wolfTPM Documentation

2024-05-06

Contents

1 始めに

wolfTPM は、組み込み用途向けに設計された後方 API 互換性を備えた移植可能なオープンソース TPM 2.0 スタックです。C 言語で書かれており、SPI ハードウェア インターフェイス用の IO コールバックは 1 つだけであり、外部依存関係がなく、リソース使用量が少なくコンパクトなコードであるため、移植性が高くなります。wolfTPM は、認証などの複雑な TPM 操作を支援する API ラッパーと、TPM を使用した証明書署名要求 (CSR) の生成などの複雑な暗号化プロセスを支援するサンプルプログラムを提供します。

1.1 プロトコル概要

トラステッド・プラットフォーム・モジュール (TPM、ISO/IEC 11889 とも呼ばれる) は、統合された暗号化鍵を通じてハードウェアを保護するように設計された専用のマイクロ コントローラーであり、安全な暗号化プロセッサの国際標準です。各 TPM チップには、製造時に固有の秘密の RSA 鍵が焼き付けられているため、コンピュータープログラムは TPM を使用してハードウェアデバイスを認証できます。

ウィキペディアによると、TPM は次の機能を提供します[1] :

- 乱数生成器
- 用途が限定された暗号鍵を安全に生成する機能
- リモート認証: ハードウェアおよびソフトウェア構成のほぼ偽造不可能なハッシュ鍵のサマリーを作成します。構成データのハッシュを担当するソフトウェアによって、要約の範囲が決まります。これにより、第三者はソフトウェアが変更されていないことを確認できます。
- バインド: ストレージ鍵から派生した一意の RSA 鍵である TPM バインド鍵を使用してデータを暗号化します。
- シール: バインドに似ていますが、さらに、復号 (アンシール) されるデータの TPM 状態を指定します。

さらに、TPM は、プラットフォームの整合性、ディスク暗号化、パスワード保護、ソフトウェアライセンス保護などのさまざまなアプリケーションにも使用できます。

1.2 階層

プラットフォーム: **TPM_RH_PLATFORM**

オーナー: **PM_RH_OWNER**

エンドースメント: **TPM_RH_ENDORSEMENT**

各階層には、製造時に生成された独自のシード (種) があります。TPM2_Create または TPM2_CreatePrimary で使用される引数はテンプレートを作成します。テンプレートは KDF に供給され、使用されるのと同じ鍵ベースの階層を生成します。生成される鍵は再起動しても毎回同一のものとなります。新しい RSA2048 ビット鍵の生成には約 15 秒かかります。

通常、これらは作成され、TPM2_EvictControl を使用して NV に格納されます。各 TPM は、シードに基づいて独自の鍵を一意に生成します。一時的な鍵を作成するために使用できる一時的な階層 (TPM_RH_NULL) もあります。

1.3 プラットフォーム構成レジスター (PCR)

プラットフォーム構成レジスタ (PCR) は、TPM の重要な機能の 1 つです。主な使用例は、ソフトウェアの状態 (プラットフォーム上で実行されているソフトウェアとそのソフトウェアによって使用される構成データの両方) を暗号化して記録 (測定) する方法を提供することです。[2]

wolfTPM には、SHA-1 および SHA-256 のインデックス 0 ～23 のハッシュ ダイジェストが含まれています。これらのハッシュダイジェストを拡張して、ブートシーケンス (セキュアブート) の整合性を証明できます。

1.4 用語

このプロジェクトでは、append と marshall および parse と unmarshall という用語を使用しています。

1.5 ハードウエア

wolfTPM は下記のハードウエアでテスト済みです:

- Infineon OPTIGA (TM) Trusted Platform Module 2.0 SLB 9670.
  - LetsTrust: [http://letstrust.de] (https://buyzero.de/collections/andere-platinen/products/letstrust-hardware-tpm-trusted-platform-module). Compact Raspberry Pi TPM 2.0 board based on Infineon SLB 9670.
- ST ST33TP* TPM 2.0 モジュール (SPI and I2C)
- Microchip ATTPM20 モジュール
- Nuvoton NPCT65X or NPCT75x TPM2.0 モジュール

1.5.1 デバイス ID

Infineon SLB9670: TIS: TPM2: Caps 0x30000697, Did 0x001b, Vid 0x15d1, Rid 0x10 Mfg IFX (1), Vendor SLB9670, Fw 7.85 (4555), FIPS 140-2 1, CC-EAL4 1

ST ST33TP SPI TPM2: Caps 0x1a7e2882, Did 0x0000, Vid 0x104a, Rid 0x4e Mfg STM (2), Vendor , Fw 74.8 (1151341959), FIPS 140-2 1, CC-EAL4 0

ST ST33TP I2C TPM2: Caps 0x1a7e2882, Did 0x0000, Vid 0x104a, Rid 0x4e Mfg STM (2), Vendor , Fw 74.9 (1151341959), FIPS 140-2 1, CC-EAL4 0

Microchip ATTPM20 TPM2: Caps 0x30000695, Did 0x3205, Vid 0x1114, Rid 0x 1 Mfg MCHP (3), Vendor , Fw 512.20481 (0), FIPS 140-2 0, CC-EAL4 0

Nations Technologies Inc. TPM 2.0 module Mfg NTZ (0), Vendor Z32H330, Fw 7.51 (419631892), FIPS 140-2 0, CC-EAL4 0

Nuvoton NPCT650 TPM2.0 Mfg NTC (0), Vendor rlsNPCT , Fw 1.3 (65536), FIPS 140-2 0, CC-EAL4 0

Nuvoton NPCT750 TPM2.0 TPM2: Caps 0x30000697, Did 0x00fc, Vid 0x1050, Rid 0x 1 Mfg NTC (0), Vendor NPCT75x"!!4rls, Fw 7.2 (131072), FIPS 140-2 1, CC-EAL4 0

2 wolfTPM のビルド

wolfTPM ライブラリをビルドするには、最初に wolfSSL ライブラリをビルドしてインストールする必要があります。この ダウンロードページ, から取得するか “git clone” コマンドを使ってクローンしてください:

```
$ git clone https://github.com/wolfssl/wolfssl
```

wolfSSL ライブラリをダウンロードしたら、configure スクリプトに次のオプションを渡してビルドします：

```
$ cd wolfssl
$ ./configure --enable-wolftpm
```

または同等の、次のオプションを使用します:

```
$ ./configure --enable-certgen --enable-certreq --enable-certext
--enable-pkcs7 --enable-cryptocb --enable-aescfb
```

次に、wolfSSL ライブラリをビルドしてインストールするだけで、ユーザーの好みに応じてインストールできます。

次のステップは、wolfTPM ライブラリをダウンロードしてインストールすることです。wolfTPM は、同様に次の Web サイトからダウンロードできます ダウンロードページ あるいは GitHub からクローンします。次のコマンドで wolfTPM をビルドします:

```
$ git clone https://github.com/wolfssl/wolftpm
$ cd wolftpm
$ ./autogen.sh
$ ./configure
$ make
```

2.0.1 ビルドオプションとマクロ定義

```
--enable-debug          デバッグ出力を有効にする/最適化をオフにする
                        --enable-debug=yes|no|verbose|io が指定可
                        これらは次のマクロ定義と同義： DEBUG_WOLFTPM,
                            WOLFTPM_DEBUG_VERBOSE, WOLFTPM_DEBUG_IO

--enable-examples       サンプルプログラムもビルドする(デフォルトで有効)

--enable-wrapper        ラッパーコードを有効にする(デフォルトで有効)
                          無効にする場合には次のマクロを定義:
    WOLFTPM2_NO_WRAPPER

--enable-wolfcrypt      wolfCryptのRNG,セッション認証、パラメータ暗号化に関す
    るフック関数を有効化する(デフォルトで有効)
                        無効にする場合には次のマクロを定義:
                            WOLFTPM2_NO_WOLFCRYPT

--enable-advio          アドバンストIOを有効にする(デフォルトで無効)
                          次のマクロ定義と同義：WOLFTPM_ADV_IO

--enable-i2c            I2C TPM のサポートを有効にする(デフォルトで無効)。加え
    てadvioを有効化も必要
                        次のマクロ定義と同義：WOLFTPM_I2C
```

