeyIndex | 38,620 |
| R_TSIP_GenerateRsa2048PublicKeyIndex | 137,504 |
| R_TSIP_GenerateRsa2048PrivateKeyIndex | 139,686 |
| R_TSIP_GenerateRsa1024RandomKeyIndex (注) | 55,269,285 |
| R_TSIP_GenerateRsa2048RandomKeyIndex (注) | 540,959,708 |
| R_TSIP_UpdateRsa1024PublicKeyIndex | 37,134 |
| R_TSIP_UpdateRsa1024PrivateKeyIndex | 38,230 |
| R_TSIP_UpdateRsa2048PublicKeyIndex | 137,132 |
| R_TSIP_UpdateRsa2048PrivateKeyIndex | 139,296 |

【注】 10 回実行時の平均値です。

表 1-41 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=SHA1)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,266,372 | 1,267,788 | 1,268,268 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 17,234 | 18,646 | 19,124 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,227,124 | 26,228,528 | 26,229,006 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 135,548 | 136,956 | 137,436 |

表 1-42 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=SHA256)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,266,472 | 1,267,948 | 1,268,356 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 17,326 | 18,802 | 19,210 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,227,212 | 26,228,686 | 26,229,094 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 135,632 | 137,108 | 137,516 |

表 1-43 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=MD5)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,266,352 | 1,267,672 | 1,268,080 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 17,216 | 18,530 | 18,938 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,227,090 | 26,228,412 | 26,228,820 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 135,524 | 136,840 | 137,246 |

表 1-44 RSAES-PKCS1-v1_5 暗号化/復号の性能 鍵サイズ 1024bit

| API | 性能 (単位 : サイクル) | |
|-----------------------------|--------------------|----------------------|
| | Message size=1byte | Message size=117byte |
| R_TSIP_RsaesPkcs1024Encrypt | 22,196 | 16,834 |
| R_TSIP_RsaesPkcs1024Decrypt | 1,265,498 | 1,265,488 |

表 1-45 RSAES-PKCS1-v1_5 暗号化/復号の性能 鍵サイズ 2048bit

| API | 性能 (単位 : サイクル) | |
|-----------------------------|--------------------|----------------------|
| | Message size=1byte | Message size=245byte |
| R_TSIP_RsaesPkcs2048Encrypt | 146,486 | 135,000 |
| R_TSIP_RsaesPkcs2048Decrypt | 26,226,422 | 26,226,424 |

表 1-46 HASH(SHA1)の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha1Init | 128 | 128 | 128 |
| R_TSIP_Sha1Update | 1,504 | 1,744 | 1,982 |
| R_TSIP_Sha1Final | 826 | 826 | 826 |

表 1-47 HASH(SHA256)の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha256Init | 182 | 182 | 182 |
| R_TSIP_Sha256Update | 1,552 | 1,756 | 1,960 |
| R_TSIP_Sha256Final | 842 | 842 | 842 |

表 1-48 HASH(MD5)の性能

| API | 性能 (単位 : サイクル) | | |
|------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Md5Init | 124 | 124 | 124 |
| R_TSIP_Md5Update | 1,406 | 1,610 | 1,814 |
| R_TSIP_Md5Final | 780 | 780 | 780 |

表 1-49 共通 API(HMAC ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|-----------------------------------|----------------|
| R_TSIP_GenerateSha1HmacKeyIndex | 2,956 |
| R_TSIP_GenerateSha256HmacKeyIndex | 2,960 |
| R_TSIP_UpdateSha1HmacKeyIndex | 2,580 |
| R_TSIP_UpdateSha256HmacKeyIndex | 2,586 |

表 1-50 HMAC(SHA1)の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha1HmacGenerateInit | 1,366 | 1,366 | 1,366 |
| R_TSIP_Sha1HmacGenerateUpdate | 962 | 1,202 | 1,444 |
| R_TSIP_Sha1HmacGenerateFinal | 1,974 | 1,974 | 1,974 |
| R_TSIP_Sha1HmacVerifyInit | 1,364 | 1,366 | 1,366 |
| R_TSIP_Sha1HmacVerifyUpdate | 972 | 1,212 | 1,452 |
| R_TSIP_Sha1HmacVerifyFinal | 3,602 | 3,602 | 3,604 |

表 1-51 HMAC(SHA256)の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha256HmacGenerateInit | 1,782 | 1,782 | 1,784 |
| R_TSIP_Sha256HmacGenerateUpdate | 900 | 1,106 | 1,308 |
| R_TSIP_Sha256HmacGenerateFinal | 1,946 | 1,946 | 1,946 |
| R_TSIP_Sha256HmacVerifyInit | 1,778 | 1,778 | 1,778 |
| R_TSIP_Sha256HmacVerifyUpdate | 904 | 1,108 | 1,312 |
| R_TSIP_Sha256HmacVerifyFinal | 3,590 | 3,590 | 3,590 |

表 1-52 共通 API(ECC ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|--|----------------|
| R_TSIP_GenerateEccP192PublicKeyIndex | 3,288 |
| R_TSIP_GenerateEccP224PublicKeyIndex | 3,282 |
| R_TSIP_GenerateEccP256PublicKeyIndex | 3,282 |
| R_TSIP_GenerateEccP384PublicKeyIndex | 3,396 |
| R_TSIP_GenerateEccP192PrivateKeyIndex | 2,946 |
| R_TSIP_GenerateEccP224PrivateKeyIndex | 2,948 |
| R_TSIP_GenerateEccP256PrivateKeyIndex | 2,948 |
| R_TSIP_GenerateEccP384PrivateKeyIndex | 2,870 |
| R_TSIP_GenerateEccP192RandomKeyIndex (注) | 143,553 |
| R_TSIP_GenerateEccP224RandomKeyIndex (注) | 154,008 |
| R_TSIP_GenerateEccP256RandomKeyIndex (注) | 154,656 |
| R_TSIP_GenerateEccP384RandomKeyIndex (注) | 1,053,934 |
| R_TSIP_UpdateEccP192PublicKeyIndex | 2,896 |
| R_TSIP_UpdateEccP224PublicKeyIndex | 2,894 |
| R_TSIP_UpdateEccP256PublicKeyIndex | 2,896 |
| R_TSIP_UpdateEccP384PublicKeyIndex | 3,010 |
| R_TSIP_UpdateEccP192PrivateKeyIndex | 2,574 |
| R_TSIP_UpdateEccP224PrivateKeyIndex | 2,576 |
| R_TSIP_UpdateEccP256PrivateKeyIndex | 2,574 |
| R_TSIP_UpdateEccP384PrivateKeyIndex | 2,480 |

【注】 10 回実行時の平均値です。

表 1-53 ECDSA 署名生成/検証の性能

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_EcdsaP192SignatureGenerate | 176,104 | 178,264 | 173,872 |
| R_TSIP_EcdsaP224SignatureGenerate | 177,366 | 175,984 | 179,824 |
| R_TSIP_EcdsaP256SignatureGenerate | 177,528 | 181,006 | 183,418 |
| R_TSIP_EcdsaP384SignatureGenerate(注) | 1,158,780 | | |
| R_TSIP_EcdsaP192SignatureVerification | 330,380 | 329,230 | 329,638 |
| R_TSIP_EcdsaP224SignatureVerification | 346,052 | 350,996 | 352,882 |
| R_TSIP_EcdsaP256SignatureVerification | 350,084 | 352,246 | 355,390 |
| R_TSIP_EcdsaP384SignatureVerification(注) | 2,233,408 | | |

【注】 SHA384 計算は含まれません

表 1-54 鍵共有の性能

| API | 性能 (単位 : サイクル) |
|---|----------------|
| R_TSIP_EcdhP256Init | 58 |
| R_TSIP_EcdhP256ReadPublicKey | 357,856 |
| R_TSIP_EcdhP256MakePublicKey | 328,834 |
| R_TSIP_EcdhP256CalculateSharedSecretIndex | 375,366 |
| R_TSIP_EcdhP256KeyDerivation | 3,772 |
| R_TSIP_EcdheP512KeyAgreement | 3,286,902 |
| R_TSIP_Rsa2048DhKeyAgreement | 52,726,694 |

(KeyAgreement を除いた)鍵共有の性能は、鍵交換形式を ECDHE、派生させる鍵の種類を AES-128 に固定して計測しました。

1.12 性能情報(RX671)

以下に RX671 の TSIP ドライバの性能情報を示します。性能はコアクロックである ICLK のサイクル単位での計測になります。TSIP の動作クロック PCLKB は ICLK : PCLKB = 2 : 1 の設定をしています。

最適化レベル 2 で実施しています。

表 1-55 共通 API の性能

| API | 性能 (単位 : サイクル) |
|--------------------------------------|----------------|
| R_TSIP_Open | 5,349,650 |
| R_TSIP_Close | 300 |
| R_TSIP_GetVersion | 22 |
| R_TSIP_GenerateAes128KeyIndex | 2,034 |
| R_TSIP_GenerateAes256KeyIndex | 2,172 |
| R_TSIP_GenerateAes128RandomKeyIndex | 1,160 |
| R_TSIP_GenerateAes256RandomKeyIndex | 1,608 |
| R_TSIP_GenerateRandomNumber | 530 |
| R_TSIP_GenerateUpdateKeyRingKeyIndex | 2,168 |
| R_TSIP_UpdateAes128KeyIndex | 1,790 |
| R_TSIP_UpdateAes256KeyIndex | 1,924 |

表 1-56 Firmware 検証の性能

| API | 性能 (単位 : サイクル) | | |
|--------------------------|----------------|-----------|-----------|
| | 8K バイト処理 | 16K バイト処理 | 24K バイト処理 |
| R_TSIP_VerifyFirmwareMAC | 16,802 | 33,180 | 49,564 |

表 1-57 AES の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128EcbEncryptInit | 1,218 | 1,200 | 1,200 |
| R_TSIP_Aes128EcbEncryptUpdate | 378 | 486 | 618 |
| R_TSIP_Aes128EcbEncryptFinal | 318 | 306 | 306 |
| R_TSIP_Aes128EcbDecryptInit | 1,216 | 1,216 | 1,216 |
| R_TSIP_Aes128EcbDecryptUpdate | 440 | 540 | 672 |
| R_TSIP_Aes128EcbDecryptFinal | 324 | 324 | 324 |
| R_TSIP_Aes256EcbEncryptInit | 1,334 | 1,312 | 1,310 |
| R_TSIP_Aes256EcbEncryptUpdate | 394 | 516 | 646 |
| R_TSIP_Aes256EcbEncryptFinal | 326 | 322 | 322 |
| R_TSIP_Aes256EcbDecryptInit | 1,324 | 1,324 | 1,324 |
| R_TSIP_Aes256EcbDecryptUpdate | 466 | 582 | 714 |
| R_TSIP_Aes256EcbDecryptFinal | 332 | 332 | 332 |
| R_TSIP_Aes128CbcEncryptInit | 1,268 | 1,262 | 1,262 |
| R_TSIP_Aes128CbcEncryptUpdate | 432 | 540 | 672 |
| R_TSIP_Aes128CbcEncryptFinal | 338 | 336 | 338 |
| R_TSIP_Aes128CbcDecryptInit | 1,286 | 1,284 | 1,286 |
| R_TSIP_Aes128CbcDecryptUpdate | 492 | 590 | 722 |
| R_TSIP_Aes128CbcDecryptFinal | 348 | 346 | 346 |
| R_TSIP_Aes256CbcEncryptInit | 1,386 | 1,376 | 1,376 |
| R_TSIP_Aes256CbcEncryptUpdate | 456 | 576 | 708 |
| R_TSIP_Aes256CbcEncryptFinal | 338 | 336 | 338 |
| R_TSIP_Aes256CbcDecryptInit | 1,394 | 1,394 | 1,396 |
| R_TSIP_Aes256CbcDecryptUpdate | 534 | 650 | 782 |
| R_TSIP_Aes256CbcDecryptFinal | 354 | 352 | 352 |

表 1-58 AES-GCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128GcmEncryptInit | 3,940 | 3,932 | 3,932 |
| R_TSIP_Aes128GcmEncryptUpdate | 1,532 | 1,612 | 1,676 |
| R_TSIP_Aes128GcmEncryptFinal | 820 | 818 | 818 |
| R_TSIP_Aes128GcmDecryptInit | 3,954 | 3,952 | 3,952 |
| R_TSIP_Aes128GcmDecryptUpdate | 1,532 | 1,592 | 1,656 |
| R_TSIP_Aes128GcmDecryptFinal | 1,430 | 1,424 | 1,426 |
| R_TSIP_Aes256GcmEncryptInit | 4,068 | 4,066 | 4,066 |
| R_TSIP_Aes256GcmEncryptUpdate | 1,568 | 1,662 | 1,726 |
| R_TSIP_Aes256GcmEncryptFinal | 832 | 820 | 820 |
| R_TSIP_Aes256GcmDecryptInit | 4,080 | 4,078 | 4,080 |
| R_TSIP_Aes256GcmDecryptUpdate | 1,564 | 1,616 | 1,688 |
| R_TSIP_Aes256GcmDecryptFinal | 1,446 | 1,438 | 1,438 |

GCM の性能は、ivec を 1024bit、追加認証データを 720bit、認証タグを 128bit に固定して計測しました。

表 1-59 AES-CCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CcmEncryptInit | 1,884 | 1,874 | 1,874 |
| R_TSIP_Aes128CcmEncryptUpdate | 880 | 958 | 1,030 |
| R_TSIP_Aes128CcmEncryptFinal | 750 | 740 | 740 |
| R_TSIP_Aes128CcmDecryptInit | 1,720 | 1,700 | 1,700 |
| R_TSIP_Aes128CcmDecryptUpdate | 790 | 856 | 936 |
| R_TSIP_Aes128CcmDecryptFinal | 1,472 | 1,462 | 1,462 |
| R_TSIP_Aes256CcmEncryptInit | 1,866 | 1,854 | 1,854 |
| R_TSIP_Aes256CcmEncryptUpdate | 928 | 1,022 | 1,110 |
| R_TSIP_Aes256CcmEncryptFinal | 768 | 760 | 760 |
| R_TSIP_Aes256CcmDecryptInit | 1,868 | 1,864 | 1,864 |
| R_TSIP_Aes256CcmDecryptUpdate | 836 | 920 | 1,016 |
| R_TSIP_Aes256CcmDecryptFinal | 1,476 | 1,470 | 1,470 |

CCM の性能は、ノンスを 104bit、追加認証データを 880bit、MAC を 128bit に固定して計測しました。

表 1-60 AES-CMAC の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CmacGenerateInit | 868 | 860 | 860 |
| R_TSIP_Aes128CmacGenerateUpdate | 482 | 512 | 554 |
| R_TSIP_Aes128CmacGenerateFinal | 618 | 610 | 610 |
| R_TSIP_Aes128CmacVerifyInit | 868 | 866 | 866 |
| R_TSIP_Aes128CmacVerifyUpdate | 484 | 514 | 556 |
| R_TSIP_Aes128CmacVerifyFinal | 1,230 | 1,226 | 1,226 |
| R_TSIP_Aes256CmacGenerateInit | 978 | 978 | 980 |
| R_TSIP_Aes256CmacGenerateUpdate | 512 | 546 | 596 |
| R_TSIP_Aes256CmacGenerateFinal | 652 | 640 | 640 |
| R_TSIP_Aes256CmacVerifyInit | 980 | 984 | 984 |
| R_TSIP_Aes256CmacVerifyUpdate | 506 | 536 | 586 |
| R_TSIP_Aes256CmacVerifyFinal | 1,254 | 1,248 | 1,248 |

表 1-61 AES Key Wrap の性能

| API | 性能 (単位 : サイクル) | |
|------------------------|----------------|----------------|
| | ラップ対象鍵 AES-128 | ラップ対象鍵 AES-256 |
| R_TSIP_Aes128KeyWrap | 6,322 | 10,012 |
| R_TSIP_Aes256KeyWrap | 6,520 | 10,310 |
| R_TSIP_Aes128KeyUnwrap | 7,102 | 10,708 |
| R_TSIP_Aes256KeyUnwrap | 7,322 | 11,026 |

表 1-62 共通 API(TDES ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|-----------------------------------|----------------|
| R_TSIP_GenerateTdesKeyIndex | 2,174 |
| R_TSIP_GenerateTdesRandomKeyIndex | 1,614 |
| R_TSIP_UpdateTdesKeyIndex | 1,928 |

表 1-63 TDES の性能

| API | 性能 (単位 : サイクル) | | |
|-----------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_TdesEcbEncryptInit | 810 | 802 | 802 |
| R_TSIP_TdesEcbEncryptUpdate | 418 | 602 | 794 |
| R_TSIP_TdesEcbEncryptFinal | 314 | 306 | 304 |
| R_TSIP_TdesEcbDecryptInit | 810 | 812 | 812 |
| R_TSIP_TdesEcbDecryptUpdate | 440 | 628 | 820 |
| R_TSIP_TdesEcbDecryptFinal | 316 | 314 | 314 |
| R_TSIP_TdesCbcEncryptInit | 866 | 854 | 854 |
| R_TSIP_TdesCbcEncryptUpdate | 482 | 676 | 868 |
| R_TSIP_TdesCbcEncryptFinal | 328 | 328 | 328 |
| R_TSIP_TdesCbcDecryptInit | 866 | 866 | 866 |
| R_TSIP_TdesCbcDecryptUpdate | 516 | 704 | 896 |
| R_TSIP_TdesCbcDecryptFinal | 342 | 342 | 342 |

表 1-64 共通 API(RSA ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|--|----------------|
| R_TSIP_GenerateRsa1024PublicKeyIndex | 36,594 |
| R_TSIP_GenerateRsa1024PrivateKeyIndex | 37,598 |
| R_TSIP_GenerateRsa2048PublicKeyIndex | 136,338 |
| R_TSIP_GenerateRsa2048PrivateKeyIndex | 138,330 |
| R_TSIP_GenerateRsa1024RandomKeyIndex (注) | 54,125,760 |
| R_TSIP_GenerateRsa2048RandomKeyIndex (注) | 294,773,725 |
| R_TSIP_UpdateRsa1024PublicKeyIndex | 36,342 |
| R_TSIP_UpdateRsa1024PrivateKeyIndex | 37,332 |
| R_TSIP_UpdateRsa2048PublicKeyIndex | 136,078 |
| R_TSIP_UpdateRsa2048PrivateKeyIndex | 138,064 |

【注】 10 回実行時の平均値です。

表 1-65 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=SHA1)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,232,778 | 1,233,882 | 1,234,296 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 15,960 | 17,086 | 17,502 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,094,764 | 26,095,882 | 26,096,298 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 133,510 | 134,626 | 135,042 |

表 1-66 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=SHA256)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,232,832 | 1,233,972 | 1,234,324 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 16,026 | 17,178 | 17,530 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,094,808 | 26,095,966 | 26,096,322 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 133,566 | 134,716 | 135,068 |

表 1-67 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=MD5)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,232,734 | 1,233,750 | 1,234,102 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 15,928 | 16,962 | 17,314 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,094,712 | 26,095,754 | 26,096,102 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 133,474 | 134,516 | 134,870 |

表 1-68 RSAES-PKCS1-v1_5 暗号化/復号の性能 鍵サイズ 1024bit

| API | 性能 (単位 : サイクル) | |
|-----------------------------|--------------------|----------------------|
| | Message size=1byte | Message size=117byte |
| R_TSIP_RsaesPkcs1024Encrypt | 19,662 | 15,488 |
| R_TSIP_RsaesPkcs1024Decrypt | 1,232,066 | 1,232,064 |

表 1-69 RSAES-PKCS1-v1_5 暗号化/復号の性能 鍵サイズ 2048bit

| API | 性能 (単位 : サイクル) | |
|-----------------------------|--------------------|----------------------|
| | Message size=1byte | Message size=245byte |
| R_TSIP_RsaesPkcs2048Encrypt | 141,756 | 132,866 |
| R_TSIP_RsaesPkcs2048Decrypt | 26,094,272 | 26,094,260 |

表 1-70 HASH(SHA1)の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha1Init | 112 | 110 | 108 |
| R_TSIP_Sha1Update | 1,208 | 1,414 | 1,622 |
| R_TSIP_Sha1Final | 660 | 660 | 658 |

表 1-71 HASH(SHA256)の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha256Init | 154 | 154 | 152 |
| R_TSIP_Sha256Update | 1,226 | 1,398 | 1,574 |
| R_TSIP_Sha256Final | 664 | 662 | 662 |

表 1-72 HASH(MD5)の性能

| API | 性能 (単位 : サイクル) | | |
|------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Md5Init | 108 | 100 | 102 |
| R_TSIP_Md5Update | 1,122 | 1,290 | 1,466 |
| R_TSIP_Md5Final | 626 | 626 | 626 |

表 1-73 共通 API(HMAC ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|-----------------------------------|----------------|
| R_TSIP_GenerateSha1HmacKeyIndex | 2,254 |
| R_TSIP_GenerateSha256HmacKeyIndex | 2,238 |
| R_TSIP_UpdateSha1HmacKeyIndex | 2,006 |
| R_TSIP_UpdateSha256HmacKeyIndex | 1,984 |

表 1-74 HMAC(SHA1)の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha1HmacGenerateInit | 1,038 | 1,036 | 1,032 |
| R_TSIP_Sha1HmacGenerateUpdate | 786 | 992 | 1,200 |
| R_TSIP_Sha1HmacGenerateFinal | 1,584 | 1,578 | 1,576 |
| R_TSIP_Sha1HmacVerifyInit | 1,038 | 1,038 | 1,038 |
| R_TSIP_Sha1HmacVerifyUpdate | 784 | 996 | 1,204 |
| R_TSIP_Sha1HmacVerifyFinal | 2,694 | 2,692 | 2,692 |

表 1-75 HMAC(SHA256)の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha256HmacGenerateInit | 1,234 | 1,228 | 1,226 |
| R_TSIP_Sha256HmacGenerateUpdate | 724 | 894 | 1,070 |
| R_TSIP_Sha256HmacGenerateFinal | 1,546 | 1,534 | 1,534 |
| R_TSIP_Sha256HmacVerifyInit | 1,220 | 1,222 | 1,222 |
| R_TSIP_Sha256HmacVerifyUpdate | 722 | 898 | 1,074 |
| R_TSIP_Sha256HmacVerifyFinal | 2,668 | 2,662 | 2,662 |

表 1-76 共通 API(ECC ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|--|----------------|
| R_TSIP_GenerateEccP192PublicKeyIndex | 2,546 |
| R_TSIP_GenerateEccP224PublicKeyIndex | 2,538 |
| R_TSIP_GenerateEccP256PublicKeyIndex | 2,538 |
| R_TSIP_GenerateEccP384PublicKeyIndex | 2,696 |
| R_TSIP_GenerateEccP192PrivateKeyIndex | 2,252 |
| R_TSIP_GenerateEccP224PrivateKeyIndex | 2,238 |
| R_TSIP_GenerateEccP256PrivateKeyIndex | 2,238 |
| R_TSIP_GenerateEccP384PrivateKeyIndex | 2,288 |
| R_TSIP_GenerateEccP192RandomKeyIndex (注) | 132,597 |
| R_TSIP_GenerateEccP224RandomKeyIndex (注) | 142,060 |
| R_TSIP_GenerateEccP256RandomKeyIndex (注) | 142,198 |
| R_TSIP_GenerateEccP384RandomKeyIndex (注) | 1,009,854 |
| R_TSIP_UpdateEccP192PublicKeyIndex | 2,312 |
| R_TSIP_UpdateEccP224PublicKeyIndex | 2,290 |
| R_TSIP_UpdateEccP256PublicKeyIndex | 2,288 |
| R_TSIP_UpdateEccP384PublicKeyIndex | 2,460 |
| R_TSIP_UpdateEccP192PrivateKeyIndex | 2,004 |
| R_TSIP_UpdateEccP224PrivateKeyIndex | 1,990 |
| R_TSIP_UpdateEccP256PrivateKeyIndex | 1,990 |
| R_TSIP_UpdateEccP384PrivateKeyIndex | 2,046 |

【注】 10 回実行時の平均値です。

表 1-77 ECDSA 署名生成/検証の性能

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_EcdsaP192SignatureGenerate | 160,126 | 164,428 | 166,008 |
| R_TSIP_EcdsaP224SignatureGenerate | 163,544 | 163,420 | 160,816 |
| R_TSIP_EcdsaP256SignatureGenerate | 164,380 | 168,060 | 165,996 |
| R_TSIP_EcdsaP384SignatureGenerate(注) | 1,133,010 | | |
| R_TSIP_EcdsaP192SignatureVerification | 298,614 | 303,594 | 303,346 |
| R_TSIP_EcdsaP224SignatureVerification | 322,254 | 320,898 | 324,414 |
| R_TSIP_EcdsaP256SignatureVerification | 329,212 | 329,672 | 330,672 |
| R_TSIP_EcdsaP384SignatureVerification(注) | 2,101,300 | | |

【注】 SHA384 計算は含まれません

表 1-78 鍵共有の性能

| API | 性能 (単位 : サイクル) |
|---|----------------|
| R_TSIP_EcdhP256Init | 42 |
| R_TSIP_EcdhP256ReadPublicKey | 330,888 |
| R_TSIP_EcdhP256MakePublicKey | 307,876 |
| R_TSIP_EcdhP256CalculateSharedSecretIndex | 350,378 |
| R_TSIP_EcdhP256KeyDerivation | 2,992 |
| R_TSIP_EcdheP512KeyAgreement | 3,158,968 |
| R_TSIP_Rsa2048DhKeyAgreement | 52,461,504 |

(KeyAgreement を除いた)鍵共有の性能は、鍵交換形式を ECDHE、派生させる鍵の種類を AES-128 に固定して計測しました。

1.13 性能情報(RX72M)

以下に RX72M の TSIP ドライバの性能情報を示します。性能はコアクロックである ICLK のサイクル単位での計測になります。TSIP の動作クロック PCLKB は ICLK : PCLKB = 2 : 1 の設定をしています。

最適化レベル 2 で実施しています。

表 1-79 共通 API の性能

| API | 性能 (単位 : サイクル) |
|--------------------------------------|----------------|
| R_TSIP_Open | 6,311,502 |
| R_TSIP_Close | 296 |
| R_TSIP_GetVersion | 18 |
| R_TSIP_GenerateAes128KeyIndex | 2,124 |
| R_TSIP_GenerateAes256KeyIndex | 2,260 |
| R_TSIP_GenerateAes128RandomKeyIndex | 1,238 |
| R_TSIP_GenerateAes256RandomKeyIndex | 1,724 |
| R_TSIP_GenerateRandomNumber | 552 |
| R_TSIP_GenerateUpdateKeyRingKeyIndex | 2,264 |
| R_TSIP_UpdateAes128KeyIndex | 1,868 |
| R_TSIP_UpdateAes256KeyIndex | 2,004 |

表 1-80 Firmware 検証の性能

| API | 性能 (単位 : サイクル) | | |
|--------------------------|----------------|-----------|-----------|
| | 8K バイト処理 | 16K バイト処理 | 24K バイト処理 |
| R_TSIP_VerifyFirmwareMAC | 18,842 | 37,272 | 55,704 |

表 1-81 AES の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128EcbEncryptInit | 1,266 | 1,264 | 1,264 |
| R_TSIP_Aes128EcbEncryptUpdate | 386 | 504 | 640 |
| R_TSIP_Aes128EcbEncryptFinal | 330 | 328 | 328 |
| R_TSIP_Aes128EcbDecryptInit | 1,276 | 1,278 | 1,278 |
| R_TSIP_Aes128EcbDecryptUpdate | 452 | 562 | 698 |
| R_TSIP_Aes128EcbDecryptFinal | 342 | 342 | 342 |
| R_TSIP_Aes256EcbEncryptInit | 1,380 | 1,376 | 1,376 |
| R_TSIP_Aes256EcbEncryptUpdate | 404 | 524 | 662 |
| R_TSIP_Aes256EcbEncryptFinal | 328 | 330 | 330 |
| R_TSIP_Aes256EcbDecryptInit | 1,390 | 1,388 | 1,390 |
| R_TSIP_Aes256EcbDecryptUpdate | 474 | 596 | 732 |
| R_TSIP_Aes256EcbDecryptFinal | 344 | 346 | 346 |
| R_TSIP_Aes128CbcEncryptInit | 1,326 | 1,324 | 1,324 |
| R_TSIP_Aes128CbcEncryptUpdate | 454 | 574 | 708 |
| R_TSIP_Aes128CbcEncryptFinal | 356 | 358 | 356 |
| R_TSIP_Aes128CbcDecryptInit | 1,342 | 1,342 | 1,342 |
| R_TSIP_Aes128CbcDecryptUpdate | 516 | 626 | 762 |
| R_TSIP_Aes128CbcDecryptFinal | 370 | 370 | 372 |
| R_TSIP_Aes256CbcEncryptInit | 1,442 | 1,442 | 1,440 |
| R_TSIP_Aes256CbcEncryptUpdate | 470 | 590 | 726 |
| R_TSIP_Aes256CbcEncryptFinal | 358 | 358 | 358 |
| R_TSIP_Aes256CbcDecryptInit | 1,458 | 1,458 | 1,458 |
| R_TSIP_Aes256CbcDecryptUpdate | 538 | 660 | 796 |
| R_TSIP_Aes256CbcDecryptFinal | 370 | 370 | 368 |

表 1-82 AES-GCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128GcmEncryptInit | 4,324 | 4,324 | 4,322 |
| R_TSIP_Aes128GcmEncryptUpdate | 1,564 | 1,650 | 1,716 |
| R_TSIP_Aes128GcmEncryptFinal | 1,050 | 1,050 | 1,050 |
| R_TSIP_Aes128GcmDecryptInit | 4,226 | 4,226 | 4,226 |
| R_TSIP_Aes128GcmDecryptUpdate | 1,572 | 1,640 | 1,706 |
| R_TSIP_Aes128GcmDecryptFinal | 1,658 | 1,658 | 1,658 |
| R_TSIP_Aes256GcmEncryptInit | 4,278 | 4,278 | 4,278 |
| R_TSIP_Aes256GcmEncryptUpdate | 1,602 | 1,698 | 1,768 |
| R_TSIP_Aes256GcmEncryptFinal | 858 | 858 | 858 |
| R_TSIP_Aes256GcmDecryptInit | 4,284 | 4,282 | 4,284 |
| R_TSIP_Aes256GcmDecryptUpdate | 1,598 | 1,664 | 1,732 |
| R_TSIP_Aes256GcmDecryptFinal | 1,488 | 1,488 | 1,486 |

GCM の性能は、ivec を 1024bit、追加認証データを 720bit、認証タグを 128bit に固定して計測しました。

表 1-83 AES-CCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CcmEncryptInit | 2,356 | 2,350 | 2,350 |
| R_TSIP_Aes128CcmEncryptUpdate | 894 | 962 | 1,040 |
| R_TSIP_Aes128CcmEncryptFinal | 752 | 748 | 750 |
| R_TSIP_Aes128CcmDecryptInit | 2,454 | 2,450 | 2,450 |
| R_TSIP_Aes128CcmDecryptUpdate | 812 | 888 | 966 |
| R_TSIP_Aes128CcmDecryptFinal | 1,440 | 1,440 | 1,440 |
| R_TSIP_Aes256CcmEncryptInit | 1,920 | 1,920 | 1,920 |
| R_TSIP_Aes256CcmEncryptUpdate | 952 | 1,040 | 1,138 |
| R_TSIP_Aes256CcmEncryptFinal | 790 | 790 | 790 |
| R_TSIP_Aes256CcmDecryptInit | 1,926 | 1,926 | 1,926 |
| R_TSIP_Aes256CcmDecryptUpdate | 856 | 952 | 1,040 |
| R_TSIP_Aes256CcmDecryptFinal | 1,512 | 1,510 | 1,510 |

CCM の性能は、ノンスを 104bit、追加認証データを 880bit、MAC を 128bit に固定して計測しました。

表 1-84 AES-CMAC の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CmacGenerateInit | 904 | 902 | 902 |
| R_TSIP_Aes128CmacGenerateUpdate | 490 | 526 | 564 |
| R_TSIP_Aes128CmacGenerateFinal | 630 | 636 | 638 |
| R_TSIP_Aes128CmacVerifyInit | 898 | 902 | 902 |
| R_TSIP_Aes128CmacVerifyUpdate | 486 | 524 | 562 |
| R_TSIP_Aes128CmacVerifyFinal | 1,268 | 1,268 | 1,268 |
| R_TSIP_Aes256CmacGenerateInit | 1,012 | 1,014 | 1,014 |
| R_TSIP_Aes256CmacGenerateUpdate | 512 | 556 | 602 |
| R_TSIP_Aes256CmacGenerateFinal | 664 | 660 | 660 |
| R_TSIP_Aes256CmacVerifyInit | 1,012 | 1,014 | 1,012 |
| R_TSIP_Aes256CmacVerifyUpdate | 512 | 556 | 602 |
| R_TSIP_Aes256CmacVerifyFinal | 1,290 | 1,292 | 1,290 |

表 1-85 AES Key Wrap の性能

| API | 性能 (単位 : サイクル) | |
|------------------------|----------------|----------------|
| | ラップ対象鍵 AES-128 | ラップ対象鍵 AES-256 |
| R_TSIP_Aes128KeyWrap | 6,476 | 10,244 |
| R_TSIP_Aes256KeyWrap | 6,692 | 10,586 |
| R_TSIP_Aes128KeyUnwrap | 7,342 | 11,058 |
| R_TSIP_Aes256KeyUnwrap | 7,566 | 11,398 |

表 1-86 共通 API(TDES ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|-----------------------------------|----------------|
| R_TSIP_GenerateTdesKeyIndex | 2,260 |
| R_TSIP_GenerateTdesRandomKeyIndex | 1,726 |
| R_TSIP_UpdateTdesKeyIndex | 2,016 |

表 1-87 TDES の性能

| API | 性能 (単位 : サイクル) | | |
|-----------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_TdesEcbEncryptInit | 828 | 826 | 826 |
| R_TSIP_TdesEcbEncryptUpdate | 430 | 628 | 828 |
| R_TSIP_TdesEcbEncryptFinal | 336 | 330 | 330 |
| R_TSIP_TdesEcbDecryptInit | 828 | 828 | 828 |
| R_TSIP_TdesEcbDecryptUpdate | 456 | 656 | 856 |
| R_TSIP_TdesEcbDecryptFinal | 350 | 350 | 350 |
| R_TSIP_TdesCbcEncryptInit | 880 | 878 | 878 |
| R_TSIP_TdesCbcEncryptUpdate | 496 | 696 | 896 |
| R_TSIP_TdesCbcEncryptFinal | 360 | 360 | 362 |
| R_TSIP_TdesCbcDecryptInit | 888 | 888 | 888 |
| R_TSIP_TdesCbcDecryptUpdate | 520 | 722 | 922 |
| R_TSIP_TdesCbcDecryptFinal | 370 | 370 | 370 |

表 1-88 共通 API(RSA ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|--|----------------|
| R_TSIP_GenerateRsa1024PublicKeyIndex | 36,738 |
| R_TSIP_GenerateRsa1024PrivateKeyIndex | 37,730 |
| R_TSIP_GenerateRsa2048PublicKeyIndex | 136,500 |
| R_TSIP_GenerateRsa2048PrivateKeyIndex | 138,462 |
| R_TSIP_GenerateRsa1024RandomKeyIndex (注) | 36,044,236 |
| R_TSIP_GenerateRsa2048RandomKeyIndex (注) | 547,787,593 |
| R_TSIP_UpdateRsa1024PublicKeyIndex | 36,492 |
| R_TSIP_UpdateRsa1024PrivateKeyIndex | 37,460 |
| R_TSIP_UpdateRsa2048PublicKeyIndex | 136,252 |
| R_TSIP_UpdateRsa2048PrivateKeyIndex | 138,208 |

【注】 10 回実行時の平均値です。

表 1-89 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=SHA1)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,232,974 | 1,234,158 | 1,234,562 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 16,074 | 17,266 | 17,672 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,095,154 | 26,096,344 | 26,096,750 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 133,728 | 134,912 | 135,322 |

表 1-90 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=SHA256)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,233,052 | 1,234,258 | 1,234,604 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 16,160 | 17,362 | 17,710 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,095,240 | 26,096,444 | 26,096,790 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 133,806 | 135,008 | 135,356 |

表 1-91 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=MD5)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,232,932 | 1,234,016 | 1,234,364 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 16,048 | 17,128 | 17,478 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,095,122 | 26,096,206 | 26,096,552 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 133,690 | 134,766 | 135,114 |

表 1-92 RSAES-PKCS1-v1_5 暗号化/復号の性能 鍵サイズ 1024bit

| API | 性能 (単位 : サイクル) | |
|-----------------------------|--------------------|----------------------|
| | Message size=1byte | Message size=117byte |
| R_TSIP_RsaesPkcs1024Encrypt | 20,108 | 15,644 |
| R_TSIP_RsaesPkcs1024Decrypt | 1,232,284 | 1,232,284 |

表 1-93 RSAES-PKCS1-v1_5 暗号化/復号の性能 鍵サイズ 2048bit

| API | 性能 (単位 : サイクル) | |
|-----------------------------|--------------------|----------------------|
| | Message size=1byte | Message size=245byte |
| R_TSIP_RsaesPkcs2048Encrypt | 142,666 | 133,122 |
| R_TSIP_RsaesPkcs2048Decrypt | 26,094,668 | 26,094,668 |

表 1-94 HASH(SHA1)の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha1Init | 106 | 104 | 104 |
| R_TSIP_Sha1Update | 1,242 | 1,446 | 1,650 |
| R_TSIP_Sha1Final | 664 | 662 | 662 |

表 1-95 HASH(SHA256)の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha256Init | 148 | 148 | 148 |
| R_TSIP_Sha256Update | 1,276 | 1,448 | 1,624 |
| R_TSIP_Sha256Final | 686 | 684 | 684 |

表 1-96 HASH(MD5)の性能

| API | 性能 (単位 : サイクル) | | |
|------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Md5Init | 100 | 98 | 98 |
| R_TSIP_Md5Update | 1,148 | 1,322 | 1,496 |
| R_TSIP_Md5Final | 628 | 628 | 628 |

表 1-97 共通 API(HMAC ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|-----------------------------------|----------------|
| R_TSIP_GenerateSha1HmacKeyIndex | 2,346 |
| R_TSIP_GenerateSha256HmacKeyIndex | 2,342 |
| R_TSIP_UpdateSha1HmacKeyIndex | 2,106 |
| R_TSIP_UpdateSha256HmacKeyIndex | 2,100 |

表 1-98 HMAC(SHA1)の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha1HmacGenerateInit | 1,080 | 1,080 | 1,080 |
| R_TSIP_Sha1HmacGenerateUpdate | 804 | 1,006 | 1,210 |
| R_TSIP_Sha1HmacGenerateFinal | 1,616 | 1,614 | 1,614 |
| R_TSIP_Sha1HmacVerifyInit | 1,080 | 1,080 | 1,082 |
| R_TSIP_Sha1HmacVerifyUpdate | 804 | 1,010 | 1,212 |
| R_TSIP_Sha1HmacVerifyFinal | 2,748 | 2,746 | 2,746 |

表 1-99 HMAC(SHA256)の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha256HmacGenerateInit | 1,408 | 1,406 | 1,406 |
| R_TSIP_Sha256HmacGenerateUpdate | 734 | 906 | 1,078 |
| R_TSIP_Sha256HmacGenerateFinal | 1,582 | 1,580 | 1,580 |
| R_TSIP_Sha256HmacVerifyInit | 1,408 | 1,408 | 1,408 |
| R_TSIP_Sha256HmacVerifyUpdate | 732 | 906 | 1,080 |
| R_TSIP_Sha256HmacVerifyFinal | 2,736 | 2,736 | 2,734 |

表 1-100 共通 API(ECC ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|--|----------------|
| R_TSIP_GenerateEccP192PublicKeyIndex | 2,650 |
| R_TSIP_GenerateEccP224PublicKeyIndex | 2,644 |
| R_TSIP_GenerateEccP256PublicKeyIndex | 2,642 |
| R_TSIP_GenerateEccP384PublicKeyIndex | 2,812 |
| R_TSIP_GenerateEccP192PrivateKeyIndex | 2,350 |
| R_TSIP_GenerateEccP224PrivateKeyIndex | 2,348 |
| R_TSIP_GenerateEccP256PrivateKeyIndex | 2,352 |
| R_TSIP_GenerateEccP384PrivateKeyIndex | 2,380 |
| R_TSIP_GenerateEccP192RandomKeyIndex (注) | 133,181 |
| R_TSIP_GenerateEccP224RandomKeyIndex (注) | 142,484 |
| R_TSIP_GenerateEccP256RandomKeyIndex (注) | 143,212 |
| R_TSIP_GenerateEccP384RandomKeyIndex (注) | 1,001,983 |
| R_TSIP_UpdateEccP192PublicKeyIndex | 2,400 |
| R_TSIP_UpdateEccP224PublicKeyIndex | 2,398 |
| R_TSIP_UpdateEccP256PublicKeyIndex | 2,398 |
| R_TSIP_UpdateEccP384PublicKeyIndex | 2,568 |
| R_TSIP_UpdateEccP192PrivateKeyIndex | 2,108 |
| R_TSIP_UpdateEccP224PrivateKeyIndex | 2,102 |
| R_TSIP_UpdateEccP256PrivateKeyIndex | 2,102 |
| R_TSIP_UpdateEccP384PrivateKeyIndex | 2,124 |

【注】 10 回実行時の平均値です。

表 1-101 ECDSA 署名生成/検証の性能

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_EcdsaP192SignatureGenerate | 163,264 | 162,488 | 163,534 |
| R_TSIP_EcdsaP224SignatureGenerate | 163,642 | 165,936 | 163,324 |
| R_TSIP_EcdsaP256SignatureGenerate | 166,324 | 168,118 | 167,926 |
| R_TSIP_EcdsaP384SignatureGenerate(注) | 1,082,840 | | |
| R_TSIP_EcdsaP192SignatureVerification | 306,388 | 306,412 | 301,572 |
| R_TSIP_EcdsaP224SignatureVerification | 326,756 | 327,964 | 319,370 |
| R_TSIP_EcdsaP256SignatureVerification | 327,288 | 329,824 | 333,924 |
| R_TSIP_EcdsaP384SignatureVerification(注) | 2,099,972 | | |

【注】 SHA384 計算は含まれません

表 1-102 鍵共有の性能

| API | 性能 (単位 : サイクル) |
|---|----------------|
| R_TSIP_EcdhP256Init | 38 |
| R_TSIP_EcdhP256ReadPublicKey | 332,290 |
| R_TSIP_EcdhP256MakePublicKey | 307,870 |
| R_TSIP_EcdhP256CalculateSharedSecretIndex | 350,762 |
| R_TSIP_EcdhP256KeyDerivation | 3,122 |
| R_TSIP_EcdheP512KeyAgreement | 3,299,252 |
| R_TSIP_Rsa2048DhKeyAgreement | 52,462,430 |

(KeyAgreement を除いた)鍵共有の性能は、鍵交換形式を ECDHE、派生させる鍵の種類を AES-128 に固定して計測しました。

1.14 性能情報(RX72N)

以下に RX72N の TSIP ドライバの性能情報を示します。性能はコアクロックである ICLK のサイクル単位での計測になります。TSIP の動作クロック PCLKB は ICLK : PCLKB = 2 : 1 の設定をしています。

最適化レベル 2 で実施しています。

表 1-103 共通 API の性能

| API | 性能 (単位 : サイクル) |
|--------------------------------------|----------------|
| R_TSIP_Open | 6,209,080 |
| R_TSIP_Close | 302 |
| R_TSIP_GetVersion | 20 |
| R_TSIP_GenerateAes128KeyIndex | 2,136 |
| R_TSIP_GenerateAes256KeyIndex | 2,256 |
| R_TSIP_GenerateAes128RandomKeyIndex | 1,246 |
| R_TSIP_GenerateAes256RandomKeyIndex | 1,730 |
| R_TSIP_GenerateRandomNumber | 560 |
| R_TSIP_GenerateUpdateKeyRingKeyIndex | 2,260 |
| R_TSIP_UpdateAes128KeyIndex | 1,874 |
| R_TSIP_UpdateAes256KeyIndex | 2,006 |

表 1-104 Firmware 検証の性能

| API | 性能 (単位 : サイクル) | | |
|--------------------------|----------------|-----------|-----------|
| | 8K バイト処理 | 16K バイト処理 | 24K バイト処理 |
| R_TSIP_VerifyFirmwareMAC | 18,850 | 37,280 | 55,712 |

表 1-105 AES の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128EcbEncryptInit | 1,266 | 1,264 | 1,264 |
| R_TSIP_Aes128EcbEncryptUpdate | 380 | 498 | 634 |
| R_TSIP_Aes128EcbEncryptFinal | 334 | 332 | 332 |
| R_TSIP_Aes128EcbDecryptInit | 1,278 | 1,278 | 1,276 |
| R_TSIP_Aes128EcbDecryptUpdate | 448 | 558 | 694 |
| R_TSIP_Aes128EcbDecryptFinal | 348 | 346 | 348 |
| R_TSIP_Aes256EcbEncryptInit | 1,386 | 1,380 | 1,380 |
| R_TSIP_Aes256EcbEncryptUpdate | 396 | 520 | 656 |
| R_TSIP_Aes256EcbEncryptFinal | 332 | 332 | 332 |
| R_TSIP_Aes256EcbDecryptInit | 1,392 | 1,392 | 1,392 |
| R_TSIP_Aes256EcbDecryptUpdate | 470 | 592 | 728 |
| R_TSIP_Aes256EcbDecryptFinal | 346 | 346 | 348 |
| R_TSIP_Aes128CbcEncryptInit | 1,322 | 1,324 | 1,324 |
| R_TSIP_Aes128CbcEncryptUpdate | 448 | 566 | 702 |
| R_TSIP_Aes128CbcEncryptFinal | 362 | 362 | 362 |
| R_TSIP_Aes128CbcDecryptInit | 1,342 | 1,344 | 1,344 |
| R_TSIP_Aes128CbcDecryptUpdate | 514 | 626 | 762 |
| R_TSIP_Aes128CbcDecryptFinal | 374 | 372 | 372 |
| R_TSIP_Aes256CbcEncryptInit | 1,442 | 1,440 | 1,442 |
| R_TSIP_Aes256CbcEncryptUpdate | 466 | 590 | 726 |
| R_TSIP_Aes256CbcEncryptFinal | 362 | 362 | 360 |
| R_TSIP_Aes256CbcDecryptInit | 1,460 | 1,458 | 1,460 |
| R_TSIP_Aes256CbcDecryptUpdate | 534 | 660 | 796 |
| R_TSIP_Aes256CbcDecryptFinal | 372 | 372 | 372 |