--enable-checkwaitstate TIS / SPI Check Wait Stateを有効化する(デフォルトではチップに依存する)
次のマクロ定義と同義：WOLFTPM_CHECK_WAIT_STATE

--enable-smallstack スタック使用量を削減する

--enable-tislock Linuxの名前付きセマフォをSPIドライバーのプロセス間排他ロックに使用
次のマクロ定義と同義：WOLFTPM_TIS_LOCK

--enable-autodetect 実行時モジュール検出を有効にする(デフォルトで有効。モジュールが指定されない場合)
次のマクロ定義と同義：WOLFTPM_AUTODETECT

--enable-infineon Infineon SLB9670 TPM のサポートを有効化する(デフォルトで無効)

--enable-st ST ST33TPM のサポートを有効化する(デフォルトで無効)
次のマクロ定義と同義：WOLFTPM_ST33

--enable-microchip Enable Microchip ATTPM20のサポートを有効化する(デフォルトで無効)
次のマクロ定義と同義：WOLFTPM_MCHP

--enable-nuvoton Nuvoton NPCT65x/NPCT75x のサポートを有効化する(デフォルトで無効)
次のマクロ定義と同義： WOLFTPM_NUVOTON

--enable-devtpm Linux /dev/tpmX用カーネルドライバーのサポートを有効化する(デフォルトで無効)
次のマクロ定義と同義：WOLFTPM_LINUX_DEV

--enable-swtpm SWTPM TCP protocolのサポートを有効化する(デフォルトで無効)
次のマクロ定義と同義： WOLFTPM_SWTPM

--enable-winapi Windows TBS APIを使用する (デフォルトで無効)
次のマクロ定義と同義： WOLFTPM_WINAPI

WOLFTPM_USE_SYMMETRIC TLSサンプルプログラムで対称 AES/Hashing/HMAC のサポートを有効化する

WOLFTPM2_USE_SW_ECDHE TLSサンプルプログラムでECCエフェメラル鍵の生成とシェアードシークレットの生成にTPMを使用しない

TLS_BENCH_MODE TLSベンチマークを有効にする

NO_TPM_BENCH TPMのベンチマークサンプルプログラムを無効にする

2.0.2 Infineon SLB9670 向けビルド

以下のコマンドで wolfTPM をビルド:

```
git clone https://github.com/wolfSSL/wolfTPM.git
cd wolfTPM
./autogen.sh
./configure
make
```

2.0.3 ST ST33TP* 向けビルド

以下のコマンドで wolfTPM をビルド:

```
./autogen.sh
./configure --enable-st33 [--enable-i2c]
make
```

Raspberry Pi 上の I2C 向けにビルドする際には IC2 を有効にする必要があります。以下がその手順です：1. `sudo vim /boot/config.txt` で編集します。2. `dtparam=i2c_arm=on` の行のコメントを外し有効にします。3. `sudo reboot` で再起動します。

2.0.4 Microchip ATTPM20 向けビルド

以下のコマンドで wolfTPM をビルド:

```
./autogen.sh
./configure --enable-microchip
make
```

2.0.5 Nuvoton 向けビルド

以下のコマンドで wolfTPM をビルド:

```
./autogen.sh
./configure --enable-nuvoton
make
```

2.0.6 “/dev/tpmX” 向けビルド

次のビルドオプションは Linux TIS カーネルドライバーでサポートされているいずれの TPM ベンダーに対して使用できます。

以下のコマンドで wolfTPM をビルド:

```
./autogen.sh
./configure --enable-devtpm
make
```

注意：Linux カーネルドライバーを通して TPM デバイスを使用する際には wolfTPM を使用するアプリケーションに対して必要なパーミッションが付与されていることを確認してください。何故なら通常、“/dev/tpmX” は “tss” ユーザーグループにのみ Read/Write パーミッションが与えられているからです。wolfTPM のサンプルプログラムを “sudo” コマンドと共に実行するかあるいはユーザーを次のようにして “tss” グループに追加する必要があります：

```
sudo adduser yourusername tss
```


2.0.6.1 QEMU と swtpm を使ってのビルド

以下では QEMU 上で wolfTPM を使い linux カーネルデバイスの "/dev/tpmX" を使うデモンストレーションを行います。これにはswtpmのインストールかビルドが必要です。以下の手順は短いビルドの手順の一つを示したものです。操作にはlibtpmsとswtpmを参照する必要があります。

```
PREFIX=$PWD/inst
git clone git@github.com:stefanberger/libtpms.git
cd libtpms/
./autogen.sh --with-openssl --with-tpm2 --prefix=$PREFIX && make install
cd ..
git clone git@github.com:stefanberger/swtpm.git
cd swtpm
PKG_CONFIG_PATH=$PREFIX/lib/pkgconfig/ ./autogen.sh --with-openssl --with-tpm2
    \
    --prefix=$PREFIX && \
  make install
cd ..
```

基本的な Linux インストールをセットアップします。他のインストールベースをインストールしてもかまいません。このセットアップには時間を要します。

```
# ミニインストールイメージをダウンロード
curl -O http://archive.ubuntu.com/ubuntu/dists/bionic-updates/main/installer-
   amd64/current/images/netboot/mini.iso
# qemuイメージファイルを作成
qemu-img create -f qcow2 lubuntu.qcow2 5G
# tpm用にディレクトリを作成
mkdir $PREFIX/mytpm
# swtpmを実行
$PREFIX/bin/swtpm socket --tpm2 --tpmstate dir=$PREFIX/mytpm \
  --ctrl type=unixio,path=$PREFIX/mytpm/swtpm-sock --log level=20 &
# インストールのためにqemuを起動
qemu-system-x86_64 -m 1024 -boot d -bios bios-256k.bin -boot menu=on \
  -chardev socket,id=chrtpm,path=$PREFIX/mytpm/swtpm-sock \
  -tpmdev emulator,id=tpm0,chardev=chrtpm \
  -device tpm-tis,tpmdev=tpm0 -hda lubuntu.qcow2 -cdrom mini.iso
```

ベースシステムがインストールされたら、qemu を実行する準備が整ったので、qemu インスタンス中から wolfSSL と wolfTPM をビルドします。

```
# swtpm を再スタート
$PREFIX/bin/swtpm socket --tpm2 --tpmstate dir=$PREFIX/mytpm \
  --ctrl type=unixio,path=$PREFIX/mytpm/swtpm-sock --log level=20 &
# wolfTPMをインストールして起動するためにqemu システムをスタート
qemu-system-x86_64 -m 1024 -boot d -bios bios-256k.bin -boot menu=on \
  -chardev socket,id=chrtpm,path=$PREFIX/mytpm/swtpm-sock \
  -tpmdev emulator,id=tpm0,chardev=chrtpm \
  -device tpm-tis,tpmdev=tpm0 -hda lubuntu.qcow2
```

以下のコマンドで QEMU ターミナルで wolfTPM をチェックアウトしてビルドします：

```
sudo apt install automake libtool gcc git make

# wolfSSLを取得してビルド
git clone https://github.com/wolfssl/wolfssl.git
pushd wolfssl
```

```
./autogen.sh && \
  ./configure --enable-wolftpm --disable-examples --prefix=$PWD/../inst && \
  make install
popd

# wolfTPMを取得してビルド
git clone https://github.com/wolfssl/wolftpm.git
pushd wolftpm
./autogen.sh && \
  ./configure --enable-devtpm --prefix=$PWD/../inst --enable-debug && \
  make install
sudo make check
popd
```

次のコマンドで QEMU でサンプルプログラムを実行できます。sudo ./examples/wrap/wrap /dev/tpm0 へのアクセスには “sudo” を付加する必要になる場合があります。

2.0.7 SWTPM 向けにビルド

wolfTPM は次の仕様書の D.3 章に記載の SW TPM とインターフェイスすることが可能。TPM-Rev-2.0-Part-4-Supporting-Routines-01.38-code

SWTPM に接続するソケットは排他的に使用しなければならず、TIS あるいは devtpm とはコンパチブルではありません。

wolTPM のテストでは機能のサブセットのみがサポートされています。プラットフォームの要求事項は wolfTPM では使用されません。

テストでは次の二つの実装が使用されます：

- https://sourceforge.net/projects/ibmswtpm2/files/
- https://github.com/stefanberger/swtpm

この機能を有効にするには wolfTPM を以下のオプションとともにビルド：

```
./configure --enable-swtpm
make
```

2.0.7.1 SWTPM シミュレーターのセットアップ

2.0.7.1.1 ibmswtpm2 チェックアウトとビルド

```
git clone https://github.com/kgoldman/ibmswtpm2.git
cd ibmswtpm2/src/
make
```

実行

```
./tpm_server --rm
```

“rm” スイッチはオプションで、指定するとキャッシュファイル (NVChip) を削除します。あるいは “rm NVChip” を実行しても同様です。

2.0.7.1.2 swtpm libtpms をビルド

```
git clone git@github.com:stefanberger/libtpms.git
(cd libtpms && ./autogen.sh --with-tpm2 --with-openssl --prefix=/usr && make
    install)
```

swtpm をビルド

```
git clone git@github.com:stefanberger/swtpm.git
(cd swtpm && ./autogen.sh && make install)
```

注意：Mac OS X では以下を最初に実行すること：

```
brew install openssl socat
pip3 install cryptography

export LDFLAGS="-L/usr/local/opt/openssl@1.1/lib"
export CPPFLAGS="-I/usr/local/opt/openssl@1.1/include"

# libtpms had to use --prefix=/usr/local
```

swtpm を実行

```
mkdir -p /tmp/myvtpm
swtpm socket --tpmstate dir=/tmp/myvtpm --tpm2 --ctrl type=tcp,port=2322 --
    server type=tcp,port=2321 --flags not-need-init
```

2.0.7.2 サンプルプログラムを実行

```
./examples/pcr/extend
./examples/wrap/wrap_test
```

2.0.8 Windows TBS API 向けにビルド

wolfTPM は Windows ネイティブ TBS(TPM Base Services) を使用するようにビルドできます。

Windows TBS インターフェイスを使用する際には NV アクセスは既定でブロックされます。TPM NV ストレージ空間は非常に制限されていてデータ書き込みにより鍵ハンドルのロードに失敗するなどの未定義のふるまいを引き起こす可能性があります。このような NV ストレージ空間への書き込みは TBS によっては管理されていません。

TPM は “TPM2_Create” を使って鍵を生成した場合には、ディスク上に格納したり必要に応じたロードが安全にできるように暗号化した秘密鍵ブロブを返すように設計されています。秘密鍵ブロブを保護するのに使用されるのは対称暗号化鍵だけです。“TPM2_Load” を使って鍵をロードする場合には一時的なハンドルを得て署名や暗号化/復号に使用します。

“TPM2_CreatePrimary” を使って生成した鍵はハンドルで返されます。暗号化された鍵データが返されるわけではありません。このハンドルは “TPM2_FlushContext” が呼び出されるまでロードされた状態が保たれます。

2.0.8.1 制限事項 wolfTPM は TPM 2.0 デバイスを搭載した Windows 10 でテストされています。Windows が TPM1.2 の機能をサポートしている場合は wolfTPM ではサポートされません。TPM 2.0 が搭載されていることを確認する場合には、PowerShell を開き “Get-PnpDevice -Class SecurityDevices” を実行してください。

```
Status     Class           FriendlyName
------     -----           ------------
OK         SecurityDevices Trusted Platform Module 2.0
Unknown    SecurityDevices Trusted Platform Module 2.0
```

2.0.8.2 MSYS2 上でビルド

MSYS2 を使ってテスト済み

```
export PREFIX=$PWD/tmp_install

cd wolfssl
./autogen.sh
./configure --prefix="$PREFIX" --enable-wolftpm
make
make install

cd ../wolftpm/
./autogen.sh
./configure --prefix="$PREFIX" --enable-winapi
make
```

2.0.8.3 linux 上でのビルド

mingw-w32-bin_x86_64-linux_20131221.tar.bz2 を使ってテスト済み。
ソースはこちら。

ツールを抽出して PATH 変数に追加します。

```
mkdir mingw_tools
cd mingw_tools
tar xjvf ../mingw-w32-bin_x86_64-linux_20131221.tar.bz2
export PATH=$PWD/bin/:$PWD/i686-w64-mingw32/bin:$PATH
cd ..
```

以下でビルドします。

```
export PREFIX=$PWD/tmp_install
export CFLAGS="-DWIN32 -DMINGW -D_WIN32_WINNT=0x0600 -DUSE_WOLF_STRTOK"
export LIBS="-lws2_32"

cd wolfssl
./autogen.sh
./configure --host=i686 CC=i686-w64-mingw32-gcc --prefix="$PREFIX" --enable-
   wolftpm
make
make install

cd ../wolftpm/
./autogen.sh
./configure --host=i686 CC=i686-w64-mingw32-gcc --prefix="$PREFIX" --enable-
   winapi
make
cd ..
```

2.0.8.4 Windows 上での実行

マシン上での TPM の存在とその状態を確認する為には “tpm.msc” を実行してください。

3 始めましょう

wolfTPM ライブラリには、wolfTPM のインストールが成功すると、すぐに使用できるようになる TPM 2.0 ラッパー テスト、ネイティブ テスト、およびサンプル ベンチマーク アプリケーションが含まれています。以下に、サンプル アプリケーションを実行する方法について説明します。

これらのアプリケーションを実行しているハードウェア プラットフォームとインターフェイスするには、tpm_io.c 内の関数 TPM2_IoCb を参照してください。

3.1 サンプルプログラム

これらのサンプルプログラムは、TPM 2.0 モジュールの機能をデモンストレーションしています。

またサンプルプログラムは、./examples/tpm_test.h で定義されたハンドルを使用して、テスト用に NV に RSA および ECC 鍵を作成します。

PKCS #7 と TLS のサンプルプログラムでは、CSR を生成し、テスト スクリプトを使用して署名する必要があります。以下の CSR と証明書の署名を参照してください。

パラメータの暗号化を有効にするには、AES-CFB モードの場合は “-aes” を使用し、XOR モードの場合は “-xor" を使用します。一部の TPM コマンド/応答のみがパラメーターの暗号化をサポートします。TPM2_ API に.flags”CMD_FLAG_ENC2" または”CMD_FLAG_DEC2" が設定されている場合、コマンドはパラメーターの暗号化/復号を使用します。