表 1-106 AES-GCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128GcmEncryptInit | 4,332 | 4,334 | 4,334 |
| R_TSIP_Aes128GcmEncryptUpdate | 1,574 | 1,656 | 1,726 |
| R_TSIP_Aes128GcmEncryptFinal | 1,048 | 1,046 | 1,046 |
| R_TSIP_Aes128GcmDecryptInit | 4,216 | 4,214 | 4,214 |
| R_TSIP_Aes128GcmDecryptUpdate | 1,566 | 1,632 | 1,700 |
| R_TSIP_Aes128GcmDecryptFinal | 1,654 | 1,654 | 1,652 |
| R_TSIP_Aes256GcmEncryptInit | 4,280 | 4,276 | 4,276 |
| R_TSIP_Aes256GcmEncryptUpdate | 1,586 | 1,686 | 1,754 |
| R_TSIP_Aes256GcmEncryptFinal | 864 | 860 | 860 |
| R_TSIP_Aes256GcmDecryptInit | 4,290 | 4,292 | 4,292 |
| R_TSIP_Aes256GcmDecryptUpdate | 1,590 | 1,654 | 1,722 |
| R_TSIP_Aes256GcmDecryptFinal | 1,476 | 1,478 | 1,476 |

GCM の性能は、ivec を 1024bit、追加認証データを 720bit、認証タグを 128bit に固定して計測しました。

表 1-107 AES-CCM の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CcmEncryptInit | 2,360 | 2,350 | 2,350 |
| R_TSIP_Aes128CcmEncryptUpdate | 890 | 958 | 1,036 |
| R_TSIP_Aes128CcmEncryptFinal | 746 | 746 | 746 |
| R_TSIP_Aes128CcmDecryptInit | 2,458 | 2,456 | 2,456 |
| R_TSIP_Aes128CcmDecryptUpdate | 814 | 890 | 968 |
| R_TSIP_Aes128CcmDecryptFinal | 1,440 | 1,440 | 1,440 |
| R_TSIP_Aes256CcmEncryptInit | 1,924 | 1,924 | 1,924 |
| R_TSIP_Aes256CcmEncryptUpdate | 954 | 1,042 | 1,140 |
| R_TSIP_Aes256CcmEncryptFinal | 790 | 790 | 790 |
| R_TSIP_Aes256CcmDecryptInit | 1,924 | 1,920 | 1,922 |
| R_TSIP_Aes256CcmDecryptUpdate | 852 | 950 | 1,038 |
| R_TSIP_Aes256CcmDecryptFinal | 1,512 | 1,508 | 1,510 |

CCM の性能は、ノンスを 104bit、追加認証データを 880bit、MAC を 128bit に固定して計測しました。

表 1-108 AES-CMAC の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|----------|----------|
| | 48 バイト処理 | 64 バイト処理 | 80 バイト処理 |
| R_TSIP_Aes128CmacGenerateInit | 912 | 908 | 908 |
| R_TSIP_Aes128CmacGenerateUpdate | 480 | 516 | 552 |
| R_TSIP_Aes128CmacGenerateFinal | 624 | 622 | 622 |
| R_TSIP_Aes128CmacVerifyInit | 904 | 910 | 908 |
| R_TSIP_Aes128CmacVerifyUpdate | 482 | 518 | 554 |
| R_TSIP_Aes128CmacVerifyFinal | 1,252 | 1,252 | 1,252 |
| R_TSIP_Aes256CmacGenerateInit | 1,022 | 1,020 | 1,020 |
| R_TSIP_Aes256CmacGenerateUpdate | 512 | 556 | 604 |
| R_TSIP_Aes256CmacGenerateFinal | 656 | 654 | 654 |
| R_TSIP_Aes256CmacVerifyInit | 1,020 | 1,020 | 1,020 |
| R_TSIP_Aes256CmacVerifyUpdate | 514 | 558 | 604 |
| R_TSIP_Aes256CmacVerifyFinal | 1,286 | 1,286 | 1,284 |

表 1-109 AES Key Wrap の性能

| API | 性能 (単位 : サイクル) | |
|------------------------|----------------|----------------|
| | ラップ対象鍵 AES-128 | ラップ対象鍵 AES-256 |
| R_TSIP_Aes128KeyWrap | 6,476 | 10,248 |
| R_TSIP_Aes256KeyWrap | 6,698 | 10,598 |
| R_TSIP_Aes128KeyUnwrap | 7,342 | 11,060 |
| R_TSIP_Aes256KeyUnwrap | 7,570 | 11,410 |

表 1-110 共通 API(TDES ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|-----------------------------------|----------------|
| R_TSIP_GenerateTdesKeyIndex | 2,252 |
| R_TSIP_GenerateTdesRandomKeyIndex | 1,730 |
| R_TSIP_UpdateTdesKeyIndex | 2,002 |

表 1-111 TDES の性能

| API | 性能 (単位 : サイクル) | | |
|-----------------------------|----------------|----------|----------|
| | 16 バイト処理 | 48 バイト処理 | 80 バイト処理 |
| R_TSIP_TdesEcbEncryptInit | 826 | 820 | 822 |
| R_TSIP_TdesEcbEncryptUpdate | 430 | 628 | 826 |
| R_TSIP_TdesEcbEncryptFinal | 320 | 320 | 320 |
| R_TSIP_TdesEcbDecryptInit | 832 | 830 | 830 |
| R_TSIP_TdesEcbDecryptUpdate | 456 | 656 | 856 |
| R_TSIP_TdesEcbDecryptFinal | 338 | 340 | 340 |
| R_TSIP_TdesCbcEncryptInit | 880 | 878 | 878 |
| R_TSIP_TdesCbcEncryptUpdate | 488 | 688 | 888 |
| R_TSIP_TdesCbcEncryptFinal | 348 | 348 | 346 |
| R_TSIP_TdesCbcDecryptInit | 888 | 886 | 888 |
| R_TSIP_TdesCbcDecryptUpdate | 520 | 722 | 922 |
| R_TSIP_TdesCbcDecryptFinal | 360 | 360 | 360 |

表 1-112 共通 API(RSA ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|--|----------------|
| R_TSIP_GenerateRsa1024PublicKeyIndex | 36,742 |
| R_TSIP_GenerateRsa1024PrivateKeyIndex | 37,724 |
| R_TSIP_GenerateRsa2048PublicKeyIndex | 136,500 |
| R_TSIP_GenerateRsa2048PrivateKeyIndex | 138,464 |
| R_TSIP_GenerateRsa1024RandomKeyIndex (注) | 35,336,114 |
| R_TSIP_GenerateRsa2048RandomKeyIndex (注) | 327,733,742 |
| R_TSIP_UpdateRsa1024PublicKeyIndex | 36,486 |
| R_TSIP_UpdateRsa1024PrivateKeyIndex | 37,454 |
| R_TSIP_UpdateRsa2048PublicKeyIndex | 136,244 |
| R_TSIP_UpdateRsa2048PrivateKeyIndex | 138,196 |

【注】 10 回実行時の平均値です。

表 1-113 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=SHA1)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,232,996 | 1,234,172 | 1,234,584 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 16,086 | 17,276 | 17,684 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,095,164 | 26,096,356 | 26,096,764 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 133,738 | 134,928 | 135,336 |

表 1-114 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=SHA256)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,233,054 | 1,234,242 | 1,234,590 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 16,150 | 17,346 | 17,694 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,095,226 | 26,096,426 | 26,096,776 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 133,806 | 135,000 | 135,348 |

表 1-115 RSASSA-PKCS1-v1_5 署名生成/検証の性能(HASH=MD5)

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_RsassaPkcs1024SignatureGenerate | 1,232,946 | 1,234,026 | 1,234,374 |
| R_TSIP_RsassaPkcs1024SignatureVerification | 16,054 | 17,136 | 17,484 |
| R_TSIP_RsassaPkcs2048SignatureGenerate | 26,095,128 | 26,096,212 | 26,096,560 |
| R_TSIP_RsassaPkcs2048SignatureVerification | 133,704 | 134,786 | 135,134 |

表 1-116 RSAES-PKCS1-v1_5 暗号化/復号の性能 鍵サイズ 1024bit

| API | 性能 (単位 : サイクル) | |
|-----------------------------|--------------------|----------------------|
| | Message size=1byte | Message size=117byte |
| R_TSIP_RsaesPkcs1024Encrypt | 20,198 | 15,704 |
| R_TSIP_RsaesPkcs1024Decrypt | 1,232,282 | 1,232,282 |

表 1-117 RSAES-PKCS1-v1_5 暗号化/復号の性能 鍵サイズ 2048bit

| API | 性能 (単位 : サイクル) | |
|-----------------------------|--------------------|----------------------|
| | Message size=1byte | Message size=245byte |
| R_TSIP_RsaesPkcs2048Encrypt | 142,724 | 133,130 |
| R_TSIP_RsaesPkcs2048Decrypt | 26,094,662 | 26,094,662 |

表 1-118 HASH(SHA1)の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha1Init | 102 | 102 | 104 |
| R_TSIP_Sha1Update | 1,252 | 1,456 | 1,660 |
| R_TSIP_Sha1Final | 668 | 668 | 668 |

表 1-119 HASH(SHA256)の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha256Init | 146 | 146 | 146 |
| R_TSIP_Sha256Update | 1,262 | 1,434 | 1,608 |
| R_TSIP_Sha256Final | 682 | 678 | 678 |

表 1-120 HASH(MD5)の性能

| API | 性能 (単位 : サイクル) | | |
|------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Md5Init | 100 | 98 | 96 |
| R_TSIP_Md5Update | 1,156 | 1,330 | 1,504 |
| R_TSIP_Md5Final | 636 | 636 | 636 |

表 1-121 共通 API(HMAC ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|-----------------------------------|----------------|
| R_TSIP_GenerateSha1HmacKeyIndex | 2,346 |
| R_TSIP_GenerateSha256HmacKeyIndex | 2,340 |
| R_TSIP_UpdateSha1HmacKeyIndex | 2,102 |
| R_TSIP_UpdateSha256HmacKeyIndex | 2,098 |

表 1-122 HMAC(SHA1)の性能

| API | 性能 (単位 : サイクル) | | |
|-------------------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha1HmacGenerateInit | 1,084 | 1,080 | 1,080 |
| R_TSIP_Sha1HmacGenerateUpdate | 802 | 1,002 | 1,206 |
| R_TSIP_Sha1HmacGenerateFinal | 1,616 | 1,616 | 1,616 |
| R_TSIP_Sha1HmacVerifyInit | 1,080 | 1,082 | 1,082 |
| R_TSIP_Sha1HmacVerifyUpdate | 802 | 1,006 | 1,210 |
| R_TSIP_Sha1HmacVerifyFinal | 2,752 | 2,750 | 2,750 |

表 1-123 HMAC(SHA256)の性能

| API | 性能 (単位 : サイクル) | | |
|---------------------------------|----------------|-----------|-----------|
| | 128 バイト処理 | 192 バイト処理 | 256 バイト処理 |
| R_TSIP_Sha256HmacGenerateInit | 1,414 | 1,410 | 1,410 |
| R_TSIP_Sha256HmacGenerateUpdate | 732 | 904 | 1,076 |
| R_TSIP_Sha256HmacGenerateFinal | 1,582 | 1,580 | 1,580 |
| R_TSIP_Sha256HmacVerifyInit | 1,408 | 1,408 | 1,408 |
| R_TSIP_Sha256HmacVerifyUpdate | 730 | 904 | 1,078 |
| R_TSIP_Sha256HmacVerifyFinal | 2,742 | 2,744 | 2,742 |

表 1-124 共通 API(ECC ユーザ鍵生成情報生成)の性能

| API | 性能 (単位 : サイクル) |
|--|----------------|
| R_TSIP_GenerateEccP192PublicKeyIndex | 2,646 |
| R_TSIP_GenerateEccP224PublicKeyIndex | 2,644 |
| R_TSIP_GenerateEccP256PublicKeyIndex | 2,644 |
| R_TSIP_GenerateEccP384PublicKeyIndex | 2,816 |
| R_TSIP_GenerateEccP192PrivateKeyIndex | 2,342 |
| R_TSIP_GenerateEccP224PrivateKeyIndex | 2,340 |
| R_TSIP_GenerateEccP256PrivateKeyIndex | 2,342 |
| R_TSIP_GenerateEccP384PrivateKeyIndex | 2,374 |
| R_TSIP_GenerateEccP192RandomKeyIndex (注) | 133,526 |
| R_TSIP_GenerateEccP224RandomKeyIndex (注) | 141,792 |
| R_TSIP_GenerateEccP256RandomKeyIndex (注) | 143,159 |
| R_TSIP_GenerateEccP384RandomKeyIndex (注) | 1,013,024 |
| R_TSIP_UpdateEccP192PublicKeyIndex | 2,412 |
| R_TSIP_UpdateEccP224PublicKeyIndex | 2,402 |
| R_TSIP_UpdateEccP256PublicKeyIndex | 2,400 |
| R_TSIP_UpdateEccP384PublicKeyIndex | 2,562 |
| R_TSIP_UpdateEccP192PrivateKeyIndex | 2,102 |
| R_TSIP_UpdateEccP224PrivateKeyIndex | 2,096 |
| R_TSIP_UpdateEccP256PrivateKeyIndex | 2,098 |
| R_TSIP_UpdateEccP384PrivateKeyIndex | 2,130 |

【注】 10 回実行時の平均値です。

表 1-125 ECDSA 署名生成/検証の性能

| API | 性能 (単位 : サイクル) | | |
|--|--------------------|----------------------|----------------------|
| | Message size=1byte | Message size=128byte | Message size=256byte |
| R_TSIP_EcdsaP192SignatureGenerate | 163,274 | 165,056 | 164,224 |
| R_TSIP_EcdsaP224SignatureGenerate | 163,644 | 163,514 | 165,142 |
| R_TSIP_EcdsaP256SignatureGenerate | 163,764 | 168,714 | 169,204 |
| R_TSIP_EcdsaP384SignatureGenerate(注) | 1,112,166 | | |
| R_TSIP_EcdsaP192SignatureVerification | 306,430 | 307,040 | 308,616 |
| R_TSIP_EcdsaP224SignatureVerification | 321,672 | 322,232 | 322,478 |
| R_TSIP_EcdsaP256SignatureVerification | 333,054 | 332,974 | 330,710 |
| R_TSIP_EcdsaP384SignatureVerification(注) | 2,080,726 | | |

【注】 SHA384 計算は含まれません

表 1-126 鍵共有の性能

| API | 性能 (単位 : サイクル) |
|---|----------------|
| R_TSIP_EcdhP256Init | 40 |
| R_TSIP_EcdhP256ReadPublicKey | 332,294 |
| R_TSIP_EcdhP256MakePublicKey | 309,134 |
| R_TSIP_EcdhP256CalculateSharedSecretIndex | 350,764 |
| R_TSIP_EcdhP256KeyDerivation | 3,118 |
| R_TSIP_EcdheP512KeyAgreement | 3,196,194 |
| R_TSIP_Rsa2048DhKeyAgreement | 52,462,412 |

(KeyAgreement を除いた)鍵共有の性能は、鍵交換形式を ECDHE、派生させる鍵の種類を AES-128 に固定して計測しました。

2. API 情報

2.1 ハードウェアの要求

TSIP ドライバは、MCU 内蔵の TSIP 機能に依存します。RX231 グループ、RX23W グループ、RX65N、RX651 グループ、RX66N グループ、RX66T グループ、RX671 グループ、RX72M グループ、RX72N グループ、または RX72T グループの内、TSIP を搭載している型名のものをご使用ください。

2.2 ソフトウェアの要求

TSIP ドライバは、以下モジュールに依存します。

- r_bsp V7.00 以降をご使用ください。(BSP=Board Support Package)

■RX231、RX23W を使用する場合(RX231 では、下記コメントの" = Chip"以降が一部異なります。)

r_config フォルダの r_bsp_config.h の以下マクロの値を 0xB、0xD(RX23W のみ)のいずれかに変更してください。

```
/* Chip version.
Character(s) = Value for macro =
A          = 0xA          = Chip version A
                        = Security function not included.
B          = 0xB          = Chip version B
                        = Security function included.
C          = 0xC          = Chip version C
                        = Security function not included.
D          = 0xD          = Chip version D
                        = Security function included.
*/
#define BSP_CFG_MCU_PART_VERSION    (0xB)
```

■RX66T、RX72T を使用する場合(RX72T では、下記コメントの"= PGA"以降が一部異なります。)

r_config フォルダの r_bsp_config.h の以下マクロの値を 0xE、0xF、0x10 のいずれかに変更してください。

```
/* Whether PGA differential input, Encryption and USB are included or not.
Character(s) = Value for macro = Description
A          = 0xA          = PGA differential input included, Encryption module not included,
                        USB module not included
B          = 0xB          = PGA differential input not included, Encryption module not included,
                        USB module not included
C          = 0xC          = PGA differential input included, Encryption module not included,
                        USB module included
E          = 0xE          = PGA differential input included, Encryption module included,
                        USB module not included
F          = 0xF          = PGA differential input not included, Encryption module included,
                        USB module not included
G          = 0x10         = PGA differential input included, Encryption module included,
                        USB module included
*/
#define BSP_CFG_MCU_PART_FUNCTION    (0xE)
```

■RX66N、RX671、RX72M、RX72N を使用する場合

r_config フォルダの r_bsp_config.h の以下マクロの値を 0x11 に変更してください。

```
/* Whether Encryption is included or not.
Character(s) = Value for macro = Description
D           = 0xD           = Encryption module not included
H           = 0x11          = Encryption module included
*/
#define BSP_CFG_MCU_PART_FUNCTION    (0x11)
```

■RX65N を使用する場合

r_config フォルダの r_bsp_config.h の以下マクロの値を true に変更してください。

```
/* Whether Encryption and SDHI/SDSI are included or not.
Character(s) = Value for macro = Description
A           = false           = Encryption module not included, SDHI/SDSI module not included
B           = false           = Encryption module not included, SDHI/SDSI module included
D           = false           = Encryption module not included, SDHI/SDSI module included
E           = true            = Encryption module included, SDHI/SDSI module not included
F           = true            = Encryption module included, SDHI/SDSI module included
H           = true            = Encryption module included, SDHI/SDSI module included
*/
#define BSP_CFG_MCU_PART_ENCRYPTION_INCLUDED (true)
```

2.3 サポートされているツールチェーン

TSIP ドライバは、以下のツールチェーンで動作を確認しています。

RX ファミリ用 C/C++コンパイラパッケージ V3.04.00

2.4 ヘッダファイル

すべての API 呼び出しとそれをサポートするインタフェース定義は r_tsip_rx_if.h に記載しています。

2.5 整数型

このプロジェクトは ANSI C99 を使用しています。

2.6 API データ構造

TSIP ドライバが使用するデータ構造体についての情報は r_tsip_rx_if.h を参照してください。

2.7 戻り値

以下に本モジュールの API 関数で使える戻り値を示します。戻り値の列挙型は、API 関数の宣言と共に `r_tsip_rx_if.h` に記載されています。

```
typedef enum e_tsip_err
{
    TSIP_SUCCESS=0,
    TSIP_ERR_FAIL, // 自己診断が異常終了
                  // R_TSIP_VerifyFirmwareMAC による MAC 異常検出
                  // または R_TSIP_各 API の内部エラー
    TSIP_ERR_RESOURCE_CONFLICT, // 本処理に必要なリソースが他の処理で利用されている
                  // ことによるリソース衝突が発生
    TSIP_ERR_RETRY, // 自己診断が異常終了。本関数を再実行してください。
    TSIP_ERR_KEY_SET, // 異常な鍵生成情報が入力された
    TSIP_ERR_AUTHENTICATION, // 認証が失敗
                  // または RSASSA-PKCS1-V.1.5 による署名文検証失敗
    TSIP_ERR_CALLBACK_UNREGIST, // コールバック関数未登録
    TSIP_ERR_PARAMETER, // 入力データが不正
    TSIP_ERR_PROHIBIT_FUNCTION, // 不正な関数呼び出しが発生した
    TSIP_RESUME_FIRMWARE_GENERATE_MAC, // 処理の続きがあります。API の再呼び出しが必要
    TSIP_ERR_VERIFICATION_FAIL, // TLS1.3 のハンドシェイク検証が失敗
}e_tsip_err_t
```

2.8 FIT モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。ルネサスでは、Smart Configurator を使用した(1)、(3)の追加方法を推奨しています。ただし、Smart Configurator は、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

- (1) e² studio 上で Smart Configurator を使用して FIT モジュールを追加する場合
e² studio の Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。
- (2) e² studio 上で FIT Configurator を使用して FIT モジュールを追加する場合
e² studio の FIT Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加することができます。詳細は、アプリケーションノート「RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。
- (3) CS+上で Smart Configurator を使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版 Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。
- (4) CS+上で FIT モジュールを追加する場合
CS+上で、手動でユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」を参照してください。

3. API 関数

3.1 API 一覧

TSIP ドライバでは、以下の API を実装しています。

- ① TSIP 初期化関連の API
- ② AES/DES/ARC4/RSA/ECC 暗号および HMAC で使用するユーザ鍵生成情報を生成する API、鍵更新用の鍵生成情報を生成する API、および、ユーザ鍵生成情報を更新する API
- ③ AES、DES、ARC4、RSA、ECC のユーザ鍵生成情報を乱数から自動生成するための API
- ④ 乱数を生成するための API
- ⑤ 各種暗号アルゴリズムの API
- ⑥ ファームウェアアップデートやブートをセキュアに行うための API
- ⑦ SSL/TLS 連携機能 API
- ⑧ 鍵共有のための API
- ⑨ Key Wrap のための API

表 3-1 API

| 一 覧 分 類 | API | 説明 | TSIP -Lite | TSIP |
|------------------|---------------------------------------|--------------------------------|---------------|------|
| ① | R_TSIP_Open | TSIP 機能を有効にします | ✓ | ✓ |
| | R_TSIP_Close | TSIP 機能を無効にします | ✓ | ✓ |
| | R_TSIP_SoftwareReset | TSIP モジュールをリセットします。 | ✓ | ✓ |
| | R_TSIP_GetVersion | TSIP ドライバのバージョンを出力します。 | ✓ | ✓ |
| ② | R_TSIP_GenerateAes128KeyIndex | AES 128 ビット用ユーザ鍵生成情報を生成します。 | ✓ | ✓ |
| | R_TSIP_GenerateAes256KeyIndex | AES256 ビット用ユーザ鍵生成情報を生成します。 | ✓ | ✓ |
| | R_TSIP_GenerateUpdateKeyRingKeyIndex | 鍵更新用鍵束用の鍵生成情報を生成します。 | ✓ | ✓ |
| | R_TSIP_GenerateTdesKeyIndex | Triple-DES 用のユーザ鍵生成情報を生成します。 | | ✓ |
| | R_TSIP_GenerateArc4KeyIndex | ARC4 用のユーザ鍵生成情報を生成します。 | | ✓ |
| | R_TSIP_GenerateRsa1024PrivateKeyIndex | RSA1024 ビット秘密鍵用ユーザ鍵生成情報を生成します。 | | ✓ |
| | R_TSIP_GenerateRsa1024PublicKeyIndex | RSA1024 ビット公開鍵用ユーザ鍵生成情報を生成します。 | | ✓ |
| | R_TSIP_GenerateRsa2048PrivateKeyIndex | RSA2048 ビット秘密鍵用ユーザ鍵生成情報を生成します。 | | ✓ |

| | | | |
|---------------------------------------|--------------------------------|---|---|
| R_TSIP_GenerateRsa2048PublicKeyIndex | RSA2048 ビット公開鍵用ユーザ鍵生成情報を生成します。 | | ✓ |
| R_TSIP_GenerateTlsRsaPublicKeyIndex | TLS 連携で使用する RSA 公開鍵生成情報を生成します。 | | ✓ |
| R_TSIP_GenerateEccP192PublicKeyIndex | ECC P-192 公開鍵用ユーザ鍵生成情報を生成します。 | | ✓ |
| R_TSIP_GenerateEccP224PublicKeyIndex | ECC P-224 公開鍵用ユーザ鍵生成情報を生成します。 | | ✓ |
| R_TSIP_GenerateEccP256PublicKeyIndex | ECC P-256 公開鍵用ユーザ鍵生成情報を生成します。 | | ✓ |
| R_TSIP_GenerateEccP384PublicKeyIndex | ECC P-384 公開鍵用ユーザ鍵生成情報を生成します。 | | ✓ |
| R_TSIP_GenerateEccP192PrivateKeyIndex | ECC P-192 秘密鍵用ユーザ鍵生成情報を生成します。 | | ✓ |
| R_TSIP_GenerateEccP224PrivateKeyIndex | ECC P-224 秘密鍵用ユーザ鍵生成情報を生成します。 | | ✓ |
| R_TSIP_GenerateEccP256PrivateKeyIndex | ECC P-256 秘密鍵用ユーザ鍵生成情報を生成します。 | | ✓ |
| R_TSIP_GenerateEccP384PrivateKeyIndex | ECC P-384 秘密鍵用ユーザ鍵生成情報を生成します。 | | ✓ |
| R_TSIP_GenerateSha1HmacKeyIndex | SHA1-HMAC 用ユーザ鍵生成情報を生成します。 | | ✓ |
| R_TSIP_GenerateSha256HmacKeyIndex | SHA256-HMAC 用ユーザ鍵生成情報を生成します。 | | ✓ |
| R_TSIP_UpdateAes128KeyIndex | AES 128 ビット用ユーザ鍵生成情報を更新します。 | ✓ | ✓ |
| R_TSIP_UpdateAes256KeyIndex | AES256 ビット用ユーザ鍵生成情報を更新します。 | ✓ | ✓ |
| R_TSIP_UpdateTdesKeyIndex | TDES 用ユーザ鍵生成情報を更新します。 | | ✓ |
| R_TSIP_UpdateArc4KeyIndex | ARC4 用ユーザ鍵生成情報を更新します。 | | ✓ |
| R_TSIP_UpdateRsa1024PrivateKeyIndex | RSA1024 ビット秘密鍵用ユーザ鍵生成情報を更新します。 | | ✓ |
| R_TSIP_UpdateRsa1024PublicKeyIndex | RSA1024 ビット公開鍵用ユーザ鍵生成情報を更新します。 | | ✓ |
| R_TSIP_UpdateRsa2048PrivateKeyIndex | RSA2048 ビット秘密鍵 | | ✓ |

| | | | | |
|---|--------------------------------------|---|---|---|
| | | 用ユーザ鍵生成情報を更新します。 | | |
| | R_TSIP_UpdateRsa2048PublicKeyIndex | RSA2048 ビット公開鍵用ユーザ鍵生成情報を更新します。 | | ✓ |
| | R_TSIP_UpdateTlsRsaPublicKeyIndex | TLS 連携で使用する RSA 公開鍵鍵生成情報を更新します。 | | ✓ |
| | R_TSIP_UpdateEccP192PublicKeyIndex | ECC P-192 公開鍵用ユーザ鍵生成情報を更新します。 | | ✓ |
| | R_TSIP_UpdateEccP224PublicKeyIndex | ECC P-224 公開鍵用ユーザ鍵生成情報を更新します。 | | ✓ |
| | R_TSIP_UpdateEccP256PublicKeyIndex | ECC P-256 公開鍵用ユーザ鍵生成情報を更新します。 | | ✓ |
| | R_TSIP_UpdateEccP384PublicKeyIndex | ECC P-384 公開鍵用ユーザ鍵生成情報を更新します。 | | ✓ |
| | R_TSIP_UpdateEccP192PrivateKeyIndex | ECC P-192 秘密鍵用ユーザ鍵生成情報を更新します。 | | ✓ |
| | R_TSIP_UpdateEccP224PrivateKeyIndex | ECC P-224 秘密鍵用ユーザ鍵生成情報を更新します。 | | ✓ |
| | R_TSIP_UpdateEccP256PrivateKeyIndex | ECC P-256 秘密鍵用ユーザ鍵生成情報を更新します。 | | ✓ |
| | R_TSIP_UpdateEccP384PrivateKeyIndex | ECC P-384 秘密鍵用ユーザ鍵生成情報を更新します。 | | ✓ |
| | R_TSIP_UpdateSha1HmacKeyIndex | SHA1-HMAC 用ユーザ鍵生成情報を更新します。 | | ✓ |
| | R_TSIP_UpdateSha256HmacKeyIndex | SHA256-HMAC 用ユーザ鍵生成情報を更新します。 | | ✓ |
| ③ | R_TSIP_GenerateAes128RandomKeyIndex | AES128 ビット用ユーザ鍵生成情報を生成します。 | ✓ | ✓ |
| | R_TSIP_GenerateAes256RandomKeyIndex | AES256 ビット用ユーザ鍵生成情報を生成します。 | ✓ | ✓ |
| | R_TSIP_GenerateTdesRandomKeyIndex | Triple-DES 用ユーザ鍵生成情報を生成します。 | | ✓ |
| | R_TSIP_GenerateArc4RandomKeyIndex | ARC4 用ユーザ鍵生成情報を生成します。 | | ✓ |
| | R_TSIP_GenerateRsa1024RandomKeyIndex | RSA1024 ビット秘密鍵用ユーザ鍵生成情報と対応する公開鍵を生成します。公開鍵指数部は 0x10001 固定です。 | | ✓ |

| | | | | |
|---|--------------------------------------|---|---|---|
| | R_TSIP_GenerateRsa2048RandomKeyIndex | RSA2048 ビット秘密鍵用ユーザ鍵生成情報と対応する公開鍵を生成します。公開鍵指数部は 0x10001 固定です。 | | ✓ |
| | R_TSIP_GenerateTlsP256EccKeyIndex | TLS 連携機能で使用する乱数から 256bit 素体上の楕円曲線暗号のための鍵ペアを生成します。 | | ✓ |
| | R_TSIP_GenerateTls13P256EccKeyIndex | TLS1.3 連携機能で使用する乱数から 256bit 素体上の楕円曲線暗号のための鍵ペアを生成します。 | | ✓ |
| | R_TSIP_GenerateEccP192RandomKeyIndex | ECC P-192 秘密鍵用ユーザ鍵生成情報と対応する公開鍵を生成します。 | | ✓ |
| | R_TSIP_GenerateEccP224RandomKeyIndex | ECC P-224 秘密鍵用ユーザ鍵生成情報と対応する公開鍵を生成します。 | | ✓ |
| | R_TSIP_GenerateEccP256RandomKeyIndex | ECC P-256 秘密鍵用ユーザ鍵生成情報と対応する公開鍵を生成します。 | | ✓ |
| | R_TSIP_GenerateEccP384RandomKeyIndex | ECC P-384 秘密鍵用ユーザ鍵生成情報と対応する公開鍵を生成します。 | | ✓ |
| ④ | R_TSIP_GenerateRandomNumber | 乱数を生成します。 | ✓ | ✓ |
| ⑤ | R_TSIP_Aes128EcbEncryptInit | AES128 ビット用ユーザ鍵生成情報を用いて AES128-ECB モード暗号化を行う準備をします。 | ✓ | ✓ |
| | R_TSIP_Aes128EcbEncryptUpdate | AES128-ECB モード暗号化をします。 | ✓ | ✓ |
| | R_TSIP_Aes128EcbEncryptFinal | AES128-ECB モード暗号化の終了処理を行います。 | ✓ | ✓ |
| | R_TSIP_Aes128EcbDecryptInit | AES128 ビット用ユーザ鍵生成情報を用いて AES128-ECB モード復号を行う準備をします。 | ✓ | ✓ |
| | R_TSIP_Aes128EcbDecryptUpdate | AES128-ECB モード復号をします。 | ✓ | ✓ |
| | R_TSIP_Aes128EcbDecryptFinal | AES128-ECB モード復号の終了処理をします。 | ✓ | ✓ |
| | R_TSIP_Aes256EcbEncryptInit | AES256 ビット用ユーザ鍵生成情報を用いて AES256-ECB モード暗号化を行う準備をしま | ✓ | ✓ |

| | | | | |
|-------------------------------|--|--|---|---|
| | | す。 | | |
| R_TSIP_Aes256EcbEncryptUpdate | | AES256-ECB モード暗号化します。 | ✓ | ✓ |
| R_TSIP_Aes256EcbEncryptFinal | | AES256-ECB モード暗号化の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes256EcbDecryptInit | | AES256 ビット用ユーザ鍵生成情報を用いて AES256-ECB モード復号を行う準備をします。 | ✓ | ✓ |
| R_TSIP_Aes256EcbDecryptUpdate | | AES256-ECB モード復号をします。 | ✓ | ✓ |
| R_TSIP_Aes256EcbDecryptFinal | | AES256-ECB モード復号の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes128CbcEncryptInit | | AES128 ビット用ユーザ鍵生成情報を用いて AES128-CBC モードで暗号化を行う準備をします。 | ✓ | ✓ |
| R_TSIP_Aes128CbcEncryptUpdate | | AES128-CBC モードで暗号化します。 | ✓ | ✓ |
| R_TSIP_Aes128CbcEncryptFinal | | AES128-CBC モード暗号化の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes128CbcDecryptInit | | AES128 ビット用ユーザ鍵生成情報を用いて AES128-CBC モード復号を行う準備をします。 | ✓ | ✓ |
| R_TSIP_Aes128CbcDecryptUpdate | | AES128-CBC モード復号をします。 | ✓ | ✓ |
| R_TSIP_Aes128CbcDecryptFinal | | AES128-CBC モード復号の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes256CbcEncryptInit | | AES256 ビット用ユーザ鍵生成情報を用いて AES256-CBC モード暗号化を行う準備をします。 | ✓ | ✓ |
| R_TSIP_Aes256CbcEncryptUpdate | | AES256-CBC モード暗号化をします。 | ✓ | ✓ |
| R_TSIP_Aes256CbcEncryptFinal | | AES256-CBC モード暗号化の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes256CbcDecryptInit | | AES256 ビット用ユーザ鍵生成情報を用いて AES256-CBC モード復号を行う準備をします。 | ✓ | ✓ |
| R_TSIP_Aes256CbcDecryptUpdate | | AES256-CBC モード復号をします。 | ✓ | ✓ |
| R_TSIP_Aes256CbcDecryptFinal | | AES256-CBC モード復号の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes128GcmEncryptInit | | AES128 ビット用ユーザ鍵生成情報を用いて | ✓ | ✓ |

| | | | | |
|-------------------------------|--|--|---|---|
| | | AES128-GCM 暗号化の準備をします。 | | |
| R_TSIP_Aes128GcmEncryptUpdate | | AES128-GCM 暗号化をします。 | ✓ | ✓ |
| R_TSIP_Aes128GcmEncryptFinal | | AES128-GCM 暗号化の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes128GcmDecryptInit | | AES128 ビット用ユーザ鍵生成情報を用いて AES128-GCM 復号の準備をします。 | ✓ | ✓ |
| R_TSIP_Aes128GcmDecryptUpdate | | AES128-GCM 復号をします。 | ✓ | ✓ |
| R_TSIP_Aes128GcmDecryptFinal | | AES128-GCM 復号の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes256GcmEncryptInit | | AES256 ビット用ユーザ鍵生成情報を用いて AES256-GCM 暗号化の準備をします。 | ✓ | ✓ |
| R_TSIP_Aes256GcmEncryptUpdate | | AES256-GCM 暗号化をします。 | ✓ | ✓ |
| R_TSIP_Aes256GcmEncryptFinal | | AES256-GCM 暗号化の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes256GcmDecryptInit | | AES256 ビット用ユーザ鍵生成情報を用いて AES256-GCM 復号の準備をします。 | ✓ | ✓ |
| R_TSIP_Aes256GcmDecryptUpdate | | AES256-GCM 復号をします。 | ✓ | ✓ |
| R_TSIP_Aes256GcmDecryptFinal | | AES256-GCM 復号の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes128CcmEncryptInit | | AES128 ビット用ユーザ鍵生成情報を用いて AES128-CCM 暗号化の準備をします。 | ✓ | ✓ |
| R_TSIP_Aes128CcmEncryptUpdate | | AES128-CCM の暗号化をします。 | ✓ | ✓ |
| R_TSIP_Aes128CcmEncryptFinal | | AES128-CCM 暗号化の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes128CcmDecryptInit | | AES128 ビット用ユーザ鍵生成情報を用いて AES128-CCM 復号の準備をします。 | ✓ | ✓ |
| R_TSIP_Aes128CcmDecryptUpdate | | AES-128CCM の復号処理をします。 | ✓ | ✓ |
| R_TSIP_Aes128CcmDecryptFinal | | AES-128CCM 復号の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes256CcmEncryptInit | | AES256 ビット用ユーザ鍵生成情報を用いて AES256-CCM 暗号化の準備をします。 | ✓ | ✓ |
| R_TSIP_Aes256CcmEncryptUpdate | | AES256-CCM の暗号化をします。 | ✓ | ✓ |

| | | | |
|---------------------------------|---|---|---|
| R_TSIP_Aes256CcmEncryptFinal | AES256-CCM 暗号化の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes256CcmDecryptInit | AES256 ビット用ユーザ鍵生成情報を用いて AES256-CCM 復号の準備をします。 | ✓ | ✓ |
| R_TSIP_Aes256CcmDecryptUpdate | AES-256CCM の復号処理をします。 | ✓ | ✓ |
| R_TSIP_Aes256CcmDecryptFinal | AES-256CCM 復号の終了処理をします。 | ✓ | ✓ |
| R_TSIP_Aes128CmacGenerateInit | AES128 ビット用ユーザ鍵生成情報を用いて AES128-CMAC モード MAC 生成を行う準備をします。 | ✓ | ✓ |
| R_TSIP_Aes128CmacGenerateUpdate | AES128-CMAC モード MAC 生成を行います。 | ✓ | ✓ |
| R_TSIP_Aes128CmacGenerateFinal | AES128-CMAC モード MAC 生成の終了処理を行います。 | ✓ | ✓ |
| R_TSIP_Aes128CmacVerifyInit | AES128 ビット用ユーザ鍵生成情報を用いて AES128-CMAC モードで生成された MAC の検証を行う準備をします。 | ✓ | ✓ |
| R_TSIP_Aes128CmacVerifyUpdate | AES128-CMAC モードで生成された MAC の検証を行います。 | ✓ | ✓ |
| R_TSIP_Aes128CmacVerifyFinal | AES128-CMAC モードで生成された MAC の検証の終了処理を行います。 | ✓ | ✓ |
| R_TSIP_Aes256CmacGenerateInit | AES256 ビット用ユーザ鍵生成情報を用いて AES256-CMAC モード MAC 生成を行う準備をします。 | ✓ | ✓ |
| R_TSIP_Aes256CmacGenerateUpdate | AES256-CMAC モード MAC 生成を行います。 | ✓ | ✓ |
| R_TSIP_Aes256CmacGenerateFinal | AES256-CMAC モード MAC 生成の終了処理を行います。 | ✓ | ✓ |
| R_TSIP_Aes256CmacVerifyInit | AES256 ビット用ユーザ鍵生成情報を用いて AES256-CMAC モードで生成された MAC の検証を行う準備をします。 | ✓ | ✓ |
| R_TSIP_Aes256CmacVerifyUpdate | AES256-CMAC モードで生成された MAC の検証を行います。 | ✓ | ✓ |
| R_TSIP_Aes256CmacVerifyFinal | AES256 ビット鍵を用いて CMAC モードで生成された MAC の検証の終 | ✓ | ✓ |

| | | | | |
|--|--|---|--|---|
| | | 了処理を行います。 | | |
| R_TSIP_TdesEcbEncryptInit | | TDES-ECB モード暗号化を行う準備をします。 | | ✓ |
| R_TSIP_TdesEcbEncryptUpdate | | TDES-ECB モード暗号化します。 | | ✓ |
| R_TSIP_TdesEcbEncryptFinal | | TDES-ECB モード暗号化の終了処理をします。 | | ✓ |
| R_TSIP_TdesEcbDecryptInit | | TDES-ECB モード復号を行う準備をします。 | | ✓ |
| R_TSIP_TdesEcbDecryptUpdate | | TDES-ECB モード復号をします。 | | ✓ |
| R_TSIP_TdesEcbDecryptFinal | | TDES-ECB モード復号の終了処理をします。 | | ✓ |
| R_TSIP_TdesCbcEncryptInit | | TDES-CBC モードで暗号化を行う準備をします。 | | ✓ |
| R_TSIP_TdesCbcEncryptUpdate | | TDES-CBC モードで暗号化します。 | | ✓ |
| R_TSIP_TdesCbcEncryptFinal | | TDES-CBC モード暗号化の終了処理をします。 | | ✓ |
| R_TSIP_TdesCbcDecryptInit | | TDES-CBC モード復号を行う準備をします。 | | ✓ |
| R_TSIP_TdesCbcDecryptUpdate | | TDES-CBC モード復号をします。 | | ✓ |
| R_TSIP_TdesCbcDecryptFinal | | TDES-CBC モード復号の終了処理をします。 | | ✓ |
| R_TSIP_Arc4EncryptInit | | ARC4 暗号化を行う準備をします。 | | ✓ |
| R_TSIP_Arc4EncryptUpdate | | ARC4 暗号化します。 | | ✓ |
| R_TSIP_Arc4EncryptFinal | | ARC4 暗号化の終了処理をします。 | | ✓ |
| R_TSIP_Arc4DecryptInit | | ARC4 復号を行う準備をします。 | | ✓ |
| R_TSIP_Arc4DecryptUpdate | | ARC4 復号をします。 | | ✓ |
| R_TSIP_Arc4DecryptFinal | | ARC4 復号の終了処理をします。 | | ✓ |
| R_TSIP_RsaesPkcs1024Encrypt | | RSAES-PKCS1-V1_5 による 1024bit RSA 暗号化をします。 | | ✓ |
| R_TSIP_RsaesPkcs1024Decrypt | | RSAES-PKCS1-V1_5 による 1024bit RSA 復号をします。 | | ✓ |
| R_TSIP_RsaesPkcs2048Encrypt | | RSAES-PKCS1-V1_5 による 2048bit RSA 暗号化をします。 | | ✓ |
| R_TSIP_RsaesPkcs2048Decrypt | | RSAES-PKCS1-V1_5 による 2048bit RSA 復号をします。 | | ✓ |
| R_TSIP_RsassaPkcs1024SignatureGenerate | | RSASSA-PKCS1-V1_5 による 1024bit 電子署名を生成します。 | | ✓ |

| | | | |
|--|---|--|---|
| R_TSIP_RsassaPkcs1024SignatureVerification | RSASSA-PKCS1-V1_5 による 1024bit 電子署名 の検証をします。 | | ✓ |
| R_TSIP_RsassaPkcs2048SignatureGenerate | RSASSA-PKCS1-V1_5 による 2048bit 電子署名 を生成します。 | | ✓ |
| R_TSIP_RsassaPkcs2048SignatureVerification | RSASSA-PKCS1-V1_5 による 2048bit 電子署名 の検証をします。 | | ✓ |
| R_TSIP_Sha1Init | SHA-1 によるハッシュ値 生成を行う準備をしま す。 | | ✓ |
| R_TSIP_Sha1Update | SHA-1 によるハッシュ値 生成を行います。 | | ✓ |
| R_TSIP_Sha1Final | SHA-1 によるハッシュ値 生成の終了処理をしま す。 | | ✓ |
| R_TSIP_Sha256Init | SHA-256 によるハッ シュ値生成を行う準備を します。 | | ✓ |
| R_TSIP_Sha256Update | SHA-256 によるハッ シュ値生成を行います。 | | ✓ |
| R_TSIP_Sha256Final | SHA-256 によるハッ シュ値生成の終了処理を します。 | | ✓ |
| R_TSIP_Sha1HmacGenerateInit | SHA1-HMAC 演算をする 準備をします。 | | ✓ |
| R_TSIP_Sha1HmacGenerateUpdate | SHA1-HMAC 演算をしま す。 | | ✓ |
| R_TSIP_Sha1HmacGenerateFinal | SHA1-HMAC 演算の終了 処理をします。 | | ✓ |
| R_TSIP_Sha256HmacGenerateInit | SHA256-HMAC 演算を する準備をします。 | | ✓ |
| R_TSIP_Sha256HmacGenerateUpdate | SHA256-HMAC 演算を します。 | | ✓ |
| R_TSIP_Sha256HmacGenerateFinal | SHA256-HMAC 演算の 終了処理をします。 | | ✓ |
| R_TSIP_Sha1HmacVerifyInit | SHA1-HMAC 演算検証を する準備をします。 | | ✓ |
| R_TSIP_Sha1HmacVerifyUpdate | SHA1-HMAC 演算検証を します。 | | ✓ |
| R_TSIP_Sha1HmacVerifyFinal | SHA1-HMAC 演算検証の 終了処理をします。 | | ✓ |
| R_TSIP_Sha256HmacVerifyInit | SHA256-HMAC 演算検 証をする準備をします。 | | ✓ |
| R_TSIP_Sha256HmacVerifyUpdate | SHA256-HMAC 演算検 証をします。 | | ✓ |
| R_TSIP_Sha256HmacVerifyFinal | SHA256-HMAC 演算検 証の終了処理をします。 | | ✓ |
| R_TSIP_Md5Init | MD5 によるハッシュ値 生成を行う準備をしま す。 | | ✓ |

| | | | | |
|---|--|------------------------------------|---|---|
| | R_TSIP_Md5Update | MD5 によるハッシュ値生成を行います。 | | ✓ |
| | R_TSIP_Md5Final | MD5 によるハッシュ値生成の終了処理をします。 | | ✓ |
| | R_TSIP_GetCurrentHashDigestValue | ハッシュ値演算途中経過を取得します。 | | ✓ |
| | R_TSIP_EcdsaP192SignatureGenerate | ECDSA P-192 による電子署名を生成します。 | | ✓ |
| | R_TSIP_EcdsaP224SignatureGenerate | ECDSA P-224 による電子署名を生成します。 | | ✓ |
| | R_TSIP_EcdsaP256SignatureGenerate | ECDSA P-256 による電子署名を生成します。 | | ✓ |
| | R_TSIP_EcdsaP384SignatureGenerate | ECDSA P-384 による電子署名を生成します。 | | ✓ |
| | R_TSIP_EcdsaP192SignatureVerification | ECDSA P-192 による電子署名の検証をします。 | | ✓ |
| | R_TSIP_EcdsaP224SignatureVerification | ECDSA P-224 による電子署名の検証をします。 | | ✓ |
| | R_TSIP_EcdsaP256SignatureVerification | ECDSA P-256 による電子署名の検証をします。 | | ✓ |
| | R_TSIP_EcdsaP384SignatureVerification | ECDSA P-384 による電子署名の検証をします。 | | ✓ |
| ⑥ | R_TSIP_StartUpdateFirmware | ファームウェアアップデートモードに遷移します。 | ✓ | ✓ |
| | R_TSIP_GenerateFirmwareMAC | 暗号化されたファームウェアの復号と MAC 生成を行います。 | ✓ | ✓ |
| | R_TSIP_VerifyFirmwareMAC | ファームウェアの MAC チェックを行います。 | ✓ | ✓ |
| ⑦ | R_TSIP_TlsRootCertificateVerification | ルート CA 証明書の束を検証します。 | | ✓ |
| | R_TSIP_TlsCertificateVerification | サーバ証明書、中間証明書の署名を検証します。 | | ✓ |
| | R_TSIP_TlsCertificateVerificationExtension | サーバ証明書、中間証明書の署名を検証します。 | | ✓ |
| | R_TSIP_TlsGeneratePreMasterSecret | 暗号化された PreMasterSecret を生成します。 | | ✓ |
| | R_TSIP_TlsEncryptPreMasterSecretWithRsa2048PublicKey | PreMasterSecret を RSA2048 で暗号化します。 | | ✓ |
| | R_TSIP_TlsGenerateMasterSecret | 暗号化された MasterSecret を生成します。 | | ✓ |
| | R_TSIP_TlsGenerateSessionKey | TLS 通信の各種鍵を出力します。 | | ✓ |
| | R_TSIP_TlsGenerateVerifyData | Verify データを生成します。 | | ✓ |
| | R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrieves | ServerKeyExchange の | | ✓ |

| | | | | |
|---|--|--|--|---|
| | | 署名を検証します。 | | |
| R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key | | ECC で暗号化された PreMasterSecret を生成します。 | | ✓ |
| R_TSIP_Tls13GenerateEcdheSharedSecret | | Shared Secret 鍵生成情報を生成します。 | | ✓ |
| R_TSIP_Tls13GenerateHandshakeSecret | | Handshake Secret 鍵生成情報を生成します。 | | ✓ |
| R_TSIP_Tls13GenerateServerHandshakeTrafficKey | | Server Write Key 及び Server Finished Key の鍵生成情報を生成します。 | | ✓ |
| R_TSIP_Tls13ServerHandshakeVerification | | サーバから提供される Finished の情報を検証します。 | | ✓ |
| R_TSIP_Tls13GenerateClientHandshakeTrafficKey | | Client Write Key 及び Client Finished Key の鍵生成情報を生成します。 | | ✓ |
| R_TSIP_Tls13GenerateMasterSecret | | Master Secret の鍵生成情報を生成します。 | | ✓ |
| R_TSIP_Tls13GenerateApplicationTrafficKey | | Application Traffic Secret と Application Traffic Key の鍵生成情報を生成します。 | | ✓ |
| R_TSIP_Tls13UpdateApplicationTrafficKey | | Application Traffic Secret と Application Traffic Key の鍵生成情報を更新します。 | | ✓ |
| R_TSIP_Tls13EncryptInit | | TLS1.3 通信データの暗号化を行う準備をします。 | | ✓ |
| R_TSIP_Tls13EncryptUpdate | | TLS1.3 通信データの暗号化をします。 | | ✓ |
| R_TSIP_Tls13EncryptFinal | | TLS1.3 通信データの暗号化の終了処理をします。 | | ✓ |
| R_TSIP_Tls13DecryptInit | | TLS1.3 通信データの復号を行う準備をします。 | | ✓ |
| R_TSIP_Tls13DecryptUpdate | | TLS1.3 通信データの復号をします。 | | ✓ |
| R_TSIP_Tls13DecryptFinal | | TLS1.3 通信データの復号の終了処理をします。 | | ✓ |
| R_TSIP_Tls13CertificateVerifyGenerate | | サーバに送信する CertificateVerify を生成します。 | | ✓ |
| R_TSIP_Tls13CertificateVerifyVerification | | サーバから受信した | | ✓ |

| | | | | |
|---|---|--------------------------------------|---|---|
| | | CertificateVerify を検証します。 | | |
| ⑧ | R_TSIP_EcdhP256Init | ECDH P-256 鍵交換演算の準備をします | | ✓ |
| | R_TSIP_EcdhP256ReadPublicKey | 鍵共有相手の ECC P-256 公開鍵の署名を検証します。 | | ✓ |
| | R_TSIP_EcdhP256MakePublicKey | ECC P-256 秘密鍵に署名をつけます。 | | ✓ |
| | R_TSIP_EcdhP256CalculateSharedSecretIndex | 鍵共有相手の公開鍵と自分の秘密鍵から、共有秘密 Z を計算します。 | | ✓ |
| | R_TSIP_EcdhP256KeyDerivation | Z から共有鍵を導出します。 | | ✓ |
| | R_TSIP_EcdheP512KeyAgreement | Brainpool P512r1 を用いて ECDHE 演算を行います。 | | ✓ |
| | R_TSIP_Rsa2048DhKeyAgreement | RSA-2048 による DH 演算を実施します。 | | ✓ |
| ⑨ | R_TSIP_Aes128KeyWrap | AES 128 鍵で、鍵をラップします。 | ✓ | ✓ |
| | R_TSIP_Aes256KeyWrap | AES 128 鍵で、鍵をアンラップします。 | ✓ | ✓ |
| | R_TSIP_Aes128KeyUnwrap | AES 256 鍵で、鍵をラップします。 | ✓ | ✓ |
| | R_TSIP_Aes256KeyUnwrap | AES 256 鍵で、鍵をアンラップします。 | ✓ | ✓ |

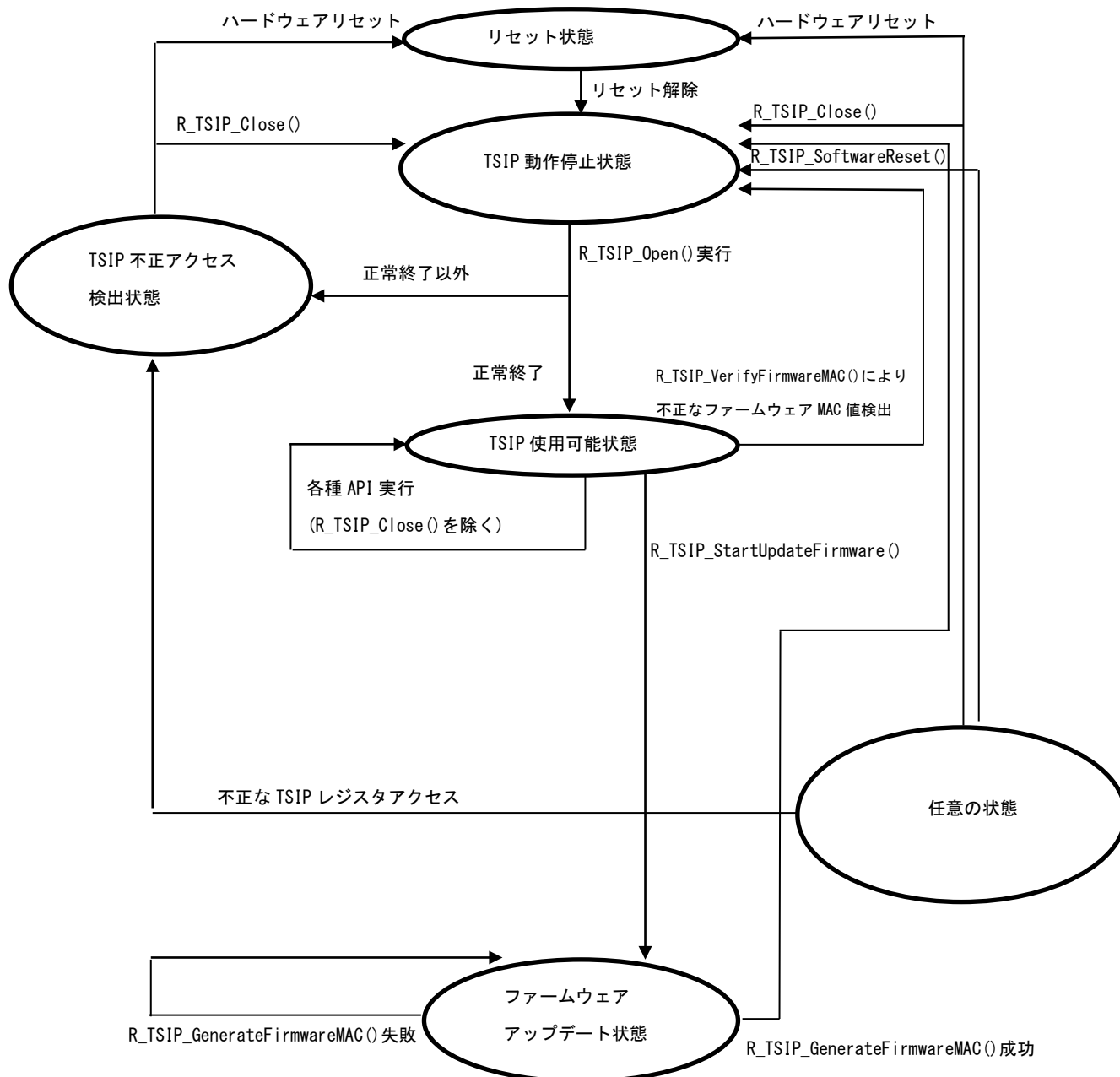
3.2 状態遷移図

TSIP はソフトウェアによる TSIP レジスタアクセスを監視しています。

TSIP は適切な状態遷移と制御手順の元、API 関数の実行を許可します。

TSIP は不正な TSIP レジスタアクセスを検出すると TSIP 不正アクセス検出状態に遷移し、処理途中で無限ループとなります。ウォッチドッグタイマ等を使用してこの無限ループを検出し、システム動作を復旧させることを推奨します。

以下に TSIP の状態遷移図を示します。



【注】 R_TSIP_Open()実行中に RX をスタンバイモードに遷移させないでください。これを防ぐため、R_TSIP_Open()では、割り込み禁止 API の R_BSP_InterruptsDisable()と割り込み許可 API の R_BSP_InterruptsEnable()を呼び出しています。

3.3 API 使用時の注意事項

3.3.1 各 API の呼び出し方法

TSIP ドライバは各アルゴリズム API を実行するときに、アルゴリズムごとに Init API→Update API→Final API を呼ぶ必要があります。複数のアルゴリズムを同時に使用することができません。例えば AES-ECB 128key の暗号化と復号を同時に使用する場合、R_TSIP_Aes128EcbEncryptInit()を呼び出し後、R_TSIP_Aes128EcbEncryptFinal()呼び出し前に R_TSIP_Aes128EcbDecryptInit()を呼び出すような使用方法はできません。呼び出し順が正常に行われなかった場合は、戻り値で TSIP_ERR_RESOURCE_CONFLICT もしくは TSIP_ERR_PROHIBIT_FUNCTION を返します。

ただし、ハッシュ演算(SHA-1, SHA-256, MD5)API は AES などの他のアルゴリズムと同時に使用することが可能です。例えば、R_TSIP_Sha1Init()→R_TSIP_Sha1Update()→R_TSIP_Aes128EcbEncryptInit()→R_TSIP_Aes128EcbEncryptUpdate()→R_TSIP_Aes128EcbEncryptFinal()→R_TSIP_Sha1Update()→R_TSIP_Sha1Final()のような呼び方をすることが可能です。

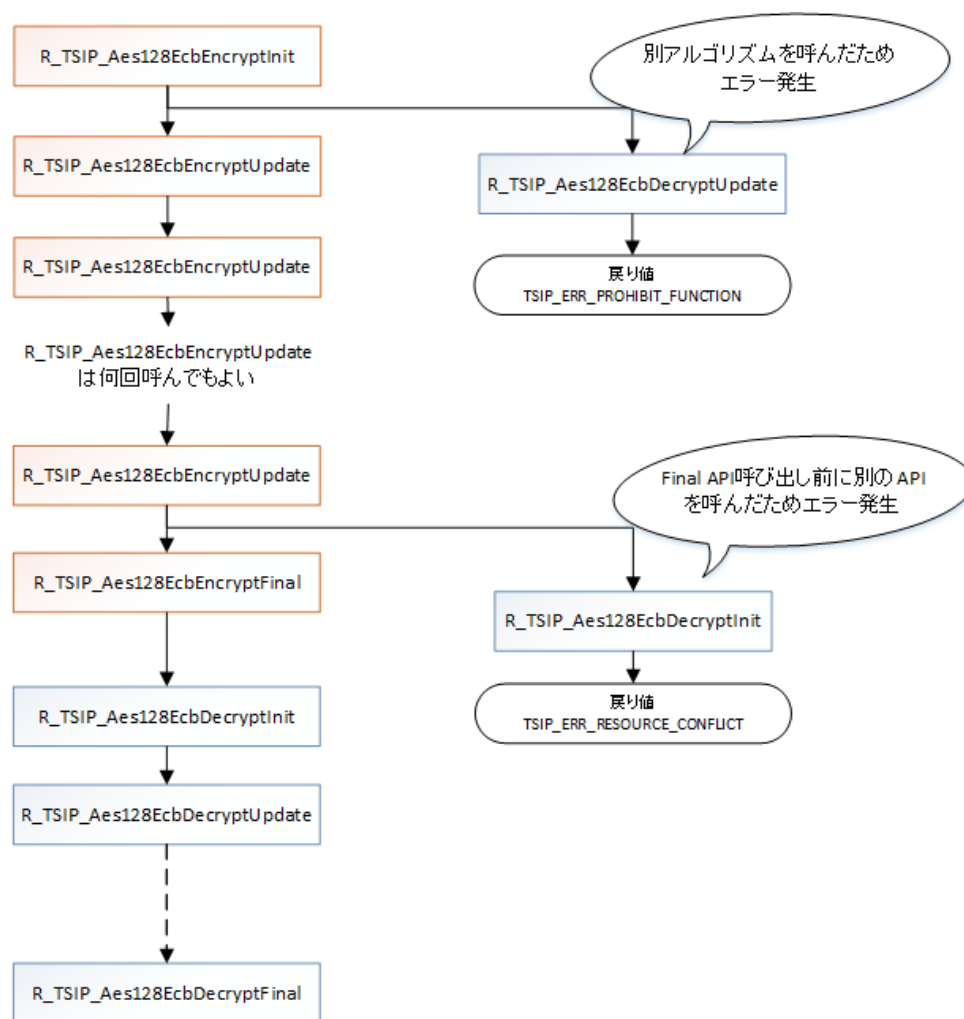


図 3-2 AES-ECB 128 の暗号化、復号を使用する例

3.3.2 BSP FIT モジュールに関する注意事項

TSIP ドライバは、2.2 章にあるように、内部で BSP FIT モジュールを使用しています。TSIP ドライバを使用する際には、以下の API をリンクしてください。詳細は、「ボードサポートパッケージモジュール Firmware Integration Technology アプリケーションノート(R01AN1685xJxxxx)」を参照してください。

- R_BSP_RegisterProtectEnable()
- R_BSP_RegisterProtectDisable()
- R_BSP_InterruptsEnable()
- R_BSP_InterruptsDisable()

また、これらの API が呼び出される前に、BSP のスタートアップが完了していることを想定しています。BSP のスタートアップを使用しない場合、事前に R_BSP_StartupOpen()を呼び出してください。上記 API 内で使用する内部変数の初期化を行います。

4. API 関数詳細説明(TSIP-Lite/TSIP 共通)

4.1 R_TSIP_Open

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Open (
    tsip_tls_ca_certification_public_key_index_t *key_index_1,
    tsip_update_key_ring_t *key_index_2
)
```

Parameters

| | | |
|-------------|----|----------------------|
| key_index_1 | 入力 | TLS 連携 RSA 公開鍵束鍵生成情報 |
| key_index_2 | 入力 | 鍵更新用鍵束鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS : | 正常終了 |
| TSIP_ERR_FAIL: | 自己診断が異常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_RETRY: | 自己診断が異常終了。 本関数を再実行してください。 |

Description

TSIP 機能を使用可能にします。

key_index_1 には R_TSIP_GenerateTlsRsaPublicKeyIndex()または R_TSIP_UpdateTlsRsaPublicKeyIndex()で生成した「TLS 連携 RSA 公開鍵の鍵生成情報」を入力してください。TLS 連携機能を使用しない場合は NULL ポインタを入力してください。

key_index_2 には R_TSIP_GenerateUpdateKeyRingKeyIndex()で生成した「鍵更新用鍵束鍵生成情報」を入力してください。鍵更新機能を使用しない場合は NULL ポインタを入力してください。

<状態遷移>

有効な実行前の状態は **TSIP 動作停止状態** です。

実行前の状態は **TSIP 動作停止状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.2 R_TSIP_Close

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Close(void)
```

Parameters

なし.

Return Values

TSIP_SUCCESS: 正常終了

Description

TSIP 機能を停止します。

<状態遷移>

有効な実行前の状態は 任意 です。

実行後は TSIP 動作停止状態 に遷移します。

Reentrant

非対応

4.3 R_TSIP_SoftwareReset

Format

```
#include "r_tsip_rx_if.h"
void R_TSIP_SoftwareReset(void)
```

Parameters

なし

Return Values

なし

Description

TSIP を初期状態に戻します。

実行前の状態は任意です。

実行後の状態遷移先は TSIP 動作停止状態です。

Reentrant

非対応

4.4 R_TSIP_GetVersion

Format

```
#include "r_tsip_rx_if.h"
uint32_t R_TSIP_GetVersion(void)
```

Parameters

なし

Return Values

| | |
|-----------|--------------------|
| 上位 2 バイト: | メジャーバージョン (10 進表示) |
| 下位 2 バイト: | マイナーバージョン (10 進表示) |

Description

TSIP ドライバのバージョン情報を取得することができます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後は **TSIP 使用可能状態**に遷移します。

Reentrant

非対応

4.5 R_TSIP_GenerateAes128KeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateAes128KeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|-------|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられたユーザ鍵 |
| key_index | 入力/出力 | ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

AES128bit のユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|----------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | AES128 鍵 | | | |
| 16-31 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

encrypted_provisioning_key, iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.6 R_TSIP_GenerateAes256KeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateAes256KeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|-------|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられたユーザ鍵 |
| key_index | 入力/出力 | ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

AES256bit のユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|----------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | AES256 鍵 | | | |
| 16-31 | | | | |
| 32-47 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

encrypted_provisioning_key, iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.7 R_TSIP_GenerateUpdateKeyRingKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateUpdateKeyRingKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_update_key_ring_t *key_index
)
```

Parameters

| | | |
|----------------------------|-------|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられたユーザ鍵 |
| key_index | 入力/出力 | 鍵更新用鍵束鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

鍵更新鍵束の鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|-------|---------|-------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | 鍵更新用鍵束 | | | |
| 16-31 | | | | |
| 32-47 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

encrypted_provisioning_key, iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.8 R_TSIP_UpdateAes128KeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateAes128KeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|-------|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられたユーザ鍵 |
| key_index | 入力/出力 | ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

AES128 鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|-------|----------|-------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | AES128 鍵 | | | |
| 16-31 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.9 R_TSIP_UpdateAes256KeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateAes256KeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|-------|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられたユーザ鍵 |
| key_index | 入力/出力 | ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

AES256 鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|-------|----------|-------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | AES256 鍵 | | | |
| 16-31 | | | | |
| 32-47 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.10 R_TSIP_GenerateAes128RandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateAes128RandomKeyIndex(
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|---------------------------|
| key_index | 入力/出力 | AES128 bit の AES ユーザ鍵生成情報 |
|-----------|-------|---------------------------|

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

AES128bit のユーザ鍵生成情報を出力するための API です。

本 API は TSIP 内部にて乱数値からユーザ鍵を生成します。従ってユーザ鍵の入力は不要です。API が出力するユーザ鍵生成情報を使用しデータを暗号化することにより、データのデッドコピーを防ぐことができます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.11 R_TSIP_GenerateAes256RandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateAes256RandomKeyIndex(
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|---------------------------|
| key_index | 入力/出力 | AES256 bit の AES ユーザ鍵生成情報 |
|-----------|-------|---------------------------|

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

AES256bit のユーザ鍵生成情報を出力するための API です。

本 API は TSIP 内部にて乱数値からユーザ鍵を生成します。従ってユーザ鍵の入力は不要です。本 API が出力するユーザ鍵生成情報を使用しデータを暗号化することにより、データのデッドコピーを防ぐことができます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.12 R_TSIP_GenerateRandomNumber

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRandomNumber(
    uint32_t *random
)
```

Parameters

| | | |
|--------|-------|-------------------|
| random | 入力/出力 | 4 ワード(16 バイト)の乱数値 |
|--------|-------|-------------------|

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

NIST SP800-90A に準拠した 4 ワードの乱数値を生成することができます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後は **TSIP 使用可能状態**に遷移します。

Reentrant

非対応

4.13 R_TSIP_StartUpdateFirmware

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_StartUpdateFirmware(void)
```

Parameters

なし

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

ファームウェアアップデート状態へ移行します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後は **ファームウェアアップデート状態**に遷移します。

Reentrant

非対応

4.14 R_TSIP_GenerateFirmwareMAC

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateFirmwareMAC(
    uint32_t *InData_KeyIndex,
    uint32_t *InData_SessionKey,
    uint32_t *InData_UpProgram,
    uint32_t *InData_IV,
    uint32_t *OutData_Program,
    uint32_t MAX_CNT,
    TSIP_GEN_MAC_CB_FUNC_T p_callback,
    tsip_firmware_generate_mac_resume_handle_t *tsip_firmware_generate_mac_resume_handle
)
```

Parameters

| | | |
|--|-------|--|
| InData_KeyIndex | 入力 | InData_SessionKey の復号、ファームウェアの MAC 値を生成するためのユーザ鍵生成情報領域 |
| InData_SessionKey | 入力 | 暗号化されたファームウェアの復号、チェックサム値検証するためのセッション鍵領域 |
| InData_UpProgram | 入力 | 暗号化されたファームウェアを一時的に格納するための領域(デモプロジェクトでは、512 ワード(2048 バイト)分確保) |
| InData_IV | 入力 | 暗号化されたファームウェアを復号するための初期化ベクタ領域 |
| OutData_Program | 入力/出力 | 復号されたファームウェアを一時的に格納するための領域(デモプロジェクトでは、512 ワード(2048 バイト)分確保) |
| MAX_CNT | 入力 | 暗号化されたファームウェアのワードサイズ+MAC サイズ ファームウェアのワードサイズは 4 の倍数である必要がある。MAC は 4 ワード (128bit) 固定のため、ファームウェアのワードサイズ+4 を入力。 暗号化されたファームウェアは 16 ワードが最小であるため、MAX_CNT の最小値は 20 |
| p_callback | 入力/出力 | ユーザ側で対応が必要な場合に、複数回呼ばれる。 対応内容は、列挙型 TSIP_FW_CB_REQ_TYPE で判別する。 |
| tsip_firmware_generate_mac_resume_handle | 入力/出力 | R_TSIP_GenerateFirmwareMAC 用ハンドラ(ワーク領域) |

Return Values

| | |
|------------------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_CALLBACK_UNREGIST: | p_callback の値が不正 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_RESUME_FIRMWARE_GENERATE_MAC: | 処理の続きがあります。 API の再呼び出しが必要。 |

Description

暗号化されたファームウェアとファームウェアチェックサム値に対し、ファームウェアの復号と新たな MAC 値生成を行います。ユーザは復号されたファームウェアと新たな MAC 値をフラッシュ ROM に書き込むことでファームウェアアップデートを行うことができます。

ファームウェアの暗号化アルゴリズムは AES-CBC, MAC は AES-CMAC を使用しています。

本 API のコールバック関数呼び出しフローは以下になります。

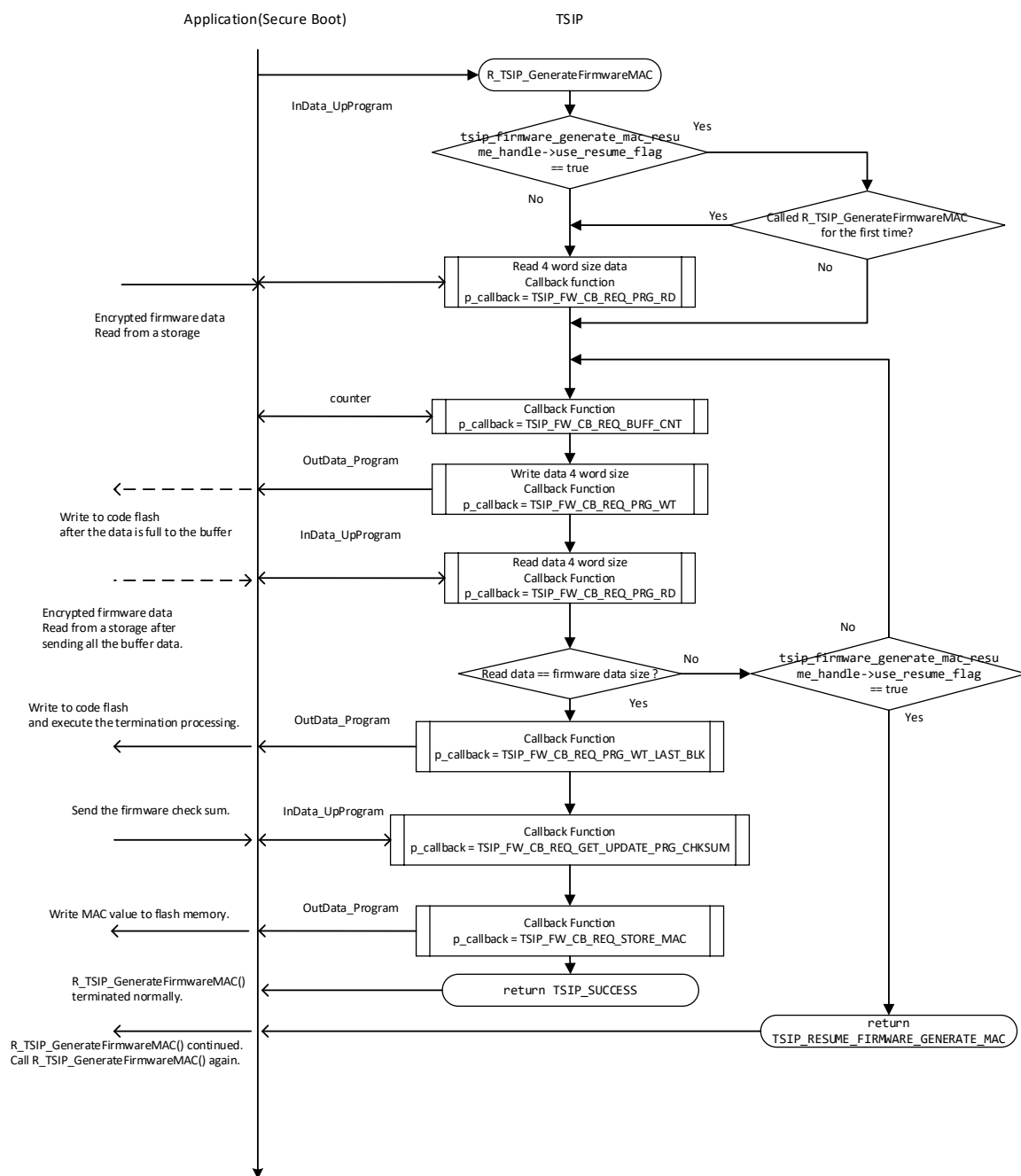


図 4-1 コールバック関数呼び出しフロー図

ファームウェアデータのリード、ライト処理を4ワード毎に行います。このため、第七引数 p_callback で登録されたコールバック関数を以下の順で呼び出します。()内はコールバック関数 p_callback 第一引数"req_type"の処理種別になります。

1. インクリメント調整(TSIP_FW_CB_REQ_BUFF_CNT)
2. 復号されたファームウェアを保存先へ書き込み(TSIP_FW_CB_REQ_PRG_WT)
3. 暗号化されたファームウェアの InData_UpProgram への格納(TSIP_FW_CB_REQ_PRG_RD)

コールバック関数内の処理は、毎回実施する必要はなく、確保した InData_Program /OutData_Program のサイズに応じて対応してください。

例えば、512 ワードのバッファを確保した場合は、 $512/4=128$ 回目にバッファ位置のインクリメント調整(TSIP_FW_CB_REQ_BUFF_CNT)、保存先への書き込み(TSIP_FW_CB_REQ_PRG_WT)、暗号化されたファームウェアを InData_UpProgram (TSIP_FW_CB_REQ_PRG_RD)に格納を実施します。

最後の保存先への書き込み要求は、TSIP_FW_CB_REQ_PRG_WT ではなく、req_type = TSIP_FW_CB_REQ_PRG_WT_LAST_BLK を指定します。

また、本 API は全ファームウェアの読み込み・書き込み完了後に、再度、コールバック関数 p_callback を呼び出します。ユーザはコールバック関数 p_callback の第一引数"req_type"が TSIP_FW_CB_REQ_GET_UPDATE_PRG_CHKSUM であることを確認後、チェックサム値を p_callback の第四引数"InData_UpProgram"に渡してください。また本 API はチェックサム値読み込み後、チェックサム値検証が正しければファームウェア MAC 値を生成します。その後、コールバック関数 p_callback の第一引数"req_type"が TSIP_FW_CB_REQ_STORE_MAC で第五引数"OutData_Program"で MAC 値をユーザに渡します。ユーザは MAC 値をフラッシュ領域に保存してください。

tsip_firmware_generate_mac_resume_handle.use_resume_flag=true に設定して呼び出した場合、ファームアップデート処理をすべて行わず、ファームアップデートの開始、更新関数として動作します。処理の続きがある場合、戻り値に TSIP_RESUME_FIRMWARE_GENERATE_MAC を返します。戻り値が TSIP_SUCCESS になるまで、R_TSIP_GenerateFirmwareMAC()を呼んでください。戻り値に TSIP_SUCCESS が返ったら、ファームアップデート処理は正常終了したことを示します。

有効な実行前の状態は ファームウェアアップデート状態です。
実行後は ファームウェアアップデート状態に遷移します。

Reentrant

非対応

4.15 R_TSIP_VerifyFirmwareMAC

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_VerifyFirmwareMAC(
    uint32_t *InData_Program,
    uint32_t MAX_CNT,
    uint32_t *InData_MAC
)
```

Parameters

| | | |
|----------------|----|--|
| InData_Program | 入力 | ファームウェア |
| MAX_CNT | 入力 | ファームウェアのワードサイズ+MAC サイズ 4 の倍数である必要がある。 MAC は 4 ワード (16byte) 固定のため、ファームウェアのワードサイズ+4 を入力。 ファームウェアは 16 ワード以上が最小であるため、MAX_CNT の最小値は 20 |
| InData_MAC | 入力 | 比較する MAC 値(16byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 不正な MAC 値 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |

Description

ファームウェアと MAC 値に対し、MAC 値の検証を行います。第三引数"InData_Mac"には R_TSIP_GenerateFirmwareMAC() で生成した MAC 値を渡してください。

MAC 検証アルゴリズムは AES-CMAC を使用しています。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

不正な MAC 値を検出すると **TSIP 不正アクセス検出状態** に遷移します。

Reentrant

非対応

4.16 R_TSIP_Aes128EcbEncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128EcbEncryptInit(
    tsip_aes_handle_t *handle,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128EcbEncryptInit() 関数は、AES 演算を実行する準備を行い、その結果を第一引数"handle"に書き出します。handle は、続く R_TSIP_Aes128EcbEncryptUpdate()関数および R_TSIP_Aes128EcbEncryptFinal()関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.17 R_TSIP_Aes128EcbEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128EcbEncryptUpdate(
    tsip_aes_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_length
)
```

Parameters

| | | |
|--------------|-------|---------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_length | 入力 | 平文データ長 (16 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128EcbEncryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"を handle に格納された key_index を用いて暗号化し、途中経過を第一引数"handle"に書き出します。また暗号化結果を第三引数"cipher"に書き出します。平文入力が完了した後は、R_TSIP_Aes128EcbEncryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.18 R_TSIP_Aes128EcbEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128EcbEncryptFinal(
    tsip_aes_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_length
)
```

Parameters

| | | |
|---------------|-------|-------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域(常に何も書き込まれません) |
| cipher_length | 入力/出力 | 暗号文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128EcbEncryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"に演算結果、第三引数"cipher_length"に演算結果の長さを書き出します。第二引数は、本来は 16 バイトの倍数に満たない分の端数について暗号化した結果が書き出されますが、Update 関数には 16 バイトの倍数でしか入力できない制限があるため、cipher には常に何も書き込まれず、cipher_length には常に 0 が書き込まれます。cipher, cipher_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.19 R_TSIP_Aes128EcbDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128EcbDecryptInit(
    tsip_aes_handle_t *handle,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128EcbDecryptInit() 関数は、AES 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Aes128EcbDecryptUpdate()関数および R_TSIP_Aes128EcbDecryptFinal()関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.20 R_TSIP_Aes128EcbDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128EcbDecryptUpdate(
    tsip_aes_handle_t *handle,
    uint8_t * cipher,
    uint8_t *plain,
    uint32_t cipher_length
)
```

Parameters

| | | |
|---------------|-------|----------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| cipher | 入力 | 暗号文データ領域 |
| plain | 入力/出力 | 平文データ領域 |
| cipher_length | 入力 | 暗号文データ長 (16 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128EcbDecryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"を handle に格納された key_index を用いて復号し、途中経過を第一引数"handle"に書き出します。また復号結果を第三引数"plain"に書き出します。暗号文入力が完了した後は、R_TSIP_Aes128EcbDecryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.21 R_TSIP_Aes128EcbDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128EcbDecryptFinal(
    tsip_aes_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_length
)
```

Parameters

| | | |
|--------------|-------|------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域(常に何も書き込まれません) |
| plain_length | 入力/出力 | 平文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128EcbDecryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"に演算結果、第三引数"plain_length"に演算結果の長さを書き出します。第二引数は、本来は 16 バイトの倍数に満たない分の端数について復号した結果が書き出されますが、Update 関数には 16 バイトの倍数でしか入力できない制限があるため、plain には常に何も書き込まれず、plain_length には常に 0 が書き込まれます。plain, plain_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.22 R_TSIP_Aes256EcbEncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256EcbEncryptInit(
    tsip_aes_handle_t *handle,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256EcbEncryptInit() 関数は、AES 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Aes256EcbEncryptUpdate()関数および R_TSIP_Aes256EcbEncryptFinal()関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.23 R_TSIP_Aes256EcbEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256EcbEncryptUpdate(
    tsip_aes_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_length
)
```

Parameters

| | | |
|--------------|-------|---------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_length | 入力 | 平文データ長 (16 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256EcbEncryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"を handle に格納された key_index を用いて暗号化し、途中経過を第一引数"handle"に書き出します。また暗号化結果を第三引数"cipher"に書き出します。平文入力が完了した後は、R_TSIP_Aes256EcbEncryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.24 R_TSIP_Aes256EcbEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256EcbEncryptFinal(
    tsip_aes_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_length
)
```

Parameters

| | | |
|---------------|-------|-------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域(常に何も書き込まれません) |
| cipher_length | 入力/出力 | 暗号文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256EcbEncryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"に演算結果、第三引数"cipher_length"に演算結果の長さを書き出します。第二引数は、本来は 16 バイトの倍数に満たない分の端数について暗号化した結果が書き出されますが、Update 関数には 16 バイトの倍数でしか入力できない制限があるため、cipher には常に何も書き込まれず、cipher_length には常に 0 が書き込まれます。cipher, cipher_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.25 R_TSIP_Aes256EcbDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256EcbDecryptInit(
    tsip_aes_handle_t *handle,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256EcbDecryptInit() 関数は、AES 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Aes256EcbDecryptUpdate()関数および R_TSIP_Aes256EcbDecryptFinal()関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.26 R_TSIP_Aes256EcbDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256EcbDecryptUpdate(
    tsip_aes_handle_t *handle,
    uint8_t * cipher,
    uint8_t *plain,
    uint32_t cipher_length
)
```

Parameters

| | | |
|---------------|-------|----------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| cipher | 入力 | 暗号文データ領域 |
| plain | 入力/出力 | 平文データ領域 |
| cipher_length | 入力 | 暗号文データ長 (16 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256EcbDecryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"を handle に格納された key_index を用いて復号し、途中経過を第一引数"handle"に書き出します。また復号結果を第三引数"plain"に書き出します。暗号文入力が完了した後は、R_TSIP_Aes256EcbDecryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.27 R_TSIP_Aes256EcbDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256EcbDecryptFinal(
    tsip_aes_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_length
)
```

Parameters

| | | |
|--------------|-------|------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域(常に何も書き込まれません) |
| plain_length | 入力/出力 | 平文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256EcbDecryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"に演算結果、第三引数"plain_length"に演算結果の長さを書き出します。第二引数は、本来は 16 バイトの倍数に満たない分の端数について復号した結果が書き出されますが、Update 関数には 16 バイトの倍数でしか入力できない制限があるため、plain には常に何も書き込まれず、plain_length には常に 0 が書き込まれます。plain, plain_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.28 R_TSIP_Aes128CbcEncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CbcEncryptInit(
    tsip_aes_handle_t *handle,
    tsip_aes_key_index_t *key_index,
    uint8_t *ivec
)
```

Parameters

| | | |
|-----------|-------|------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| ivec | 入力 | 初期化ベクタ(16 バイト) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128CbcEncryptInit() 関数は、AES 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Aes128CbcEncryptUpdate()関数および R_TSIP_Aes128CbcEncryptFinal()関数で引数として使用されます。

TLS 連携機能で使用する場合、key_index には R_TSIP_TlsGenerateSessionKey()で生成された client_crypto_key_index もしくは server_crypto_key_index を入力してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.29 R_TSIP_Aes128CbcEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CbcEncryptUpdate(
    tsip_aes_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_length
)
```

Parameters

| | | |
|--------------|-------|---------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_length | 入力 | 平文データ長 (16 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128CbcEncryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"を handle に格納された key_index を用いて暗号化し、途中経過を第一引数"handle"に書き出します。また暗号化結果を第三引数"cipher"に書き出します。平文入力が完了した後は、R_TSIP_Aes128CbcEncryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.30 R_TSIP_Aes128CbcEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CbcEncryptFinal(
    tsip_aes_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_length
)
```

Parameters

| | | |
|---------------|-------|-------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域(常に何も書き込まれません) |
| cipher_length | 入力/出力 | 暗号文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128CbcEncryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"に演算結果、第三引数"cipher_length"に演算結果の長さを書き出します。第二引数は、本来は 16 バイトの倍数に満たない分の端数について暗号化した結果が書き出されますが、Update 関数には 16 バイトの倍数でしか入力できない制限があるため、cipher には常に何も書き込まれず、cipher_length には常に 0 が書き込まれます。cipher, cipher_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.31 R_TSIP_Aes128CbcDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CbcDecryptInit(
    tsip_aes_handle_t *handle,
    tsip_aes_key_index_t *key_index,
    uint8_t *ivec
)
```

Parameters

| | | |
|-----------|-------|------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| ivec | 入力 | 初期化ベクタ(16 バイト) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128CbcDecryptInit() 関数は、AES 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Aes128CbcDecryptUpdate()関数および R_TSIP_Aes128CbcDecryptFinal()関数で引数として使用されます。

TLS 連携機能で使用する場合、key_index には R_TSIP_TlsGenerateSessionKey()で生成された client_crypto_key_index もしくは server_crypto_key_index を入力してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.32 R_TSIP_Aes128CbcDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CbcDecryptUpdate(
    tsip_aes_handle_t *handle,
    uint8_t * cipher,
    uint8_t *plain,
    uint32_t cipher_length
)
```

Parameters

| | | |
|---------------|-------|----------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| cipher | 入力 | 暗号文データ領域 |
| plain | 入力/出力 | 平文データ領域 |
| cipher_length | 入力 | 暗号文データ長 (16 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128CbcDecryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"を handle に格納された key_index を用いて復号し、途中経過を第一引数"handle"に書き出します。また復号結果を第三引数"plain"に書き出します。暗号文入力が完了した後は、R_TSIP_Aes128CbcDecryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.33 R_TSIP_Aes128CbcDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CbcDecryptFinal(
    tsip_aes_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_length
)
```

Parameters

| | | |
|--------------|-------|------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域(常に何も書き込まれません) |
| plain_length | 入力/出力 | 平文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128CbcDecryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"に演算結果、第三引数"plain_length"に演算結果の長さを書き出します。第二引数は、本来は 16 バイトの倍数に満たない分の端数について復号した結果が書き出されますが、Update 関数には 16 バイトの倍数でしか入力できない制限があるため、plain には常に何も書き込まれず、plain_length には常に 0 が書き込まれます。plain, plain_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.34 R_TSIP_Aes256CbcEncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CbcEncryptInit(
    tsip_aes_handle_t *handle,
    tsip_aes_key_index_t *key_index,
    uint8_t *ivec
)
```

Parameters

| | | |
|-----------|-------|------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| ivec | 入力 | 初期化ベクタ(16 バイト) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256CbcEncryptInit() 関数は、AES 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Aes256CbcEncryptUpdate()関数および R_TSIP_Aes256CbcEncryptFinal()関数で引数として使用されます。

TLS 連携機能で使用する場合、key_index には R_TSIP_TlsGenerateSessionKey()で生成された client_crypto_key_index もしくは server_crypto_key_index を入力してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.35 R_TSIP_Aes256CbcEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CbcEncryptUpdate(
    tsip_aes_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_length
)
```

Parameters

| | | |
|--------------|-------|---------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_length | 入力 | 平文データ長 (16 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256CbcEncryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"を handle に格納された key_index を用いて暗号化し、途中経過を第一引数"handle"に書き出します。また暗号化結果を第三引数"cipher"に書き出します。平文入力が完了した後は、R_TSIP_Aes256CbcEncryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.36 R_TSIP_Aes256CbcEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CbcEncryptFinal(
    tsip_aes_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_length
)
```

Parameters

| | | |
|---------------|-------|-------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域(常に何も書き込まれません) |
| cipher_length | 入力/出力 | 暗号文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256CbcEncryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"に演算結果、第三引数"cipher_length"に演算結果の長さを書き出します。第二引数は、本来は 16 バイトの倍数に満たない分の端数について暗号化した結果が書き出されますが、Update 関数には 16 バイトの倍数でしか入力できない制限があるため、cipher には常に何も書き込まれず、cipher_length には常に 0 が書き込まれます。cipher, cipher_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.37 R_TSIP_Aes256CbcDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CbcDecryptInit(
    tsip_aes_handle_t *handle,
    tsip_aes_key_index_t *key_index
    uint8_t *ivec
)
```

Parameters

| | | |
|-----------|-------|------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| ivec | 入力 | 初期化ベクタ(16 バイト) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256CbcDecryptInit() 関数は、AES 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Aes256CbcDecryptUpdate()関数および R_TSIP_Aes256CbcDecryptFinal()関数で引数として使用されます。

TLS 連携機能で使用する場合、key_index には R_TSIP_TlsGenerateSessionKey()で生成された client_crypto_key_index もしくは server_crypto_key_index を入力してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.38 R_TSIP_Aes256CbcDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CbcDecryptUpdate(
    tsip_aes_handle_t *handle,
    uint8_t *cipher,
    uint8_t *plain,
    uint32_t cipher_length,
)
```

Parameters

| | | |
|---------------|-------|----------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| cipher | 入力 | 暗号文データ領域 |
| plain | 入力/出力 | 平文データ領域 |
| cipher_length | 入力 | 暗号文データ長 (16 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS : | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256CbcDecryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"を handle に格納された key_index を用いて復号し、途中経過を第一引数"handle"に書き出します。また復号結果を第三引数"plain"に書き出します。暗号文入力が完了した後は、R_TSIP_Aes256CbcDecryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.39 R_TSIP_Aes256CbcDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CbcDecryptFinal(
    tsip_aes_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_length
)
```

Parameters

| | | |
|--------------|-------|------------------------|
| handle | 入力/出力 | AES 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域(常に何も書き込まれません) |
| plain_length | 入力/出力 | 平文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256CbcDecryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"に演算結果、第三引数"plain_length"に演算結果の長さを書き出します。第二引数は、本来は 16 バイトの倍数に満たない分の端数について復号した結果が書き出されますが、Update 関数には 16 バイトの倍数でしか入力できない制限があるため、plain には常に何も書き込まれず、plain_length には常に 0 が書き込まれます。plain, plain_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.40 R_TSIP_Aes128GcmEncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128GcmEncryptInit(
    tsip_gcm_handle_t *handle,
    tsip_aes_key_index_t *key_index,
    uint8_t *ivec,
    uint32_t ivec_len
)
```

Parameters

| | | |
|-----------|-------|----------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| ivec | 入力 | 初期化ベクタ領域 (iv_len byte) 【注】 |
| ivec_len | 入力 | 初期化ベクタ長 (1~任意 byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128GcmEncryptInit()関数は、GCM 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Aes128GcmEncryptUpdate()関数および R_TSIP_Aes128GcmEncryptFinal()関数で引数として使用されます。また ivec は 4 の倍数の RAM アドレスを指定してください。