ST33 および NPCT75x の TPM 2.0 追加 GPIO のサンプルプログラムなど、ベンダー固有のサンプルプログラムがいくつかあります。

3.1.1 ネイティブ API のテスト

ネイティブ TPM2_* API の呼び出しを示します。

```
./examples/native/native_test
TPM2 Demo using Native API's
TPM2: Caps 0x30000495, Did 0x0000, Vid 0x104a, Rid 0x4e
TPM2_Startup pass
TPM2_SelfTest pass
TPM2_GetTestResult: Size 12, Rc 0x0
TPM2_IncrementalSelfTest: Rc 0x0, Alg 0x1 (Todo 0)
TPM2_GetCapability: Property FamilyIndicator 0x322e3000
TPM2_GetCapability: Property PCR Count 24
TPM2_GetCapability: Property FIRMWARE_VERSION_1 0x004a0008
TPM2_GetCapability: Property FIRMWARE_VERSION_2 0x44a01587
TPM2_GetRandom: Got 32 bytes
TPM2_StirRandom: success
TPM2_PCR_Read: Index 0, Count 1
TPM2_PCR_Read: Index 0, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 1, Count 1
TPM2_PCR_Read: Index 1, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 2, Count 1
TPM2_PCR_Read: Index 2, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 3, Count 1
TPM2_PCR_Read: Index 3, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 4, Count 1
TPM2_PCR_Read: Index 4, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 5, Count 1
TPM2_PCR_Read: Index 5, Digest Sz 32, Update Counter 20
```

```
TPM2_PCR_Read: Index 6, Count 1
TPM2_PCR_Read: Index 6, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 7, Count 1
TPM2_PCR_Read: Index 7, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 8, Count 1
TPM2_PCR_Read: Index 8, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 9, Count 1
TPM2_PCR_Read: Index 9, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 10, Count 1
TPM2_PCR_Read: Index 10, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 11, Count 1
TPM2_PCR_Read: Index 11, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 12, Count 1
TPM2_PCR_Read: Index 12, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 13, Count 1
TPM2_PCR_Read: Index 13, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 14, Count 1
TPM2_PCR_Read: Index 14, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 15, Count 1
TPM2_PCR_Read: Index 15, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 16, Count 1
TPM2_PCR_Read: Index 16, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 17, Count 1
TPM2_PCR_Read: Index 17, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 18, Count 1
TPM2_PCR_Read: Index 18, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 19, Count 1
TPM2_PCR_Read: Index 19, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 20, Count 1
TPM2_PCR_Read: Index 20, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 21, Count 1
TPM2_PCR_Read: Index 21, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 22, Count 1
TPM2_PCR_Read: Index 22, Digest Sz 32, Update Counter 20
TPM2_PCR_Read: Index 23, Count 1
TPM2_PCR_Read: Index 23, Digest Sz 32, Update Counter 20
TPM2_PCR_Extend success
TPM2_PCR_Read: Index 0, Count 1
TPM2_PCR_Read: Index 0, Digest Sz 32, Update Counter 21
TPM2_StartAuthSession: sessionHandle 0x3000000
TPM2_PolicyGetDigest: size 32
TPM2_PCR_Read: Index 0, Digest Sz 20, Update Counter 21
wc_Hash of PCR[0]: size 32
TPM2_PolicyPCR failed 0x1c4: TPM_RC_AUTHSIZE
TPM2_PolicyRestart: Done
TPM2_HashSequenceStart: sequenceHandle 0x80000000
Hash SHA256 test success
TPM2_CreatePrimary: Endorsement 0x80000000 (314 bytes)
TPM2_CreatePrimary: Storage 0x80000002 (282 bytes)
TPM2_LoadExternal: 0x80000004
TPM2_MakeCredential: credentialBlob 68, secret 256
TPM2_ReadPublic Handle 0x80000004: pub 314, name 34, qualifiedName 34
Create HMAC-SHA256 Key success, public 48, Private 137
TPM2_Load New HMAC Key Handle 0x80000004
```

```
TPM2_PolicyCommandCode: success
TPM2_ObjectChangeAuth: private 137
TPM2_ECC_Parameters: CurveID 3, sz 256, p 32, a 32, b 32, gX 32, gY 32, n 32,
    h 1
TPM2_Create: New ECDSA Key: pub 88, priv 126
TPM2_Load ECDSA Key Handle 0x80000004
TPM2_Sign: ECC S 32, R 32
TPM2_VerifySignature: Tag 32802
TPM2_Create: New ECDH Key: pub 88, priv 126
TPM2_Load ECDH Key Handle 0x80000004
TPM2_ECDH_KeyGen: zPt 68, pubPt 68
TPM2_ECDH_ZGen: zPt 68
TPM2 ECC Shared Secret Pass
TPM2_Create: New RSA Key: pub 278, priv 222
TPM2_Load RSA Key Handle 0x80000004
TPM2_RSA_Encrypt: 256
TPM2_RSA_Decrypt: 68
RSA Encrypt/Decrypt test passed
TPM2_NV_DefineSpace: 0x1bffffff
TPM2_NV_ReadPublic: Sz 14, Idx 0x1bffffff, nameAlg 11, Attr 0x2020002, authPol
    0, dataSz 32, name 34
Create AES128 CFB Key success, public 50, Private 142
TPM2_Load New AES Key Handle 0x80000004
Encrypt/Decrypt test success
```

3.1.2 ラッパー API のテスト

wolfTPM2_* ラッパー API の呼び出しを示します。

```
./examples/wrap/wrap_test
TPM2 Demo for Wrapper API's
Mfg STM  (2), Vendor , Fw 74.8 (1151341959), FIPS 140-2 1, CC-EAL4 0
RSA Encrypt/Decrypt Test Passed
RSA Encrypt/Decrypt OAEP Test Passed
RSA Key 0x80000000 Exported to wolf RsaKey
wolf RsaKey loaded into TPM: Handle 0x80000000
RSA Private Key Loaded into TPM: Handle 0x80000000
ECC Sign/Verify Passed
ECC DH Test Passed
ECC Verify Test Passed
ECC Key 0x80000000 Exported to wolf ecc_key
wolf ecc_key loaded into TPM: Handle 0x80000000
ECC Private Key Loaded into TPM: Handle 0x80000000
NV Test on index 0x1800200 with 1024 bytes passed
Hash SHA256 test success
HMAC SHA256 test success
Encrypt/Decrypt (known key) test success
Encrypt/Decrypt test success
```

3.1.3 認証のユースケース

3.1.3.1 TPM 署名のタイムスタンプ, TPM2.0 GetTime Attestation Identity Keys (AIK) の作成と、現在のシステム アップタイムの保護されたレポートとして後で使用できる TPM 署名付きタイムスタンプの生成を示します。

このサンプルプログラムでは、“authSession”(承認セッション) と “policySession”(ポリシー承認) を使用して、AIK の作成に必要な承認階層を有効にする方法を示します。AIK は、TPM 2.0 ネイティブ API を使用して “TPM2_GetTime” コマンドを発行するために使用されます。これにより、稼働時間のシステム レポートとして使用できる、TPM によって生成され署名されたタイムスタンプが提供されます。

```
./examples/timestamp/signed_timestamp
```

3.1.3.2 TPM 署名 PCR(system) 計測, TPM2.0 Quote TPM 署名済み構造に PCR 値を配置することにより、システム状態の証明に使用される TPM2.0 Quote の生成を示します。

3.1.3.2.1 サンプルプログラムのリスト “./examples/pcr/” フォルダには、プラットフォーム構成レジスタ (PCR) を操作するためのツールが含まれています。より多くのログ出力を表示するには、make の前に “./configure --enable-debug” を使用してデバッグ出力を有効にして wolfTPM をビルドすることをお勧めします。これらの PCR の例を使用して表示するサンプル スクリプトがあります。

サンプルプログラム:

- ./examples/pcr/reset: PCR の内容をクリアするために使用されます (制限が適用されます。以下を参照してください)。
- ./examples/pcr/extend: PCR の内容を変更するために使用されます (拡張は暗号化操作です。以下を参照してください)。
- ./examples/pcr/quote: PCR ダイジェストと TPM によって生成された署名を含む TPM2.0 Quote 構造を生成するために使用されます