【注】

key_index->type が”TSIP_KEY_INDEX_TYPE_AES128_FOR_TLS”の場合

R_TSIP_TlsGenerateSessionKey ()関数で select_cipher:6, 7 を指定して生成した key_index は、96bit の IV を含んでいます。第三引数の ivec には NULL ポインタを入力してください。第四引数の ivec_len に 0 を指定してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.41 R_TSIP_Aes128GcmEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128GcmEncryptUpdate(
    tsip_gcm_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_data_len,
    uint8_t *aad,
    uint32_t aad_len
)
```

Parameters

| | | |
|----------------|-------|------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_data_len | 入力 | 平文データ長 (0～任意 byte) |
| aad | 入力 | 追加認証データ (aad_len byte) |
| aad_len | 入力 | 追加認証データ長 (0～任意 byte) |

Return Values

| | |
|-----------------------------|---|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | plain データの入力の後に、aad が入力された 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128GcmEncryptUpdate()関数は、第二引数"plain"で指定された平文から R_TSIP_Aes128GcmEncryptInit()で指定された"key_index"と"ivec"、第五引数で指定された"aad"を用いて GCM で暗号化します。本関数内部で、aad, plain の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。暗号化結果は"plain"入力データが 16byte 以上になってから、第三引数で指定された"cipher"に出力します。入力する"plain", "aad"データ長はそれぞれ第四引数の"plain_data_len", 第六引数の"aad_len"で指定します。ここでは、"aad", "plain"入力データの総バイト数ではなく、ユーザが本関数を呼ぶ際に入力するデータ長を指定してください。入力値の"plain"および"aad"は 16byte で割り切れない場合、パディング処理は関数内部で実施します。データの入れ方は"aad", "plain"の順で処理してください。"plain"データ入力開始後、"aad"データを入力するとエラーとなります。"aad"データと"plain"データが同時に本関数に入力された場合、"aad"データ処理後、"plain"データ入力状態に移行します。plain と cipher は領域が重ならないように配置してください。また plain と cipher と aad は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.42 R_TSIP_Aes128GcmEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128GcmEncryptFinal(
    tsip_gcm_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_data_len,
    uint8_t *atag
)
```

Parameters

| | | |
|-----------------|-------|--------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域 (data_len byte) |
| cipher_data_len | 入力/出力 | 暗号文データ長 (0～任意 byte) |
| atag | 入力/出力 | 認証タグ領域(16byte) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS : | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128GcmEncryptFinal () 関数は、R_TSIP_Aes128GcmEncryptUpdate()で入力した plain の総データ長に 16byte の端数データがある場合、第二引数で指定された”cipher”に端数分の暗号化したデータを出力します。このとき、16byte に満たない部分は 0 padding されています。認証タグは第四引数の”atag”に出力します。また cipher と atag は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.43 R_TSIP_Aes128GcmDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128GcmDecryptInit(
    tsip_gcm_handle_t *handle,
    tsip_aes_key_index_t *key_index,
    uint8_t *ivec,
    uint32_t ivec_len
)
```

Parameters

| | | |
|-----------|-------|----------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| ivec | 入力 | 初期化ベクタ領域 (iv_len byte) 【注】 |
| ivec_len | 入力 | 初期化ベクタ長 (1~任意 byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS : | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128GcmDecryptInit() 関数は、GCM 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Aes128GcmDecryptUpdate()関数および R_TSIP_Aes128GcmDecryptFinal()関数で引数として使用されます。また ivec は 4 の倍数の RAM アドレスを指定してください。

【注】

key_index->type が”TSIP_KEY_INDEX_TYPE_AES128_FOR_TLS”の場合

R_TSIP_TlsGenerateSessionKey ()関数で select_cipher:6, 7 を指定して生成した key_index は、96bit の IV を含んでいます。第三引数の ivec には NULL ポインタを入力してください。第四引数の ivec_len に 0 を指定してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.44 R_TSIP_Aes128GcmDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128GcmDecryptUpdate(
    tsip_gcm_handle_t *handle,
    uint8_t *cipher,
    uint8_t *plain,
    uint32_t cipher_data_len,
    uint8_t *aad,
    uint32_t aad_len
)
```

Parameters

| | | |
|-----------------|-------|------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| cipher | 入力 | 暗号文データ領域 |
| plain | 入力/出力 | 平文データ領域 |
| cipher_data_len | 入力 | 暗号文データ長 (0～任意 byte) |
| aad | 入力 | 追加認証データ (aad_len byte) |
| aad_len | 入力 | 追加認証データ長 (0～任意 byte) |

Return Values

| | |
|-----------------------------|---|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | plain データの入力の後に、aad が入力された 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128GcmDecryptUpdate() 関数は、第二引数"cipher"で指定された暗号文から R_TSIP_Aes128GcmDecryptInit()で指定された"key_index"と"ivec"、第五引数で指定された"aad"を用いて GCM で復号します。本関数内部で、aad, plain の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。復号結果は"cipher"入力データが 16byte 以上になってから、第三引数で指定された"plain"に出力します。入力する"cipher", "aad"データ長はそれぞれ第四引数の"cipher_data_len", 第六引数の"aad_len"で指定します。ここでは、"aad", "cipher"入力データの総バイト数ではなく、ユーザが本関数を呼ぶ際に入力するデータ長を指定してください。入力値の"cipher"および"aad"は 16byte で割り切れない場合、パディング処理は関数内部で実施します。データの inputs は"aad", "cipher"の順で処理してください。"cipher"データ入力開始後、"aad"データを inputs するとエラーとなります。"aad"データと"cipher"データが同時に本関数に入力された場合、"aad"データ処理後、"cipher"データ入力状態に移行します。plain と cipher は領域が重ならないように配置してください。また plain と cipher と aad は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.45 R_TSIP_Aes128GcmDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128GcmDecryptFinal(
    tsip_gcm_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_data_len,
    uint8_t *atag,
    uint32_t atag_len
)
```

Parameters

| | | |
|----------------|-------|--------------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域 (data_len byte) |
| plain_data_len | 入力/出力 | 平文データ長 (0~任意 byte) |
| atag | 入力/出力 | 認証タグ領域 (atag_len byte) |
| atag_len | 入力 | 認証タグ長 (4,8,12,13,14,15,16byte) |

Return Values

| | |
|-----------------------------|--------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_AUTHENTICATION: | 認証が失敗 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128GcmDecryptFinal()関数は、R_TSIP_Aes128GcmDecryptUpdate()で指定された 16byte に満たない端数の暗号文を GCM で復号し、GCM 復号機能を終了させます。復号データ、認証タグはそれぞれ第二引数で指定された"plain"および、第四引数の"atag"に出力します。復号された総データ長は第三引数の"plain_data_len"に出力します。認証に失敗した場合は、戻り値 TSIP_ERR_AUTHENTICATION が返ります。第四引数で指定する"atag"は 16byte 以下で入力してください。16byte に満たない場合は、本関数内で 0padding を実施します。また plain と atag は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.46 R_TSIP_Aes256GcmEncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256GcmEncryptInit(
    tsip_gcm_handle_t *handle,
    tsip_aes_key_index_t *key_index,
    uint8_t *ivec,
    uint32_t ivec_len
)
```

Parameters

| | | |
|-----------|-------|------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| ivec | 入力 | 初期化ベクタ領域 (iv_len byte) |
| ivec_len | 入力 | 初期化ベクタ長 (1~任意 byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS : | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256GcmEncryptInit()関数は、GCM 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handleは続く R_TSIP_Aes128GcmEncryptUpdate()関数および R_TSIP_Aes256GcmEncryptFinal()関数で引数として使用されます。また ivec は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.47 R_TSIP_Aes256GcmEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256GcmEncryptUpdate(
    tsip_gcm_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_data_len,
    uint8_t *aad,
    uint32_t aad_len
)
```

Parameters

| | | |
|----------------|-------|------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_data_len | 入力 | 平文データ長 (0～任意 byte) |
| aad | 入力 | 追加認証データ (aad_len byte) |
| aad_len | 入力 | 追加認証データ長 (0～任意 byte) |

Return Values

| | |
|-----------------------------|---|
| TSIP_SUCCESS : | 正常終了 |
| TSIP_ERR_PARAMETER: | plain データの入力の後に、aad が入力された 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256GcmEncryptUpdate()関数は、第二引数"plain"で指定された平文から R_TSIP_Aes256GcmEncryptInit()で指定された"key_index"と"ivec"、第五引数で指定された"aad"を用いて GCM で暗号化します。本関数内部で、aad, plain の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。暗号化結果は"plain"入力データが 16byte 以上になってから、第三引数で指定された"cipher"に出力します。入力する"plain", "aad"データ長はそれぞれ第四引数の"plain_data_len", 第六引数の"aad_len"で指定します。ここでは、"aad", "plain"入力データの総バイト数ではなく、ユーザが本関数を呼ぶ際に入力するデータ長を指定してください。入力値の"plain"および"aad"は 16byte で割り切れない場合、パディング処理は関数内部で実施します。データの入れ方は"aad", "plain"の順で処理してください。"plain"データ入力開始後、"aad"データを入力するとエラーとなります。"aad"データと"plain"データが同時に本関数に入力された場合、"aad"データ処理後、"plain"データ入力状態に移行します。plain と cipher は領域が重ならないように配置してください。また plain と cipher と aad は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.48 R_TSIP_Aes256GcmEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256GcmEncryptFinal(
    tsip_gcm_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_data_len,
    uint8_t *atag
)
```

Parameters

| | | |
|-----------------|-------|--------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域 (data_len byte) |
| cipher_data_len | 入力/出力 | 暗号文データ長 (0～任意 byte) |
| atag | 入力/出力 | 認証タグ領域 (16byte) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS : | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256GcmEncryptFinal () 関数は、R_TSIP_Aes256GcmEncryptUpdate()で入力した plain の総データ長が 16byte に満たない場合、第二引数で指定された"cipher"に端数分の暗号化したデータが出力されます。このとき、16byte に満たない部分は 0padding されています。認証タグは第四引数の"atag"に出力します。また cipher と atag は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.49 R_TSIP_Aes256GcmDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256GcmDecryptInit(
    tsip_gcm_handle_t *handle,
    tsip_aes_key_index_t *key_index,
    uint8_t *ivec,
    uint32_t ivec_len
)
```

Parameters

| | | |
|-----------|-------|------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| ivec | 入力 | 初期化ベクタ領域 (iv_len byte) |
| ivec_len | 入力 | 初期化ベクタ長 (1~任意 byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256GcmDecryptInit()関数は、GCM 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handleは続く R_TSIP_Aes256GcmDecryptUpdate()関数および R_TSIP_Aes256GcmDecryptFinal()関数で引数として使用されます。また ivec は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.50 R_TSIP_Aes256GcmDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256GcmDecryptUpdate(
    tsip_gcm_handle_t *handle,
    uint8_t *cipher,
    uint8_t *plain,
    uint32_t cipher_data_len,
    uint8_t *aad,
    uint32_t aad_len
)
```

Parameters

| | | |
|-----------------|-------|------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| cipher | 入力 | 暗号文データ領域 |
| plain | 入力/出力 | 平文データ領域 |
| cipher_data_len | 入力 | 暗号文データ長 (0～任意 byte) |
| aad | 入力 | 追加認証データ (aad_len byte) |
| aad_len | 入力 | 追加認証データ長 (0～任意 byte) |

Return Values

| | |
|-----------------------------|---|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | plain データの入力の後に、aad が入力された 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256GcmDecryptUpdate()関数は、第二引数"cipher"で指定された暗号文から R_TSIP_Aes256GcmDecryptInit()で指定された"key_index"と"ivec"、第五引数で指定された"aad"を用いて GCM で復号します。本関数内部で、aad, plain の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。復号結果は"cipher"入力データが 16byte 以上になってから、第三引数で指定された"plain"に出力します。入力する"cipher", "aad"データ長はそれぞれ第四引数の"cipher_data_len", 第六引数の"aad_len"で指定します。ここでは、"aad", "cipher"入力データの総バイト数ではなく、ユーザが本関数を呼ぶ際に入力するデータ長を指定してください。入力値の"cipher"および"aad"は 16byte で割り切れない場合、パディング処理は関数内部で実施します。データの inputs は"aad", "cipher"の順で処理してください。"cipher"データ入力開始後、"aad"データを inputs するとエラーとなります。"aad"データと"cipher"データが同時に本関数に入力された場合、"aad"データ処理後、"cipher"データ入力状態に移行します。plain と cipher は領域が重ならないように配置してください。また plain と cipher と aad は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.51 R_TSIP_Aes256GcmDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256GcmDecryptFinal(
    tsip_gcm_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_data_len,
    uint8_t *atag,
    uint32_t atag_len
)
```

Parameters

| | | |
|----------------|-------|--------------------------------|
| handle | 入力/出力 | AES-GCM 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域 (data_len byte) |
| plain_data_len | 入力/出力 | 平文データ長 (0~任意 byte) |
| atag | 入力/出力 | 認証タグ領域 (atag_len byte) |
| atag_len | 入力 | 認証タグ長 (4,8,12,13,14,15,16byte) |

Return Values

| | |
|-----------------------------|--------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_AUTHENTICATION: | 認証が失敗 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256GcmDecryptFinal()関数は、R_TSIP_Aes256GcmDecryptUpdate()で指定された 16byte に満たない端数の暗号文を GCM で復号し、GCM 復号機能を終了させます。復号データ、認証タグはそれぞれ第二引数で指定された"plain"および、第四引数の"atag"に出力します。復号された総データ長は第三引数の"plain_data_len"に出力します。認証に失敗した場合は、戻り値 TSIP_ERR_AUTHENTICATION が返ります。第四引数で指定する"atag"は 16byte 以下で入力してください。16byte に満たない場合は、本関数内で 0padding を実施します。また plain と atag は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.52 R_TSIP_Aes128CcmEncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CcmEncryptInit(
    tsip_ccm_handle_t *handle,
    tsip_aes_key_index_t *key_index,
    uint8_t *nonce,
    uint32_t nonce_len,
    uint8_t *adata,
    uint8_t a_len,
    uint32_t payload_len,
    uint32_t mac_len
)
```

Parameters

| | | |
|-------------|-------|-------------------------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| nonce | 入力 | ノンス |
| nonce_len | 入力 | ノンスデータ長(7~13 byte) |
| adata | 入力 | 追加認証データ |
| a_len | 入力 | 追加認証データ長(0~110byte) |
| payload_len | 入力 | ペイロード長(任意 byte) |
| mac_len | 入力 | MAC 長(4, 6, 8, 10, 12, 14, 16 byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128CcmEncryptInit()関数は、CCM 演算を実行する準備を行い、その結果を第一引数“handle“に書き出します。handle は、続く R_TSIP_Aes128CcmEncryptUpdate()関数および R_TSIP_Aes128CcmEncryptFinal()関数で引数として使用されます。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.53 R_TSIP_Aes128CcmEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CcmEncryptUpdate(
    tsip_ccm_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_length
)
```

Parameters

| | | |
|--------------|-------|----------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_length | 入力 | 平文データ長 |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |

Description

R_TSIP_Aes128CcmEncryptUpdate()関数は、第二引数“plain”で指定された平文から R_TSIP_Aes128CcmEncryptInit()で指定された“key_index”, “nonce”, “adata”を用いて CCM を用いて暗号化します。本関数内部で plain の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。暗号化結果は“cipher”出力します。入力する plain の総データ長は R_TSIP_Aes128CcmEncryptInit() の payload_len で指定してください。本関数の plain_length には、ユーザが本関数を呼ぶ際に入力するデータ長を指定してください。入力値の plain は 16byte で割り切れない場合、パディング処理は関数内部で実施します。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.54 R_TSIP_Aes128CcmEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CcmEncryptFinal(
    tsip_ccm_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_length,
    uint8_t *mac,
    uint32_t mac_length
)
```

Parameters

| | | |
|---------------|-------|-------------------------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域 |
| cipher_length | 入力/出力 | 暗号文データ長 |
| mac | 入力/出力 | MAC 領域 |
| mac_length | 入力 | MAC 長(4, 6, 8, 10, 12, 14, 16 byte) |

Return Values

| | |
|-----------------------------|--------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_FAIL | 内部エラーが発生 |

Description

R_TSIP_Aes128CcmEncryptFinal()関数は、R_TSIP_Aes128CcmEncryptUpdate()で入力した plain のデータ長に 16byte の端数データがある場合、第二引数で指定された“cipher”に端数分の暗号化したデータを出力します。MAC 値は第四引数の“mac”に出力します。第五引数の“mac_length”には、Aes128CcmEncryptInit()の引数“mac_len”と同じ値を指定してください。また cipher と mac は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.55 R_TSIP_Aes128CcmDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CcmDecryptInit(
    tsip_ccm_handle_t *handle,
    tsip_aes_key_index_t *key_index,
    uint8_t *nonce,
    uint32_t nonce_len,
    uint8_t *adata,
    uint8_t a_len,
    uint32_t payload_len,
    uint32_t mac_len
)
```

Parameters

| | | |
|-------------|-------|-------------------------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| nonce | 入力 | ノンス |
| nonce_len | 入力 | ノンスデータ長(7~13byte) |
| adata | 入力 | 追加認証データ |
| a_len | 入力 | 追加認証データ長(0~110byte) |
| payload_len | 入力 | ペイロード長(任意 byte) |
| mac_len | 入力 | MAC 長(4, 6, 8, 10, 12, 14, 16 byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128CcmDecryptInit()関数は、CCM 演算を実行する準備を行い、その結果を第一引数“handle“に書き出します。handle は、続く R_TSIP_Aes128CcmDecryptUpdate 関数および R_TSIP_Aes128CcmDecryptFinal()関数で引数として使用されます。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.56 R_TSIP_Aes128CcmDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CcmDecryptUpdate(
    tsip_ccm_handle_t *handle,
    uint8_t *cipher,
    uint8_t *plain,
    uint32_t cipher_length
)
```

Parameters

| | | |
|---------------|-------|----------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| cipher | 入力 | 平文データ領域 |
| plain | 入力/出力 | 暗号文データ領域 |
| cipher_length | 入力 | 暗号文データ長 |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |

Description

R_TSIP_Aes128CcmDecryptUpdate()関数は、第二引数“cipher”で指定された暗号文から R_TSIP_Aes128CcmDecryptInit()で指定された“key_index”, “nonce”, “adata”を用いて CCM を用いて復号します。本関数内部で cipher の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。暗号化結果は“cipher”入力データが 16byte 以上になってから、第三引数で指定された“plain”に出力します。入力する cipher の総データ長は R_TSIP_Aes128CcmDecryptInit() の payload_len で指定してください。本関数の cipher_length には、ユーザが本関数を呼ぶ際に入力するデータ長を指定してください。入力値の cipher は 16byte で割り切れない場合、パディング処理は関数内部で実施します。

cipher と plain は領域が重ならないように配置してください。また cipher と plain は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.57 R_TSIP_Aes128CcmDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CcmDecryptFinal(
    tsip_ccm_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_length,
    uint8_t *mac,
    uint32_t mac_length
)
```

Parameters

| | | |
|--------------|-------|-------------------------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域 |
| plain_length | 入力/出力 | 平文データ長 |
| mac | 入力 | MAC 領域 |
| mac_length | 入力 | MAC 長(4, 6, 8, 10, 12, 14, 16 byte) |

Return Values

| | |
|-----------------------------|--------------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_FAIL | 内部エラーが発生、もしくは認証が失敗 |

Description

R_TSIP_Aes128CcmDecryptFinal()関数は、R_TSIP_Aes128CcmDecryptUpdate()で入力した cipher のデータ長に 16byte の端数データがある場合、第二引数で指定された“cipher”に端数分の復号したデータを出力します。また、第四引数の“mac”を検証します。第五引数の“mac_length”には、Aes128CcmDecryptInit()の引数“mac_len”と同じ値を指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.58 R_TSIP_Aes256CcmEncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CcmEncryptInit(
    tsip_ccm_handle_t *handle,
    tsip_aes_key_index_t *key_index,
    uint8_t *nonce,
    uint32_t nonce_len,
    uint8_t *adata,
    uint8_t a_len,
    uint32_t payload_len,
    uint32_t mac_len
)
```

Parameters

| | | |
|-------------|-------|-------------------------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| nonce | 入力 | ノンス |
| nonce_len | 入力 | ノンスデータ長(7~13 byte) |
| adata | 入力 | 追加認証データ |
| a_len | 入力 | 追加認証データ長(0~110byte) |
| payload_len | 入力 | ペイロード長(任意 byte) |
| mac_len | 入力 | MAC 長(4, 6, 8, 10, 12, 14, 16 byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256CcmEncryptInit()関数は、CCM 演算を実行する準備を行い、その結果を第一引数“handle”に書き出します。handle は、続く R_TSIP_Aes256CcmEncryptUpdate()関数および R_TSIP_Aes256CcmEncryptFinal()関数で引数として使用されます。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.59 R_TSIP_Aes256CcmEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CcmEncryptUpdate(
    tsip_ccm_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_length
)
```

Parameters

| | | |
|--------------|-------|----------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_length | 入力 | 平文データ長 |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |

Description

R_TSIP_Aes256CcmEncryptUpdate()関数は、第二引数“plain”で指定された平文から R_TSIP_Aes256CcmEncryptInit()で指定された“key_index”, “nonce”, “adata”を用いて CCM を用いて暗号化します。本関数内部で plain の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。暗号化結果は“cipher”に入力データが 16byte 以上になってから、第三引数で指定された“cipher”に出力します。入力する plain の総データ長は R_TSIP_Aes256CcmEncryptInit() の payload_len で指定してください。本関数の plain_length には、ユーザが本関数を呼ぶ際に入力するデータ長を指定してください。入力値の plain は 16byte で割り切れない場合、パディング処理は関数内部で実施します。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.60 R_TSIP_Aes256CcmEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CcmEncryptFinal(
    tsip_ccm_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_length,
    uint8_t *mac,
    uint32_t mac_length
)
```

Parameters

| | | |
|---------------|-------|-------------------------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域 |
| cipher_length | 入力/出力 | 暗号文データ長 |
| mac | 入力/出力 | MAC 領域 |
| mac_length | 入力 | MAC 長(4, 6, 8, 10, 12, 14, 16 byte) |

Return Values

| | |
|-----------------------------|--------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_FAIL | 内部エラーが発生 |

Description

R_TSIP_Aes256CcmEncryptFinal()関数は、R_TSIP_Aes256CcmEncryptUpdate()で入力した plain のデータ長に 16byte の端数データがある場合、第二引数で指定された“cipher”に端数分の暗号化したデータを出力します。MAC 値は第四引数の“mac”に出力します。第五引数の“mac_length”には、Aes256CcmEncryptInit()の引数“mac_len”と同じ値を指定してください。また cipher と mac は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.61 R_TSIP_Aes256CcmDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CcmDecryptInit(
    tsip_ccm_handle_t *handle,
    tsip_aes_key_index_t *key_index,
    uint8_t *nonce,
    uint32_t nonce_len,
    uint8_t *adata,
    uint8_t a_len,
    uint32_t payload_len,
    uint32_t mac_len
)
```

Parameters

| | | |
|-------------|-------|-------------------------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |
| nonce | 入力 | ノンス |
| nonce_len | 入力 | ノンスデータ長(7~13byte) |
| adata | 入力 | 追加認証データ |
| a_len | 入力 | 追加認証データ長(0~110byte) |
| payload_len | 入力 | ペイロード長(任意 byte) |
| mac_len | 入力 | MAC 長(4, 6, 8, 10, 12, 14, 16 byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256CcmDecryptInit()関数は、CCM 演算を実行する準備を行い、その結果を第一引数“handle”に書き出します。handle は、続く R_TSIP_Aes256CcmDecryptUpdate 関数および R_TSIP_Aes256CcmDecryptFinal()関数で引数として使用されます。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.62 R_TSIP_Aes256CcmDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CcmDecryptUpdate(
    tsip_ccm_handle_t *handle,
    uint8_t *cipher,
    uint8_t *plain,
    uint32_t cipher_length
)
```

Parameters

| | | |
|---------------|-------|----------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| cipher | 入力 | 平文データ領域 |
| plain | 入力/出力 | 暗号文データ領域 |
| cipher_length | 入力 | 暗号文データ長 |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |

Description

R_TSIP_Aes256CcmDecryptUpdate()関数は、第二引数“cipher”で指定された暗号文から R_TSIP_Aes256CcmDecryptInit()で指定された“key_index”, “nonce”, “adata”を用いて CCM を用いて復号します。本関数内部で cipher の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。暗号化結果は“cipher”入力データが 16byte 以上になってから、第三引数で指定された“plain”に出力します。入力する cipher の総データ長は R_TSIP_Aes256CcmDecryptInit() の payload_len で指定してください。本関数の cipher_length には、ユーザが本関数を呼ぶ際に入力するデータ長を指定してください。入力値の cipher は 16byte で割り切れない場合、パディング処理は関数内部で実施します。

cipher と plain は領域が重ならないように配置してください。また cipher と plain は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.63 R_TSIP_Aes256CcmDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CcmDecryptFinal(
    tsip_ccm_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_length,
    uint8_t *mac,
    uint32_t mac_length
)
```

Parameters

| | | |
|--------------|-------|-------------------------------------|
| handle | 入力/出力 | AES-CCM 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域 |
| plain_length | 入力/出力 | 平文データ長 |
| mac | 入力 | MAC 領域 |
| mac_length | 入力 | MAC 長(4, 6, 8, 10, 12, 14, 16 byte) |

Return Values

| | |
|-----------------------------|--------------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_FAIL | 内部エラーが発生、もしくは認証が失敗 |

Description

R_TSIP_Aes256CcmDecryptFinal()関数は、R_TSIP_Aes256CcmDecryptUpdate()で入力した cipher のデータ長に 16byte の端数データがある場合、第二引数で指定された“cipher”に端数分の復号したデータを出力します。また、第四引数の“mac”を検証します。第五引数の“mac_length”には、Aes256CcmDecryptInit()の引数“mac_len”と同じ値を指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.64 R_TSIP_Aes128CmacGenerateInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CmacGenerateInit(
    tsip_cmac_handle_t *handle,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-----------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128CmacGenerateInit() 関数は、CMAC 演算を実行する準備を行い、その結果を第一引数 "handle" に書き出します。handle は続く R_TSIP_Aes128CmacGenerateUpdate()関数や、R_TSIP_Aes128CmacGenerateFinal()関数の引数で使します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.65 R_TSIP_Aes128CmacGenerateUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CmacGenerateUpdate(
    tsip_cmac_handle_t *handle,
    uint8_t *message,
    uint32_t message_length
)
```

Parameters

| | | |
|----------------|-------|----------------------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| message | 入力 | メッセージデータ領域 (message_length byte) |
| message_length | 入力 | メッセージデータ長 (0~任意 byte) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128CmacGenerateUpdate 関数は、第二引数"message"で指定された message から R_TSIP_Aes128CmacGenerateInit()で指定された"key_index"を用いて MAC 値を生成します。本関数内部で、"message"の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。入力する"message"データ長は第三引数の"message_len"で指定します。ここでは、"message"入力データの総バイト数ではなく、ユーザが本関数を呼ぶ際に入力するメッセージのデータ長を入力してください。入力値の"message"は 16byte で割り切れない場合、パディング処理は関数内部で実施します。また"message"は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.66 R_TSIP_Aes128CmacGenerateFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CmacGenerateFinal(
    tsip_cmac_handle_t *handle,
    uint8_t *mac
)
```

Parameters

| | | |
|--------|-------|-----------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| mac | 入力/出力 | MAC データ領域(16byte) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128CmacGenerateFinal() 関数は、第二引数で指定された"mac"に Mac 値を出力し、CMAC の動作を終了させます。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.67 R_TSIP_Aes256CmacGenerateInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CmacGenerateInit(
    tsip_cmac_handle_t *handle,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-----------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256CmacGenerateInit() 関数は、CMAC 演算を実行する準備を行い、その結果を第一引数 "handle" に書き出します。handle は続く R_TSIP_Aes256CmacGenerateUpdate()関数や、R_TSIP_Aes256CmacGenerateFinal()関数の引数で使⽤します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.68 R_TSIP_Aes256CmacGenerateUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CmacGenerateUpdate(
    tsip_cmac_handle_t *handle,
    uint8_t *message,
    uint32_t message_length
)
```

Parameters

| | | |
|----------------|-------|----------------------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| message | 入力 | メッセージデータ領域 (message_length byte) |
| message_length | 入力 | メッセージデータ長 (0~任意 byte) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256CmacGenerateUpdate 関数は、第二引数"message"で指定された message から R_TSIP_Aes256CmacGenerateInit()で指定された"key_index"を用いて MAC 値を生成します。本関数内部で、"message"の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。入力する"message"データ長はそれぞれ第三引数の"message_len"で指定します。ここでは、"message"入力データの総バイト数ではなく、ユーザが本関数を呼ぶ際に入力するメッセージのデータ長を入力してください。入力値の"message"は 16byte で割り切れない場合、パディング処理は関数内部で実施します。また"message"は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.69 R_TSIP_Aes256CmacGenerateFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CmacGenerateFinal(
    tsip_cmac_handle_t *handle,
    uint8_t *mac
)
```

Parameters

| | | |
|--------|-------|-----------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| mac | 入力/出力 | MAC データ領域(16byte) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256CmacGenerateFinal() 関数は、第二引数で指定された"mac"に Mac 値を出力し、CMAC の動作を終了させます。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.70 R_TSIP_Aes128CmacVerifyInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CmacVerifyInit(
    tsip_cmac_handle_t *handle,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-----------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128CmacVerifyInit() 関数は、CMAC 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handleは続く R_TSIP_Aes128CmacVerifyUpdate()関数や、R_TSIP_Aes128CmacVerifyFinal()関数の引数で使します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.71 R_TSIP_Aes128CmacVerifyUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CmacVerifyUpdate(
    tsip_cmac_handle_t *handle,
    uint8_t *message,
    uint32_t message_length
)
```

Parameters

| | | |
|----------------|-------|----------------------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| message | 入力 | メッセージデータ領域 (message_length byte) |
| message_length | 入力 | メッセージデータ長 (0~任意 byte) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128CmacVerifyUpdate 関数は、第二引数"message"で指定された message から R_TSIP_Aes128CmacVerifyInit()で指定された"key_index"を用いて MAC 値を生成します。本関数内部で、"message"の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。入力する"message"データ長はそれぞれ第三引数の"message_len"で指定します。ここでは、"message"入力データの総バイト数ではなく、ユーザが本関数を呼ぶ際に入力するメッセージのデータ長を入力してください。入力値の"message"は 16byte で割り切れない場合、パディング処理は関数内部で実施します。また"message"は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.72 R_TSIP_Aes128CmacVerifyFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128CmacVerifyFinal(
    tsip_cmac_handle_t *handle,
    uint8_t *mac,
    uint32_t mac_length
)
```

Parameters

| | | |
|------------|-------|-----------------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| mac | 入力 | MAC データ領域 (mac_length byte) |
| mac_length | 入力 | MAC データ長(2~16byte) |

Return Values

| | |
|-----------------------------|--------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_AUTHENTICATION: | 認証が失敗 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes128CmacVerifyFinal()関数は、第二引数で指定された"mac"に Mac 値を入力し、Mac 値を検証します。認証が失敗した場合は、戻り値 TSIP_ERR_AUTHENTICATION が返ります。Mac 値が 16byte 以下の場合は、本関数内で 0padding をします。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.73 R_TSIP_Aes256CmacVerifyInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CmacVerifyInit(
    tsip_cmac_handle_t *handle,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-----------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| key_index | 入力 | ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256CmacVerifyInit() 関数は、CMAC 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handleは続く R_TSIP_Aes256CmacVerifyUpdate()関数や、R_TSIP_Aes256CmacVerifyFinal()関数の引数で使します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

4.74 R_TSIP_Aes256CmacVerifyUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CmacVerifyUpdate(
    tsip_cmac_handle_t *handle,
    uint8_t *message,
    uint32_t message_length
)
```

Parameters

| | | |
|----------------|-------|----------------------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| message | 入力 | メッセージデータ領域 (message_length byte) |
| message_length | 入力 | メッセージデータ長 (0~任意 byte) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256CmacVerifyUpdate 関数は、第二引数"message"で指定された message から R_TSIP_Aes256CmacVerifyInit()で指定された"key_index"を用いて MAC 値を生成します。本関数内部で、"message"の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。入力する"message"データ長はそれぞれ第三引数の"message_len"で指定します。ここでは、"message"入力データの総バイト数ではなく、ユーザが本関数を呼ぶ際に入力するメッセージのデータ長を入力してください。入力値の"message"は 16byte で割り切れない場合、パディング処理は関数内部で実施します。また"message"は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

4.75 R_TSIP_Aes256CmacVerifyFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256CmacVerifyFinal(
    tsip_cmac_handle_t *handle,
    uint8_t *mac,
    uint32_t mac_length
)
```

Parameters

| | | |
|------------|-------|-----------------------------|
| handle | 入力/出力 | AES-CMAC 用ハンドラ(ワーク領域) |
| mac | 入力 | MAC データ領域 (mac_length byte) |
| mac_length | 入力 | MAC データ長(2~16byte) |

Return Values

| | |
|-----------------------------|--------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_AUTHENTICATION: | 認証が失敗 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Aes256CmacVerifyFinal()関数は、第二引数で指定された"mac"に Mac 値を入力し、Mac 値を検証します。認証が失敗した場合は、戻り値 TSIP_ERR_AUTHENTICATION が返ります。Mac 値が 16byte 以下の場合は、本関数内で 0padding をします。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

4.76 R_TSIP_Aes128KeyWrap

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128KeyWrap (
    tsip_aes_key_index_t *wrap_key_index,
    uint32_t target_key_type,
    tsip_aes_key_index_t *target_key_index,
    uint32_t *wrapped_key
)
```

Parameters

| | | |
|------------------|----|---|
| wrap_key_index | 入力 | ラップに使用する AES-128 鍵インデックス |
| target_key_type | 入力 | ラップする対象の鍵の選択 0(R_TSIP_KEYWRAP_AES128): AES-128 2(R_TSIP_KEYWRAP_AES256): AES-256 他は Reserved |
| target_key_index | 入力 | ラップする対象の鍵インデックス target_key_type 0 : 13 word size target_key_type 2 : 17 word size |
| wrapped_key | 出力 | ラップされた鍵 target_key_type 0 : 6 word size target_key_type 2 : 10 word size |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128KeyWrap()関数は、第三引数に入力した target_key_index を第一引数の wrap_key_index を使いラップします。ラップされた鍵は第四引数の wrapped_key に書き出します。ラップのアルゴリズム RFC3394 に準拠します。ラップする対象の鍵は、第二引数の target_key_type で選択してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7 章 鍵データの運用](#)」を参照してください

Reentrant

非対応

4.77 R_TSIP_Aes256KeyWrap

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256KeyWrap (
    tsip_aes_key_index_t *wrap_key_index,
    uint32_t target_key_type,
    tsip_aes_key_index_t *target_key_index,
    uint32_t *wrapped_key
)
```

Parameters

| | | |
|------------------|----|---|
| wrap_key_index | 入力 | ラップに使用する AES-256 鍵インデックス |
| target_key_type | 入力 | ラップする対象の鍵の選択 0(R_TSIP_KEYWRAP_AES128): AES-128 2(R_TSIP_KEYWRAP_AES256): AES-256 他は Reserved |
| target_key_index | 入力 | ラップする対象の鍵インデックス target_key_type 0 : 13 word size target_key_type 2 : 17 word size |
| wrapped_key | 出力 | ラップされた鍵 target_key_type 0 : 6 word size target_key_type 2 : 10 word size |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256KeyWrap()関数は、第三引数に入力した target_key_index を第一引数の wrap_key_index を使いラップします。ラップされた鍵は第四引数の wrapped_key に書き出します。ラップのアルゴリズム RFC3394 に準拠します。ラップする対象の鍵は、第二引数の target_key_type で選択してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7 章 鍵データの運用](#)」を参照してください

Reentrant

非対応

4.78 R_TSIP_Aes128KeyUnwrap

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes128KeyUnwrap (
    tsip_aes_key_index_t *wrap_key_index,
    uint32_t target_key_type,
    uint32_t *wrapped_key,
    tsip_aes_key_index_t *target_key_index
)
```

Parameters

| | | |
|------------------|----|---|
| wrap_key_index | 入力 | アンラップに使用する AES-128 鍵インデックス |
| target_key_type | 入力 | アンラップする対象の鍵の選択 0(R_TSIP_KEYWRAP_AES128): AES-128 2(R_TSIP_KEYWRAP_AES256): AES-256 他は Reserved |
| wrapped_key | 入力 | ラップされた鍵 target_key_type 0 : 6 word size target_key_type 2 : 10 word size |
| target_key_index | 出力 | 鍵インデックス target_key_type 0 : 13 word size target_key_type 2 : 17 word size |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes128KeyUnwrap 関数は、第三引数に入力した wrapped_key を第一引数の wrap_key_index を使いアンラップします。アンラップされた鍵は第四引数の target_key_index に書き出します。アンラップのアルゴリズム RFC3394 に準拠します。アンラップする対象の鍵は、第二引数の target_key_type で選択してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7 章 鍵データの運用](#)」を参照してください

Reentrant

非対応

4.79 R_TSIP_Aes256KeyUnwrap

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Aes256KeyUnwrap (
    tsip_aes_key_index_t *wrap_key_index,
    uint32_t target_key_type,
    uint32_t *wrapped_key,
    tsip_aes_key_index_t *target_key_index
)
```

Parameters

| | | |
|------------------|----|---|
| wrap_key_index | 入力 | アンラップに使用する AES-256 鍵インデックス |
| target_key_type | 入力 | アンラップする対象の鍵の選択 0(R_TSIP_KEYWRAP_AES128): AES-128 2(R_TSIP_KEYWRAP_AES256): AES-256 他は Reserved |
| wrapped_key | 入力 | ラップされた鍵 target_key_type 0 : 6 word size target_key_type 2 : 10 word size |
| target_key_index | 出力 | 鍵インデックス target_key_type 0 : 13 word size target_key_type 2 : 17 word size |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

R_TSIP_Aes256KeyUnwrap 関数は、第三引数に入力した wrapped_key を第一引数の wrap_key_index を使いアンラップします。アンラップされた鍵は第四引数の target_key_index に書き出します。アンラップのアルゴリズム RFC3394 に準拠します。アンラップする対象の鍵は、第二引数の target_key_type で選択してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7 章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5. API 関数詳細説明(TSIP 用)

5.1 R_TSIP_Sha1Init

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Sha1Init(
    tsip_sha_md5_handle_t *handle
)
```

Parameters

| | | |
|--------|-------|------------------|
| handle | 入力/出力 | SHA 用ハンドラ(ワーク領域) |
|--------|-------|------------------|

Return Values

| | |
|---------------|------|
| TSIP_SUCCESS: | 正常終了 |
|---------------|------|

Description

R_TSIP_Sha1Init() 関数は、SHA1 ハッシュ演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。”handle”は、続く R_TSIP_Sha1Update() 関数および R_TSIP_Sha1Final() 関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.2 R_TSIP_Sha1Update

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Sha1Update(
    tsip_sha_md5_handle_t *handle,
    uint8_t *message,
    uint32_t message_length
)
```

Parameters

| | | |
|----------------|-------|------------------|
| handle | 入力/出力 | SHA 用ハンドラ(ワーク領域) |
| message | 入力 | メッセージデータ領域 |
| message_length | 入力 | メッセージバイトデータ長 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha1Update()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"message"と第三引数の"message_length"からハッシュ値を演算し、途中経過を第一引数"handle"に書き出します(R_TSIP_GetCurrentHashDigestValue()関数で取得可能)。メッセージ入力が完了した後は、R_TSIP_Sha1Final()を呼び出してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.3 R_TSIP_Sha1Final

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Sha1Final(
    tsip_sha_md5_handle_t *handle,
    uint8_t *digest,
    uint32_t *digest_length
)
```

Parameters

| | | |
|---------------|-------|-------------------|
| handle | 入力/出力 | SHA 用ハンドラ(ワーク領域) |
| digest | 入力/出力 | hash データ領域 |
| digest_length | 入力/出力 | hash データ長(20byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha1Final() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の" digest"に演算結果、第三引数" digest_length"に演算結果の長さを書き出します。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.4 R_TSIP_Sha256Init

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Sha256Init(
    tsip_sha_md5_handle_t *handle
)
```

Parameters

| | | |
|--------|-------|------------------|
| handle | 入力/出力 | SHA 用ハンドラ(ワーク領域) |
|--------|-------|------------------|

Return Values

| | |
|---------------|------|
| TSIP_SUCCESS: | 正常終了 |
|---------------|------|

Description

R_TSIP_Sha256Init() 関数は、SHA-256 ハッシュ演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Sha256Update() 関数および R_TSIP_Sha256Final() 関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.5 R_TSIP_Sha256Update

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Sha256Update(
    tsip_sha_md5_handle_t *handle,
    uint8_t *message,
    uint32_t message_length
)
```

Parameters

| | | |
|----------------|-------|------------------|
| handle | 入力/出力 | SHA 用ハンドラ(ワーク領域) |
| message | 入力 | メッセージデータ領域 |
| message_length | 入力 | メッセージバイトデータ長 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha256Update()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"message"と第三引数の"message_length"からハッシュ値を演算し、途中経過を第一引数"handle"に書き出します(R_TSIP_GetCurrentHashDigestValue()関数で取得可能)。メッセージ入力が完了した後は、R_TSIP_Sha256Final()を呼び出してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.6 R_TSIP_Sha256Final

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Sha256Final(
    tsip_sha_md5_handle_t *handle,
    uint8_t *digest,
    uint32_t *digest_length
)
```

Parameters

| | | |
|---------------|-------|-------------------|
| handle | 入力/出力 | SHA 用ハンドラ(ワーク領域) |
| digest | 入力/出力 | hash データ領域 |
| digest_length | 入力/出力 | hash データ長(32byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha256Final()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"digest"に演算結果、第三引数"digest_length"に演算結果の長さを書き出します。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.7 R_TSIP_Md5Init

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Md5Init(
    tsip_sha_md5_handle_t *handle
)
```

Parameters

| | | |
|--------|-------|------------------|
| handle | 入力/出力 | MD5 用ハンドラ(ワーク領域) |
|--------|-------|------------------|

Return Values

| | |
|---------------|------|
| TSIP_SUCCESS: | 正常終了 |
|---------------|------|

Description

R_TSIP_Md5Init() 関数は、MD5 ハッシュ演算を実行する準備を行い、その結果を第一引数” handle” に書き出します。handle は、続く R_TSIP_Md5Update() 関数および R_TSIP_Md5Final() 関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.8 R_TSIP_Md5Update

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Md5Update(
    tsip_sha_md5_handle_t *handle,
    uint8_t *message,
    uint32_t message_length
)
```

Parameters

| | | |
|----------------|-------|------------------|
| handle | 入力/出力 | MD5 用ハンドラ(ワーク領域) |
| message | 入力 | メッセージデータ領域 |
| message_length | 入力 | メッセージバイトデータ長 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Md5Update()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"message"と第三引数の"message_length"からハッシュ値を演算し、途中経過を第一引数"handle"に書き出します(R_TSIP_GetCurrentHashDigestValue()関数で取得可能)。メッセージ入力が完了した後は、R_TSIP_Md5Final()を呼び出してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.9 R_TSIP_Md5Final

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Md5Final(
    tsip_sha_md5_handle_t *handle,
    uint8_t *digest,
    uint32_t *digest_length
)
```

Parameters

| | | |
|---------------|-------|-------------------|
| handle | 入力/出力 | MD5 用ハンドラ(ワーク領域) |
| digest | 入力/出力 | hash データ領域 |
| digest_length | 入力/出力 | hash データ長(16byte) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Md5Final()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"digest"に演算結果、第三引数"digest_length"に演算結果の長さを書き出します。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.10 R_TSIP_GetCurrentHashDigestValue

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GetCurrentHashDigestValue(
    tsip_sha_md5_handle_t *handle,
    uint8_t *digest,
    uint32_t *digest_length,
    uint32_t hash_type
)
```

Parameters

| | | |
|---------------|----|---|
| handle | 入力 | SHA,MD5 用ハンドラ(ワーク領域) |
| digest | 出力 | ハッシュ値演算途中経過データ領域 |
| digest_length | 出力 | ハッシュ値演算途中経過データ長(16, 20, 32 byte) |
| hash_type | 入力 | ハッシュの種類 : R_TSIP_HASH_MD5, R_TSIP_HASH_SHA1 または R_TSIP_HASH_SHA256 |

Return Values

| | |
|-----------------------------|--------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

本関数は、引数"handle"で指定されたハンドルを使用し、引数"digest"に各 Update()関数(注)実行後のハッシュ値演算途中経過データ、引数"digest_length"にデータ長を出力します。

【注】 R_TSIP_Sha1Update()、R_TSIP_Sha256Update()、または R_TSIP_Md5Update()関数

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

5.11 R_TSIP_GenerateTdesKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateTdesKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_tdes_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|-------|----------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられた Triple-DES ユーザ鍵 |
| key_index | 入力/出力 | Triple-DES ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

Triple-DES のユーザ鍵生成情報を出力するための API です。

encrypted_key には以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|---------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | 暗号化された Triple-DES 鍵 | | | |
| 16-31 | | | | |
| 32-47 | MAC | | | |

DES もしくは 2TDES(2key-TDES)として使用する場合は、

「[7 章 鍵データの運用](#)」を参照してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後は **TSIP 使用可能状態**に遷移します。

encrypted_key, iv および encrypted_provisioning_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.12 R_TSIP_GenerateTdesRandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateTdesRandomKeyIndex(
    tsip_tdes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|---------------------|
| key_index | 入力/出力 | Triple-DES ユーザ鍵生成情報 |
|-----------|-------|---------------------|

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

Triple-DES のユーザ鍵生成情報を出力するための API です。

本 API は TSIP 内部にて乱数値からユーザ鍵を生成します。従ってユーザ鍵の入力は不要です。本 API が出力するユーザ鍵生成情報を使用しデータを暗号化することにより、データのデッドコピーを防ぐことができます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.13 R_TSIP_UpdateTdesKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateTdesKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_tdes_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|-------|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられたユーザ鍵 |
| key_index | 入力/出力 | Triple-DES ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

Triple-DES 鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|-------|--------------|-------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | Triple-DES 鍵 | | | |
| 16-31 | | | | |
| 32-47 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.14 R_TSIP_TdesEcbEncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesEcbEncryptInit(
    tsip_tdes_handle_t *handle,
    tsip_tdes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-------------------------|
| handle | 入力/出力 | Triple-DES 用ハンドラ(ワーク領域) |
| key_index | 入力 | Triple-DES ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |

Description

R_TSIP_TdesEcbEncryptInit() 関数は、DES 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_TdesEcbEncryptUpdate()関数および R_TSIP_TdesEcbEncryptFinal()関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.15 R_TSIP_TdesEcbEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesEcbEncryptUpdate(
    tsip_tdes_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_length
)
```

Parameters

| | | |
|--------------|-------|--------------------------|
| handle | 入力/出力 | Triple-DES 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_length | 入力 | 平文データ長 (8 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_TdesEcbEncryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"を handle に格納された key_index を用いて暗号化し、途中経過を第一引数"handle"に書き出します。また暗号化結果を第三引数"cipher"に書き出します。平文入力が完了した後は、R_TSIP_TdesEcbEncryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.16 R_TSIP_TdesEcbEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesEcbEncryptFinal(
    tsip_tdes_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_length
)
```

Parameters

| | | |
|---------------|-------|-------------------------|
| handle | 入力/出力 | TDES 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域(常に何も書き込まれません) |
| cipher_length | 入力/出力 | 暗号文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_TdesEcbEncryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"に演算結果、第三引数"cipher_length"に演算結果の長さを書き出します。第二引数は、本来は 8 バイトの倍数に満たない分の端数について暗号化した結果が書き出されますが、Update 関数には 8 バイトの倍数でしか入力できない制限があるため、cipher には常に何も書き込まれず、cipher_length には常に 0 が書き込まれます。cipher, cipher_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.17 R_TSIP_TdesEcbDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesEcbDecryptInit(
    tsip_tdes_handle_t *handle,
    tsip_tdes_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-------------------------|
| handle | 入力/出力 | Triple-DES 用ハンドラ(ワーク領域) |
| key_index | 入力 | Triple-DES ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

R_TSIP_TdesEcbDecryptInit() 関数は、DES 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_TdesEcbDecryptUpdate()関数および R_TSIP_TdesEcbDecryptFinal()関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.18 R_TSIP_TdesEcbDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesEcbDecryptUpdate(
    tsip_tdes_handle_t *handle,
    uint8_t *cipher,
    uint8_t *plain,
    uint32_t cipher_length
)
```

Parameters

| | | |
|---------------|-------|---------------------------|
| handle | 入力/出力 | Triple-DES 用ハンドラ(ワーク領域) |
| cipher | 入力 | 暗号文データ領域 |
| plain | 入力/出力 | 平文データ領域 |
| cipher_length | 入力 | 暗号文データ長 (8 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_TdesEcbDecryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"を handle に格納された key_index を用いて復号し、途中経過を第一引数"handle"に書き出します。また復号結果を第三引数"plain"に書き出します。暗号文入力が完了した後は、R_TSIP_TdesEcbDecryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.19 R_TSIP_TdesEcbDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesEcbDecryptFinal(
    tsip_tdes_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_length
)
```

Parameters

| | | |
|--------------|-------|-------------------------|
| handle | 入力/出力 | Triple-DES 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域(常に何も書き込まれません) |
| plain_length | 入力/出力 | 平文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_TdesEcbDecryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"に演算結果、第三引数"plain_length"に演算結果の長さを書き出します。第二引数は、本来は 8 バイトの倍数に満たない分の端数について復号した結果が書き出されますが、Update 関数には 8 バイトの倍数でしか入力できない制限があるため、plain には常に何も書き込まれず、plain_length には常に 0 が書き込まれます。plain, plain_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.20 R_TSIP_TdesCbcEncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcEncryptInit(
    tsip_tdes_handle_t *handle,
    tsip_tdes_key_index_t *key_index,
    uint8_t *ivec
)
```

Parameters

| | | |
|-----------|-------|-------------------------|
| handle | 入力/出力 | Triple-DES 用ハンドラ(ワーク領域) |
| key_index | 入力 | Triple-DES ユーザ鍵生成情報領域 |
| ivec | 入力 | 初期化ベクタ(8 バイト) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

R_TSIP_TdesCbcEncryptInit() 関数は、DES 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_TdesCbcEncryptUpdate()関数および R_TSIP_TdesCbcEncryptFinal()関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.21 R_TSIP_TdesCbcEncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcEncryptUpdate(
    tsip_tdes_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_length
)
```

Parameters

| | | |
|--------------|-------|--------------------------|
| handle | 入力/出力 | Trile-des 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_length | 入力 | 平文データ長 (8 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_TdesCbcEncryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"を handle に格納された key_index を用いて暗号化し、途中経過を第一引数"handle"に書き出します。また暗号化結果を第三引数"cipher"に書き出します。平文入力が完了した後は、R_TSIP_TdesCbcEncryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.22 R_TSIP_TdesCbcEncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcEncryptFinal(
    tsip_tdes_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_length
)
```

Parameters

| | | |
|---------------|-------|-------------------------|
| handle | 入力/出力 | Triple-DES 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域(常に何も書き込まれません) |
| cipher_length | 入力/出力 | 暗号文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_TdesCbcEncryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"に演算結果、第三引数"cipher_length"に演算結果の長さを書き出します。第二引数は、本来は 8 バイトの倍数に満たない分の端数について暗号化した結果が書き出されますが、Update 関数には 8 バイトの倍数でしか入力できない制限があるため、cipher には常に何も書き込まれず、cipher_length には常に 0 が書き込まれます。cipher, cipher_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.23 R_TSIP_TdesCbcDecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcDecryptInit(
    tsip_tdes_handle_t *handle,
    tsip_tdes_key_index_t *key_index,
    uint8_t *ivec
)
```

Parameters

| | | |
|-----------|-------|-------------------------|
| handle | 入力/出力 | Triple-DES 用ハンドラ(ワーク領域) |
| key_index | 入力 | Triple-DES ユーザ鍵生成情報領域 |
| ivec | 入力 | 初期化ベクタ(8 バイト) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

R_TSIP_TdesCbcDecryptInit() 関数は、DES 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_TdesCbcDecryptUpdate()関数および R_TSIP_TdesCbcDecryptFinal()関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.24 R_TSIP_TdesCbcDecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcDecryptUpdate(
    tsip_tdes_handle_t *handle,
    uint8_t *cipher,
    uint8_t *plain,
    uint32_t cipher_length
)
```

Parameters

| | | |
|---------------|-------|---------------------------|
| handle | 入力/出力 | Triple-DES 用ハンドラ(ワーク領域) |
| cipher | 入力 | 暗号文データ領域 |
| plain | 入力/出力 | 平文データ領域 |
| cipher_length | 入力 | 暗号文データ長 (8 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS : | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_TdesCbcDecryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"を handle に格納された key_index を用いて復号し、途中経過を第一引数"handle"に書き出します。また復号結果を第三引数"plain"に書き出します。暗号文入力が完了した後は、R_TSIP_TdesCbcDecryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.25 R_TSIP_TdesCbcDecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TdesCbcDecryptFinal(
    tsip_tdes_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_length
)
```

Parameters

| | | |
|--------------|-------|-------------------------|
| handle | 入力/出力 | Triple-DES 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域(常に何も書き込まれません) |
| plain_length | 入力/出力 | 平文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_TdesCbcDecryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"に演算結果、第三引数"plain_length"に演算結果の長さを書き出します。第二引数は、本来は 8 バイトの倍数に満たない分の端数について復号した結果が書き出されますが、Update 関数には 8 バイトの倍数でしか入力できない制限があるため、plain には常に何も書き込まれず、plain_length には常に 0 が書き込まれます。plain, plain_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.26 R_TSIP_GenerateArc4KeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateArc4KeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_arc4_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|-------|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられた ARC4 ユーザ鍵 |
| key_index | 入力/出力 | ARC4 ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ARC4 のユーザ鍵生成情報を出力するための API です。

encrypted_key には以下のフォーマットのデータを入力してください。

| | | | | |
|---------|---------------|-------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-255 | 暗号化された ARC4 鍵 | | | |
| 256-271 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

encrypted_key, iv および encrypted_provisioning_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.27 R_TSIP_GenerateArc4RandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateArc4RandomKeyIndex(
    tsip_arc4_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|---------------|
| key_index | 入力/出力 | ARC4 ユーザ鍵生成情報 |
|-----------|-------|---------------|

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

ARC4 のユーザ鍵生成情報を出力するための API です。

本 API は TSIP 内部にて乱数値からユーザ鍵を生成します。従ってユーザ鍵の入力は不要です。本 API が出力するユーザ鍵生成情報を使用しデータを暗号化することにより、データのデッドコピーを防ぐことができます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.28 R_TSIP_UpdateArc4KeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateArc4KeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_arc4_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|-------|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられたユーザ鍵 |
| key_index | 入力/出力 | ARC4 ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ARC4 鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|---------|---------|-------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-255 | ARC4 鍵 | | | |
| 256-271 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.29 R_TSIP_Arc4EncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Arc4EcbEncryptInit(
    tsip_arc4_handle_t *handle,
    tsip_arc4_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-------------------|
| handle | 入力/出力 | ARC4 用ハンドラ(ワーク領域) |
| key_index | 入力 | ARC4 ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |

Description

R_TSIP_Arc4EncryptInit() 関数は、ARC4 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Arc4EncryptUpdate()関数および R_TSIP_Arc4EncryptFinal()関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.30 R_TSIP_Arc4EncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Arc4EncryptUpdate(
    tsip_arc4_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_length
)
```

Parameters

| | | |
|--------------|-------|---------------------------|
| handle | 入力/出力 | ARC4 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_length | 入力 | 平文データ長 (16 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Arc4EncryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"を handle に格納された key_index を用いて暗号化し、途中経過を第一引数"handle"に書き出します。また暗号化結果を第三引数"cipher"に書き出します。平文入力が完了した後は、R_TSIP_Arc4EncryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.31 R_TSIP_Arc4EncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Arc4EncryptFinal(
    tsip_arc4_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_length
)
```

Parameters

| | | |
|---------------|-------|-------------------------|
| handle | 入力/出力 | ARC4 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域(常に何も書き込まれません) |
| cipher_length | 入力/出力 | 暗号文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Arc4EncryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"に演算結果、第三引数"cipher_length"に演算結果の長さを書き出します。第二引数は、本来は 16 バイトの倍数に満たない分の端数について暗号化した結果が書き出されますが、Update 関数には 16 バイトの倍数でしか入力できない制限があるため、cipher には常に何も書き込まれず、cipher_length には常に 0 が書き込まれます。cipher, cipher_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

5.32 R_TSIP_Arc4DecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Arc4DecryptInit(
    tsip_arc4_handle_t *handle,
    tsip_arc4_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-------------------|
| handle | 入力/出力 | ARC4 用ハンドラ(ワーク領域) |
| key_index | 入力 | ARC4 ユーザ鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

R_TSIP_Arc4DecryptInit() 関数は、ARC4 演算を実行する準備を行い、その結果を第一引数”handle”に書き出します。handle は、続く R_TSIP_Arc4DecryptUpdate()関数および R_TSIP_Arc4DecryptFinal()関数で引数として使用されます。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の使用方法については、「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.33 R_TSIP_Arc4DecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Arc4DecryptUpdate(
    tsip_arc4_handle_t *handle,
    uint8_t *cipher,
    uint8_t *plain,
    uint32_t cipher_length
)
```

Parameters

| | | |
|---------------|-------|----------------------------|
| handle | 入力/出力 | ARC4 用ハンドラ(ワーク領域) |
| cipher | 入力 | 暗号文データ領域 |
| plain | 入力/出力 | 平文データ領域 |
| cipher_length | 入力 | 暗号文データ長 (16 の倍数である必要があります) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Arc4DecryptUpdate() 関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"cipher"を handle に格納された key_index を用いて復号し、途中経過を第一引数"handle"に書き出します。また復号結果を第三引数"plain"に書き出します。暗号文入力が完了した後は、R_TSIP_Arc4DecryptFinal()を呼び出してください。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.34 R_TSIP_Arc4DecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Arc4DecryptFinal(
    tsip_arc4_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_length
)
```

Parameters

| | | |
|--------------|-------|------------------------|
| handle | 入力/出力 | ARC4 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域(常に何も書き込まれません) |
| plain_length | 入力/出力 | 平文データ長 (常に 0 が書き込まれます) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Arc4DecryptFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"plain"に演算結果、第三引数"plain_length"に演算結果の長さを書き出します。第二引数は、本来は 16 バイトの倍数に満たない分の端数について復号した結果が書き出されますが、Update 関数には 16 バイトの倍数でしか入力できない制限があるため、plain には常に何も書き込まれず、plain_length には常に 0 が書き込まれます。plain, plain_length は将来この制限が解除された際の互換性のための引数です。

<状態遷移>

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

5.35 R_TSIP_GenerateRsa1024PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa1024PublicKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_rsa1024_public_key_index_t *key_index
)
```

Parameters

| | | |
|---------------------------------------|-------------------------|----------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC をつけられた RSA 1024bit 公開鍵 |
| key_index | 入力/出力 | RSA 1024bit 公開鍵ユーザ鍵生成情報 |
| key_index->value.key_management_info1 | : 鍵管理情報 | |
| key_index->value.key_n | : RSA 1024bit 公開鍵 n(平文) | |
| key_index->value.key_e | : RSA 1024bit 公開鍵 e(平文) | |
| key_index->value.dummy | : ダミー | |
| key_index->value.key_management_info2 | : 鍵管理情報 | |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

1024 bit の RSA 公開鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|---------|--------------------|-----------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-127 | RSA 1024 bit 公開鍵 n | | | |
| 128-143 | RSA 1024 bit 公開鍵 e | 0 padding | | |
| 144-159 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key 生成方法、key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.36 R_TSIP_GenerateRsa1024PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa1024PrivateKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_rsa1024_private_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|-------|----------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられた RSA 1024bit 秘密鍵 |
| key_index | 入力/出力 | RSA 1024bit 秘密鍵ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

1024 bit の RSA 秘密鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|---------|--------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-127 | RSA 1024 bit 公開鍵 n | | | |
| 128-255 | RSA 1024 bit 秘密鍵 d | | | |
| 256-271 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key の生成方法、key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.37 R_TSIP_GenerateRsa2048PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa2048PublicKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_rsa2048_public_key_index_t *key_index
)
```

Parameters

| | | |
|---------------------------------------|-------------------------|----------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられた RSA 2048bit 公開鍵 |
| key_index | 入力/出力 | RSA 2048bit 公開鍵ユーザ鍵生成情報 |
| key_index->value.key_management_info1 | : 鍵管理情報 | |
| key_index->value.key_n | : RSA 2048bit 公開鍵 n(平文) | |
| key_index->value.key_e | : RSA 2048bit 公開鍵 e(平文) | |
| key_index->value.dummy | : ダミー | |
| key_index->value.key_management_info2 | : 鍵管理情報 | |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

2048 bit の RSA 公開鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|---------|--------------------|-----------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-255 | RSA 2048bit 公開鍵 n | | | |
| 256-271 | RSA 2048 bit 公開鍵 e | 0 padding | | |
| 272-287 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key の生成方法、key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.38 R_TSIP_GenerateRsa2048PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa2048PrivateKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_rsa2048_private_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|-------|----------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられた RSA 2048bit 秘密鍵 |
| key_index | 入力/出力 | RSA 2048bit 秘密鍵ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

2048 bit の RSA 秘密鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|---------|--------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-255 | RSA 2048bit 公開鍵 n | | | |
| 256-511 | RSA 2048 bit 秘密鍵 d | | | |
| 512-527 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

encrypted_provisioning_key, iv および encrypted_key の生成方法、key_index, install_key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.39 R_TSIP_GenerateRsa1024RandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa1024RandomKeyIndex(
    tsip_rsa1024_key_pair_index_t *key_pair_index
)
```

Parameters

| | | |
|---|-------|--------------------------------|
| key_pair_index | 入力/出力 | RSA 1024bit 公開鍵、秘密鍵ペアのユーザ鍵生成情報 |
| key_pair_index->public | | : RSA1024bit 公開鍵ユーザ鍵生成情報 |
| key_pair_index->public.value.key_management_info1 | | : 鍵管理情報 |
| key_pair_index->public.value.key_n | | : RSA 1024bit 公開鍵 n(平文) |
| key_pair_index->public.value.key_e | | : RSA 1024bit 公開鍵 e(平文) |
| key_pair_index->public.value.dummy | | : ダミー |
| key_pair_index->public.value.key_management_info2 | | : 鍵管理情報 |
| key_pair_index->private | | : RSA1024bit 秘密鍵ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生。鍵生成に失敗 |

Description

1024 bit の RSA 公開鍵、秘密鍵ペアのユーザ鍵生成情報を出力するための API です。本 API は TSIP 内部にて乱数値からユーザ鍵を生成します。従ってユーザ鍵の入力は不要です。本 API が出力するユーザ鍵生成情報を使用しデータを暗号化することにより、データのデッドコピーを防ぐことができます。key_pair_index->public に公開鍵の鍵生成情報、key_pair_index->private に秘密鍵の鍵生成情報を生成します。公開鍵の exponent は 0x00010001 のみを生成しています。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_pair_index->public ならびに key_pair_index->private 使用方法については「[7 章 鍵データの運用](#)」を参照してください。key_pair_index->public は R_TSIP_GenerateRsa1024PublicKeyIndex() から出力される公開鍵のユーザ鍵生成情報、key_pair_index->private は R_TSIP_GenerateRsa1024PrivateKeyIndex() から出力される秘密鍵のユーザ鍵生成情報と同様の運用になります。

Reentrant

非対応

5.40 R_TSIP_GenerateRsa2048RandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateRsa2048RandomKeyIndex(
    tsip_rsa2048_key_pair_index_t *key_pair_index
)
```

Parameters

| | | |
|---|-------|--------------------------------|
| key_pair_index | 入力/出力 | RSA 2048bit 公開鍵、秘密鍵ペアのユーザ鍵生成情報 |
| key_pair_index->public | | : RSA2048bit 公開鍵ユーザ鍵生成情報 |
| key_pair_index->public.value.key_management_info1 | | : 鍵管理情報 |
| key_pair_index->public.value.key_n | | : RSA 2048bit 公開鍵 n(平文) |
| key_pair_index->public.value.key_e | | : RSA 2048bit 公開鍵 e(平文) |
| key_pair_index->public.value.dummy | | : ダミー |
| key_pair_index->public.value.key_management_info2 | | : 鍵管理情報 |
| key_pair_index->private | | : RSA2048bit 秘密鍵ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生。鍵生成に失敗 |

Description

2048 bit の RSA 公開鍵、秘密鍵ペアのユーザ鍵生成情報を出力するための API です。本 API は TSIP 内部にて乱数値からユーザ鍵を生成します。従ってユーザ鍵の入力は不要です。本 API が出力するユーザ鍵生成情報を使用しデータを暗号化することにより、データのデッドコピーを防ぐことができます。key_pair_index->public に公開鍵の鍵生成情報、key_pair_index->private に秘密鍵の鍵生成情報を生成します。公開鍵の exponent は 0x00010001 のみを生成しています。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_pair_index->public ならびに key_pair_index->private 使用方法については「[7 章 鍵データの運用](#)」を参照してください。key_pair_index->public は R_TSIP_GenerateRsa2048PublicKeyIndex() から出力される公開鍵のユーザ鍵生成情報、key_pair_index->private は R_TSIP_GenerateRsa2048PrivateKeyIndex() から出力される秘密鍵のユーザ鍵生成情報と同様の運用になります。

Reentrant

非対応

5.41 R_TSIP_UpdateRsa1024PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateRsa1024PublicKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_rsa1024_public_key_index_t *key_index
)
```

Parameters

| | | |
|--|-------|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた公開鍵 |
| key_index | 入力/出力 | RSA 1024bit 公開鍵のユーザ鍵生成情報 |
| key_index->value.key_management_info1 : 鍵管理情報 | | |
| key_index->value.key_n : RSA 1024bit 公開鍵 n(平文) | | |
| key_index->value.key_e : RSA 1024bit 公開鍵 e(平文) | | |
| key_index->value.dummy : ダミー | | |
| key_index->value.key_management_info2 : 鍵管理情報 | | |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

RSA 1024bit 公開鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|---------|--------------------|-----------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-127 | RSA 1024 bit 公開鍵 n | | | |
| 128-143 | RSA 1024 bit 公開鍵 e | 0 padding | | |
| 144-159 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.42 R_TSIP_UpdateRsa1024PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateRsa1024PrivateKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_rsa1024_private_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|-------|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた秘密鍵 |
| key_index | 入力/出力 | RSA 1024bit 秘密鍵のユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

RSA 1024bit 秘密鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|---------|--------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-127 | RSA 1024 bit 公開鍵 n | | | |
| 128-255 | RSA 1024 bit 秘密鍵 d | | | |
| 256-271 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.43 R_TSIP_UpdateRsa2048PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateRsa2048PublicKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_rsa2048_public_key_index_t *key_index
)
```

Parameters

| | | |
|--|-------|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた公開鍵 |
| key_index | 入力/出力 | RSA 2048bit 公開鍵のユーザ鍵生成情報 |
| key_index->value.key_management_info1 : 鍵管理情報 | | |
| key_index->value.key_n : RSA 2048bit 公開鍵 n(平文) | | |
| key_index->value.key_e : RSA 2048bit 公開鍵 e(平文) | | |
| key_index->value.dummy : ダミー | | |
| key_index->value.key_management_info2 : 鍵管理情報 | | |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

RSA 2048bit 公開鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|---------|--------------------|-----------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-255 | RSA 2048 bit 公開鍵 n | | | |
| 256-271 | RSA 2048 bit 公開鍵 e | 0 padding | | |
| 272-287 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.44 R_TSIP_UpdateRsa2048PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateRsa2048PrivateKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_rsa2048_private_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|-------|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた秘密鍵 |
| key_index | 入力/出力 | RSA 2048bit 秘密鍵のユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

RSA 2048bit 秘密鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|---------|--------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-255 | RSA 2048 bit 公開鍵 n | | | |
| 256-511 | RSA 2048 bit 秘密鍵 d | | | |
| 512-527 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.45 R_TSIP_RsaesPkcs1024Encrypt

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsaesPkcs1024Encrypt(
    tsip_rsa_byte_data_t *plain,
    tsip_rsa_byte_data_t *cipher,
    tsip_rsa1024_public_key_index_t *key_index
)
```

Parameters

| | | | |
|---------------------|-------|--------|---|
| plain | 入力 | 平文 | |
| plain->pdata | | | : 平文を格納している配列のポインタを指定 |
| plain->data_length | | | : 平文配列の有効データ長を指定 データサイズ <= 公開鍵 n サイズ-11 |
| cipher | 入力/出力 | 暗号文 | |
| cipher->pdata | | | : 暗号文を格納する配列のポインタを指定 |
| cipher->data_length | | | : 暗号文のバッファサイズを入力 暗号化後、有効データ長を出力(公開鍵 n サイズ) |
| key_index | 入力 | 鍵データ領域 | : 1024bit RSA 公開鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |

Description

R_TSIP_RsaesPkcs1024Encrypt()関数は、第一引数"plain"に入力された平文を RSAES-PKCS1-V1_5 に従って、RSA 暗号化をします。暗号化結果を第二引数"cipher"に書き出します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.46 R_TSIP_RsaesPkcs1024Decrypt

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsaesPkcs1024Decrypt(
    tsip_rsa_byte_data_t *cipher,
    tsip_rsa_byte_data_t *plain,
    tsip_rsa1024_private_key_index_t *key_index
)
```

Parameters

| | | | |
|---------------------|-------|--------|--|
| cipher | 入力 | 暗号文 | |
| cipher->pdata | | | : 暗号文を格納している配列のポインタを指定 |
| cipher->data_length | | | : 暗号文配列の有効データ長を指定 (公開鍵 n サイズ) |
| plain | 入力/出力 | 平文 | |
| plain->pdata | | | : 平文を格納する配列のポインタを指定 |
| plain->data_length | | | : 平文バッファサイズ入力 平文バッファサイズ >= 公開鍵 n サイズ-11 を満たすバッファを用意してください 復号後、有効データ長を出力 |
| key_index | 入力 | 鍵データ領域 | : 1024bit RSA 秘密鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |

Description

R_TSIP_RsaesPkcs1024Decrypt()関数は、第一引数"cipher"に入力された暗号文を RSAES-PKCS1-V1_5 に従って、RSA 復号を行います。復号結果を第二引数"plain"に出力します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.47 R_TSIP_RsaesPkcs2048Encrypt

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsaesPkcs2048Encrypt(
    tsip_rsa_byte_data_t *plain,
    tsip_rsa_byte_data_t *cipher,
    tsip_rsa2048_public_key_index_t *key_index
)
```

Parameters

| | | | |
|---------------------|-------|--------|---|
| plain | 入力 | 平文 | |
| plain->pdata | | | : 平文を格納している配列のポインタを指定 |
| plain->data_length | | | : 平文配列の有効データ長を指定 データサイズ <= 公開鍵 n サイズ-11 |
| cipher | 入力/出力 | 暗号文 | |
| cipher->pdata | | | : 暗号文を格納する配列のポインタを指定 |
| cipher->data_length | | | : 暗号文のバッファサイズを入力 暗号化後、有効データ長を出力(公開鍵 n サイズ) |
| key_index | 入力 | 鍵データ領域 | : 2048bit RSA 公開鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |

Description

R_TSIP_RsaesPkcs2048Encrypt()関数は、第一引数"plain"に入力された平文を RSAES-PKCS1-V1_5 に従って、RSA 暗号化をします。暗号化結果を第二引数"cipher"に書き出します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.48 R_TSIP_RsaesPkcs2048Decrypt

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsaesPkcs2048Decrypt(
    tsip_rsa_byte_data_t *cipher,
    tsip_rsa_byte_data_t *plain,
    tsip_rsa2048_private_key_index_t *key_index
)
```

Parameters

| | | | |
|---------------------|-------|--------|--|
| cipher | 入力 | 暗号文 | |
| cipher->pdata | | | : 暗号文を格納している配列のポインタを指定 |
| cipher->data_length | | | : 暗号文配列の有効データ長を指定 (公開鍵 n サイズ) |
| plain | 入力/出力 | 平文 | |
| plain->pdata | | | : 平文を格納する配列のポインタを指定 |
| plain->data_length | | | : 平文バッファサイズ入力 平文バッファサイズ >= 公開鍵 n サイズ-11 を満たすバッファを用意してください 復号後、有効データ長を出力 |
| key_index | 入力 | 鍵データ領域 | : 2048bit RSA 秘密鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |

Description

R_TSIP_RsaesPkcs2048Decrypt()関数は、第一引数"cipher"に入力された暗号文を RSAES-PKCS1-V1_5 に従って、RSA 復号を行います。復号結果を第二引数"plain"に出力します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.49 R_TSIP_RsassaPkcs1024SignatureGenerate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsassaPkcs1024SignatureGenerate(
    tsip_rsa_byte_data_t *message_hash,
    tsip_rsa_byte_data_t *signature,
    tsip_rsa1024_private_key_index_t *key_index,
    uint8_t hash_type
)
```

Parameters

| | | |
|---------------------------|-------|---|
| message_hash | 入力 | 署名を付けるメッセージまたはハッシュ値情報 |
| message_hash->pdata | | : メッセージまたはハッシュ値を格納している配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(メッセージの場合のみ指定) |
| message_hash->data_type | | : message_hash のデータ種別を選択 メッセージ: 0 ハッシュ値: 1 |
| signature | 入力/出力 | 署名文格納先情報 |
| signature->pdata | | : 署名文を格納する配列のポインタを指定 |
| signature->data_length | | : データ長(バイト単位) |
| key_index | 入力 | 鍵データ領域 : 1024bit RSA 秘密鍵のユーザ鍵生成情報を入力 |
| hash_type | 入力 | hash の種類 : R_TSIP_RSA_HASH_MD5, R_TSIP_RSA_HASH_SHA1 または R_TSIP_RSA_HASH_SHA256 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| 上記の Return Values 以外 | ハッシュ演算を行う内部関数からの戻り値 |

Description

R_TSIP_RsassaPkcs1024SignatureGenerate()関数は、RSASSA-PKCS1-V1_5に従って、第一引数"message_hash"に入力されたメッセージ文またはハッシュ値から、第三引数"key_index"に入力された秘密鍵ユーザ鍵生成情報を使って署名文を計算し、第二引数"signature"に書き出します。第一引数"message_hash->data_type"でメッセージを指定した場合、メッセージに対して第四引数"hash_type"で指定された HASH 計算を行います。第一引数"message_hash->data_type"でハッシュ値を指定した場合、第四引数"hash_type"で指定したハッシュアルゴリズムで計算したハッシュ値を"message_hash->pdata"へ入力してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.50 R_TSIP_RsassaPkcs1024SignatureVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsassaPkcs1024SignatureVerification(
    tsip_rsa_byte_data_t *signature,
    tsip_rsa_byte_data_t *message_hash,
    tsip_rsa1024_public_key_index_t *key_index,
    uint8_t hash_type
)
```

Parameters

| | | |
|---------------------------|----|---|
| signature | 入力 | 検証する署名文情報 |
| signature->pdata | | : 署名文を格納している配列のポインタを指定 |
| signature->data_length | | : 配列の有効データ長を指定 |
| message_hash | 入力 | 検証するメッセージ文またはハッシュ値情報 |
| message_hash->pdata | | : メッセージまたはハッシュ値を格納している配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(メッセージの場合のみ指定) |
| message_hash->data_type | | : message_hash のデータ種別を選択 メッセージ: 0 ハッシュ値: 1 |
| key_index | 入力 | 鍵データ領域 : 1024bit RSA 公開鍵のユーザ鍵生成情報を入力 |
| hash_type | 入力 | hash の種類 : R_TSIP_RSA_HASH_MD5, R_TSIP_RSA_HASH_SHA1 または R_TSIP_RSA_HASH_SHA256 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_AUTHENTICATION: | 署名検証失敗 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| 上記の Return Values 以外 | ハッシュ演算を行う内部関数からの戻り値 |

Description

R_TSIP_RsassaPkcs1024SignatureVerification()関数は、RSASSA-PKCS1-V1_5に従って、第三引数"key_index"に入力された公開鍵ユーザ鍵生成情報を使い第一引数"signature"に入力された署名文と第二引数"message_hash"に入力されたメッセージ文またはハッシュ値の検証をします。第二引数"message_hash->data_type"でメッセージを指定した場合、第三引数"key_index"に入力された公開鍵ユーザ鍵生成情報と第四引数"hash_type"で指定された HASH 計算を行います。第二引数"message_hash->data_type"でハッシュ値を指定した場合、第四引数"hash_type"で指定したハッシュアルゴリズムで計算したハッシュ値を"message_hash->pdata"へ入力してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.51 R_TSIP_RsassaPkcs2048SignatureGenerate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsassaPkcs2048SignatureGenerate(
    tsip_rsa_byte_data_t *message_hash,
    tsip_rsa_byte_data_t *signature,
    tsip_rsa2048_private_key_index_t *key_index,
    uint8_t hash_type
)
```

Parameters

| | | |
|---------------------------|-------|---|
| message_hash | 入力 | 署名を付けるメッセージまたはハッシュ値情報 |
| message_hash->pdata | | : メッセージまたはハッシュ値を格納している配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(メッセージの場合のみ指定) |
| message_hash->data_type | | : message_hash のデータ種別を選択 メッセージ: 0 ハッシュ値: 1 |
| signature | 入力/出力 | 署名文格納先情報 |
| signature->pdata | | : 署名文を格納する配列のポインタを指定 |
| signature->data_length | | : データ長(バイト単位) |
| key_index | 入力 | 鍵データ領域 : 2048bit RSA 秘密鍵のユーザ鍵生成情報を入力 |
| hash_type | 入力 | hash の種類 : R_TSIP_RSA_HASH_MD5, R_TSIP_RSA_HASH_SHA1 または R_TSIP_RSA_HASH_SHA256 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| 上記の Return Values 以外 | ハッシュ演算を行う内部関数からの戻り値 |

Description

R_TSIP_RsassaPkcs2048SignatureGenerate()関数は、RSASSA-PKCS1-V1_5に従って、第一引数"message_hash"に入力されたメッセージ文またはハッシュ値から、第三引数"key_index"に入力された秘密鍵ユーザ鍵生成情報を使って署名文を計算し、第二引数"signature"に書き出します。第一引数"message_hash->data_type"でメッセージを指定した場合、メッセージに対して第四引数"hash_type"で指定された HASH 計算を行います。第一引数"message_hash->data_type"でハッシュ値を指定した場合、第四引数"hash_type"で指定したハッシュアルゴリズムで計算したハッシュ値を"message_hash->pdata"へ入力してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.52 R_TSIP_RsassaPkcs2048SignatureVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_RsassaPkcs2048SignatureVerification(
    tsip_rsa_byte_data_t *signature,
    tsip_rsa_byte_data_t *message_hash,
    tsip_rsa2048_public_key_index_t *key_index,
    uint8_t hash_type
)
```

Parameters

| | | |
|---------------------------|----|---|
| signature | 入力 | 検証する署名文情報 |
| signature->pdata | | : 署名文を格納している配列のポインタを指定 |
| signature->data_length | | : 配列の有効データ長を指定 |
| message_hash | 入力 | 検証するメッセージ文またはハッシュ値情報 |
| message_hash->pdata | | : メッセージまたはハッシュ値を格納している配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(メッセージの場合のみ指定) |
| message_hash->data_type | | : message_hash のデータ種別を選択 メッセージ: 0 ハッシュ値: 1 |
| key_index | 入力 | 鍵データ領域 : 1024bit RSA 公開鍵のユーザ鍵生成情報を入力 |
| hash_type | 入力 | hash の種類 : R_TSIP_RSA_HASH_MD5, R_TSIP_RSA_HASH_SHA1 または R_TSIP_RSA_HASH_SHA256 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_AUTHENTICATION: | 署名検証失敗 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| 上記の Return Values 以外 | ハッシュ演算を行う内部関数からの戻り値 |

Description

R_TSIP_RsassaPkcs2048SignatureVerification()関数は、RSASSA-PKCS1-V1_5に従って、第三引数"key_index"に入力された公開鍵ユーザ鍵生成情報を使い第一引数"signature"に入力された署名文と第二引数"message_hash"に入力されたメッセージ文またはハッシュ値の検証をします。第二引数"message_hash->data_type"でメッセージを指定した場合、第三引数"key_index"に入力された公開鍵ユーザ鍵生成情報と第四引数"hash_type"で指定された HASH 計算を行います。第二引数"message_hash->data_type"でハッシュ値を指定した場合、第四引数"hash_type"で指定したハッシュアルゴリズムで計算したハッシュ値を"message_hash->pdata"へ入力してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.53 R_TSIP_Rsa2048DhKeyAgreement

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Rsa2048DhKeyAgreement(
    tsip_aes_key_index_t *key_index,
    tsip_rsa2048_private_key_index_t *sender_private_key_index,
    uint8_t *message,
    uint8_t *receiver_modulus,
    uint8_t *sender_modulus
)
```

Parameters

| | | |
|--------------------------|-------|--|
| key_index | 入力 | AES-128 CMAC 演算用ユーザ鍵生成情報領域 |
| sender_private_key_index | 入力 | DH 演算で使用する秘密鍵生成情報 秘密鍵生成情報に含まれる秘密鍵 d を TSIP 内部で 復号し、利用します |
| message | 入力 | メッセージ(2048bit) sender_private_key_index に含まれる素数(d)より 小さい値を設定してください |
| receiver_modulus | 入力 | Receiver が計算したべき乗剰余演算結果 + MAC 2048bit べき乗剰余演算 128bit |
| sender_modulus | 入力/出力 | Sender が計算したべき乗剰余演算結果 + MAC 2048bit べき乗剰余演算 128bit |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

RSA-2048 による DH 演算を実施します。

なお、Sender は TSIP、Receiver は鍵交換相手を示します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

5.54 R_TSIP_Sha1HmacGenerateInit

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha1HmacGenerateInit(
    tsip_hmac_sha_handle_t *handle,
    tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドラ(ワーク領域) |
| key_index | 入力 | MAC 鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常な MAC 鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

R_TSIP_Sha1HmacGenerateInit()関数は、第二引数の"key_index"を用い SHA1-HMAC 演算を実行する準備を行い、その結果を第一引数"handle"に書き出します。"key_index"には、TLS 連携機能の場合、R_TSIP_TlsGenerateSessionKey()関数で生成された MAC 鍵生成情報を使用してください。"handle"は続く R_TSIP_Sha1HmacGenerateUpdate()関数や、R_TSIP_Sha1HmacGenerateFinal()関数の引数で使用します。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.55 R_TSIP_Sha1HmacGenerateUpdate

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha1HmacGenerateUpdate(
    tsip_hmac_sha_handle_t *handle,
    uint8_t *message,
    uint32_t message_length
)
```

Parameters

| | | |
|----------------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドル(ワーク領域) |
| message | 入力 | メッセージ領域 |
| message_length | 入力 | メッセージ長 |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha1HmacGenerateUpdate()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"message"と第三引数の"message_length"からハッシュ値を演算し、途中経過を第一引数"handle"に書き出します。メッセージ入力が完了した後は、R_TSIP_Sha1HmacGenerateFinal()を呼び出してください。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.56 R_TSIP_Sha1HmacGenerateFinal

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha1HmacGenerateFinal(
    tsip_hmac_sha_handle_t *handle,
    uint8_t *mac
)
```

Parameters

| | | |
|--------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドル(ワーク領域) |
| mac | 入力/出力 | HMAC 領域(20 バイト) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha1HmacGenerateFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"mac"に演算結果を書き出します。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.57 R_TSIP_Sha256HmacGenerateInit

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha256HmacGenerateInit(
    tsip_hmac_sha_handle_t *handle,
    tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドラ(ワーク領域) |
| key_index | 入力 | MAC 鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常な MAC 鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

R_TSIP_Sha256HmacGenerateInit()関数は、第二引数の"key_index"を用い SHA256-HMAC 演算を実行する準備を行い、その結果を第一引数"handle"に書き出します。"key_index"には、TLS 連携機能で使用する場合は R_TSIP_TlsGenerateSessionKey()関数で生成された MAC 鍵生成情報を使用してください。"handle"は続く R_TSIP_Sha256HmacGenerateUpdate()関数や、R_TSIP_Sha256HmacGenerateFinal()関数の引数で使します。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.58 R_TSIP_Sha256HmacGenerateUpdate

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha256HmacGenerateUpdate(
    tsip_hmac_sha_handle_t *handle,
    uint8_t *message,
    uint32_t message_length
)
```

Parameters

| | | |
|----------------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドル(ワーク領域) |
| message | 入力 | メッセージ領域 |
| message_length | 入力 | メッセージ長 |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha256HmacGenerateUpdate()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"message"と第三引数の"message_length"からハッシュ値を演算し、途中経過を第一引数"handle"に書き出します。メッセージ入力が完了した後は、R_TSIP_Sha256HmacGenerateFinal()を呼び出してください。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.59 R_TSIP_Sha256HmacGenerateFinal

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha256HmacGenerateFinal(
    tsip_hmac_sha_handle_t *handle,
    uint8_t *mac
)
```

Parameters

| | | |
|--------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドル(ワーク領域) |
| mac | 入力/出力 | HMAC 領域(32 バイト) |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha256HmacGenerateFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"mac"に演算結果を書き出します。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.60 R_TSIP_Sha1HmacVerifyInit

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha1HmacVerifyInit(
    tsip_hmac_sha_handle_t *handle,
    tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドラ(ワーク領域) |
| key_index | 入力 | MAC 鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常な MAC 鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

R_TSIP_Sha1HmacVerifyInit()関数は、第一引数の"key_index"を用い SHA1-HMAC 演算を実行する準備を行い、その結果を第一引数"handle"に書き出します。"key_index"には、TLS 連携機能で使用する場合、R_TSIP_TlsGenerateSessionKey()関数で生成された MAC 鍵生成情報を使用してください。"handle"は続く R_TSIP_Sha1HmacVerifyUpdate()関数や、R_TSIP_Sha1HmacVerifyFinal()関数の引数で使用します。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.61 R_TSIP_Sha1HmacVerifyUpdate

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha1HmacVerifyUpdate(
    tsip_hmac_sha_handle_t *handle,
    uint8_t *message,
    uint32_t message_length
)
```

Parameters

| | | |
|----------------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドル(ワーク領域) |
| message | 入力 | メッセージ領域 |
| message_length | 入力 | メッセージ長 |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha1HmacVerifyUpdate()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"message"と第三引数の"message_length"からハッシュ値を演算し、途中経過を第一引数"handle"に書き出します。メッセージ入力が完了した後は、R_TSIP_Sha1HmacVerifyFinal()を呼び出してください。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.62 R_TSIP_Sha1HmacVerifyFinal

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha1HmacVerifyFinal(
    tsip_hmac_sha_handle_t *handle,
    uint8_t *mac,
    uint32_t mac_length
)
```

Parameters

| | | |
|------------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドル(ワーク領域) |
| mac | 入力 | HMAC 領域 |
| mac_length | 入力 | HMAC 長 |

Return Values

| | |
|-----------------------------|--------------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生、もしくは認証が失敗 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha1HmacVerifyFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"mac"と第三引数の"mac_length"から mac 値の検証を行います。"mac_length"の単位は byte で 4 以上 20 以下の値を入力してください。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.63 R_TSIP_Sha256HmacVerifyInit

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha256HmacVerifyInit(
    tsip_hmac_sha_handle_t *handle,
    tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

| | | |
|-----------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドラ(ワーク領域) |
| key_index | 入力 | MAC 鍵生成情報領域 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常な MAC 鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

R_TSIP_Sha256HmacVerifyInit()関数は、第二引数の"key_index"を用い SHA256-HMAC 演算を実行する準備を行い、その結果を第一引数"handle"に書き出します。"key_index"には TLS 連携機能で使用する場合、R_TSIP_TlsGenerateSessionKey()関数で生成された MAC 鍵生成情報を使用してください。"handle"は続く R_TSIP_Sha256HmacVerifyUpdate()関数や、R_TSIP_Sha256HmacVerifyFinal()関数の引数で使します。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.64 R_TSIP_Sha256HmacVerifyUpdate

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha256HmacVerifyUpdate(
    tsip_hmac_sha_handle_t *handle,
    uint8_t *message,
    uint32_t message_length
)
```

Parameters

| | | |
|----------------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドル(ワーク領域) |
| message | 入力 | メッセージ領域 |
| message_length | 入力 | メッセージ長 |

Return Values

| | |
|-----------------------------|---------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha256HmacVerifyUpdate()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"message"と第三引数の"message_length"からハッシュ値を演算し、途中経過を第一引数"handle"に書き出します。メッセージ入力が完了した後は、R_TSIP_Sha256HmacVerifyFinal()を呼び出してください。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.65 R_TSIP_Sha256HmacVerifyFinal

Format

```
#include "r_tsip_rx_if.h"

e_tsip_err_t R_TSIP_Sha256HmacVerifyFinal(
    tsip_hmac_sha_handle_t *handle,
    uint8_t *mac,
    uint32_t mac_length
)
```

Parameters

| | | |
|------------|-------|-----------------------|
| handle | 入力/出力 | SHA-HMAC 用ハンドル(ワーク領域) |
| mac | 入力 | HMAC 領域 |
| mac_length | 入力 | HMAC 長 |

Return Values

| | |
|-----------------------------|--------------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生、もしくは認証が失敗 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_Sha256HmacVerifyFinal()関数は、第一引数"handle"で指定されたハンドルを使用し、第二引数の"mac"と第三引数の"mac_length"から mac 値の検証を行います。"mac_length"の単位は byte で 4 以上 32 以下の値を入力してください。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.66 R_TSIP_GenerateTlsRsaPublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateTlsRsaPublicKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_tls_ca_certification_public_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|-------|---------------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | AES128-ECB モードで暗号化された 2048bit RSA 公開鍵 |
| key_index | 入力/出力 | TLS 連携機能で使用する 2048bit 長の RSA 公開鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

TLS 連携機能で使用する 2048bit RSA の公開鍵のユーザ鍵生成情報を出力するための API です。

encrypted_key には以下のフォーマットのデータを入力してください。

| | | | | |
|---------|--------------------|-----------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-255 | RSA 2048 bit 公開鍵 n | | | |
| 256-271 | RSA 2048 bit 公開鍵 e | 0 padding | | |
| 272-287 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.67 R_TSIP_UpdateTlsRsaPublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateTlsRsaPublicKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_tls_ca_certification_public_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|-------|-----------------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた公開鍵 |
| key_index | 入力/出力 | TLS 連携機能で使用する RSA 2048bit 公開鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

TLS 連携機能で使用する RSA 2048bit 公開鍵の鍵生成情報を更新するための API です。

encrypted_key には以下のフォーマットのデータを入力してください。

| | | | | |
|---------|--------------------|-----------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-255 | RSA 2048 bit 公開鍵 n | | | |
| 256-271 | RSA 2048 bit 公開鍵 e | 0 padding | | |
| 272-287 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.68 R_TSIP_TlsRootCertificateVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsRootCertificateVerification(
    uint32_t public_key_type,
    uint8_t *certificate,
    uint32_t certificate_length,
    uint32_t public_key_n_start_position,
    uint32_t public_key_n_end_position,
    uint32_t public_key_e_start_position,
    uint32_t public_key_e_end_position,
    uint8_t *signature,
    uint32_t *encrypted_root_public_key
)
```