スクリプト:

- ./examples/pcr/demo.sh ：上記ツールをデモンストレーションするスクリプト
- ./examples/pcr/demo-quote-zip.sh ：上記のツールを使用してシステム ファイルを測定し、その測定値を使用して TPM 署名付き証明を生成する方法を示すスクリプト

3.1.3.2.2 技術解説 プラットフォーム構成レジスタ（Platform Configuration Registers = PCR)

TPM2.0 の PCR は、1 種類の書き込み操作のみを実行できる特別なレジスタです。TPM 2.0 拡張操作は、PCR を更新する唯一の方法です。

電源投入時に、TPM はすべての PCR をデフォルト状態 (PCR に応じてすべてゼロまたはすべて 1) にリセットします。この状態から、同じハッシュ ダイジェストで PCR を拡張した場合にのみ、TPM は同じ PCR 値を生成できます。複数の値 (複数の拡張操作) の場合、値を正しい順序で指定する必要があります。そうしないと、最終的な PCR 値が異なります。

たとえば、カーネルがすべてのブートで同じである場合、Linux でメジャー ブートを実行すると、同じ PCR ダイジェストが生成されます。ただし、同じ (A) Linux カーネル、(B) initrd イメージ、および (C) 構成ファイルをロードすると、拡張操作の順序が一貫している場合 (A-B-C など) にのみ、同じ PCR ダイジェストが生成されます。順序が同じである限り、どちらの拡張操作が最初か最後かは問題ではありません。たとえば、C-B-A は再現可能なダイジェストになりますが、A-B-C ダイジェストとは異なります。

リセット

すべての PCR が等しいわけではありません。ユーザーはすべての PCR で拡張操作を実行できますが、通常の実行時にそのうちの 1 つだけをリセットできます。これが PCR を非常に便利なものにしている理由です。

- PCR0-15 は起動時にリセットされ、再起動サイクルからのみ再度クリア (リセット) できます。
- PCR16 はデバッグ用の PCR です。これは、上記のすべてのツールでデフォルトで使用される PCR です。PCR16 でのテストと作業は安全です。
- PCR17-22 は、ダイナミック ルート オブ トラスト測定 (DRTM) 用に予約されています。これは、個別に説明する高度なトピックです。

拡張

TPM 2.0 の TPM2_Extend API は、SHA1 または SHA256 暗号化操作を使用して、PCR の現在の値と、新しく提供されたハッシュ ダイジェストを組み合わせます。

クオート

TPM 2.0 の TPM2_Quote API は、TPMS_ATTEST と呼ばれる TCG 定義の構造体に PCR ダイジェストを TPM 署名とともにカプセル化する標準操作です。署名は、TPM のみが使用できる Attestation Identity Key (AIK) と呼ばれる TPM によって生成されたキーから生成されます。これにより、Quote および PCR ダイジェストのソースが保証されます。Quote と PCR は共に、システムの測定と完全性のための手段を提供します。

3.1.3.2.3 サンプルプログラムの使用方法

Reset サンプルプログラムの使用方法

```
$ ./examples/pcr/reset -?
PCR index is out of range (0-23)
Expected usage:
./examples/pcr/reset [pcr]
* pcr is a PCR index between 0-23 (default 16)
Demo usage without parameters, resets PCR16.
```

Extend サンプルプログラムの使用方法

```
$ ./examples/pcr/extend -?
Incorrect arguments
Expected usage:
./examples/pcr/extend [pcr] [filename]
* pcr is a PCR index between 0-23 (default 16)
* filename points to file(data) to measure
    If wolfTPM is built with --disable-wolfcrypt the file
    must contain SHA256 digest ready for extend operation.
    Otherwise, the extend tool computes the hash using wolfcrypt.
Demo usage without parameters, extends PCR16 with known hash.
```

Quote サンプルプログラムの使用方法

```
$ ./examples/pcr/quote -?
Incorrect arguments
Expected usage:
./examples/pcr/quote [pcr] [filename]
* pcr is a PCR index between 0-23 (default 16)
* filename for saving the TPMS_ATTEST structure to a file
Demo usage without parameters, generates quote over PCR16 and
saves the output TPMS_ATTEST structure to "quote.blob" file.
```

3.1.3.2.4 一般的なデモ出力

すべての PCR の例は、引数なしで使用できます。以下は ./examples/pcr/demo.sh スクリプトの出力です:

```
$ ./examples/pcr/reset
Demo how to reset a PCR (clear the PCR value)
wolfTPM2_Init: success
Trying to reset PCR16...
TPM2_PCR_Reset success
PCR16 digest:
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
```

予想どおり、PCR16 の内容はすべてゼロに戻されました。この時点から、システム測定用の予測可能な PCR ダイジェスト (値) を生成できます。PCR7 はシステムの起動時にリセットされるため、起動後に PCR7 を使用する場合と同様です。PCR16 を使用すると、システムの再起動をスキップして安全にテストできます。

```
$ ./examples/pcr/extend
Demo how to extend data into a PCR (TPM2.0 measurement)
wolfTPM2_Init: success
Hash to be used for measurement:
000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
TPM2_PCR_Extend success
PCR16 digest:
    bb 22 75 c4 9f 28 ad 52 ca e6 d5 5e 34 a9 74 a5 | ."u..(.R...^4.t.
    8c 7a 3b a2 6f 97 6e 8e cb be 7a 53 69 18 dc 73 | .z;.o.n...zSi..s
```

PCR の古い内容 (すべてゼロ) と提供されたハッシュ (SHA256 32 バイト ダイジェスト) に基づいて、PCR は extend の例の最後に出力された新しい値を取得します。この値は、reset が extend の前に起動された場合、常に同じになります。カスタム ハッシュ ダイジェストを渡すために、extend ツールは、PCR インデックスを最初の引数として受け入れ (PCR16 には 16 を使用することをお勧めします)、ユーザー ファイルを 2 番目の引数として受け入れます。

```
$ ./examples/pcr/quote
Demo of generating signed PCR measurement (TPM2.0 Quote)
wolfTPM2_Init: success
TPM2_CreatePrimary: 0x80000000 (314 bytes)
wolfTPM2_CreateEK: Endorsement 0x80000000 (314 bytes)
TPM2_CreatePrimary: 0x80000001 (282 bytes)
wolfTPM2_CreateSRK: Storage 0x80000001 (282 bytes)
TPM2_StartAuthSession: sessionHandle 0x3000000
TPM2_Create key: pub 280, priv 212
TPM2_Load Key Handle 0x80000002
wolfTPM2_CreateAndLoadAIK: AIK 0x80000002 (280 bytes)
TPM2_Quote: success
TPM with signature attests (type 0x8018):
    TPM signed 1 count of PCRs
    PCR digest:
    c7 d4 27 2a 57 97 7f 66 1f bd 79 30 0a 1b bf ff | ..'*W..f..y0....
    2e 43 57 cc 44 14 7a 82 11 aa 76 3f 9f 1b 3a 6c | .CW.D.z...v?..:l
    TPM generated signature:
    28 dc da 76 33 35 a5 85 2a 0c 0b e8 25 d0 f8 8d | (..v35..*...%...
    1f ce c3 3b 71 64 ed 54 e6 4d 82 af f3 83 18 8e | ...;qd.T.M......
    6e 2d 9f 9e 5a 86 4f 11 fe 13 84 94 cf 05 b9 d5 | n-..Z.O.........
    eb 5a 34 39 b2 a5 7a 5f 52 c0 f4 e7 2b 70 b7 62 | .Z49..z_R...+p.b
    6a fe 79 4e 2e 46 2e 43 d7 1c ef 2c 14 21 11 14 | j.yN.F.C...,.!..
    95 01 93 a9 85 0d 02 c7 b2 f8 75 1a bd 59 da 56 | ..........u..Y.V
    cc 43 e3 d2 aa 14 49 2a 59 26 09 9e c9 4b 1a 66 | .C....I*Y&...K.f
    cb 77 65 95 79 69 89 bd 46 46 13 3d 2c a9 78 f8 | .we.yi..FF.=,.x.
    2c ab 8a 4a 6b f2 97 67 86 37 f8 f6 9d 85 cd cf | ,..Jk..g.7......
    a4 ae c6 d3 cf c1 63 92 8c 7b 88 79 90 54 0a ba | ......c..{.y.T..
    8d c6 1c 8f 6e 6d 61 bc a9 2f 35 b0 1a 46 74 9a | ....nma../5..Ft.
    e3 7d 39 33 52 1a f5 4b 07 8d 30 53 75 b5 68 40 | .}93R..K..0Su.h@
    04 e7 a1 fc b1 93 5d 1e bc ca f4 a9 fa 75 d3 f6 | ......]......u..
    3d 4a 5b 07 23 0e f0 f4 1f 97 23 76 1a ee 66 93 | =J[.#.....#v..f.
    cd fd 9e 6f 2b d3 95 c5 51 cf f6 81 5b 97 a1 d2 | ...o+...Q...[...
    06 45 c0 30 70 ad bd 36 66 9f 95 af 60 7c d5 a2 | .E.0p..6f...`|..
```

PCR 測定値を含む TPM 署名付き構造を生成する前に、クオートのサンプルプログラムでは、TPM が動作するために必要な保証鍵 (EK) を作成することから始めます。基本的に、他のすべての鍵のプライマリー鍵として機能します。次に、ストレージ鍵 (SRK) が生成され、その SRK の下に特別な Attestation Identity Key (AIK) が追加されます。AIK を使用して、TPM はクオート構造に署名できます。

3.1.3.2.5 システムファイルの測定手順（ローカル認証の実施） システム管理者は、ユーザーの zip ツールが本物であること (正当なソフトウェア、正しいバージョン、改ざんされていないこと) を確認したいと考えています。これを行うために、システム管理者は PCR16 をリセットし、その後、ファイルが変更された場合に将来の参照に使用できる zip バイナリに基づいて PCR ダイジェストを生成できます。

これは ./examples/pcr/demo-quote-zip.sh スクリプトからの出力です。

```
$ ./examples/pcr/reset 16
...
Trying to reset PCR16...
TPM2_PCR_Reset success
...
```

これは、よく知られた PCR の初期状態です。extend ツールを使用することにより、SysAdmin は/usr/bin/zip バイナリを SHA256 ハッシュ計算のために wolfCrypt にフィードします。これは、wolfTPM が PCR16 で TPM2_Extend 操作を発行するために使用されます。

```
$ ./examples/pcr/extend 16 /usr/bin/zip
...
TPM2_PCR_Extend success
PCR16 digest:
    2b bd 54 ae 08 5b 59 ef 90 42 d5 ca 5d df b5 b5 | +.T..[Y..B..]...
    74 3a 26 76 d4 39 37 eb b0 53 f5 82 67 6f b4 aa | t:&v.97..S..go..
```

拡張操作が完了すると、SysAdmin は PCR16 での測定の証明として TPM2.0 Quote を作成したいと考えています。

```
$ ./examples/pcr/quote 16 zip.quote
...
TPM2_Quote: success
TPM with signature attests (type 0x8018):
    TPM signed 1 count of PCRs
...
```

TPM2.0 Quote 操作の結果は、zip.quote バイナリ ファイルに保存されます。TPM 2.0 Quote の TPMS_ATTEST 構造には、有用なクロックと時刻の情報も含まれています。TPM 時刻認証の詳細については、./examples/timestamp/signed_timestamp の例を確認してください。

3.1.3.3 リモート認証チャレンジ PM 2.0 を使用してリモート認証チャレンジを作成し、その後応答を準備する方法を示します。

3.1.3.3.1 サンプルプログラムのリスト ./examples/attestation/ フォルダーには、特にリモート認証に関連するサンプルプログラムが含まれています。ただし、デモンストレーションでは、wolfTPM のソース コードにも含まれている keygen のサンプルプログラムを使用して TPM 2.0 鍵を作成する必要があります。

必要なサンプルプログラムの完全なリストを以下に示します：

- ./examples/attestation/make_credential: リモート認証チャレンジを作成するためにサーバーによって使用されます
- ./examples/attestation/activate_credential: クライアントがチャレンジを復号して応答するために使用

- ./examples/keygen/keygen: プライマリー鍵 (PK) と認証鍵 (AK) を作成するために使用されます

注: これらのサンプルプログラムではすべて、保証階層の下で保証鍵と認証鍵を使用できます。これは、上記の 3 つの例のいずれかを実行するときに-eh オプションを追加することによって行われます。EK/EH を使用する利点は、EK の秘密鍵マテリアルが TPM から離れないことです。EK の公開部分を使用して暗号化されたものはすべて、EK の TPM 所有者によって内部的にのみ暗号化でき、EK はすべての TPM チップに対して一意です。したがって、EK/EH を使用してリモート認証のチャレンジを作成することは、シナリオによってはより大きな価値があります。欠点の 1 つは、EK を使用すると、認証対象のホストの ID が常に知られることです。これは、EK の秘密鍵と公開鍵のペアが TPM を識別し、シナリオによっては、プライバシーの問題が発生する可能性があるためです。リモート認証の例では、SRK の下の AK と EK の下の AK の両方をサポートしています。どちらを使用するかは開発者次第です。

3.1.3.3.2 技術解説 リモート認証は、クライアントが既知の状態にあるかどうかを検証する認証サーバーに証拠を提供するクライアントのプロセスです。

このプロセスを実行するには、クライアントとサーバーが最初の信頼を確立する必要があります。これは、標準の TPM 2.0 コマンド MakeCredential および ActivateCredential を使用して実現されます。

1. ライアントは、TPM 2.0 Primary Attestation Key (PAK) と引用署名 Attestation Key (AK) の公開部分を転送します。
2. MakeCredential は、PAK の公開部分を使用してチャレンジ (シークレット) を暗号化します。通常、チャレンジは認証キー (AK) の公開部分のダイジェストです。

この方法では、PAK と AK のプライベート部分をロードできる TPM によってのみ、チャレンジを復号できます。PAK と AK は fixedTPM 鍵属性を使用して TPM にバインドされているため、これらの鍵をロードできる唯一の TPM は、それらが最初に作成された TPM です。

3. チャレンジが作成されると、サーバーからクライアントに転送されます。
4. ActivateCredential は、TPM 2.0 がロードされた PAK と AK を使用してチャレンジを復号し、シークレットを取得します。取得すると、クライアントはサーバー チャレンジに応答できます。

このようにして、クライアントはサーバーに対して、予想される TPM 2.0 システム ID と認証キーを所有していることを確認します。

ノート：

- チャレンジとレスポンスを交換するためのトランスポート プロトコルは、実装固有であるため、開発者が選択する必要があります。1 つのアプローチは、wolfSSL を使用した TLS1.3 クライアント/サーバー接続の使用です。