Parameters

| | | |
|-----------------------------|-------|--|
| public_key_type | 入力 | 証明書に含まれている公開鍵の種類 0 : RSA 2048bit, 2 : ECC P-256, 他は Reserved |
| certificate | 入力 | ルート CA 証明書の束(DER 形式) |
| certificate_length | 入力 | ルート CA 証明書の束のバイト長 |
| public_key_n_start_position | 入力 | 引数 certificate のアドレスを起点とした 公開鍵の開始バイト位置 |
| public_key_n_end_position | 入力 | 公開鍵 public_key_type 0 : n, 2 : Qx 引数 certificate のアドレスを起点とした 公開鍵の終了バイト位置 |
| public_key_e_start_position | 入力 | 公開鍵 public_key_type 0 : n, 2 : Qx 引数 certificate のアドレスを起点とした 公開鍵の開始バイト位置 |
| public_key_e_end_position | 入力 | 公開鍵 public_key_type 0 : e, 2 : Qy 引数 certificate のアドレスを起点とした 公開鍵の終了バイト位置 |
| signature | 入力 | 公開鍵 public_key_type 0 : e, 2 : Qy ルート CA 証明書の束に対する署名データ 署名データは 256 バイト入力してください 署名方式は「RSA2048 PSS with SHA256」 |
| encrypted_root_public_key | 入力/出力 | R_TSIP_TlsCertificateVerification または R_TSIP_TlsCertificateVerificationExtension で 使用する暗号化された ECDSA P256 もしくは RSA2048 公開鍵 public_key_type が 0 の場合 560 バイト, 2 の場合 96 バイト出力されます |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |

Description

ルート CA 証明書の束を検証するための API です。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は TSIP 使用可能状態です。

Reentrant

非対応

5.69 R_TSIP_TlsCertificateVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsCertificateVerification(
    uint32_t public_key_type,
    uint32_t *encrypted_input_public_key,
    uint8_t *certificate,
    uint32_t certificate_length,
    uint8_t *signature,
    uint32_t public_key_n_start_position,
    uint32_t public_key_n_end_position,
    uint32_t public_key_e_start_position,
    uint32_t public_key_e_end_position,
    uint32_t *encrypted_output_public_key
)
```

Parameters

| | | |
|-----------------------------|----|--|
| public_key_type | 入力 | 証明書に含まれている公開鍵の種類 0 : RSA 2048bit, 1 : RSA 4096bit, 2 : ECC P-256, 他は Reserved ただし 1 : RSA 4096bit は RX651/RX65N のみ対応 |
| encrypted_input_public_key | 入力 | R_TSIP_TlsRootCertificateVerification、 R_TSIP_TlsCertificateVerification または、 R_TSIP_TlsCertificateVerificationExtension で出力 された、暗号化された公開鍵 データサイズ public_key_type 0 : 140 ワード, 2 : 24 ワード |
| certificate | 入力 | 証明書の束(DER 形式) |
| certificate_length | 入力 | 証明書の束のバイト長 |
| signature | 入力 | 証明書の束に対する署名データ public_key_type:0 データサイズ 256 バイト 署名アルゴリズムは sha256 With RSA2048 Encryption public_key_type:1 データサイズ 512 バイト 署名アルゴリズムは sha256 With RSA4096 Encryption public_key_type:2 データサイズ 64 バイト "r(256bit) s(256bit)" 署名アルゴリズムは sha256 With ECDSA P-256 Encryption |
| public_key_n_start_position | 入力 | 引数 certificate のアドレスを起点とした 公開鍵の開始バイト位置 |
| public_key_n_end_position | 入力 | 公開鍵 public_key_type 0,1 : n, 2 : Qx 引数 certificate のアドレスを起点とした 公開鍵の終了バイト位置 |
| public_key_e_start_position | 入力 | 公開鍵 public_key_type 0,1 : n, 2 : Qx 引数 certificate のアドレスを起点とした 公開鍵の開始バイト位置 |
| public_key_e_end_position | 入力 | 公開鍵 public_key_type 0,1 : e, 2 : Qy 引数 certificate のアドレスを起点とした 公開鍵の終了バイト位置 |

| | | |
|-----------------------------|-------|--|
| encrypted_output_public_key | 入力/出力 | 公開鍵 public_key_type 0,1 : e, 2 : Qy R_TSIP_TlsCertificateVerification、 R_TSIP_TlsCertificateVerificationExtension、 R_TSIP_TlsEncryptPreMasterSecretWithRsa2048PublicKey または R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrives で 使用する暗号化された公開鍵 ただし public_key_type=1 選択時は R_TSIP_TlsCertificateVerification 及び R_TSIP_TlsCertificateVerificationExtension でのみ 使用可能 データサイズ public_key_type 0,1 : 140 ワード, 2 : 24 ワード |
|-----------------------------|-------|--|

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

サーバ証明書、中間証明書の署名を検証するための API です。

R_TSIP_TlsCertificateVerificationExtension()関数と同じ用途で使いますが、署名検証をする鍵のアルゴリズムと certificate から取り出す鍵のアルゴリズムが同一の場合には、こちらの関数を使用してください。

また、pub_key_type が 1 : RSA 4096bit の場合は RX651/RX65N のみ対応しています。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.70 R_TSIP_TlsCertificateVerificationExtension

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsCertificateVerificationExtension(
    uint32_t public_key_type,
    uint32_t public_key_output_type,
    uint32_t *encrypted_input_public_key,
    uint8_t *certificate,
    uint32_t certificate_length,
    uint8_t *signature,
    uint32_t public_key_n_start_position,
    uint32_t public_key_n_end_position,
    uint32_t public_key_e_start_position,
    uint32_t public_key_e_end_position,
    uint32_t *encrypted_output_public_key
)
```

Parameters

| | | |
|-----------------------------|----|---|
| public_key_type | 入力 | 入力する証明書に含まれている公開鍵の種類 0 : RSA 2048bit, 1 : RSA 4096bit, 2 : ECC P-256, 他は Reserved |
| public_key_output_type | 入力 | certificate から出力する鍵の種類 0 : RSA 2048bit, 1 : RSA 4096bit 2 : ECC P-256, 他は Reserved |
| encrypted_input_public_key | 入力 | R_TSIP_TlsRootCertificateVerification、 R_TSIP_TlsCertificateVerification または、 R_TSIP_TlsCertificateVerificationExtension で出力 された、暗号化された公開鍵 データサイズ |
| certificate | 入力 | public_key_type 0,1 : 140 ワード, 2 : 24 ワード |
| certificate_length | 入力 | 証明書の束(DER 形式) |
| signature | 入力 | 証明書の束のバイト長 証明書の束に対する署名データ public_key_type:0 データサイズ 256 バイト 署名アルゴリズムは sha256 With RSA2048 Encryption public_key_type:1 データサイズ 512 バイト 署名アルゴリズムは sha256 With RSA4096 Encryption public_key_type:2 データサイズ 64 バイト"r(256bit) s(256bit)" 署名アルゴリズムは sha256 With ECDSA P-256 Encryption |
| public_key_n_start_position | 入力 | 引数 certificate のアドレスを起点とした 公開鍵の開始バイト位置 |
| public_key_n_end_position | 入力 | 公開鍵 public_output_key_type 0,1 : n, 2 : Qx 引数 certificate のアドレスを起点とした 公開鍵の終了バイト位置 |
| public_key_e_start_position | 入力 | 公開鍵 public_output_key_type 0,1 : n, 2 : Qx 引数 certificate のアドレスを起点とした 公開鍵の開始バイト位置 |

| | | |
|-----------------------------|-------|---|
| public_key_e_end_position | 入力 | 公開鍵 public_output_key_type 0,1 : e, 2 :Qy 引数 certificate のアドレスを起点とした 公開鍵の終了バイト位置 |
| encrypted_output_public_key | 入力/出力 | 公開鍵 public_output_key_type 0,1 : e, 2 :Qy R_TSIP_TlsCertificateVerification、 R_TSIP_TlsCertificateVerificationExtension または、 R_TSIP_TlsEncryptPreMasterSecretWithRsa2048PublicKey、 R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrives で 使用する暗号化された公開鍵 ただし public_output_key_type =1 選択時は R_TSIP_TlsCertificateVerification 及び R_TSIP_TlsCertificateVerificationExtension でのみ 使用可能 データサイズ public_key_type 0,1 : 140 ワード, 2 :24 ワード |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |

Description

サーバ証明書、中間証明書の署名を検証するための API です。R_TSIP_TlsCertificateVerification()
関数と同じ用途で使いますが、署名検証をする鍵のアルゴリズムと certificate から取り出す鍵の
アルゴリズムが異なる場合には、こちらの関数を使用してください。

本 API は、RX651/RX65N のみ対応しています。

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態遷移先は **TSIP 使用可能状態** です。

Reentrant

非対応

5.71 R_TSIP_TIsGeneratePreMasterSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TIsGeneratePreMasterSecret(
    uint32_t *tsip_pre_master_secret
)
```

Parameters

| | | |
|------------------------|-------|---|
| tsip_pre_master_secret | 入力/出力 | TSIP 固有の変換を施した pre-master secret データ 80 バイト出力されます。 |
|------------------------|-------|---|

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |

Description

暗号化された PreMasterSecret を生成するための API です。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.72 R_TSIP_TlsEncryptPreMasterSecretWithRsa2048PublicKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsEncryptPreMasterSecretWithRsa2048PublicKey(
    uint32_t *encrypted_public_key,
    uint32_t *tsip_pre_master_secret,
    uint8_t *encrypted_pre_master_secret
)
```

Parameters

| | | |
|-----------------------------|-------|--|
| encrypted_public_key | 入力 | R_TSIP_TlsCertificateVerification または R_TSIP_TlsCertificateVerificationExtension が出力 する、暗号化された公開鍵データ 140 ワードサイズ |
| tsip_pre_master_secret | 入力 | R_TSIP_TlsGeneratePreMasterSecret が出力する TSIP 固有の変換を施した pre-master secret データ |
| encrypted_pre_master_secret | 入力/出力 | public_key を用いて RSA2048 で暗号化した pre-master secret データ |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |

Description

入力データの公開鍵を用いて、PreMasterSecret を RSA2048 で暗号化するための API です。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.73 R_TSIP_TlsGenerateMasterSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsGenerateMasterSecret(
    uint32_t select_cipher_suite,
    uint32_t *tsip_pre_master_secret,
    uint8_t *client_random,
    uint8_t *server_random,
    uint32_t *tsip_master_secret
)
```

Parameters

| | | | |
|------------------------|-------|--|----|
| select_cipher_suite | 入力 | 選択する cipher_suite | |
| | | R_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA | :0 |
| | | R_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA | :1 |
| | | R_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA256 | :2 |
| | | R_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA256 | :3 |
| | | R_TSIP_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 | :4 |
| | | R_TSIP_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 | :5 |
| | | R_TSIP_TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 | :6 |
| | | R_TSIP_TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 | :7 |
| tsip_pre_master_secret | 入力 | R_TSIP_TlsGeneratePreMasterSecret または R_TSIP_TlsGeneratePreMasterSecretWithEcc P256Key が出力する TSIP 固有の変換を施した pre-master secret データ | |
| client_random | 入力 | ClientHello で通知した乱数値 32 バイト | |
| server_random | 入力 | ServerHello で通知された乱数値 32 バイト | |
| tsip_master_secret | 入力/出力 | TSIP 固有の変換を施した master secret データ 20 ワードで出力されます。 | |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |

Description

暗号化された MasterSecret を生成するための API です。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.74 R_TSIP_TlsGenerateSessionKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsGenerateSessionKey (
    uint32_t select_cipher_suite,
    uint32_t *tsip_master_secret,
    uint8_t *client_random,
    uint8_t *server_random,
    uint8_t* nonce_explicit,
    tsip_hmac_sha_key_index_t *client_mac_key_index,
    tsip_hmac_sha_key_index_t *server_mac_key_index,
    tsip_aes_key_index_t *client_crypto_key_index,
    tsip_aes_key_index_t *server_crypto_key_index,
    uint8_t *client_iv,
    uint8_t *server_iv
)
```

Parameters

| | | | |
|-------------------------|-------|---|----|
| select_cipher_suite | 入力 | 選択する cipher_suite | |
| | | R_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA | :0 |
| | | R_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA | :1 |
| | | R_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA256 | :2 |
| | | R_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA256 | :3 |
| | | R_TSIP_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 | :4 |
| | | R_TSIP_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 | :5 |
| | | R_TSIP_TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 | :6 |
| | | R_TSIP_TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 | :7 |
| tsip_master_secret | 入力 | R_TSIP_TlsGenerateMasterSecret が出力する TSIP 固有の変換を施した master secret データ | |
| client_random | 入力 | ClientHello で通知した乱数値 32 バイト | |
| server_random | 入力 | ServerHello で通知された乱数値 32 バイト | |
| nonce_explicit | 入力 | cipher suite AES128GCM で使用するノンス select_cipher_suite=6-7: 8 バイト | |
| client_mac_key_index | 入力/出力 | クライアント→サーバ通信時の MAC 鍵生成情報 select_cipher_suite=0-5: 17 ワード | |
| server_mac_key_index | 入力/出力 | サーバ→クライアント通信時の MAC 鍵生成情報 select_cipher_suite=0-5: 17 ワード | |
| client_crypto_key_index | 入力/出力 | クライアント→サーバ通信時の AES 共通鍵生成情報 select_cipher_suite=0, 2, 4, 5: 13 ワード select_cipher_suite=1, 3, 6, 7: 17 ワード | |
| server_crypto_key_index | 入力/出力 | サーバ→クライアント通信時の AES 共通鍵生成情報 select_cipher_suite=0, 2, 4, 5: 13 ワード select_cipher_suite=1, 3, 6, 7: 17 ワード | |
| client_iv | 入力/出力 | select_cipher_suite が 0~5 の時に、Client から Server へ送信時に使用する IV を出力します。 (RX651,RX65N で NetX Duo を使用する場合に使用 します) | |
| server_iv | 入力/出力 | それ以外の場合には、何も出力されません select_cipher_suite が 0~5 の時に、Server から受 信時に使用する IV を出力します。 (RX651,RX65N で NetX Duo を使用する場合に使用 | |

します)
それ以外の場合には、何も出力されません

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

TLS 通信の各種鍵を出力するための API です。

client_iv、server_iv 引数には、引数の説明にある場合以外には何も出力されません。

通信で用いる鍵情報は TSIP 内部に保持します。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.75 R_TSIP_TlsGenerateVerifyData

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsGenerateVerifyData(
    uint32_t select_verify_data,
    uint32_t *tsip_master_secret,
    uint8_t *hand_shake_hash,
    uint8_t *verify_data
)
```

Parameters

| | | |
|--------------------|-------|--|
| select_verify_data | 入力 | 選択する Client/Server の種別 R_TSIP_TLS_GENERATE_CLIENT_VERIFY ClientVerifyData の生成 R_TSIP_TLS_GENERATE_SERVER_VERIFY ServerVerifyData の生成 |
| tsip_master_secret | 入力 | R_TSIP_TlsGenerateMasterSecret が出力する TSIP 固有の変換を施した master secret データ |
| hand_shake_hash | 入力 | TLS ハンドシェイクメッセージ全体の SHA256 HASH 値 |
| verify_data | 入力/出力 | Finished メッセージ用の VerifyData |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |

Description

Verify データを生成するための API です。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.76 R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrieves

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrieves(
    uint32_t public_key_type,
    uint8_t *client_random,
    uint8_t *server_random,
    uint8_t *server_ephemeral_ecdh_public_key,
    uint8_t *server_key_exchange_signature,
    uint32_t *encrypted_public_key,
    uint32_t *encrypted_ephemeral_ecdh_public_key
)
```

Parameters

| | | |
|-------------------------------------|-------|---|
| public_key_type | 入力 | 公開鍵の種類 0 : RSA 2048bit, 1 : Reserved, 2 : ECDSA P-256 |
| client_random | 入力 | ClientHello で通知した乱数値(32 バイト) |
| server_random | 入力 | ServerHello で通知された乱数値(32 バイト) |
| server_ephemeral_ecdh_public_key | 入力 | サーバから受け取った ephemeral ECDH 公開鍵 (非圧縮形式) 0padding(24bit) 04(8bit) Qx(256bit) Qy(256bit) |
| server_key_exchange_signature | 入力 | ServerKeyExchange の署名データ 公開鍵: RSA2048bit の場合 256 バイト ECDSA P-256 の場合 64 バイト |
| encrypted_public_key | 入力 | 出力された暗号化された ephemeral ECDH 公開鍵 署名検証のための暗号化された公開鍵 R_TSIP_CertificateVerification から出力された 暗号化された公開鍵情報 公開鍵: RSA2048bit の場合 140 ワードサイズ ECDSA P-256 の場合 24 ワードサイズ |
| encrypted_ephemeral_ecdh_public_key | 入力/出力 | 暗号化された ephemeral ECDH 公開鍵 R_TSIP_TlsGeneratePreMasterSecretWithEccP256 Key に入力する(24 ワードサイズ) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

入力された公開鍵データを用いて、ServerKeyExchange の署名を検証します。署名に成功した場合、R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key で使用する ephemeral ECDH public key を暗号化して出力します。

該当暗号スイート: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256、
 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256、
 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256、
 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.77 R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key(
    uint32_t *encrypted_public_key,
    tsip_tls_p256_ecc_key_index_t *tls_p256_ecc_key_index,
    uint32_t *tsip_pre_master_secret
)
```

Parameters

| | | |
|------------------------|-------|--|
| encrypted_public_key | 入力 | R_TSIP_TlsServersEphemeralEcdhPublicKey Retrieves から出力された暗号化された ephemeral ECDH 公開鍵 |
| tls_p256_ecc_key_index | 入力 | R_TSIP_GenerateTlsP256EccKeyIndex から出力された鍵情報 |
| tsip_pre_master_secret | 入力/出力 | TSIP 固有の変換を施した pre-master secret データ 64 バイト出力されます。 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

入力された鍵データを用いて、暗号化された PreMasterSecret を生成するための API です。

該当暗号スイート: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256、
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256、
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256、
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.78 R_TSIP_GenerateTlsP256EccKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateTlsP256EccKeyIndex(
    tls_p256_ecc_key_index_t *tls_p256_ecc_key_index,
    uint8_t *ephemeral_ecdh_public_key
)
```

Parameters

| | | |
|---------------------------|----|--|
| tls_p256_ecc_key_index | 出力 | PreMasterSecret 生成のための鍵情報 R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key へ 入力 |
| ephemeral_ecdh_public_key | 出力 | ephemeral ECDH 公開鍵 公開鍵 Qx(256bit) 公開鍵 Qy(256bit) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

TLS 連携機能で使用する乱数から 256bit 素体上の楕円曲線暗号のための鍵ペアを生成する API です。

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

Reentrant

非対応

5.79 R_TSIP_GenerateTls13P256EccKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateTls13P256EccKeyIndex(
    tsip_tls13_handle_t *handle,
    e_tsip_tls13_mode_t mode,
    tsip_tls_p256_ecc_key_index_t * key_index,
    uint8_t * ephemeral_ecdh_public_key
)
```

Parameters

| | | |
|---------------------------|-------|--|
| handle | 入力/出力 | 同一セッションを示すハンドル番号(ワーク領域) |
| mode | 入力 | 実施するハンドシェイクプロトコル TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake |
| key_index | 入力/出力 | Ephemeral ECC 秘密鍵生成情報 R_TSIP_Tls13GenerateEcdheSharedSecret の入力 key_index に使用してください。 |
| ephemeral_ecdh_public_key | 入力/出力 | Ephemeral ECDH 公開鍵 公開鍵 Qx(256bit) 公開鍵 Qy(256bit) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |

Description

TLS1.3 連携機能で使用する、乱数から 256bit 素体上の楕円曲線暗号のための鍵ペアを生成するための API です。

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態遷移先は **TSIP 使用可能状態** です。

Reentrant

非対応

5.80 R_TSIP_Tls13GenerateEcdheSharedSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13GenerateEcdheSharedSecret (
    e_tsip_tls13_mode_t mode,
    uint8_t * server_public_key,
    tsip_tls_p256_ecc_key_index_t * key_index,
    tsip_tls13_ephemeral_shared_secret_key_index_t * shared_secret_key_index
)
```

Parameters

| | | |
|-------------------------|-------|---|
| mode | 入力 | 実施するハンドシェイクプロトコル TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake |
| server_public_key | 入力 | サーバから提供される公開鍵 Qx(256bit) Qy(256bit) |
| key_index | 入力 | Ephemeral ECC 秘密鍵生成情報 R_TSIP_GenerateTls13P256EccKeyIndex の出力 key_index を使用してください。 |
| shared_secret_key_index | 入力/出力 | Shared Secret の Ephemeral 鍵生成情報 R_TSIP_Tls13GenerateHandshakeSecret の 入力 shared_secret_key_index に使用してください。 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |

Description

TLS1.3 連携機能で使用する、サーバから提供される公開鍵とあらかじめ演算した秘密鍵を用いて、256bit 素体上の共有鍵である Shared Secret を計算し、鍵生成情報を生成するための API です。

該当暗号スイート: TLS_AES_128_GCM_SHA256, TLS_AES_128_CCM_SHA256

鍵交換の方式: ECDHE NIST P-256

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.81 R_TSIP_Tls13GenerateHandshakeSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13GenerateHandshakeSecret(
    tsip_tls13_ephemeral_shared_secret_key_index_t * shared_secret_key_index,
    tsip_tls13_ephemeral_handshake_secret_key_index_t * handshake_secret_key_index
)
```

Parameters

| | | |
|----------------------------|-------|--|
| shared_secret_key_index | 入力 | Shared Secret の Ephemeral 鍵生成情報 R_TSIP_Tls13GenerateEcdheSharedSecret の出力 shared_secret_key_index を使用してください。 |
| handshake_secret_key_index | 入力/出力 | Handshake Secret の Ephemeral 鍵生成情報 R_TSIP_Tls13GenerateServerHandshakeTrafficKey、 R_TSIP_Tls13GenerateClientHandshakeTrafficKey 及び R_TSIP_Tls13GenerateMasterSecret の入力 handshake_secret_key_index の入力に使用してください。 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |

Description

TLS1.3 連携機能で使用する、Shared Secret の Ephemeral 鍵を用いて、Handshake Secret 鍵生成情報を生成するための API です。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.82 R_TSIP_Tls13GenerateServerHandshakeTrafficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13GenerateServerHandshakeTrafficKey(
    tsip_tls13_handle_t *handle,
    e_tsip_tls13_mode_t mode,
    tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index,
    uint8_t *digest,
    tsip_aes_key_index_t *server_write_key_index,
    tsip_tls13_ephemeral_server_finished_key_index_t *server_finished_key_index
)
```

Parameters

| | | |
|----------------------------|-------|--|
| handle | 入力/出力 | 同一セッションを示すハンドル番号(ワーク領域) |
| mode | 入力 | 実施するハンドシェイクプロトコル TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake |
| handshake_secret_key_index | 入力 | Handshake Secret の Ephemeral 鍵生成情報 R_TSIP_Tls13GenerateHandshakeSecret の 出力 handshake_secret_key_index を使用し てください。 |
| digest | 入力 | SHA256 で演算したメッセージハッシュ (ClientHello ServerHello)のハッシュ値を演算し、 R_TSIP_Sha256Final の出力 digest を使用 してください。 |
| server_write_key_index | 入力/出力 | Server Write Key の Ephemeral 鍵生成情報 R_TSIP_Tls13DecryptInit の入力 server_write_key_index に使用してください。 |
| server_finished_key_index | 入力/出力 | Server Finished Key の Ephemeral 鍵生成情報 R_TSIP_Tls13ServerHandshakeVerification の入力 server_finished_key_index に使用してください。 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |

Description

TLS1.3 連携機能で使用する、R_TSIP_Tls13GenerateHandshakeSecret で出力された Handshake Secret を用いて Server Write Key 及び Server Finished Key の鍵生成情報を生成するための API です。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.83 R_TSIP_Tls13ServerHandshakeVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13ServerHandshakeVerification(
    e_tsip_tls13_mode_t mode,
    tsip_tls13_ephemeral_server_finished_key_index_t * server_finished_key_index,
    uint8_t * digest,
    uint8_t * server_finished,
    uint32_t * verify_data_index
)
```

Parameters

| | | |
|---------------------------|-------|---|
| mode | 入力 | 実施するハンドシェイクプロトコル TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake |
| server_finished_key_index | 入力 | Server Finished Key の Ephemeral 鍵生成情報 R_TSIP_Tls13GenerateServerHandshakeTrafficKey の 出力 server_write_key_index を使用してください。 |
| digest | 入力 | SHA256 で演算したメッセージハッシュ (ClientHello ServerHello EncryptedExtensions CertificateRequest Certificate CertificateVerify) のように、ハンドシェイクメッセージを連結した 値のハッシュ値を演算して入力してください。 R_TSIP_Sha256Final の出力 digest を使用 してください。 |
| server_finished | 入力 | サーバから提供される暗号化された Finished 情報 R_TSIP_Tls13DecryptUpdate/Final により取得した ServerFinished のデータを格納している バッファの先頭アドレスを入力してください。 |
| verify_data_index | 入力/出力 | Server Handshake 検証結果 R_TSIP_Tls13GenerateMasterSecret の入力 verify_data_index に使用してください。 データを出力するバッファの先頭アドレスを入力 してください。必要サイズは 8 ワード(32 バイト) です。 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_VERIFICATION_FAIL | 検証で合格しなかった |

Description

TLS1.3 連携機能で使用する、サーバから提供される Finished の情報を検証するための API です。

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態遷移先は **TSIP 使用可能状態** です。

Reentrant

非対応

5.84 R_TSIP_Tls13GenerateClientHandshakeTrafficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13GenerateClientHandshakeTrafficKey(
    tsip_tls13_handle_t *handle,
    e_tsip_tls13_mode_t mode,
    tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index,
    uint8_t *digest,
    tsip_aes_key_index_t *client_write_key_index,
    tsip_hmac_sha_key_index_t *client_finished_key_index
)
```

Parameters

| | | |
|----------------------------|-------|---|
| handle | 入力/出力 | 同一セッションを示すハンドル番号(ワーク領域) |
| mode | 入力 | 実施するハンドシェイクプロトコル TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake |
| handshake_secret_key_index | 入力 | Handshake Secret の Ephemeral 鍵生成情報 R_TSIP_Tls13GenerateHandshakeSecret の 出力 handshake_secret_key_index を使用し てください。 |
| digest | 入力 | SHA256 で演算したメッセージハッシュ (ClientHello ServerHello)のハッシュ値を演算し、 R_TSIP_Sha256Final の出力 digest を使用 してください。 |
| client_write_key_index | 入力/出力 | Client Write Key の Ephemeral 鍵生成情報 R_TSIP_Tls13EncryptInit の入力 client_write_key_index に使用してください。 |
| client_finished_key_index | 入力/出力 | Client Finished Key の Ephemeral 鍵生成情報 Client Finished の生成に使用してください。 R_TSIP_Sha256HmacGenerateInit の入力 key_index に使用してください。 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |

Description

TLS1.3 連携機能で使用する、R_TSIP_Tls13GenerateHandshakeSecret で出力された Handshake Secret を用いて Client Write Key 及び Client Finished Key の鍵生成情報を生成するための API です。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.85 R_TSIP_Tls13GenerateMasterSecret

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13GenerateMasterSecret(
    tsip_tls13_handle_t *handle,
    e_tsip_tls13_mode_t mode,
    tsip_tls13_ephemeral_handshake_secret_key_index_t *handshake_secret_key_index,
    uint32_t *verify_data_index,
    tsip_tls13_ephemeral_master_secret_key_index_t *master_secret_key_index
)
```

Parameters

| | | |
|----------------------------|-------|---|
| handle | 入力/出力 | 同一セッションを示すハンドル番号(ワーク領域) |
| mode | 入力 | 実施するハンドシェイクプロトコル TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake |
| handshake_secret_key_index | 入力 | Handshake Secret の Ephemeral 鍵生成情報 R_TSIP_Tls13GenerateHandshakeSecret の 出力 handshake_secret_key_index を使用し てください。 |
| verify_data_index | 入力 | Server Handshake 検証結果 R_TSIP_Tls13ServerHandshakeVerification の 出力 verify_data_index を使用してください。 |
| master_secret_key_index | 入力/出力 | Master Secret の Ephemeral 鍵生成情報 R_TSIP_Tls13GenerateApplicationTrafficKey の 入力 master_secret_key_index に使用してください。 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |

Description

TLS1.3 連携機能で使用する、Handshake Secret の Ephemeral 鍵を用いて、Master Secret の Ephemeral 鍵生成情報を生成するための API です。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.86 R_TSIP_Tls13GenerateApplicationTrafficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13GenerateApplicationTrafficKey(
    tsip_tls13_handle_t *handle,
    e_tsip_tls13_mode_t mode,
    tsip_tls13_ephemeral_master_secret_key_index_t * master_secret_key_index,
    uint8_t * digest,
    tsip_tls13_ephemeral_app_secret_key_index_t * server_app_secret_key_index,
    tsip_tls13_ephemeral_app_secret_key_index_t * client_app_secret_key_index,
    tsip_aes_key_index_t * server_write_key_index,
    tsip_aes_key_index_t * client_write_key_index
)
```

Parameters

| | | |
|-----------------------------|-------|---|
| handle | 入力/出力 | 同一セッションを示すハンドル番号(ワーク領域) |
| mode | 入力 | 実施するハンドシェイクプロトコル TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake |
| master_secret_key_index | 入力 | Master Secret の Ephemeral 鍵生成情報 R_TSIP_Tls13GenerateMasterSecret の出力 master_secret_key_index を使用してください。 |
| digest | 入力 | SHA256 で演算したメッセージハッシュ (ClientHello ServerHello EncryptedExtensions CertificateRequest Certificate CertificateVerify ServerFinished) のように、ハンドシェイク メッセージを連結した値のハッシュ値を演算し、 R_TSIP_Sha256Final の出力 digest を使用 してください。 |
| server_app_secret_key_index | 入力/出力 | Server Application Traffic Secret の Ephemeral 鍵生成情報 R_TSIP_Tls13UpdateApplicationTrafficKey の入力 input_app_secret_key_index に使用してください。 |
| client_app_secret_key_index | 入力/出力 | Client Application Traffic Secret の Ephemeral 鍵生成情報 R_TSIP_Tls13UpdateApplicationTrafficKey の入力 input_app_secret_key_index に使用してください。 |
| server_write_key_index | 入力/出力 | Server Write Key の Ephemeral 鍵生成情報 R_TSIP_Tls13DecryptInit の入力 server_write_key_index に使用してください。 |
| client_write_key_index | 入力/出力 | Client Write Key の Ephemeral 鍵生成情報 R_TSIP_Tls13EncryptInit の入力 client_write_key_index に使用してください。 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |

Description

TLS1.3 連携機能で使用する、Master Secret の Ephemeral 鍵を用いて、Application Traffic Secret 鍵生成情報を生成するための API です。併せて、Server Write Key 及び Client Write Key の Ephemeral 鍵生成情報を生成します。

実行前の状態は TSIP 使用可能状態です。

実行後の状態遷移先は TSIP 使用可能状態です。

Reentrant

非対応

5.87 R_TSIP_Tls13UpdateApplicationTrafficKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13UpdateApplicationTrafficKey(
    tsip_tls13_handle_t *handle,
    e_tsip_tls13_mode_t mode,
    e_tsip_tls13_update_key_type_t key_type,
    tsip_tls13_ephemeral_app_secret_key_index_t * input_app_secret_key_index,
    tsip_tls13_ephemeral_app_secret_key_index_t * output_app_secret_key_index,
    tsip_aes_key_index_t * app_write_key_index
)
```

Parameters

| | | |
|-----------------------------|-------|--|
| handle | 入力/出力 | 同一セッションを示すハンドル番号(ワーク領域) |
| mode | 入力 | 実施するハンドシェイクプロトコル TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake |
| key_type | 入力 | 更新する鍵の種類 TSIP_TLS13_UPDATE_SERVER_KEY : Server Application Traffic Secret TSIP_TLS13_UPDATE_CLIENT_KEY : Client Application Traffic Secret |
| input_app_secret_key_index | 入力 | Server / Client Application Traffic Secret の Ephemeral 鍵生成情報 R_TSIP_Tls13GenerateApplicationTrafficKey の 出力 server/clientapp_secret_key_index または R_TSIP_Tls13UpdateApplicationTrafficKey の出力 output_app_secret_key_index のうち、key_type に 指定した鍵の種類に適合した入力を使用 してください。 |
| output_app_secret_key_index | 入力/出力 | Server / Client Application Traffic Secret の Ephemeral 鍵生成情報 key_type に指定した鍵の種類に対応した出力が 得られます。 R_TSIP_Tls13UpdateApplicationTrafficKey の 入力 input_app_secret_key_index に使用して ください。 |
| app_write_key_index | 入力/出力 | Server / Client Write Key の Ephemeral 鍵生成情報 key_type に指定した鍵の種類に対応した出力が 得られます。 Server Write Key は R_TSIP_Tls13DecryptInit の入力 server_write_key_index に使用してください。 Client Write Key は R_TSIP_Tls13EncryptInit の入力 client_write_key_index に使用してください。 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER | 入力データが不正 |

Description

TLS1.3 連携機能で使用する、Application Traffic Secret を用いて、Application Traffic Secret 鍵生成情報と対応する暗号鍵の鍵生成情報を更新するための API です。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.88 R_TSIP_Tls13EncryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13EncryptInit(
    tsip_tls13_handle_t *handle,
    e_tsip_tls13_phase_t phase,
    e_tsip_tls13_mode_t mode,
    e_tsip_tls13_cipher_suite_t cipher_suite,
    tsip_aes_key_index_t *client_write_key_index,
    uint32_t payload_length
)
```

Parameters

| | | |
|------------------------|-------|--|
| handle | 入力/出力 | TLS1.3 用ハンドラ(ワーク領域) |
| phase | 入力 | 通信フェーズ TSIP_TLS13_PHASE_HANDSHAKE : ハンドシェイクフェーズ TSIP_TLS13_PHASE_APPLICATION : アプリケーションフェーズ |
| mode | 入力 | 実施するハンドシェイクプロトコル TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake |
| cipher_suite | 入力 | 暗号スイート TSIP_TLS13_CIPHER_SUITE_AES_128_GCM_SHA256 : TLS_AES_128_GCM_SHA256 TSIP_TLS13_CIPHER_SUITE_AES_128_CCM_SHA256 : TLS_AES_128_CCM_SHA256 |
| client_write_key_index | 入力 | Client Write Key の Ephemeral 鍵生成情報 |
| payload_length | 入力 | 暗号化するデータのバイト長 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |

Description

R_TSIP_Tls13EncryptInit()関数は、TLS1.3 通信データの暗号化を実行する準備を行い、その結果を第一引数"handle"に書き出します。handle は、続く R_TSIP_Tls13EncryptUpdate()関数および R_TSIP_Tls13EncryptFinal()関数で引数として使用されます。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.89 R_TSIP_Tls13EncryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13EncryptUpdate(
    tsip_tls13_handle_t *handle,
    uint8_t *plain,
    uint8_t *cipher,
    uint32_t plain_length
)
```

Parameters

| | | |
|--------------|-------|---------------------|
| handle | 入力/出力 | TLS1.3 用ハンドラ(ワーク領域) |
| plain | 入力 | 平文データ領域 |
| cipher | 入力/出力 | 暗号文データ領域 |
| plain_length | 入力 | 平文データ長 |

Return Values

| | |
|-----------------------------|--------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER | 入力データが不正 |

Description

R_TSIP_Tls13EncryptUpdate()関数は、第二引数“plain”で指定された平文から R_TSIP_Tls13EncryptInit()で指定された“client_write_key_index”を用いて暗号化します。本関数内部で plain の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。暗号化結果は“plain”入力データが 16byte 以上になってから、第三引数で指定された“cipher”に出力します。入力する plain の総データ長は R_TSIP_Tls13EncryptInit()の payload_length で指定してください。本関数の plain_length には、ユーザが本関数を呼ぶ際に入力するデータ長を指定してください。入力値の plain は 16byte で割り切れない場合、パディング処理は関数内部で実施します。

plain と cipher は領域が重ならないように配置してください。また plain と cipher は 4 の倍数の RAM アドレスを指定してください。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.90 R_TSIP_Tls13EncryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13EncryptFinal(
    tsip_tls13_handle_t *handle,
    uint8_t *cipher,
    uint32_t *cipher_length
)
```

Parameters

| | | |
|---------------|-------|---------------------|
| handle | 入力/出力 | TLS1.3 用ハンドラ(ワーク領域) |
| cipher | 入力/出力 | 暗号文データ領域 |
| cipher_length | 入力/出力 | 暗号文データ長 |

Return Values

| | |
|-----------------------------|--------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER | 入力データが不正 |

Description

R_TSIP_Tls13EncryptFinal()関数は、R_TSIP_Tls13EncryptUpdate()で入力した plain のデータ長に 16byte の端数データがある場合、第二引数で指定された“cipher”に端数分の暗号化したデータを出します。このとき、16byte に満たない部分は 0padding されています。cipher は 4 の倍数の RAM アドレスを指定してください。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.91 R_TSIP_Tls13DecryptInit

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13DecryptInit(
    tsip_tls13_handle_t *handle,
    e_tsip_tls13_phase_t phase,
    e_tsip_tls13_mode_t mode,
    e_tsip_tls13_cipher_suite_t cipher_suite,
    tsip_aes_key_index_t *server_write_key_index,
    uint32_t payload_length
)
```

Parameters

| | | |
|------------------------|-------|--|
| handle | 入力/出力 | TLS1.3 用ハンドラ(ワーク領域) |
| phase | 入力 | 通信フェーズ TSIP_TLS13_PHASE_HANDSHAKE : ハンドシェイクフェーズ TSIP_TLS13_PHASE_APPLICATION : アプリケーションフェーズ |
| mode | 入力 | 実施するハンドシェイクプロトコル TSIP_TLS13_MODE_FULL_HANDSHAKE : Full Handshake |
| cipher_suite | 入力 | 暗号スイート TSIP_TLS13_CIPHER_SUITE_AES_128_GCM_SHA256 : TLS_AES_128_GCM_SHA256 TSIP_TLS13_CIPHER_SUITE_AES_128_CCM_SHA256 : TLS_AES_128_CCM_SHA256 |
| server_write_key_index | 入力 | Server Write Key の Ephemeral 鍵生成情報 |
| payload_length | 入力 | 復号するデータのバイト長 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |

Description

R_TSIP_Tls13DecryptInit()関数は、TLS1.3 通信データの復号を実行する準備を行い、その結果を第一引数"handle"に書き出します。handle は、続く R_TSIP_Tls13DecryptUpdate 関数および R_TSIP_Tls13DecryptFinal()関数で引数として使用されます。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.92 R_TSIP_Tls13DecryptUpdate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13DecryptUpdate(
    tsip_tls13_handle_t *handle,
    uint8_t *cipher,
    uint8_t *plain,
    uint32_t cipher_length
)
```

Parameters

| | | |
|---------------|-------|---------------------|
| handle | 入力/出力 | TLS1.3 用ハンドラ(ワーク領域) |
| cipher | 入力 | 暗号文データ領域 |
| plain | 入力/出力 | 平文データ領域 |
| cipher_length | 入力 | 暗号文データ長 |

Return Values

| | |
|-----------------------------|--------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER | 入力データが不正 |

Description

R_TSIP_Tls13DecryptUpdate()関数は、第二引数“cipher”で指定された暗号文から R_TSIP_Tls13DecryptInit()で指定された“server_write_key_index”を用いて復号します。本関数内部で cipher の入力値が 16byte を超えるまでユーザが入力したデータをバッファリングします。暗号化結果は“cipher”入力データが 16byte 以上になってから、第三引数で指定された“plain”に出力します。入力する cipher の総データ長は R_TSIP_Tls13DecryptInit()の payload_length で指定してください。本関数の cipher_length には、ユーザが本関数を呼ぶ際に入力するデータ長を指定してください。入力値の cipher は 16byte で割り切れない場合、パディング処理は関数内部で実施します。

cipher と plain は領域が重ならないように配置してください。また cipher と plain は 4 の倍数の RAM アドレスを指定してください。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.93 R_TSIP_Tls13DecryptFinal

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13DecryptFinal(
    tsip_tls13_handle_t *handle,
    uint8_t *plain,
    uint32_t *plain_length
)
```

Parameters

| | | |
|--------------|-------|---------------------|
| handle | 入力/出力 | TLS1.3 用ハンドラ(ワーク領域) |
| plain | 入力/出力 | 平文データ領域 |
| plain_length | 入力/出力 | 平文データ長 |

Return Values

| | |
|-----------------------------|--------------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |
| TSIP_ERR_PARAMETER | 入力データが不正 |
| TSIP_ERR_AUTHENTICATION | 認証が失敗 |

Description

R_TSIP_Tls13DecryptFinal()関数は、R_TSIP_Tls13DecryptUpdate()で入力した cipher のデータ長に 16byte の端数データがある場合、第二引数で指定された“plain”に端数分の復号したデータを出力します。このとき、16byte に満たない部分は 0padding されています。plain は 4 の倍数の RAM アドレスを指定してください。

実行前の状態は **TSIP 使用可能状態**です。

実行後の状態遷移先は **TSIP 使用可能状態**です。

Reentrant

非対応

5.94 R_TSIP_Tls13CertificateVerifyGenerate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13CertificateVerifyGenerate(
    uint32_t * key_index,
    e_tsip_tls13_signature_scheme_type_t signature_scheme,
    uint8_t * digest,
    uint8_t * certificate_verify,
    uint32_t * certificate_verify_len
)
```

Parameters

| | | |
|------------------------|-------|---|
| key_index | 入力 | ECC P-256 秘密鍵ユーザ鍵生成情報 R_TSIP_GenerateEccP256PrivateKeyIndex の 出力 key_index を使用してください。 引数は uint32_t * でキャストしてから入力して ください。 |
| signature_scheme | 入力 | 使用する署名アルゴリズム TSIP_TLS13_SIGNATURE_SCHEME_ECDSA_SECP256R1_SHA256 : ecdsa_secp256r1_sha256 |
| digest | 入力 | SHA256 で演算したメッセージハッシュ (ClientHello ServerHello EncryptedExtensions CertificateRequest Certificate CertificateVerify ServerFinished Certificate) のように、 ハンドシェイクメッセージを連結した値の ハッシュ演算をし、R_TSIP_Sha256Final の 出力 digest を使用してください。 |
| certificate_verify | 入力/出力 | CertificateVerify データは RFC8446 4.4.3 章の CertificateVerify の 形式で出力されます。データ格納に十分な領域を 確保してください。 |
| certificate_verify_len | 入力/出力 | certificate_verify のバイト長 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER | 入力データが不正 |

Description

TLS1.3 連携機能で使用する、サーバに送信する CertificateVerify を生成するための API です。署名アルゴリズムは ECDSA P-256、ハッシュアルゴリズムは SHA256 を使用します。

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態遷移先は **TSIP 使用可能状態** です。

Reentrant

非対応

5.95 R_TSIP_Tls13CertificateVerifyVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_Tls13CertificateVerifyVerification(
    uint32_t * key_index,
    e_tsip_tls13_signature_scheme_type_t signature_scheme,
    uint8_t * digest,
    uint8_t * certificate_verify,
    uint32_t certificate_verify_len
)
```

Parameters

| | | |
|------------------------|----|--|
| key_index | 入力 | 暗号化された公開鍵 R_TSIP_GenerateEccP256PublicKeyIndex の出力 key_index を使用してください。 引数は uint32_t * でキャストしてから入力してください。 |
| signature_scheme | 入力 | 使用する署名アルゴリズム TSIP_TLS13_SIGNATURE_SCHEME_ECDSA_SECP256R1_SHA256 : ecdsa_secp256r1_sha256 |
| digest | 入力 | SHA256 で演算したメッセージハッシュ (ClientHello ServerHello EncryptedExtensions CertificateRequest Certificate)のように、 ハンドシェイクメッセージを連結した値のハッシュ値を演算して入力してください。 R_TSIP_Sha256Final の出力 digest を使用してください。 |
| certificate_verify | 入力 | CertificateVerify RFC8446 4.4.3 章の CertificateVerify の形式のデータを格納しているバッファの先頭アドレスを入力してください。 |
| certificate_verify_len | 入力 | certificate_verify のバイト長 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_FAIL: | 内部エラーが発生、もしくは署名検証失敗 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER | 入力データが不正 |

Description

TLS1.3 連携機能で使用する、サーバから受信した CertificateVerify を検証するための API です。署名アルゴリズムは ECDSA P-256、ハッシュアルゴリズムは SHA256 を使用します。

実行前の状態は **TSIP 使用可能状態** です。

実行後の状態遷移先は **TSIP 使用可能状態** です。

Reentrant

非対応

5.96 R_TSIP_GenerateEccP192PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP192PublicKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|--------------------------------------|----|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC をつけられた ECC P-192 公開鍵 |
| key_index | 出力 | ECC P-192 公開鍵ユーザ鍵生成情報 |
| key_index->value.key_management_info | | : 鍵管理情報 |
| key_index->value.key_q | | : ECC P-192 公開鍵 Q(平文) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-192 公開鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|----------------------|-------|------------------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | 0 padding | | ECC P-192 公開鍵 Qx | |
| 16-31 | ECC P-192 公開鍵 Qx(続き) | | | |
| 32-47 | 0 padding | | ECC P-192 公開鍵 Qy | |
| 48-63 | ECC P-192 公開鍵 Qy(続き) | | | |
| 64-79 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key 生成方法、key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.97 R_TSIP_GenerateEccP224PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP224PublicKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|--------------------------------------|----|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC をつけられた ECC P-224 公開鍵 |
| key_index | 出力 | ECC P-224 公開鍵ユーザ鍵生成情報 |
| key_index->value.key_management_info | | : 鍵管理情報 |
| key_index->value.key_q | | : ECC P-224 公開鍵 Q(平文) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-224 公開鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|----------------------|------------------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | 0 padding | ECC P-224 公開鍵 Qx | | |
| 16-31 | ECC P-224 公開鍵 Qx(続き) | | | |
| 32-47 | 0 padding | ECC P-224 公開鍵 Qy | | |
| 48-63 | ECC P-224 公開鍵 Qy(続き) | | | |
| 64-79 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key 生成方法、key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.98 R_TSIP_GenerateEccP256PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP256PublicKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|--------------------------------------|----|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC をつけられた ECC P-256 公開鍵 |
| key_index | 出力 | ECC P-256 公開鍵ユーザ鍵生成情報 |
| key_index->value.key_management_info | | : 鍵管理情報 |
| key_index->value.key_q | | : ECC P-256 公開鍵 Q(平文) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-256 公開鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-31 | ECC P-256 公開鍵 Qx | | | |
| 32-63 | ECC P-256 公開鍵 Qy | | | |
| 64-79 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key 生成方法、key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.99 R_TSIP_GenerateEccP384PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP384PublicKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|--------------------------------------|----|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC をつけられた ECC P-384 公開鍵 |
| key_index | 出力 | ECC P-384 公開鍵ユーザ鍵生成情報 |
| key_index->value.key_management_info | | : 鍵管理情報 |
| key_index->value.key_q | | : ECC P-384 公開鍵 Q(平文) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-384 公開鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|--------|------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-47 | ECC P-384 公開鍵 Qx | | | |
| 48-95 | ECC P-256 公開鍵 Qy | | | |
| 96-111 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key 生成方法、key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.100 R_TSIP_GenerateEccP192PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP192PrivateKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|----|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられた ECC P-192 秘密鍵 |
| key_index | 出力 | ECC P-192 秘密鍵ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-192 秘密鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|-------|-------------------|-------|---------------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | 0 padding | | ECC P-192 秘密鍵 | |
| 16-31 | ECC P-192 秘密鍵(続き) | | | |
| 32-47 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key の生成方法、key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.101 R_TSIP_GenerateEccP224PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP224PrivateKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|----|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられた ECC P-224 秘密鍵 |
| key_index | 出力 | ECC P-224 秘密鍵ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-224 秘密鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|-------|-------------------|---------------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | 0 padding | ECC P-224 秘密鍵 | | |
| 16-31 | ECC P-224 秘密鍵(続き) | | | |
| 32-47 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key の生成方法、key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.102 R_TSIP_GenerateEccP256PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP256PrivateKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|----|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられた ECC P-256 秘密鍵 |
| key_index | 出力 | ECC P-256 秘密鍵ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-256 秘密鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|-------|---------------|-------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-31 | ECC P-256 秘密鍵 | | | |
| 32-47 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key の生成方法、key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.103 R_TSIP_GenerateEccP384PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP384PrivateKeyIndex(
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|----|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられた ECC P-384 秘密鍵 |
| key_index | 出力 | ECC P-384 秘密鍵ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-384 秘密鍵ユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|-------|---------------|-------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-47 | ECC P-384 秘密鍵 | | | |
| 48-63 | MAC | | | |

encrypted_key と key_index は領域が重ならないように配置してください。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

encrypted_provisioning_key, iv および encrypted_key の生成方法、key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.104 R_TSIP_GenerateEccP192RandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP192RandomKeyIndex(
    tsip_ecc_key_pair_index_t *key_pair_index
)
```

Parameters

| | | |
|--|----|------------------------------|
| key_pair_index | 出力 | ECC P-192 公開鍵、秘密鍵ペアのユーザ鍵生成情報 |
| key_pair_index->public | | : ECC P-192 公開鍵ユーザ鍵生成情報 |
| key_pair_index->public.value.key_management_info | | : 鍵管理情報 |
| key_pair_index->public.value.key_q | | : ECC P-192 公開鍵 Q(平文) |
| key_pair_index->private | | : ECC P-192 秘密鍵ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-192 公開鍵、秘密鍵ペアのユーザ鍵生成情報を出力するための API です。本 API は TSIP 内部にて乱数値からユーザ鍵を生成します。従ってユーザ鍵の入力は不要です。本 API が出力するユーザ鍵生成情報を使用しデータを暗号処理することにより、データのデッドコピーを防ぐことができます。key_pair_index->public に公開鍵の鍵生成情報、key_pair_index->private に秘密鍵の鍵生成情報を生成します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_pair_index->public ならびに key_pair_index->private 使用方法については「[7 章 鍵データの運用](#)」を参照してください。key_pair_index->public は R_TSIP_GenerateEccP192PublicKeyIndex() から出力される公開鍵のユーザ鍵生成情報、key_pair_index->private は R_TSIP_GenerateEccP192PrivateKeyIndex() から出力される秘密鍵のユーザ鍵生成情報と同様の運用になります。

Reentrant

非対応

5.105 R_TSIP_GenerateEccP224RandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP224RandomKeyIndex(
    tsip_ecc_key_pair_index_t *key_pair_index
)
```

Parameters

| | | |
|--|---------|------------------------------|
| key_pair_index | 出力 | ECC P-224 公開鍵、秘密鍵ペアのユーザ鍵生成情報 |
| key_pair_index->public | | : ECC P-224 公開鍵ユーザ鍵生成情報 |
| key_pair_index->private | | : ECC P-224 秘密鍵ユーザ鍵生成情報 |
| key_pair_index->public.value.key_management_info | : 鍵管理情報 | |
| key_pair_index->public.value.key_q | | : ECC P-224 公開鍵 Q(平文) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-224 公開鍵、秘密鍵ペアのユーザ鍵生成情報を出力するための API です。本 API は TSIP 内部にて乱数値からユーザ鍵を生成します。従ってユーザ鍵の入力は不要です。本 API が出力するユーザ鍵生成情報を使用しデータを暗号処理することにより、データのデッドコピーを防ぐことができます。key_pair_index->public に公開鍵の鍵生成情報、key_pair_index->private に秘密鍵の鍵生成情報を生成します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_pair_index->public ならびに key_pair_index->private 使用方法については「[7 章 鍵データの運用](#)」を参照してください。key_pair_index->public は R_TSIP_GenerateEccP224PublicKeyIndex() から出力される公開鍵のユーザ鍵生成情報、key_pair_index->private は R_TSIP_GenerateEccP224PrivateKeyIndex() から出力される秘密鍵のユーザ鍵生成情報と同様の運用になります。

Reentrant

非対応

5.106 R_TSIP_GenerateEccP256RandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP256RandomKeyIndex(
    tsip_ecc_key_pair_index_t *key_pair_index
)
```

Parameters

| | | |
|--|----|------------------------------|
| key_pair_index | 出力 | ECC P-256 公開鍵、秘密鍵ペアのユーザ鍵生成情報 |
| key_pair_index->public | | : ECC P-256 公開鍵ユーザ鍵生成情報 |
| key_pair_index->public.value.key_management_info | | : 鍵管理情報 |
| key_pair_index->public.value.key_q | | : ECC P-256 公開鍵 Q(平文) |
| key_pair_index->private | | : ECC P-256 秘密鍵ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-256 公開鍵、秘密鍵ペアのユーザ鍵生成情報を出力するための API です。本 API は TSIP 内部にて乱数値からユーザ鍵を生成します。従ってユーザ鍵の入力は不要です。本 API が出力するユーザ鍵生成情報を使用しデータを暗号処理することにより、データのデッドコピーを防ぐことができます。key_pair_index->public に公開鍵の鍵生成情報、key_pair_index->private に秘密鍵の鍵生成情報を生成します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_pair_index->public ならびに key_pair_index->private 使用方法については「[7章 鍵データの運用](#)」を参照してください。key_pair_index->public は R_TSIP_GenerateEccP256PublicKeyIndex() から出力される公開鍵のユーザ鍵生成情報、key_pair_index->private は R_TSIP_GenerateEccP256PrivateKeyIndex() から出力される秘密鍵のユーザ鍵生成情報と同様の運用になります。

Reentrant

非対応

5.107 R_TSIP_GenerateEccP384RandomKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateEccP384RandomKeyIndex(
    tsip_ecc_key_pair_index_t *key_pair_index
)
```

Parameters

| | | |
|--|----|------------------------------|
| key_pair_index | 出力 | ECC P-384 公開鍵、秘密鍵ペアのユーザ鍵生成情報 |
| key_pair_index->public | | : ECC P-384 公開鍵ユーザ鍵生成情報 |
| key_pair_index->public.value.key_management_info | | : 鍵管理情報 |
| key_pair_index->public.value.key_q | | : ECC P-384 公開鍵 Q(平文) |
| key_pair_index->private | | : ECC P-384 秘密鍵ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-384 公開鍵、秘密鍵ペアのユーザ鍵生成情報を出力するための API です。本 API は TSIP 内部にて乱数値からユーザ鍵を生成します。従ってユーザ鍵の入力は不要です。本 API が出力するユーザ鍵生成情報を使用しデータを暗号処理することにより、データのデッドコピーを防ぐことができます。key_pair_index->public に公開鍵の鍵生成情報、key_pair_index->private に秘密鍵の鍵生成情報を生成します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_pair_index->public ならびに key_pair_index->private 使用方法については「[7章 鍵データの運用](#)」を参照してください。key_pair_index->public は R_TSIP_GenerateEccP384PublicKeyIndex() から出力される公開鍵のユーザ鍵生成情報、key_pair_index->private は R_TSIP_GenerateEccP384PrivateKeyIndex() から出力される秘密鍵のユーザ鍵生成情報と同様の運用になります。

Reentrant

非対応

5.108 R_TSIP_GenerateSha1HmacKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateSha1HmacKeyIndex (
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|-------|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられたユーザ鍵 |
| key_index | 入力/出力 | ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

SHA1-HMAC のユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|--------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | SHA1-HMAC 160bit 鍵 | | | |
| 16-31 | | | | |
| | 0 padding | | | |
| 32-47 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

encrypted_provisioning_key, iv, encrypted_key の生成方法および key_index の使用方法については「[7章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.109 R_TSIP_GenerateSha256HmacKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_GenerateSha256HmacKeyIndex (
    uint8_t *encrypted_provisioning_key,
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

| | | |
|----------------------------|-------|--------------------------------|
| encrypted_provisioning_key | 入力 | DLM でラッピングされた provisioning key |
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 暗号化され MAC を付けられたユーザ鍵 |
| key_index | 入力/出力 | ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

SHA256-HMAC のユーザ鍵生成情報を出力するための API です。

encrypted_key には provisioning key で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|----------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | SHA256-HMAC 256bit 鍵 | | | |
| 16-31 | | | | |
| 32-47 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

encrypted_provisioning_key, iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.110 R_TSIP_UpdateEccP192PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP192PublicKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|--------------------------------------|----|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた公開鍵 |
| key_index | 出力 | ECC P-192 公開鍵のユーザ鍵生成情報 |
| key_index->value.key_management_info | | : 鍵管理情報 |
| key_index->value.key_q | | : ECC P-192 公開鍵 Q(平文) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-192 公開鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|----------------------|-------|------------------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | 0 padding | | ECC P-192 公開鍵 Qx | |
| 16-31 | ECC P-192 公開鍵 Qx(続き) | | | |
| 32-47 | 0 padding | | ECC P-192 公開鍵 Qy | |
| 48-63 | ECC P-192 公開鍵 Qy(続き) | | | |
| 64-79 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.111 R_TSIP_UpdateEccP224PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP224PublicKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|--------------------------------------|----|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた公開鍵 |
| key_index | 出力 | ECC P-224 公開鍵のユーザ鍵生成情報 |
| key_index->value.key_management_info | | : 鍵管理情報 |
| key_index->value.key_q | | : ECC P-224 公開鍵 Q(平文) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-224 公開鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|----------------------|------------------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | 0 padding | ECC P-224 公開鍵 Qx | | |
| 16-31 | ECC P-224 公開鍵 Qx(続き) | | | |
| 32-47 | 0 padding | ECC P-224 公開鍵 Qy | | |
| 48-63 | ECC P-224 公開鍵 Qy(続き) | | | |
| 64-79 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.112 R_TSIP_UpdateEccP256PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP256PublicKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|--------------------------------------|----|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた公開鍵 |
| key_index | 出力 | ECC P-256 公開鍵のユーザ鍵生成情報 |
| key_index->value.key_management_info | | : 鍵管理情報 |
| key_index->value.key_q | | : ECC P-256 公開鍵 Q(平文) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-256 公開鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-31 | ECC P-256 公開鍵 Qx | | | |
| 32-63 | ECC P-256 公開鍵 Qy | | | |
| 64-79 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.113 R_TSIP_UpdateEccP384PublicKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP384PublicKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|--------------------------------------|----|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた公開鍵 |
| key_index | 出力 | ECC P-384 公開鍵のユーザ鍵生成情報 |
| key_index->value.key_management_info | | : 鍵管理情報 |
| key_index->value.key_q | | : ECC P-384 公開鍵 Q(平文) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-384 公開鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|--------|------------------|-------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-47 | ECC P-384 公開鍵 Qx | | | |
| 48-95 | ECC P-384 公開鍵 Qy | | | |
| 96-111 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.114 R_TSIP_UpdateEccP192PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP192PrivateKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|----|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた秘密鍵 |
| key_index | 出力 | ECC P-192 秘密鍵のユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-192 秘密鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|-------------------|-------|---------------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | 0 padding | | ECC P-192 秘密鍵 | |
| 16-31 | ECC P-192 秘密鍵(続き) | | | |
| 32-47 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.115 R_TSIP_UpdateEccP224PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP224PrivateKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|----|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた秘密鍵 |
| key_index | 出力 | ECC P-224 秘密鍵のユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-224 秘密鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|-------------------|---------------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | 0 padding | ECC P-224 秘密鍵 | | |
| 16-31 | ECC P-224 秘密鍵(続き) | | | |
| 32-47 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.116 R_TSIP_UpdateEccP256PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP256PrivateKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|----|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた秘密鍵 |
| key_index | 出力 | ECC P-256 秘密鍵のユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-256 秘密鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|-------|---------------|-------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-31 | ECC P-256 秘密鍵 | | | |
| 32-47 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.117 R_TSIP_UpdateEccP384PrivateKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateEccP384PrivateKeyIndex(
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|----|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられた秘密鍵 |
| key_index | 出力 | ECC P-384 秘密鍵のユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

ECC P-384 秘密鍵の鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| | | | | |
|-------|---------------|-------|-------|-------|
| byte | 128 bit | | | |
| | 32bit | 32bit | 32bit | 32bit |
| 0-47 | ECC P-384 秘密鍵 | | | |
| 48-63 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.118 R_TSIP_UpdateSha1HmacKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateSha1HmacKeyIndex (
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|-------|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられたユーザ鍵 |
| key_index | 入力/出力 | ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

SHA1-HMAC のユーザ鍵生成情報を更新するための API です。
encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|--------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | SHA1-HMAC 160bit 鍵 | | | |
| 16-31 | | | | |
| | 0 padding | | | |
| 32-47 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。
実行後は **TSIP 使用可能状態**に遷移します。
encrypted_provisioning_key, iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.119 R_TSIP_UpdateSha256HmacKeyIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_UpdateSha256HmacKeyIndex (
    uint8_t *iv,
    uint8_t *encrypted_key,
    tsip_hmac_sha_key_index_t *key_index
)
```

Parameters

| | | |
|---------------|-------|-----------------------------|
| iv | 入力 | encrypted_key 生成時に使用した初期ベクタ |
| encrypted_key | 入力 | 鍵更新用鍵束で暗号化され MAC を付けられたユーザ鍵 |
| key_index | 入力/出力 | ユーザ鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

SHA256-HMAC のユーザ鍵生成情報を更新するための API です。

encrypted_key には鍵更新用鍵束で暗号化した以下のフォーマットのデータを入力してください。

| byte | 128 bit | | | |
|-------|----------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | SHA256-HMAC 256bit 鍵 | | | |
| 16-31 | | | | |
| 32-47 | MAC | | | |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後は **TSIP 使用可能状態** に遷移します。

encrypted_provisioning_key, iv, encrypted_key の生成方法および key_index の使用方法については「[7 章 鍵データの運用](#)」を参照してください。

Reentrant

非対応

5.120 R_TSIP_EcdsaP192SignatureGenerate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP192SignatureGenerate(
    tsip_ecdsa_byte_data_t *message_hash,
    tsip_ecdsa_byte_data_t *signature,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|---------------------------|----|---|
| message_hash | 入力 | 署名を付けるメッセージまたはハッシュ値情報 |
| message_hash->pdata | | : メッセージまたはハッシュ値を格納している配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(メッセージの場合のみ指定) |
| message_hash->data_type | | : message_hash のデータ種別を選択 メッセージ: 0 ハッシュ値: 1 |
| signature | 出力 | 署名文格納先情報 |
| signature->pdata | | : 署名文を格納する配列のポインタを指定 署名形式は"0 padding(64bit) 署名 r(192bit) 0 padding(64bit) 署名 s(192bit)" |
| signature->data_length | | : データ長(バイト単位) |
| key_index | 入力 | 鍵データ領域 : ECC P-192 秘密鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| 上記の Return Values 以外 | ハッシュ演算を行う内部関数からの戻り値 |

Description

第一引数"message_hash->data_type"でメッセージを指定した場合、第一引数"message_hash->pdata"に入力されたメッセージ文を SHA-256 ハッシュ計算し、第三引数"key_index"に入力された秘密鍵ユーザ鍵生成情報から、ECDSA P-192 に従い署名文を第二引数"signature"に書き出します。

第一引数"message_hash->data_type"でハッシュ値を指定した場合、第一引数"message_hash->pdata"に入力された SHA-256 ハッシュ値の先頭 24 バイトに対して、第三引数"key_index"に入力された秘密鍵ユーザ鍵生成情報から、ECDSA P-192 に従い署名文を第二引数"signature"に書き出します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.121 R_TSIP_EcdsaP224SignatureGenerate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP224SignatureGenerate(
    tsip_ecdsa_byte_data_t *message_hash,
    tsip_ecdsa_byte_data_t *signature,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|---------------------------|----|---|
| message_hash | 入力 | 署名を付けるメッセージまたはハッシュ値情報 |
| message_hash->pdata | | : メッセージまたはハッシュ値を格納している配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(メッセージの場合のみ指定) |
| message_hash->data_type | | : message_hash のデータ種別を選択 メッセージ: 0 ハッシュ値: 1 |
| signature | 出力 | 署名文格納先情報 |
| signature->pdata | | : 署名文を格納する配列のポインタを指定 署名形式は"0 padding(32bit) 署名 r(224bit) 0 padding(32bit) 署名 s(224bit)" |
| signature->data_length | | : データ長(バイト単位) |
| key_index | 入力 | 鍵データ領域 : ECC P-224 秘密鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| 上記の Return Values 以外 | ハッシュ演算を行う内部関数からの戻り値 |

Description

第一引数"message_hash->data_type"でメッセージを指定した場合、第一引数"message_hash->pdata"に入力されたメッセージ文を SHA-256 ハッシュ計算し、第三引数"key_index"に入力された秘密鍵ユーザ鍵生成情報から、ECDSA P-224 に従い署名文を第二引数"signature"に書き出します。

第一引数"message_hash->data_type"でハッシュ値を指定した場合、第一引数"message_hash->pdata"に入力された SHA-256 ハッシュ値の先頭 28 バイトに対して、第三引数"key_index"に入力された秘密鍵ユーザ鍵生成情報から、ECDSA P-224 に従い署名文を第二引数"signature"に書き出します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.122 R_TSIP_EcdsaP256SignatureGenerate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP256SignatureGenerate(
    tsip_ecdsa_byte_data_t *message_hash,
    tsip_ecdsa_byte_data_t *signature,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|---------------------------|----|---|
| message_hash | 入力 | 署名を付けるメッセージまたはハッシュ値情報 |
| message_hash->pdata | | : メッセージまたはハッシュ値を格納している配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(メッセージの場合のみ指定) |
| message_hash->data_type | | : message_hash のデータ種別を選択 メッセージ: 0 ハッシュ値: 1 |
| signature | 出力 | 署名文格納先情報 |
| signature->pdata | | : 署名文を格納する配列のポインタを指定 署名形式は"署名 r(256bit) 署名 s(256bit)" |
| signature->data_length | | : データ長(バイト単位) |
| key_index | 入力 | 鍵データ領域 : ECC P-256 秘密鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| 上記の Return Values 以外 | ハッシュ演算を行う内部関数からの戻り値 |

Description

第一引数"message_hash->data_type"でメッセージを指定した場合、第一引数"message_hash->pdata"に入力されたメッセージ文を SHA-256 ハッシュ計算し、第三引数"key_index"に入力された秘密鍵ユーザ鍵生成情報から、ECDSA P-256 に従い署名文を第二引数"signature"に書き出します。

第一引数"message_hash->data_type"でハッシュ値を指定した場合、第一引数"message_hash->pdata"に入力された SHA-256 ハッシュ値の 32 バイト全てに対して、第三引数"key_index"に入力された秘密鍵ユーザ鍵生成情報から、ECDSA P-256 に従い署名文を第二引数"signature"に書き出します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

key_index の生成方法については「[7 章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.123 R_TSIP_EcdsaP384SignatureGenerate

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP384SignatureGenerate(
    tsip_ecdsa_byte_data_t *message_hash,
    tsip_ecdsa_byte_data_t *signature,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|---------------------------|----|---|
| message_hash | 入力 | 署名を付けるハッシュ値情報 |
| message_hash->pdata | | : ハッシュ値を格納している配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(不使用) |
| message_hash->data_type | | : 1 のみ指定可能 |
| signature | 出力 | 署名文格納先情報 |
| signature->pdata | | : 署名文を格納する配列のポインタを指定 署名形式は"署名 r(384bit) 署名 s(384bit)" |
| signature->data_length | | : データ長(バイト単位) |
| key_index | 入力 | 鍵データ領域 : ECC P-384 秘密鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |

Description

第一引数"message_hash->pdata"に入力された SHA-384 ハッシュ値の 48 バイト全てに対して、第三引数"key_index"に入力された秘密鍵ユーザ鍵生成情報から、ECDSA P-384 に従い署名文を第二引数"signature"に書き出します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7 章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.124 R_TSIP_EcdsaP192SignatureVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP192SignatureVerification(
    tsip_ecdsa_byte_data_t *signature,
    tsip_ecdsa_byte_data_t *message_hash,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|---------------------------|----|--|
| signature | 入力 | 検証する署名文情報 |
| signature->pdata | | : 署名文を格納している配列のポインタを指定 署名形式は"0 padding(64bit) 署名 r(192bit) 0 padding(64bit) 署名 s(192bit)" |
| signature->data_length | | : データ長(バイト単位)を指定(不使用) |
| message_hash | 入力 | 検証するメッセージ文またはハッシュ値情報 |
| message_hash->pdata | | : メッセージまたはハッシュ値を格納している 配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(メッセージの場合のみ指定) |
| message_hash->data_type | | : message_hash のデータ種別を選択 メッセージ: 0 ハッシュ値: 1 |
| key_index | 入力 | 鍵データ領域 : ECC P-192 公開鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生、もしくは署名検証失敗 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| 上記の Return Values 以外 | ハッシュ演算を行う内部関数からの戻り値 |

Description

第二引数"message_hash->data_type"でメッセージを指定した場合、第二引数"message_hash->pdata"に入力されたメッセージ文を SHA-256 ハッシュ計算し、第三引数"key_index"に入力された公開鍵ユーザ鍵生成情報から、ECDSA P-192 に従い第一引数"signature"に入力された署名文との検証をします。

第二引数"message_hash->data_type"でハッシュ値を指定した場合、第二引数"message_hash->pdata"に入力された SHA-256 ハッシュ値の先頭 24 バイトに対して、第三引数"key_index"に入力された公開鍵ユーザ鍵生成情報から、ECDSA P-192 に従い第一引数"signature"に入力された署名文との検証をします。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.125 R_TSIP_EcdsaP224SignatureVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP224SignatureVerification(
    tsip_ecdsa_byte_data_t *signature,
    tsip_ecdsa_byte_data_t *message_hash,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|---------------------------|----|--|
| signature | 入力 | 検証する署名文情報 |
| signature->pdata | | : 署名文を格納している配列のポインタを指定 署名形式は"0 padding(32bit) 署名 r(224bit) 0 padding(32bit) 署名 s(224bit)" |
| signature->data_length | | : データ長(バイト単位)を指定(不使用) |
| message_hash | 入力 | 検証するメッセージ文またはハッシュ値情報 |
| message_hash->pdata | | : メッセージまたはハッシュ値を格納している 配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(メッセージの場合のみ指定) |
| message_hash->data_type | | : message_hash のデータ種別を選択 メッセージ: 0 ハッシュ値: 1 |
| key_index | 入力 | 鍵データ領域 : ECC P-224 公開鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生、もしくは署名検証失敗 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| 上記の Return Values 以外 | ハッシュ演算を行う内部関数からの戻り値 |

Description

第二引数"message_hash->data_type"でメッセージを指定した場合、第二引数"message_hash->pdata"に入力されたメッセージ文を SHA-256 ハッシュ計算し、第三引数"key_index"に入力された公開鍵ユーザ鍵生成情報から、ECDSA P-224 に従い第一引数"signature"に入力された署名文との検証をします。

第二引数"message_hash->data_type"でハッシュ値を指定した場合、第二引数"message_hash->pdata"に入力された SHA-256 ハッシュ値の先頭 28 バイトに対して、第三引数"key_index"に入力された公開鍵ユーザ鍵生成情報から、ECDSA P-224 に従い第一引数"signature"に入力された署名文との検証をします。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.126 R_TSIP_EcdsaP256SignatureVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP256SignatureVerification(
    tsip_ecdsa_byte_data_t *signature,
    tsip_ecdsa_byte_data_t *message_hash,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|---------------------------|----|---|
| signature | 入力 | 検証する署名文情報 |
| signature->pdata | | : 署名文を格納している配列のポインタを指定 署名形式は"署名 r(256bit) 署名 s(256bit)" |
| signature->data_length | | : データ長(バイト単位)を指定(不使用) |
| message_hash | 入力 | 検証するメッセージ文またはハッシュ値情報 |
| message_hash->pdata | | : メッセージまたはハッシュ値を格納している 配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(メッセージの場合のみ指定) |
| message_hash->data_type | | : message_hash のデータ種別を選択 メッセージ: 0 ハッシュ値: 1 |
| key_index | 入力 | 鍵データ領域 : ECC P-256 公開鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生、もしくは署名検証失敗 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| 上記の Return Values 以外 | ハッシュ演算を行う内部関数からの戻り値 |

Description

第二引数"message_hash->data_type"でメッセージを指定した場合、第二引数"message_hash->pdata"に入力されたメッセージ文を SHA-256 ハッシュ計算し、第三引数"key_index"に入力された公開鍵ユーザ鍵生成情報から、ECDSA P-256 に従い第一引数"signature"に入力された署名文との検証をします。

第二引数"message_hash->data_type"でハッシュ値を指定した場合、第二引数"message_hash->pdata"に入力された SHA-256 ハッシュ値の 32 バイト全てに対して、第三引数"key_index"に入力された公開鍵ユーザ鍵生成情報から、ECDSA P-256 に従い第一引数"signature"に入力された署名文との検証をします。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.127 R_TSIP_EcdsaP384SignatureVerification

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdsaP384SignatureVerification(
    tsip_ecdsa_byte_data_t *signature,
    tsip_ecdsa_byte_data_t *message_hash,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|---------------------------|----|---|
| signature | 入力 | 検証する署名文情報 |
| signature->pdata | | : 署名文を格納している配列のポインタを指定 署名形式は"署名 r(384bit) 署名 s(384bit)" |
| signature->data_length | | : データ長(バイト単位)を指定(不使用) |
| message_hash | 入力 | 検証するハッシュ値情報 |
| message_hash->pdata | | : ハッシュ値を格納している 配列のポインタを指定 |
| message_hash->data_length | | : 配列の有効データ長(不使用) |
| message_hash->data_type | | : 1 のみ指定可能 |
| key_index | 入力 | 鍵データ領域 : ECC P-384 公開鍵のユーザ鍵生成情報を入力 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生、もしくは署名検証失敗 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |

Description

第二引数"message_hash->pdata"に入力された SHA-384 ハッシュ値の 48 バイト全てに対して、第三引数"key_index"に入力された公開鍵ユーザ鍵生成情報から、ECDSA P-384 に従い第一引数"signature"に入力された署名文との検証をします。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7 章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.128 R_TSIP_EcdhP256Init

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdhP256Init (
    tsip_ecdh_handle_t *handle,
    uint32_t key_type,
    uint32_t use_key_id
)
```

Parameters

| | | |
|------------|-------|-------------------------------|
| handle | 入力/出力 | ECDH 用ハンドラ(ワーク領域) |
| key_type | 入力 | 鍵交換の種類 0 : ECDHE 1 : ECDH |
| use_key_id | 入力 | 0 : key_id 不使用, 1 : key_id 使用 |

Return Values

| | |
|---------------------|----------|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_PARAMETER: | 入力データが不正 |

Description

R_TSIP_EcdhP256Init 関数は、ECDH 鍵交換を演算する準備を行い、その結果を第一引数"handle"に書き出します。"handle"は、続く R_TSIP_EcdhP256ReadPublicKey、R_TSIP_EcdhP256MakePublicKey、R_TSIP_EcdhP256CalculateSharedSecretIndex、R_TSIP_EcdhP256KeyDerivation 関数で引数として使用されます。

第二引数の"key_type"では ECDH 鍵交換の種類を選択してください。ECDHE では、R_TSIP_EcdhP256MakePublicKey 関数で TSIP の乱数生成機能を使い ECC P-256 の鍵ペアを生成します。ECDH では、鍵交換では予めインストールした鍵を使用します。

第三引数の"use_key_id"は、鍵交換の際に key_id を使用する場合"1"を入力してください。key_id はスマートメータ向け規格の DLMS/COSEM 用途です。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7 章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.129 R_TSIP_EcdhP256ReadPublicKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdhP256ReadPublicKey (
    tsip_ecdh_handle_t *handle,
    tsip_ecc_public_key_index_t *public_key_index,
    uint8_t *public_key_data,
    tsip_ecdsa_byte_data_t *signature,
    tsip_ecc_public_key_index_t *key_index
)
```

Parameters

| | | |
|------------------|-------|--|
| handle | 入力/出力 | ECDH 用ハンドラ(ワーク領域) |
| public_key_index | 入力 | 署名検証向けの公開鍵生成情報領域 |
| public_key_data | 入力 | key_id を使用しない場合 ECC P-256 公開鍵(512bit) key_id を使用する場合 key_id (8bit) 公開鍵 s(512bit) |
| signature | 入力 | public_key_data の ECDSA P-256 署名 |
| key_index | 出力 | public_key_data の鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生、もしくは署名検証失敗 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_EcdhP256ReadPublicKey()関数は ECDH 鍵交換相手の ECC P-256 public key の署名を検証し、署名が正しければ第 5 引数に public_key_data の鍵生成情報を出力します。

第一引数"handle"は続く R_TSIP_EcdhP256CalculateSharedSecretIndex()関数の引数で使します。

key_index は R_TSIP_EcdhP256CalculateSharedSecretIndex で Z を計算するための入力として使します

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7 章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.130 R_TSIP_EcdhP256MakePublicKey

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdhP256MakePublicKey (
    tsip_ecdh_handle_t *handle,
    tsip_ecc_public_key_index_t *public_key_index,
    tsip_ecc_private_key_index_t *private_key_index,
    uint8_t *public_key,
    tsip_ecdsa_byte_data_t *signature,
    tsip_ecc_private_key_index_t *key_index
)
```

Parameters

| | | |
|-------------------|-------|---|
| handle | 入力/出力 | ECDH 用ハンドラ(ワーク領域) key_id を使用する場合は、R_TSIP_Ecdh256Init()の 実行後、handle->key_id に入力してください。 |
| public_key_index | 入力 | ECDHE の場合は NULL ポインタを入力してください。 ECDH の場合は、ECC P-256 公開鍵の鍵生成情報を入力してください。 |
| private_key_index | 入力 | 署名生成向けの ECC P-256 秘密鍵 |
| public_key | 出力 | 鍵交換用ユーザ公開鍵(512bit) key_id を使用する場合 key_id (8bit) 公開鍵(512bit) 0 padding(24bit) |
| signature | 出力 | 署名文格納先情報 |
| ->pdata | | : 署名文を格納する配列のポインタを指定 署名形式は"署名 r(256bit) 署名 s(256bit)" |
| ->data_length | | : データ長(バイト単位) |
| key_index | 出力 | ECDHE の場合は乱数から生成された秘密鍵生成情報 ECDH の場合は何も出力されません。 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_EcdhP256MakePublicKey()関数は、ECDH 鍵交換のための公開鍵のユーザ鍵生成情報の署名を計算します。

R_TSIP_EcdhP256Init()関数の key_type で ECDHE を指定した場合、TSIP の乱数生成機能を使い ECC P-256 の鍵ペアを生成します。公開鍵は public_key へ出力し、秘密鍵は key_index に出力されます。

R_TSIP_EcdhP256Init()関数の key_type で ECDH を指定した場合、public_key には public_key_index で入力した公開鍵を出力します。key_index には何も出力されません。

第一引数"handle"は続く R_TSIP_EcdhP256CalculateSharedSecretIndex()関数の引数で使います。

key_index は R_TSIP_EcdhP256CalculateSharedSecretIndex で Z を計算するための入力として使います。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.131 R_TSIP_EcdhP256CalculateSharedSecretIndex

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdhP256CalculateSharedSecretIndex (
    tsip_ecdh_handle_t *handle,
    tsip_ecc_public_key_index_t *public_key_index,
    tsip_ecc_private_key_index_t *private_key_index,
    tsip_ecdh_key_index_t *shared_secret_index
)
```

Parameters

| | | |
|---------------------|-------|--|
| handle | 入力/出力 | ECDH 用ハンドラ(ワーク領域) |
| public_key_index | 入力 | R_TSIP_EcdhP256ReadPublicKey()で署名検証した公開鍵の鍵生成情報 |
| private_key_index | 入力 | 秘密鍵の鍵生成情報 |
| shared_secret_index | 出力 | ECDH 鍵共有で計算した共有秘密 "Z"の鍵生成情報 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_FAIL: | 内部エラーが発生 |
| TSIP_ERR_PARAMETER: | 不正なハンドルが入力された |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_EcdhP256CalculateSharedSecretIndex()関数は、鍵交換相手の公開鍵と自身の秘密鍵からECDH 鍵交換アルゴリズムで共有秘密"Z"の鍵生成情報を出力します。

第二引数の public_key_index には、R_TSIP_EcdhP256ReadPublicKey()で署名検証した公開鍵の鍵生成情報を入力してください。

第三引数の private_key_index には、R_TSIP_EcdhP256Init()の key_type が 0 の場合には、R_TSIP_EcdhP256MakePublicKey()の出力の乱数から生成された秘密鍵の鍵生成情報、key_type が 0 以外の場合には、R_TSIP_EcdhP256MakePublicKey()の第二引数と対になる秘密鍵の鍵生成情報を入力してください。

shared_secret_index は、続く R_TSIP_EcdhP256KeyDerivation()でユーザ鍵生成情報を出力するための鍵材料として使用します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7 章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.132 R_TSIP_EcdhP256KeyDerivation

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdhP256KeyDerivation (
    tsip_ecdh_handle_t *handle,
    tsip_ecdh_key_index_t *shared_secret_index,
    uint32_t key_type,
    uint32_t kdf_type,
    uint8_t *other_info,
    uint32_t other_info_length,
    tsip_hmac_sha_key_index_t *salt_key_index,
    tsip_aes_key_index_t *key_index
)
```

Parameters

| | | |
|---------------------|-------|--|
| handle | 入力/出力 | ECDH 用ハンドラ(ワーク領域) |
| shared_secret_index | 入力 | R_TSIP_EcdhP256CalculateSharedSecretIndex で計算した"Z"の鍵生成情報 |
| key_type | 入力 | 派生させる鍵の種類 0: AES-128 1: AES-256 2: SHA256-HMAC |
| kdf_type | 入力 | 鍵導出の計算で使用するアルゴリズム 0: SHA256 1: SHA256-HMAC |
| other_info | 入力 | 鍵導出の計算で使用する追加データ AlgorithmID PartyUInfo PartyVInfo |
| other_info_length | 入力 | other_info のデータ長(147 以下のバイト単位) |
| salt_key_index | 入力 | Salt の鍵生成情報(kdf_type が 0 の場合は NULL を入力) |
| key_index | 出力 | key_type に対応した鍵生成情報 key_type:2 の場合、SHA256-HMAC 鍵生成情報を出力しま す。tsip_hmac_sha_key_index_t 型で事前に確保された領 域の先頭アドレスを、(tsip_aes_key_index_t*)型でキャスト して指定することが可能です。 |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で 使用されていることによるリソース衝突が発生 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_PARAMETER: | 入力データが不正 |
| TSIP_ERR_PROHIBIT_FUNCTION: | 不正な関数が呼び出された |

Description

R_TSIP_EcdhP256KeyDerivation()関数は、R_TSIP_EcdhP256CalculateSharedSecretIndex()関数で計算した共有秘密"Z(shared_secret_index)"を鍵材料として、第三引数の key_type で指定した鍵生成情報を導出します。鍵導出のアルゴリズムは、NIST SP800-56C の One-Step Key Derivation です。第四引数 kdf_type で、SHA-256 または SHA-256 HMAC を指定します。SHA-256 HMAC を指定する場合、第七引数 salt_key_index に、R_TSIP_GenerateSha256HmacKeyIndex()関数または R_TSIP_UpdateSha256HmacKeyIndex()関数で出力した鍵生成情報を指定します。

第五引数の other_info には鍵交換相手と共有している鍵導出のための固定値を入力してください。

第八引数の key_index は key_type に対応した鍵生成情報が出力されます。導出する key_index と、使用可能な関数の組合せを以下に示します。

| 導出する key index | 使用可能な関数 |
|----------------|---|
| AES-128 | AES128 全ての Init 関数、R_TSIP_Aes128KeyUnwrap() |
| AES-256 | AES256 全ての Init 関数、R_TSIP_Aes256KeyUnwrap() |
| SHA256-HMAC | R_TSIP_Sha256HmacGenerateInit()、R_TSIP_Sha256HmacVerifyInit() |

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態**です。

実行後の状態は **TSIP 使用可能状態**です。

key_index の生成方法については「[7 章 鍵データの運用](#)」を参照してください

Reentrant

非対応

5.133 R_TSIP_EcdheP512KeyAgreement

Format

```
#include "r_tsip_rx_if.h"
e_tsip_err_t R_TSIP_EcdheP512KeyAgreement(
    tsip_aes_key_index_t *key_index,
    uint8_t *receiver_public_key,
    uint8_t *sender_public_key
)
```

Parameters

| | | |
|---------------------|-------|--|
| key_index | 入力 | AES-128 CMAC 演算用ユーザ鍵生成情報領域 |
| receiver_public_key | 入力 | Receiver の Brainpool P512r1 公開鍵 Q(1024bit) MAC(128bit) |
| sender_public_key | 入力/出力 | Sender の Brainpool P512r1 公開鍵 Q(1024bit) MAC(128bit) |

Return Values

| | |
|-----------------------------|--|
| TSIP_SUCCESS: | 正常終了 |
| TSIP_ERR_KEY_SET: | 異常なユーザ鍵生成情報が入力された |
| TSIP_ERR_RESOURCE_CONFLICT: | 本処理に必要なハードウェアリソースが他の処理で使用されていることによるリソース衝突が発生 |
| TSIP_ERR_FAIL: | 内部エラーが発生 |

Description

Brainpool P512r1 を用いて鍵ペア生成の後、ECDHE 演算を行います。

なお、Sender は TSIP、Receiver は鍵交換相手を示します。

<状態遷移>

有効な実行前の状態は **TSIP 使用可能状態** です。

実行後の状態は **TSIP 使用可能状態** です。

Reentrant

非対応

6. コールバック関数

6.1 TSIP_GEN_MAC_CB_FUNC_T 型

Format

```
#include "r_tsip_rx_if.h"
```

```
typedef void (*TSIP_GEN_MAC_CB_FUNC_T)(  
    TSIP_FW_CB_REQ_TYPE req_type,  
    uint32_t iLoop,  
    uint32_t *counter,  
    uint32_t *InData_UpProgram,  
    uint32_t *OutData_Program,  
    uint32_t MAX_CNT)
```

Parameters

| | | |
|------------------|-------|---|
| req_type | 入力 | 要求内容(TSIP_FW_CB_REQ_TYPE) |
| iLoop | 入力 | ループ回数(ワード単位) |
| counter | 入力/出力 | 領域参照用のオフセット |
| InData_UpProgram | 入力/出力 | R_TSIP_GenerateFirmwareMAC()の第三引数 "InData_UpProgram"と同アドレス |
| OutData_Program | 入力/出力 | R_TSIP_GenerateFirmwareMAC()の第五引数 "OutData_Program"と同アドレス |
| MAX_CNT | 入力 | R_TSIP_GenerateFirmwareMAC()の第六引数"MAX_CNT" と同値 |

Return Values

none

Description

R_TSIP_GenerateFirmwareMAC 関数で使用されます。同関数の第七引数で登録します。

復号されたファームウェアと MAC をユーザ側で保存するために使用します。

InData_UpProgram と OutData_Program の領域サイズは、4 の倍数であり、かつ、最低 4 ワード必要です。InData_UpProgram と OutData_Program は、同じサイズにしてください。デモプロジェクトは、コードフラッシュの最小書き込み単位にしています。

本コールバック関数では、R_TSIP_GenerateFirmwareMAC 関数の中で、複数の要求内容で呼び出されます。要求内容は、第一引数"req_type"に格納されます。

第一引数"req_type"には、列挙型 TSIP_FW_CB_REQ_TYPE で定義された値が入ります。