3.1.3.3.3 サンプルプログラムの使用方法 リモート認証用の TPM 2.0 キーの作成

keygen のサンプルプログラムを使用して、必要な TPM 2.0 構成証明鍵と、プライマリ構成証明鍵 (PAK) として使用される TPM 2.0 プライマリ ストレージ 鍵を作成できます。

```
$ ./examples/keygen/keygen -rsa
TPM2.0 Key generation example
    Key Blob: keyblob.bin
    Algorithm: RSA
    Template: AIK
    Use Parameter Encryption: NULL
Loading SRK: Storage 0x81000200 (282 bytes)
RSA AIK template
Creating new RSA key...
New key created and loaded (pub 280, priv 222 bytes)
Wrote 508 bytes to keyblob.bin
Wrote 288 bytes to srk.pub
```

```
Wrote AK Name digest
```

クレデンシャル作成サンプルプログラムの使用方法

make_credential のサンプルプログラムを使用して、認証サーバーはリモート認証チャレンジを生成できます。シークレットは、ランダムに生成された 32 バイトのシードであり、一部のリモート認証スキームで対称キーに使用できます。

```
$ ./examples/attestation/make_credential
Using public key from SRK to create the challenge
Demo how to create a credential challenge for remote attestation
Credential will be stored in cred.blob
wolfTPM2_Init: success
Reading 288 bytes from srk.pub
Reading the private part of the key
Public key for encryption loaded
Read AK Name digest success
TPM2_MakeCredential success
Wrote credential blob and secret to cred.blob, 648 bytes
```

クライアントと認証サーバー間の PAK および AK パブリック パーツの転送は、交換が実装固有であるため、make_credential の例の一部ではありません。

クレデンシャルのアクティベートサンプルプログラムの使用方法

activate_credential のサンプルプログラムを使用して、クライアントはリモート認証チャレンジを復号できます。シークレットは平文で公開され、認証サーバーと交換できます。

```
$ ./examples/attestation/activate_credential
Using default values
Demo how to create a credential blob for remote attestation
wolfTPM2_Init: success
Credential will be read from cred.blob
Loading SRK: Storage 0x81000200 (282 bytes)
SRK loaded
Reading 508 bytes from keyblob.bin
Reading the private part of the key
AK loaded at 0x80000001
Read credential blob and secret from cred.blob, 648 bytes
TPM2_ActivateCredential success
```

シークレットをプレーンで含む (または対称鍵シードとして使用される) チャレンジ レスポンスの転送は、activate_credential のサンプルプログラムの一部ではありません。これは、交換も実装固有であるためです。

3.1.4 パラメータ暗号化

3.1.4.1 暗号化された承認による鍵の生成 詳細情報は、“鍵生成”セクションの下にあります

3.1.4.2 暗号化された NVRAM 認証によるキーの安全なボールト 詳細情報は、このファイルのセクションの下にあります“Storing keys into the TPM’s NVRAM”

3.1.4.3 暗号化されたユーザー データを含む TPM2.0 クオート パラメータ暗号化を使用して、ホストと TPM の間のユーザー データを保護する方法を示すサンプルプログラム。

このサンプルプログラムでは、Quote 操作のためにユーザーが提供できる修飾データが保護されています。修飾データは、署名された Quote 構造に組み込まれた任意のデータです。パラメータの暗号化を使用することで、wolfTPM はホストがそのユーザー データを暗号化された形式で TPM に、またはその逆に転送できるようにします。したがって、中間者攻撃からデータを保護します。

TPM コマンドの最初のパラメーターのみを暗号化でき、パラメーターは「TPM2B_DATA」型である必要があります。たとえば、TPM 鍵のパスワード認証や TPM2.0 Quote の修飾データなどです。

コマンド要求と応答の暗号化は、一緒に実行することも、個別に実行することもできます。TPM とクライアント プログラムの間で、要求コマンドのパラメーターのみが暗号化される通信交換が可能です。

この動作は sessionAttributes に依存します:

・コマンドリクエストの TPMA_SESSION_encrypt ・コマンド応答用 TPMA_SESSION_decrypt

いずれかを個別に設定することも、両方を 1 つの認証セッションで設定することもできます。これはユーザー（開発者）次第です。

```
./examples/pcr/quote_paramenc
```

3.1.5 CSR

TPM 鍵ペアに基づいて証明書を作成するための証明書署名要求を生成します。

```
./examples/csr/csr
```

以下の 2 ファイルを生成します: ./certs/tpm-rsa-cert.csr ./certs/tpm-ecc-cert.csr

3.1.6 証明書への署名

TPM で生成された CSR に基づいてテスト証明書を生成するための外部スクリプト。通常、CSR は署名のために信頼できる CA に提供されます。

```
./certs/certreq.sh
```

このスクリプトは、次の X.509 ファイル (これも.pem 形式) を作成します: ./certs/ca-ecc-cert.der ./certs/ca-rsa-cert.der ./certs/client-rsa-cert.der ./certs/client-ecc-cert.der ./certs/server-rsa-cert.der ./certs/server-ecc-cert.der

3.1.7 PKCS #7

サンプルプログラムでは、TPM ベースの鍵を使用して PKCS #7 でデータに署名し、検証します。

- 最初に実行する必要があります:

1. ./examples/csr/csr
2. ./certs/certreq.sh
3. ./examples/pkcs7/pkcs7

結果はコンソールの stdout に表示されます。

3.1.8 TLS サンプルプログラム

TLS のサンプルプログラムでは、TPM ベースの ECDHE (ECC エフェメラル キー) サポートを使用しています。CFLAGS="-DWOLFTPM2_USE_SW_ECDHE" または #define WOLFTPM2_USE_SW_ECDHE を使用して無効にすることができます。また、パフォーマンスとスケーラビリティを向上させるために、2 フェーズの「TPM2_EC_Ephemeral」および「TPM2_ZGen_2Phase」メソッドの使用も検討しています。

RSA が有効な場合に wolfSSL で ECC の使用を強制するには、TLS_USE_ECC を定義します。

TPM で対称 AES/Hashing/HMAC を使用するには、WOLFTPM_USE_SYMMETRIC を定義します。

クライアント証明書とサーバー証明書を生成するには、次を実行する必要があります:

1. ./examples/keygen/keygen rsa_test_blob.raw -rsa -t
2. ./examples/keygen/keygen ecc_test_blob.raw -ecc -t
3. ./examples/csr/csr
4. ./certs/certreq.sh
5. CA ファイルを wolfTPM から wolfSSL certs ディレクトリにコピーします。
    a. cp ./certs/ca-ecc-cert.pem ../wolfssl/certs/tpm-ca-ecc-cert.pem
    b. cp ./certs/ca-rsa-cert.pem ../wolfssl/certs/tpm-ca-rsa-cert.pem

注:「wolf-ca-rsa-cert.pem」および「wolf-ca-ecc-cert.pem」ファイルは、こちらの wolfSSL サンプル証明書から取得されます:

```
cp ../wolfssl/certs/ca-cert.pem ./certs/wolf-ca-rsa-cert.pem
cp ../wolfssl/certs/ca-ecc-cert.pem ./certs/wolf-ca-ecc-cert.pem
```

3.1.8.1 TLS Client TLS 相互認証 (クライアント認証) に TPM キーと証明書を使用するサンプルプログラムを示します。

このサンプルプログラムのクライアントは、デフォルトでポート 11111 の localhost に接続します。これらは TLS_HOST と TLS_PORT を使用してオーバーライドできます。

次のように wolfSSL サンプル サーバーを使用して検証できます:

```
./examples/server/server -b -p 11111 -g -d -i -V
```

クライアント証明書を検証するには、次の wolfSSL サンプル サーバー コマンドを使用します: ./examples/server/server -b -p 11111 -g -A ./certs/tpm-ca-rsa-cert.pem -i -V または ./examples/server/server -b -p 11111 -g -A ./certs/tpm-ca-ecc-cert.pem -i -V

次に、wolfTPM TLS クライアントのサンプルプログラムを実行します: ./examples/tls/tls_client -rsa または ./examples/tls/tls_client -ecc