```
typedef enum
{
    TSIP_FW_CB_REQ_PRG_WT = 0u,
    TSIP_FW_CB_REQ_PRG_RD,
    TSIP_FW_CB_REQ_BUFF_CNT,
    TSIP_FW_CB_REQ_PRG_WT_LAST_BLK,
    TSIP_FW_CB_REQ_GET_UPDATE_PRG_CHKSUM,
    TSIP_FW_CB_REQ_STORE_MAC,
}TSIP_FW_CB_REQ_TYPE;
```

この値によって、ユーザ側は必要な対応を行います。

<req_type = TSIP_FW_CB_REQ_PRG_WT>

復号されたファームウェアの保存要求です。

TSIP Module は、4 ワード単位で第五引数"OutData_Program"にデータを格納した後、その都度、本要求を出します。

要求のたびに処理する必要はありません。

ユーザ側で確保した領域に応じて、復号されたファームウェアを保存してください。例えば、8 ワード分領域を確保した場合は、2 回に 1 回復号されたファームウェアを保存してください。

復号されたサイズ数の合計は、第二引数"iLoop"に格納されています。

本要求での"iLoop"最大値は、第六引数"MAX_CNT"から 4 ワード分引いた値です。最後の 4 ワードおよび保存できていないファームウェアは、<req_type = TSIP_FW_CB_REQ_PRG_WT_LAST_BLK>の要求で対応します。

<req_type = TSIP_FW_CB_REQ_PRG_RD>

更新する暗号化されたファームウェアの取得要求です。

TSIP Module は、4 ワード単位で復号処理をする前に、その都度、本要求を出します。

仕組みは、<req_type = TSIP_FW_CB_REQ_PRG_WT>と同じです。

ユーザ側で確保した領域に応じて、第四引数"InData_UpProgram"に格納してください。

<req_type = TSIP_FW_CB_REQ_BUFF_CNT,>

第四引数"InData_UpProgram"と第五引数"OutData_Program"に参照するときのオフセット値要求です。

第三引数"counter" に対して 4 ワードインクリメントした値を第三引数"counter" に戻してください。

第四引数" InData_UpProgram" と第五引数" OutData_Program" で確保したサイズを超える場合は、第三引数" counter" を初期値に戻してください。

<req_type = TSIP_FW_CB_REQ_PRG_WT_LAST_BLK>

暗号化されたファームウェアの最後のブロックに対して復号された時に要求を出します。復号されたファームウェアで保存できていない領域は、このタイミングで保存してください。

<req_type = TSIP_FW_CB_REQ_GET_UPDATE_PRG_CHKSUM>

更新するファームウェアのファームウェアチェックサム値の取得要求です。

チェックサム値を第四引数"lnData_UpProgram"に格納してください。チェックサムのサイズは、16byte です。

<req_type = req_type = TSIP_FW_CB_REQ_STORE_MAC>

復号したファームウェアに対する MAC を出力します。

第五引数" OutData_Program" に MAC が格納されています。サイズは 16byte 分です。

第六引数"MAX_CNT"は、R_TSIP_GenerateFirmwareMAC()の第六引数"MAX_CNT"と同値です。

7. 鍵データの運用

本アプリケーションノートでは provisioning key および encrypted provisioning key に関してサンプルプログラムに添付している鍵を使って説明しています。量産等に適用する場合は独自の鍵を生成する必要があり、それらの詳細が書かれたアプリケーションノートを別途ご用意しています。

ルネサスマイコンをご採用/ご採用予定のお客様に提供させていただいていますので、お取引のあるルネサスエレクトロニクス営業窓口にお問合せください。<https://www.renesas.com/contact/>

7.1 AES ユーザ鍵の運用

7.1.1 AES ユーザ鍵インストール概要

以下に AES ユーザ鍵をインストールする方法を示します。

AES ユーザ鍵はユーザ PC 内で生成する任意のバイト列(128 ビットまたは 256 ビット)です。

AES ユーザ鍵はユーザ毎にユニークな値です。

本インストール手順にしたがってユーザ鍵のインストールを行ってください。また、ユーザ鍵が以下処理フローを経て RX マイコン内部のデータフラッシュに書き込まれるまでの間は必ず安全なサイト内(ユーザ企業直営工場など)で処理を行ってください。

ユーザ鍵はユーザ鍵生成情報という形式でデータフラッシュに書き込みます。このユーザ鍵生成情報から TSIP 内部でユーザ鍵に復元します。復元されたユーザ鍵は、ソフトウェアでアクセスできません。

ユーザ鍵生成情報を各 API に入力することにより、TSIP 内部でユーザ鍵を復元します。ユーザ鍵生成情報はデバイス固有情報で暗号化されているため、データフラッシュ内のユーザ鍵生成情報を別の TSIP 搭載 RX マイコンにコピーして使用しようとしても、正しい復号結果/暗号化結果は得られません。また、不正なユーザ鍵生成情報を TSIP に入力すると TSIP は正常動作しません。

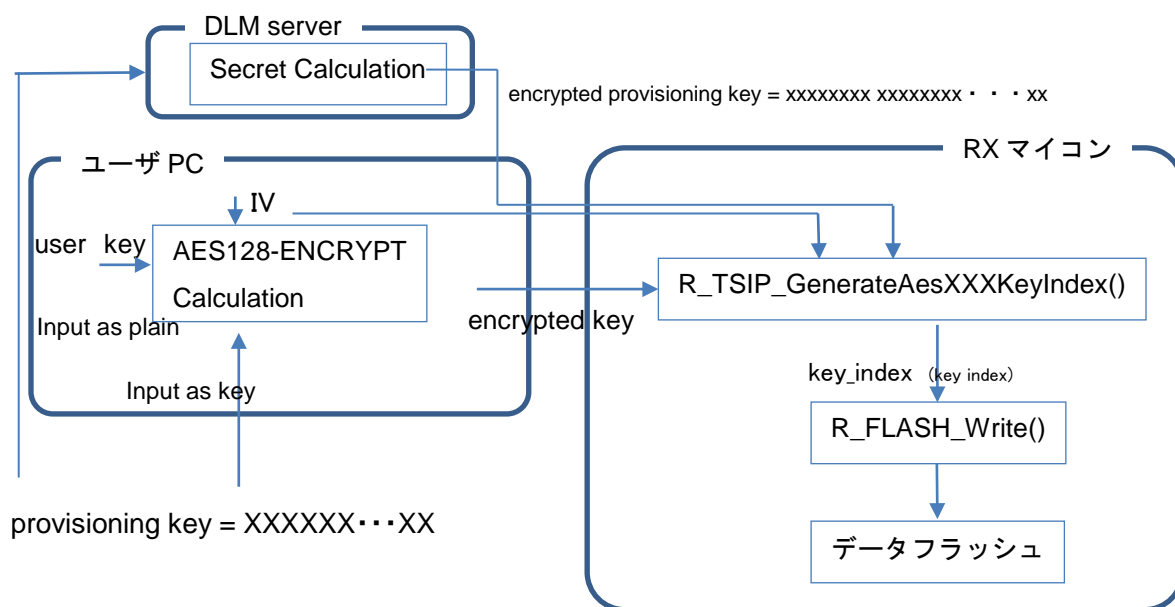


図 7-1 AES ユーザ鍵をインストールする方法

ユーザ PC 上でユーザ鍵を生成する方法の例を次ページ以降に示します。使用するユーザ PC は Windows PC です。

ユーザ鍵の生成には Renesas Secure Flash Programmer を使用します。

7.1.2 AES ユーザ鍵 encrypted key の作成方法

Renesas Secure Flash Programmer を起動します。



図 7-2 Renesas Secure Flash Programmer(Key Wrap タブ AES128bit 鍵設定時)

Key Wrap タブでユーザ鍵の設定を行います。

AES のユーザが自由に使用できる鍵(AES128bit、AES256bit)とファームウェアアップデート用の鍵(AES128bit)を出力するため設定をします。

Key Wrap タブ Key Type で AES-128bit もしくは AES-256bit を選択してください。

Key Data に AES-128bit 選択時には 16 バイト、AES-256bit 選択時には 32 バイトの鍵情報を入力してください。Register ボタンを押すと、Key List に入力された鍵情報が登録されます。Key List に入力するデータのフォーマットは以下の通りです。

・ AES-128bit データフォーマット

| | |
|------|-------------|
| byte | 128bit |
| 0-15 | AES128 鍵データ |

・ AES-256bit データフォーマット

| | |
|------|-------------|
| byte | 256bit |
| 0-31 | AES256 鍵データ |

“provisioning key”に provisioning key File Path と encrypted provisioning key File Path 情報を設定してください。

Path 情報としては、FITDemos フォルダ下に置かれている Key 情報を設定してください。provisioning key File Path には sample.key の Path を、encrypted provisioning key File Path には sample.key_enc.key の Path を設定してください。

必要であれば iv を設定後、[Generate Key File...]ボタンを押すと、R_TSIP_GenerateAesXXXKeyIndex() 関数に入力するための暗号化された鍵(encrypted key)データファイル key_data.c と key_data.h が生成されます。

7.2 TDES ユーザ鍵の運用

7.2.1 TDES ユーザ鍵インストール概要

以下に TDES ユーザ鍵をインストールする方法を示します。

TDES ユーザ鍵はユーザ PC 内で生成する 56 ビット×3 の鍵です。

TDES ユーザ鍵はユーザ毎にユニークな値です。

本インストール手順にしたがってユーザ鍵のインストールを行ってください。また、ユーザ鍵が以下処理フローを経て RX マイコン内部のデータフラッシュに書き込まれるまでの間は必ず安全なサイト内(ユーザ企業直営工場など)で処理を行ってください。

ユーザ鍵はユーザ鍵生成情報という形式でデータフラッシュに書き込みます。このユーザ鍵生成情報から TSIP 内部でユーザ鍵に復元します。復元されたユーザ鍵は、ソフトウェアでアクセスできません。

ユーザ鍵生成情報を各 API に入力することにより、TSIP 内部でユーザ鍵を復元します。ユーザ鍵生成情報はデバイス固有情報で暗号化されているため、データフラッシュ内のユーザ鍵生成情報を別の TSIP 搭載 RX マイコンにコピーして使用しようとしても、正しい復号結果/暗号化結果は得られません。また、不正なユーザ鍵生成情報を TSIP に入力すると TSIP は正常動作しません。

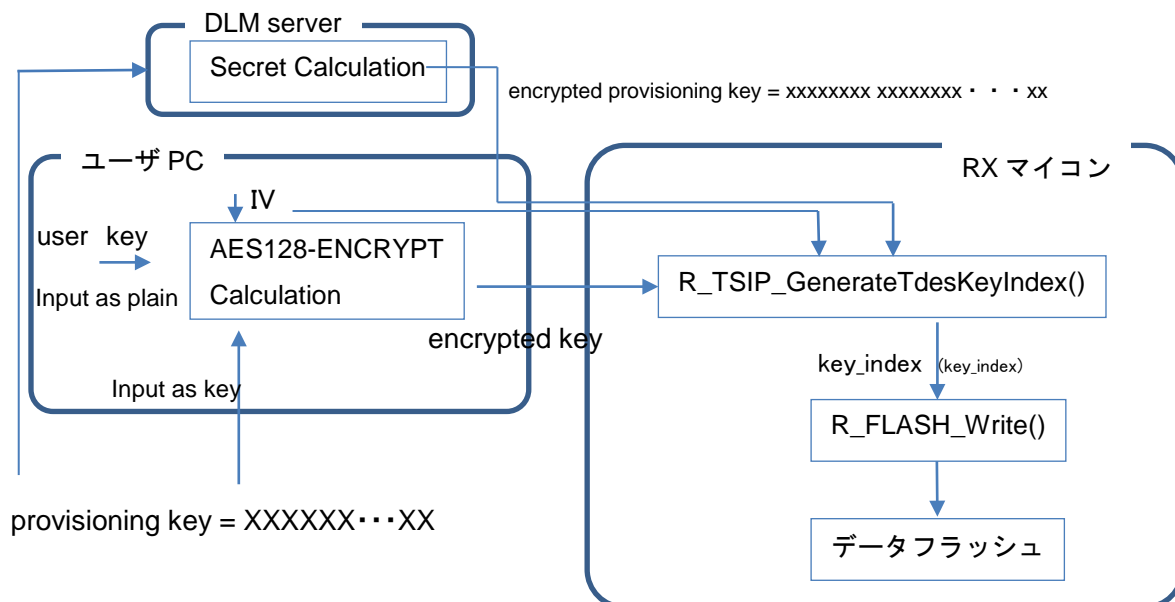


図 7-3 TDES ユーザ鍵をインストールする方法

TDES user key format data

| byte | 128bit | | | |
|-------|-------------|-------|------------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-15 | DES ユーザ鍵 1* | | DES ユーザ鍵 2 | |
| 16-31 | DES ユーザ鍵 3 | | 0padding | |

*DES ユーザ鍵 n

DES ユーザ鍵の鍵データ長は 56 ビットです。鍵データ 7 ビットに対し、1 ビットの奇数パリティが付加されるため、DES ユーザ鍵長は 64 ビットデータになります。

フォーマットは以下になります。

| DES ユーザ鍵 n | | | | | | | |
|------------|------|--------|------|--------|-----|------|--------|
| バイト No | 0 | | 1 | | ... | 8 | |
| ビット | 7-1 | 0 | 7-1 | 0 | ... | 7-1 | 0 |
| データ | 鍵データ | 奇数パリティ | 鍵データ | 奇数パリティ | ... | 鍵データ | 奇数パリティ |

例: パリティを付けた場合、DES ユーザ鍵 0x0000000000000000 は 0x0101010101010101、
0xFFFFFFFFFFFFFFFF は 0xFEFEFEFEFEFEFEFEFE、0x01020304050607 は
0x018080614029190E になります。

DES として使用する場合、DES ユーザ鍵 1= DES ユーザ鍵 2= DES ユーザ鍵 3 の値を入力してください。

2Key-TDES として使用する場合、DES ユーザ鍵 1= DES ユーザ鍵 3 かつ、DES ユーザ鍵 1≠ DES ユーザ鍵 2 の値を入力してください。

ユーザ PC 上でユーザ鍵を生成する方法の例を次ページ以降に示します。使用するユーザ PC は Windows PC です。

ユーザ鍵の生成には Renesas Secure Flash Programmer を使用します。

7.2.2 TDES ユーザ鍵 encrypted key の作成方法

Renesas Secure Flash Programmer を起動します。

図 7-4 Renesas Secure Flash Programmer(Key Wrap タブ Triple-DES 鍵設定時)

Key Wrap タブでユーザ鍵の設定を行います。

TDES のユーザが自由に使用できる鍵(Triple-DES, 2Key-TDES, DES)を出力するため設定をします。

Key Wrap タブ Key Type で Triple-DES、2Key-TDES、DES を選択してください。

Key Data に Triple-DES 選択時には 24 バイト、2Key-TDES 選択時には 16 バイト、DES 選択時には 8 バイトの鍵情報を入力してください。Register ボタンを押すと、Key List に入力された鍵情報が登録されます。Key List に入力するデータのフォーマットは以下の通りです。

・ Triple-DES データフォーマット

| byte | DES ユーザ鍵 1 | DES ユーザ鍵 2 | DES ユーザ鍵 3 |
|------|------------|------------|------------|
| 0-23 | DES 鍵データ | DES 鍵データ | DES 鍵データ |

・ 2Key-TDES データフォーマット

| byte | DES ユーザ鍵 1 | DES ユーザ鍵 2 |
|------|------------|------------|
| 0-15 | DES 鍵データ | DES 鍵データ |

・ DES データフォーマット

| byte | DES ユーザ鍵 1 |
|------|------------|
| 0-7 | DES 鍵データ |

“provisioning key”に provisioning key File Path と encrypted provisioning key File Path 情報を設定してください。

Path 情報としては、FITDemos フォルダ下に置かれている Key 情報を設定してください。provisioning key File Path には sample.key の Path を、encrypted provisioning key File Path には sample.key_enc.key の Path を設定してください。

必要であれば iv を設定後、[Generate Key File...]ボタンを押すと、R_TSIP_GenerateTdesKeyIndex()関数に入力するための暗号化された鍵(encrypted key)データファイル key_data.c と key_data.h が生成されます。

7.3 ARC4 ユーザ鍵の運用

7.3.1 ARC4 ユーザ鍵インストール概要

以下に ARC4 ユーザ鍵をインストールする方法を示します。

ARC4 ユーザ鍵はユーザ PC 内で生成する 2048 ビットの鍵です。

ARC4 ユーザ鍵はユーザ毎にユニークな値です。

本インストール手順にしたがってユーザ鍵のインストールを行ってください。また、ユーザ鍵が以下処理フローを経て RX マイコン内部のデータフラッシュに書き込まれるまでの間は必ず安全なサイト内(ユーザ企業直営工場など)で処理を行ってください。

ユーザ鍵はユーザ鍵生成情報という形式でデータフラッシュに書き込みます。このユーザ鍵生成情報から TSIP 内部でユーザ鍵に復元します。復元されたユーザ鍵は、ソフトウェアでアクセスできません。

ユーザ鍵生成情報を各 API に入力することにより、TSIP 内部でユーザ鍵を復元します。ユーザ鍵生成情報はデバイス固有情報で暗号化されているため、データフラッシュ内のユーザ鍵生成情報を別の TSIP 搭載 RX マイコンにコピーして使用しようとしても、正しい復号結果/暗号化結果は得られません。また、不正なユーザ鍵生成情報を TSIP に入力すると TSIP は正常動作しません。

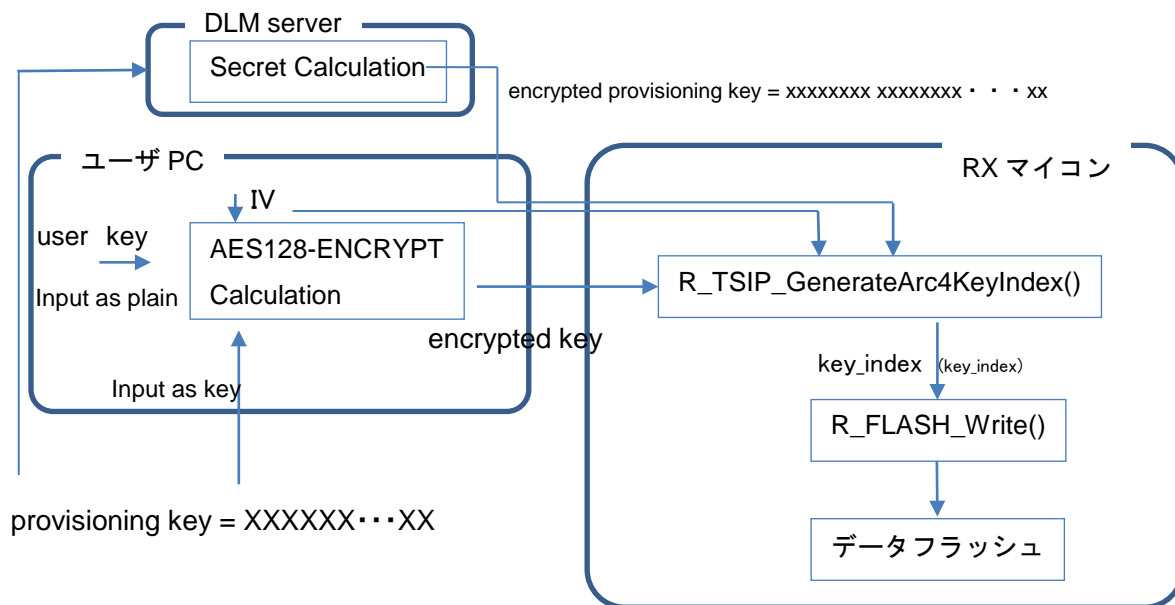


図 7-5 ARC4 ユーザ鍵をインストールする方法

ユーザ PC 上でユーザ鍵を生成する方法の例を次ページ以降に示します。使用するユーザ PC は Windows PC です。

ユーザ鍵の生成には Renesas Secure Flash Programmer を使用します。

7.3.2 ARC4 ユーザ鍵 encrypted key の作成方法

Renesas Secure Flash Programmer を起動します。

| Key Type | Key Data |
|----------|----------|
| | |

図 7-6 Renesas Secure Flash Programmer(Key Wrap タブ ARC4 鍵設定時)

Key Wrap タブでユーザ鍵の設定を行います。

ARC4 のユーザが自由に使用できる鍵を出力するため設定をします。

Key Wrap タブ Key Type で ARC4-2048bit を選択してください。

Key Data に 256 バイトの鍵情報を入力してください。Register ボタンを押すと、Key List に入力された鍵情報が登録されます。Key List に入力するデータのフォーマットは以下の通りです。

・ARC4 データフォーマット

| | |
|-------|-----------|
| byte | 2048bit |
| 0-255 | ARC4 鍵データ |

“provisioning key”に provisioning key File Path と encrypted provisioning key File Path 情報を設定してください。

Path 情報としては、FITDemos フォルダ下に置かれている Key 情報を設定してください。provisioning key File Path には sample.key の Path を、encrypted provisioning key File Path には sample.key_enc.key の Path を設定してください。

必要であれば iv を設定後、[Generate Key File...]ボタンを押すと、R_TSIP_GenerateArc4KeyIndex()関数に入力するための暗号化された鍵(encrypted key)データファイル key_data.c と key_data.h が生成されます。

7.4 HMAC ユーザ鍵の運用

7.4.1 HMAC ユーザ鍵インストール概要

以下に HMAC ユーザ鍵をインストールする方法を示します。

HMAC ユーザ鍵はユーザ PC 内で生成する 256 ビットの鍵です。

HMAC ユーザ鍵はユーザ毎にユニークな値です。

本インストール手順にしたがってユーザ鍵のインストールを行ってください。また、ユーザ鍵が以下処理フローを経て RX マイコン内部のデータフラッシュに書き込まれるまでの間は必ず安全なサイト内(ユーザ企業直営工場など)で処理を行ってください。

ユーザ鍵はユーザ鍵生成情報という形式でデータフラッシュに書き込みます。このユーザ鍵生成情報から TSIP 内部でユーザ鍵に復元します。復元されたユーザ鍵は、ソフトウェアでアクセスできません。

ユーザ鍵生成情報を各 API に入力することにより、TSIP 内部でユーザ鍵を復元します。ユーザ鍵生成情報はデバイス固有情報で暗号化されているため、データフラッシュ内のユーザ鍵生成情報を別の TSIP 搭載 RX マイコンにコピーして使用しようとしても、正しい復号結果/暗号化結果は得られません。また、不正なユーザ鍵生成情報を TSIP に入力すると TSIP は正常動作しません。

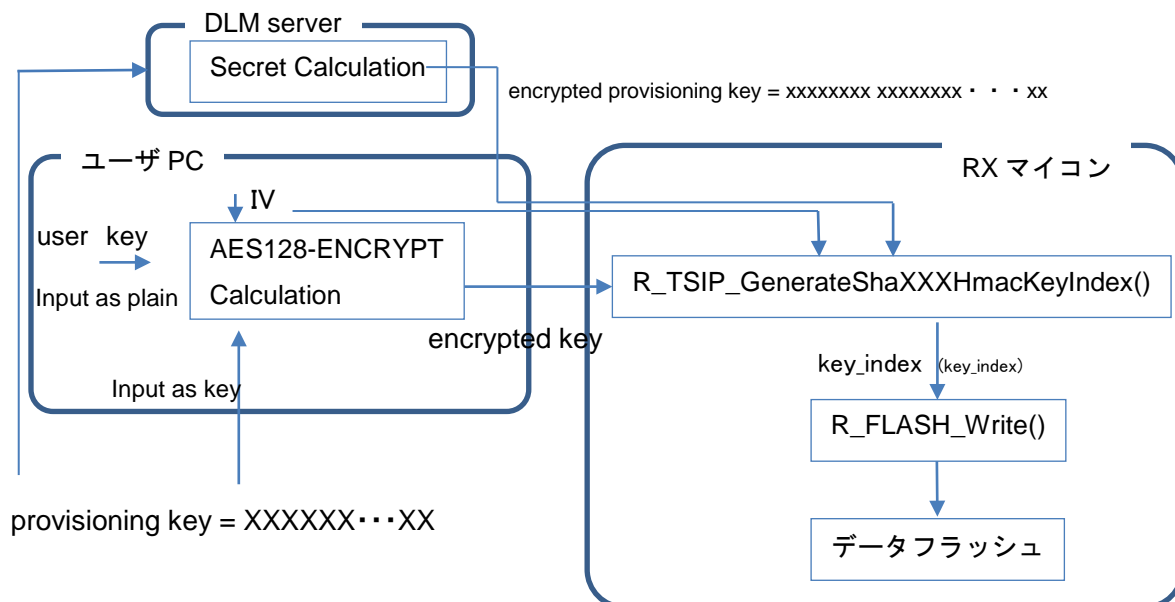


図 7-7 HMAC ユーザ鍵をインストールする方法

ユーザ PC 上でユーザ鍵を生成する方法の例を次ページ以降に示します。使用するユーザ PC は Windows PC です。

ユーザ鍵の生成には Renesas Secure Flash Programmer を使用します。

7.4.2 HMAC ユーザ鍵 encrypted key の作成方法

Renesas Secure Flash Programmer を起動します。



図 7-8 Renesas Secure Flash Programmer(Key Wrap タブ SHA256-HMAC 鍵設定時)

Key Wrap タブでユーザ鍵の設定を行います。

HMAC のユーザが自由に使用できる鍵(SHA-1、SHA-256)を出力するため設定をします。

Key Wrap タブ Key Type で SHA1-HMAC もしくは SHA256-HMAC を選択してください。

Key Data に SHA1-HMAC 選択時には 20 バイト、SHA256-HMAC 選択時には 32 バイトの鍵情報を入力してください。Register ボタンを押すと、Key List に入力された鍵情報が登録されます。Key List に入力するデータのフォーマットは以下の通りです。

・ SHA1-HMAC データフォーマット

| | |
|------|----------------|
| byte | 160bit |
| 0-19 | SHA1-HMAC 鍵データ |

・ SHA256-HMAC データフォーマット

| | |
|------|------------------|
| byte | 256bit |
| 0-31 | SHA256-HMAC 鍵データ |

“provisioning key”に provisioning key File Path と encrypted provisioning key File Path 情報を設定してください。

Path 情報としては、FITDemos フォルダ下に置かれている Key 情報を設定してください。provisioning key File Path には sample.key の Path を、encrypted provisioning key File Path には sample.key_enc.key の Path を設定してください。

必要であれば iv を設定後、[Generate Key File...]ボタンを押すと、R_TSIP_GenerateShaXXXHmacKeyIndex()関数に入力するための暗号化された鍵(encrypted key)データ ファイル key_data.c と key_data.h が生成されます。

7.5 RSA 公開鍵、秘密鍵の運用

7.5.1 RSA 公開鍵、秘密鍵データインストール概要

以下に RSA の公開鍵(public key)、秘密鍵(private key)をインストールする方法を示します。

本インストール手順にしたがって公開鍵と秘密鍵のインストールを行ってください。また、公開鍵と秘密鍵が以下処理フローを経て RX マイコン内部のデータフラッシュに書き込まれるまでの間は必ず安全なサイト内(ユーザ企業直営工場など)で処理を行ってください。

ユーザ鍵はユーザ鍵生成情報という形式でデータフラッシュに書き込みます。このユーザ鍵生成情報から TSIP 内部でユーザ鍵に復元します。復元されたユーザ鍵は、ソフトウェアでアクセスできません。

ユーザ鍵生成情報を各 API に入力することにより、TSIP 内部でユーザ鍵を復元します。ユーザ鍵生成情報はデバイス固有情報で暗号化されているため、データフラッシュ内のユーザ鍵生成情報を別の TSIP 搭載 RX マイコンにコピーして使用しようとしても、正しい復号結果/暗号化結果は得られません。また、不正なユーザ鍵生成情報を TSIP に入力すると TSIP は正常動作しません。

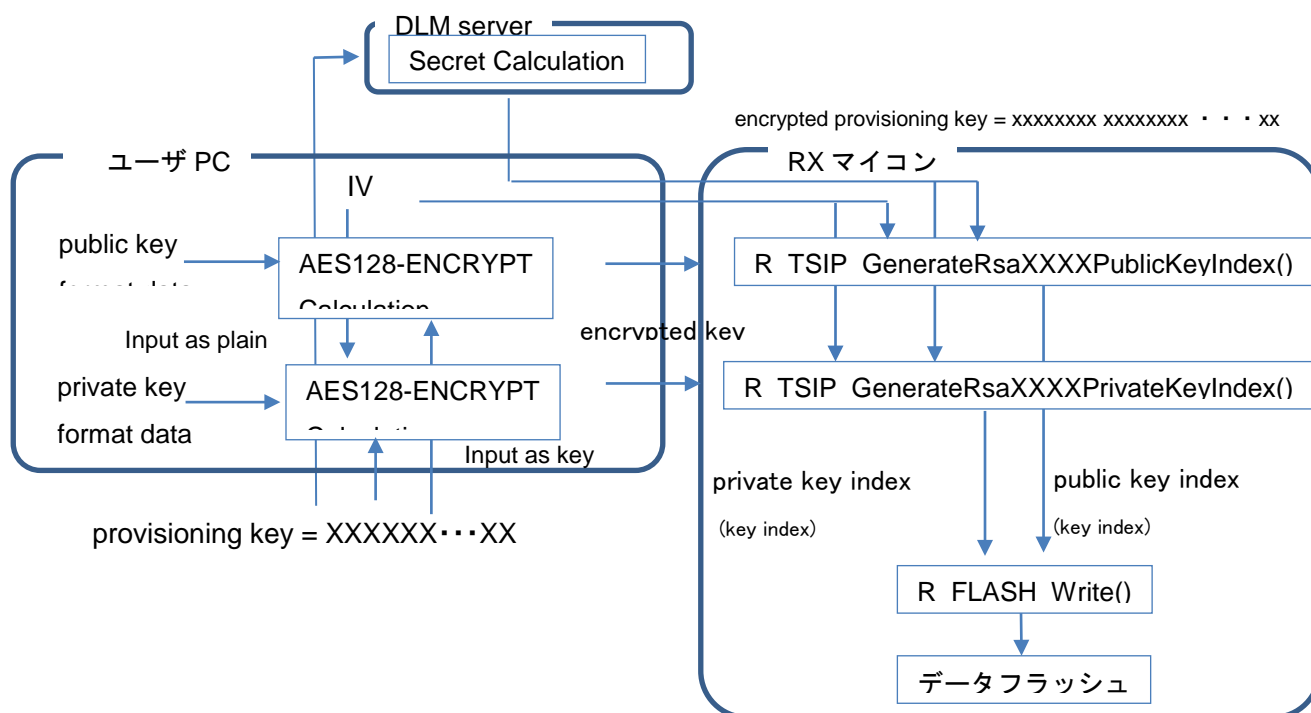


図 7-9 RSA 公開鍵、秘密鍵をインストールする方法

・ public key format data

| byte | 128bit | | | |
|------------------------------------|---------------------------|----------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 1024bit:0-127 2048bit:0-255 | RSA 1024/2048bit 公開鍵 n | | | |
| 1024bit:128-143 2048bit:256-271 | RSA 1024/2048bit 公開鍵 e | 0padding | | |

・ private key format data

| byte | 128bit | | | |
|------------------------------------|------------------------|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 1024bit:0-127 2048bit:0-255 | RSA 1024/2048bit 公開鍵 n | | | |
| 1024bit:128-255 2048bit:256-511 | RSA 1024/2048bit 秘密鍵 d | | | |

ユーザ PC 上で公開鍵、秘密鍵情報を生成する方法の例を次ページに示します。使用するユーザ PC は Windows PC です。

公開鍵、秘密鍵の生成には Renesas Secure Flash Programmer を使用します。

7.5.2 RSA 公開鍵、秘密鍵 encrypted key の作成方法

Renesas Secure Flash Programmer を起動します。

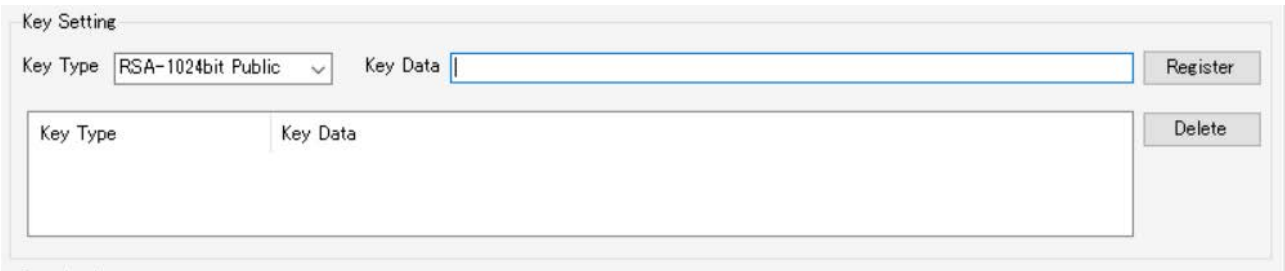


図 7-10 Renesas Secure Flash Programmer(Key Wrap タブ RSA-1024bit Public 鍵設定時)

Key Wrap タブでユーザ鍵の設定を行います。

RSA のユーザが自由に使用できる鍵(RSA-1024bit Public/Private/All, RSA-2048bit Public/Private/All)を出力するため設定をします。

Key Wrap タブ Key Type で RSA-1024bit Public、RSA-1024bit Private、RSA-1024bit All、RSA-2048bit Public、RSA-2048bit Private、RSA-2048bit All を選択してください。

Key Data に RSA-1024bit Public 選択時には 132 バイト、RSA-1024bit Private 選択時には 256 バイト、RSA-1024bit All 選択時には 260 バイト、RSA-2048bit Public 選択時には 260 バイト、RSA-2048bit Private 選択時には 512 バイト、RSA-2048bit All 選択時には 516 バイトの鍵情報を入力してください。Register ボタンを押すと、Key List に入力された鍵情報が登録されます(RSA-XXXXbit All 選択時には、RSA-XXXXbit Public と RSA-XXXXbit Private に分割して登録されます)。Key List に入力するデータのフォーマットは以下の通りです。鍵データが指定ビット長以下の場合は、上位を 0 でパディングしてください。例えば公開鍵 e に 0x10001 を使用する場合は 0x00,0x01,0x00,0x01 を入力してください。

・ RSA-1024bit Public データフォーマット

| byte | RSA 1024bit Public key n | RSA 1024bit Public key e |
|-------|--------------------------|--------------------------|
| 0-131 | 128 バイト RSA 公開鍵 n データ | 4 バイト RSA 公開鍵 e データ |

・ RSA-1024bit Pravate データフォーマット

| byte | RSA 1024bit Public key n | RSA 1024bit Private key d |
|-------|--------------------------|---------------------------|
| 0-255 | 128 バイト RSA 公開鍵 n データ | 128 バイト RSA 秘密鍵 d データ |

・ RSA-1024bit All データフォーマット

| byte | RSA 1024bit Public key n | RSA 1024bit Public key e | RSA 1024bit Private key d |
|-------|--------------------------|--------------------------|---------------------------|
| 0-259 | 128 バイト RSA 公開鍵 n データ | 4 バイト RSA 公開鍵 e データ | 128 バイト RSA 秘密鍵 d データ |

・ RSA-2048bit Public データフォーマット

| byte | RSA 2048bit Public key n | RSA 2048bit Public key e |
|-------|--------------------------|--------------------------|
| 0-259 | 256 バイト RSA 公開鍵 n データ | 4 バイト RSA 公開鍵 e データ |

・ RSA-2048bit Private データフォーマット

| byte | RSA 2048bit Public key n | RSA 2048bit Private key d |
|-------|--------------------------|---------------------------|
| 0-511 | 256 バイト RSA 公開鍵 n データ | 256 バイト RSA 秘密鍵 d データ |

・ RSA-2048bit All データフォーマット

| byte | RSA 2048bit Public key n | RSA 2048bit Public key e | RSA 2048bit Private key d |
|-------|-----------------------------|-----------------------------|------------------------------|
| 0-515 | 256 バイト RSA 公開鍵 n データ | 4 バイト RSA 公開鍵 e データ | 256 バイト RSA 秘密鍵 d データ |

“provisioning key”に provisioning key File Path と encrypted provisioning key File Path 情報を設定してください。

Path 情報としては、FITDemos フォルダ下に置かれている Key 情報を設定してください。provisioning key File Path には sample.key の Path を、encrypted provisioning key File Path には sample.key_enc.key の Path を設定してください。

必要であれば iv を設定後、[Generate Key File...]ボタンを押すと、
R_TSIP_GenerateRsaXXXXPublic/PrivateKeyIndex()関数に入力するための暗号化された鍵(encrypted key)
データファイル key_data.c と key_data.h が生成されます。

• public key format data

| byte | 128bit | | | |
|------------|---|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-31(注 1) | 0 padding(192/224bit の場合に必要) ECC-192/224/256/384bit 公開鍵 Qx | | | |
| 32-63(注 2) | 0 padding(192/224bit の場合に必要) ECC-192/224/256/384bit 公開鍵 Qy | | | |

- 【注】 1. ECC-192/224/256bit の場合です。ECC-384bit の場合は 0-47 となります。
2. ECC-192/224/256bit の場合です。ECC-384bit の場合は 48-95 となります。

• private key format data

| byte | 128bit | | | |
|-----------|--|-------|-------|-------|
| | 32bit | 32bit | 32bit | 32bit |
| 0-31(注 1) | 0 padding(192/224bit の場合に必要) ECC-192/224/256/384bit 秘密鍵 | | | |

ユーザ PC 上で公開鍵、秘密鍵情報を生成する方法の例を次ページに示します。使用するユーザ PC は Windows PC です。

公開鍵、秘密鍵の生成には Renesas Secure Flash Programmer を使用します。

7.6.2 ECC 公開鍵、秘密鍵 encrypted key の作成方法

Renesas Secure Flash Programmer を起動します。



図 7-12 Renesas Secure Flash Programmer(Key Wrap タブ ECC-256bit Public 鍵設定時)

Key Wrap タブでユーザ鍵の設定を行います。

ECC のユーザが自由に使用できる鍵(ECC-192bit Public/Private/All, ECC-224bit Public/Private/All, ECC-256bit Public/Private/All, ECC-384bit Public/Private/All)を出力するため設定をします。

Key Wrap タブ Key Type で ECC-192bit Public、ECC-192bit Private、ECC-192bit All、ECC-224bit Public、ECC-224bit Private、ECC-224bit All、ECC-256bit Public、ECC-256bit Private、ECC-256bit All、ECC-384bit Public、ECC-384bit Private、ECC-384bit All を選択してください。

Key Data に以下のデータフォーマットで示すバイト数の鍵情報を入力してください。Register ボタンを押すと、Key List に入力された鍵情報が登録されます(ECC-XXXbit All 選択時には、ECC-XXXbit Public と ECC-XXXbit Private に分割して登録されます)。Key List に入力するデータのフォーマットは以下の通りです。

- ・ ECC-192bit Public データフォーマット(48 バイト)

| byte | ECC-192bit Public key Qx | ECC-192bit Public key Qy |
|------|--------------------------|--------------------------|
| 0-47 | 24 バイト ECC 公開鍵 Qx データ | 24 バイト ECC 公開鍵 Qy データ |

- ・ ECC-192bit Pravate データフォーマット(24 バイト)

| byte | ECC-192bit Private key |
|------|------------------------|
| 0-23 | 24 バイト ECC 秘密鍵データ |

- ・ ECC-192bit All データフォーマット(72 バイト)

| byte | ECC-192bit Public key Qx | ECC-192bit Public key Qy | ECC-192bit Private key |
|------|--------------------------|--------------------------|------------------------|
| 0-71 | 24 バイト ECC 公開鍵 Qx データ | 24 バイト ECC 公開鍵 Qy データ | 24 バイト ECC 秘密鍵データ |

・ ECC-224bit Public データフォーマット(56 バイト)

| | | |
|------|--------------------------|--------------------------|
| byte | ECC-224bit Public key Qx | ECC-224bit Public key Qy |
| 0-55 | 28 バイト ECC 公開鍵 Qx データ | 28 バイト ECC 公開鍵 Qy データ |

・ ECC-224bit Private データフォーマット(28 バイト)

| | |
|------|------------------------|
| byte | ECC-224bit Private key |
| 0-27 | 28 バイト ECC 秘密鍵データ |

・ ECC-224bit All データフォーマット(84 バイト)

| | | | |
|------|-----------------------------|-----------------------------|---------------------------|
| byte | ECC-224bit Public key Qx | ECC-224bit Public key Qy | ECC-224bit Private key |
| 0-83 | 28 バイト ECC 公開鍵 Qx データ | 28 バイト ECC 公開鍵 Qy データ | 28 バイト ECC 秘密鍵データ |

・ ECC-256bit Public データフォーマット(64 バイト)

| | | |
|------|--------------------------|--------------------------|
| byte | ECC-256bit Public key Qx | ECC-256bit Public key Qy |
| 0-63 | 32 バイト ECC 公開鍵 Qx データ | 32 バイト ECC 公開鍵 Qy データ |

・ ECC-256bit Private データフォーマット(32 バイト)

| | |
|------|------------------------|
| Byte | ECC-256bit Private key |
| 0-31 | 32 バイト ECC 秘密鍵データ |

・ ECC-256bit All データフォーマット(96 バイト)

| | | | |
|------|-----------------------------|-----------------------------|---------------------------|
| byte | ECC-256bit Public key Qx | ECC-256bit Public key Qy | ECC-256bit Private key |
| 0-95 | 32 バイト ECC 公開鍵 Qx データ | 32 バイト ECC 公開鍵 Qy データ | 32 バイト ECC 秘密鍵データ |

・ ECC-384bit Public データフォーマット(96 バイト)

| byte | ECC-384bit Public key Qx | ECC-384bit Public key Qy |
|------|--------------------------|--------------------------|
| 0-95 | 48 バイト ECC 公開鍵 Qx データ | 48 バイト ECC 公開鍵 Qy データ |

・ ECC-384bit Private データフォーマット(48 バイト)

| Byte | ECC-384bit Private key |
|------|------------------------|
| 0-47 | 48 バイト ECC 秘密鍵データ |

・ ECC-384bit All データフォーマット(144 バイト)

| byte | ECC-384bit Public key Qx | ECC-384bit Public key Qy | ECC-384bit Private key |
|-------|-----------------------------|-----------------------------|---------------------------|
| 0-143 | 48 バイト ECC 公開鍵 Qx データ | 48 バイト ECC 公開鍵 Qy データ | 48 バイト ECC 秘密鍵データ |

“provisioning key”に provisioning key File Path と encrypted provisioning key File Path 情報を設定してください。

Path 情報としては、FITDemos フォルダ下に置かれている Key 情報を設定してください。provisioning key File Path には sample.key の Path を、encrypted provisioning key File Path には sample.key_enc.key の Path を設定してください。

必要であれば iv を設定後、[Generate Key File...]ボタンを押すと、
R_TSIP_GenerateEccXXXXPublic/PrivateKeyIndex()関数に入力するための暗号化された鍵(encrypted key)
データファイル key_data.c と key_data.h が生成されます。

8. 付録

8.1 動作確認環境

本ドライバの動作確認環境を以下に示します。

表 8-1 動作確認環境

| 項目 | 内容 |
|--|---|
| 統合開発環境 | ルネサスエレクトロニクス製 e ² studio 2022-01 IAR Embedded Workbench for Renesas RX 4.20.01 |
| C コンパイラ | ルネサスエレクトロニクス製 C/C++ Compiler for RX Family(CC-RX) V3.04.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99 GCC for Renesas RX 8.3.0.202104 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std = gnu99 IAR C/C++ Compiler for Renesas RX version 4.20.01 コンパイルオプション：統合開発環境のデフォルト設定 |
| Renesas Secure Flash Programmer(GUI ツール) | 以下のソフトウェアが必要 Microsoft .NET Framework 4.5 以上 |
| エンディアン | ビッグエンディアン/リトルエンディアン |
| モジュールのバージョン | Ver.1.15 |
| 使用ボード | Renesas Starter Kit for RX231(B 版) (型名：R0K505231S020BE) Renesas Solution Starter Kit for RX23W(TSIP 搭載) (型名：RTK5523W8BC00001BJ) Renesas Starter Kit+ for RX65N-2MB(TSIP 搭載) (型名：RTK50565N2S10010BE) Renesas Starter Kit for RX66T(TSIP 搭載) (型名：RTK50566T0S00010BE) Renesas Starter Kit+ for RX671 (型名：RTK55671xxxxxxxxxx) Renesas Starter Kit+ for RX72M(TSIP 搭載) (型名：RTK5572MNHS10000BE) Renesas Starter Kit+ for RX72N(TSIP 搭載) (型名：RTK5572NNHC00000BJ) Renesas Starter Kit for RX72T(TSIP 搭載) (型名：RTK5572TKCS00010BE) |

8.2 トラブルシューティング

- (1) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「Could not open source file "platform.h"」エラーが発生します。

A : FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。

- CS+を使用している場合
アプリケーションノート「RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」
- e² studio を使用している場合
アプリケーションノート「RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

また、本 FIT モジュールを使用する場合、ボードサポートパッケージ FIT モジュール(BSP モジュール)もプロジェクトに追加する必要があります。BSP モジュールの追加方法は、アプリケーションノート「ボードサポートパッケージモジュール(R01AN1685)」を参照してください。

- (2) Q : FITDemos の e²studio サンプルプロジェクトを CS+で使用したい。

A : 以下の web サイトを参照してください。

「e²studio から CS+への移行方法」

> 「既存のプロジェクトを変換して CS+の新規プロジェクトを作成」

<https://www.renesas.com/jp/ja/products/software-tools/tools/migration-tools/migration-e2studio-to-csplus.html>

【注意】 : 手順 5 で

「変換直後のプロジェクト構成ファイルをまとめてバックアップする(C)」

チェックが入っている場合に、[Q0268002]ダイアログが出る場合があります。

[Q0268002]ダイアログで [はい] ボタンを押した場合、コンパイラのインクルード・パスを設定しなおす必要があります。

9. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

(最新版をルネサス エレクトロニクスホームページから入手してください。)

テクニカルアップデート／テクニカルニュース

(最新の情報をルネサス エレクトロニクスホームページから入手してください。)

ユーザーズマニュアル：開発環境

RX ファミリ CC-RX コンパイラ ユーザーズマニュアル (R20UT3248)

(最新版をルネサス エレクトロニクスホームページから入手してください。)

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

<https://www.renesas.com/jp/ja/>

お問い合わせ先

<https://www.renesas.com/jp/ja/support/contact.html>

すべての商標および登録商標は、それぞれの所有者に帰属します。

改訂記録

| Rev. | 発行日 | 改訂内容 | |
|------|------------|------|---|
| | | ページ | ポイント |
| 1.00 | 2020.07.10 | — | 初版発行 |
| 1.11 | 2020.12.31 | — | <ul style="list-style-type: none"> ・ ECC P-384 鍵インストール、鍵生成、鍵更新機能を追加 ・ ECDSA P-384 機能追加 ・ 鍵共有機能の RX72M、RX66N、RX72N 対応追加 ・ ECDH 鍵交換関数 R_TSIP_EcdhXXX()の関数名を、R_TSIP_EcdhP256XXX()に変更 ・ ECC 公開鍵の構造体 tsip_ecc_public_key_index_t を変更 ・ R_TSIP_AesXXXKeyWrap()と R_TSIP_AesXXXKeyUnwrap()を TSIP-Lite/TSIP 共通の API 関数に変更 ・ コンフィグレーションの記載を削除 ・ R_TSIP_GenerateXXXKeyIndex()および R_TSIP_UpdateXXXKeyIndex()の Parameters において、iv の説明を統一 ・ AES 全ての Init 関数における Return Values に、TSIP_ERR_FAIL を記載 ・ TSIP_USER_HASH_ENABLED に関する記述を削除 ・ 開発環境のバージョンを、開発時に使用した番号に変更 ・ デバイス名に関する記載順を変更 <p>1.2 製品構成の表において、mdf ファイル、secure_boot のプロジェクト、rsk_tsip_rfp_project、および rsk_usb_serial_driver を削除し、RX72N のプロジェクトを追加</p> <p>1.4~1.12 本バージョンの情報を記載</p> <p>1.5 セキュアブートの記載を削除</p> <p>2.2 r_bsp のバージョンを変更</p> <p>3.4 TSIP_ERR_RESOURCE_CONFLICT のスペルを修正</p> <p>4.14 USB メモリを使用したセキュアアップデートの実装例の記載を削除</p> <p>4.40、4.43 key_index->type の違いによる IV の取り扱いについての情報を記載</p> <p>5.23 引数 cipher_length の説明を修正</p> <p>5.52 R_TSIP_Rsa2048DhKeyAgreement 関数の記載を移動</p> <p>5.113 引数 algorithm_id を key_type に(設定値も含めて)変更し、引数 kdf_type および salt_key_index を追加(併せて、戻り値 TSIP_ERR_FAIL を削除)</p> <p>8.1 Renesas Secure Flash Programmer を追加</p> |
| 1.12 | 2021.06.30 | — | <ul style="list-style-type: none"> ・ 開発環境のバージョンを、開発時に使用した番号に変更 ・ AES-GCM および RSA 復号関数の説明を変更 <p>1.2 製品構成の表において、AES 暗号プロジェクトおよび TLS 連携機能プロジェクトを追加</p> <p>1.4~1.12 本バージョンの情報を記載</p> |
| 1.13 | 2021.08.31 | — | <ul style="list-style-type: none"> ・ RX671 対応追加 ・ 開発環境のバージョンを、開発時に使用した番号に変更 ・ HMAC ユーザ鍵を追加 <p>1.2 TSIP 概要 追加(「ユーザ鍵生成のメカニズム」を削除)</p> |

| | | | |
|------|------------|---|---|
| | | | 1.3 製品構成の表において、TSIP ドライバ アプリケーションノートは日本語と英語の両方を記載 1.5~1.14 本バージョンの情報を記載 2.2 r_bsp のバージョンを変更 3.2 状態遷移図 更新 5.38, 5.39, 5.85, 5.86, 5.87, 5.88 更新 7.1.1, 7.2.1, 7.3.1, 7.4.1, 7.5.1, 7.6.1 更新 |
| 1.14 | 2021.10.22 | — | ・ TLS1.3 対応追加(RX65N のみ) |
| 1.15 | 2022.03.31 | — | ・ TLS1.3 対応追加(RX66N、RX72M、RX72N) ・ TLS1.2 RSA 4096bit 対応追加 ・ ハッシュ値演算途中経過取得関数追加 ・ 開発環境のバージョンを、開発時に使用した番号に変更 1.5~1.14 本バージョンの情報を記載 2.2 r_bsp のバージョンを変更 3.3.2 BSP FIT モジュールに関する注意事項を追加 5.49~5.52 第四引数 hash_type の各定義名称を変更 |

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策

CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、 V_{IL} (Max.) から V_{IH} (Min.) までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、 V_{IL} (Max.) から V_{IH} (Min.) までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違くと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。

7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限られません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因したまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものとしします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1 2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。