3.1.8.2 TLS Server このサンプルプログラムでは、TLS サーバーに TPM キーと証明書を使用する方法を示します。

デフォルトでは、ポート 11111 でリッスンし、ビルド時に TLS_PORT マクロを使用してオーバーライドできます。

wolfTPM TLS サーバーのサンプルプログラムを実行します: ./examples/tls/tls_server -rsa または ./examples/tls/tls_server -ecc

次に、wolfSSL サンプル クライアントを次のように実行します: ./examples/client/client -h localhost -p 11111 -g -d

サーバー証明書を検証するには、次の wolfSSL サンプル クライアント コメントを使用します: ./examples/client/client -h localhost -p 11111 -g -A ./certs/tpm-ca-rsa-cert.pem または ./examples/client/client -h localhost -p 11111 -g -A ./certs/tpm-ca-ecc-cert.pem

または、ブラウザを使用して: https://localhost:11111

ブラウザーでは、テスト CA の ./certs/ca-rsa-cert.pem および ./certs/ca-ecc-cert.pem を OS キー ストアにロードするまで、証明書の警告が表示されます。テスト用に、ほとんどのブラウザには警告を回避するためにとにかくサイトにアクセスし続ける方法があります。

3.1.9 クロック

TPM クロックの更新

TPM には、ユーザーに役立つ内部ハードウェア クロックがあります。時間に関して TPM が提供できる値は 2 つあります。

TPM 時間は、最後の電源投入シーケンス以降の現在の稼働時間です。この値は変更または修正できません。そのためのメカニズムはありません。値は電源シーケンスごとにリセットされます。

TPM クロックは、TPM の電源が投入された合計時間です。この値は、TPM2_ClockSet コマンドを使用して変更できます。TPM クロックは正転のみ設定できます。

このようにして、ユーザーは TPM クロックを使用して相対時刻と現在時刻を追跡できます。

注: 新しい時間値が TPM クロック更新間隔よりも大きな変更を行う場合、TPM は最初に時間の揮発性レジスタを更新し、次に時間の不揮発性レジスタを更新します。これにより、コマンドがユーザーに実行を返す前にわずかな遅延が発生する場合があります。TPM の製造元によっては、遅延が数ミリ秒から数ミリ秒まで異なる場合があります。

注: このサンプルプログラムでは、オプションの引数 (TPM クロックの増分に使用されるミリ秒単位の時間値) を取ることができます。デフォルト値は 50000ms (50 秒) です。

```
./examples/timestamp/clock_set
```

3.1.10 鍵生成

TPM 鍵 BLOB を生成してディスクに格納し、ディスクからロードして一時 TPM ハンドルにロードするサンプルプログラム。

```
$ ./examples/keygen/keygen keyblob.bin -rsa
TPM2.0 Key generation example
Loading SRK: Storage 0x81000200 (282 bytes)
Creating new RSA key...
Created new key (pub 280, priv 222 bytes)
Wrote 840 bytes to keyblob.bin

$ ./examples/keygen/keyload keyblob.bin
TPM2.0 Key load example
Loading SRK: Storage 0x81000200 (282 bytes)
Reading 840 bytes from keyblob.bin
Loaded key to 0x80000001


$ ./examples/keygen/keygen keyblob.bin -ecc
TPM2.0 Key generation example
Loading SRK: Storage 0x81000200 (282 bytes)
Creating new ECC key...
Created new key (pub 88, priv 126 bytes)
Wrote 744 bytes to keyblob.bin

$ ./examples/keygen/keyload keyblob.bin
TPM2.0 Key load example
Loading SRK: Storage 0x81000200 (282 bytes)
Reading 744 bytes from keyblob.bin
Loaded key to 0x80000001

./examples/keygen/keygen -sym=aescfb128
TPM2.0 Key generation example
    Key Blob: keyblob.bin
    Algorithm: SYMCIPHER
```

```
         aescfb mode, 128 keybits
    Template: Default
    Use Parameter Encryption: NULL
Loading SRK: Storage 0x81000200 (282 bytes)
Symmetric template
Creating new SYMCIPHER key...
Created new key (pub 50, priv 142 bytes)
Wrote 198 bytes to keyblob.bin

$ ./examples/keygen/keyload
TPM2.0 Key load example
    Key Blob: keyblob.bin
    Use Parameter Encryption: NULL
Loading SRK: Storage 0x81000200 (282 bytes)
Reading 198 bytes from keyblob.bin
Reading the private part of the key
Loaded key to 0x80000001
```

ファイル名が指定されていない場合、デフォルトのファイル名「keyblob.bin」が使用されるため、「keyload」と「keygen」をパラメーターを追加せずに使用して、TPM 2.0 鍵生成のデモを迅速に行うことができます。

「keygen」のサンプルプログラムでサポートされている暗号化アルゴリズムとオプションの完全なリストを表示するには、「–help」スイッチのいずれかを使用します。

秘密鍵を TPM キー BLOB としてインポートしてディスクに保存し、ディスクからロードして一時 TPM ハンドルにロードする例:

```
$ ./examples/keygen/keyimport keyblob.bin -rsa
TPM2.0 Key import example
Loading SRK: Storage 0x81000200 (282 bytes)
Imported key (pub 278, priv 222 bytes)
Wrote 840 bytes to keyblob.bin

$ ./examples/keygen/keyload keyblob.bin
TPM2.0 Key load example
Loading SRK: Storage 0x81000200 (282 bytes)
Reading 840 bytes from keyblob.bin
Loaded key to 0x80000001


$ ./examples/keygen/keyimport keyblob.bin -ecc
TPM2.0 Key Import example
Loading SRK: Storage 0x81000200 (282 bytes)
Imported key (pub 86, priv 126 bytes)
Wrote 744 bytes to keyblob.bin

$ ./examples/keygen/keyload keyblob.bin
TPM2.0 Key load example
Loading SRK: Storage 0x81000200 (282 bytes)
Reading 744 bytes from keyblob.bin
Loaded key to 0x80000001
```

keyload ツールは、格納された鍵のファイル名である引数を 1 つだけ受け取ります。キー スキーム (RSA または ECC) とは何かという情報が鍵 blob 内に含まれているためです。

3.1.11 鍵を TPM の NVRAM に保存

これらのサンプルプログラムは、TPM を鍵の安全なボールトとして使用する方法を示しています。TPM 鍵を TPM の NVRAM に格納するプログラムと、TPM の NVRAM から鍵を抽出するプログラムの 2 つのプログラムがあります。どちらのサンプルプログラムでも、パラメータ暗号化を使用して 中間者攻撃から保護できます。不揮発性メモリの場所は、コマンド ラインで「-aes」を指定すると、暗号化された形式で渡されるパスワード認証で保護されます。

サンプルを実行する前に、keygen ツールを使用して生成された keyblob.bin があることを確認してください。鍵は、RSA、ECC、または対称型の任意のタイプにすることができます。このサンプルプログラムでは、プライベート部分とパブリック部分を保存します。対称鍵の場合、公開部分は TPM からのメタ データです。上記の keygen のサンプルプログラムの説明にあるキーの生成方法。

パラメータ暗号化が有効な状態で RSA キーを保存してから読み取る場合の一般的な出力:

```
$ ./examples/nvram/store -aes
Parameter Encryption: Enabled (AES CFB).

TPM2_StartAuthSession: sessionHandle 0x2000000
Reading 840 bytes from keyblob.bin
Storing key at TPM NV index 0x1800202 with password protection

Public part = 616 bytes
NV write of public part succeeded

Private part = 222 bytes
Stored 2-byte size marker before the private part
NV write of private part succeeded


$ ./examples/nvram/read -aes
Parameter Encryption: Enabled (AES CFB).

TPM2_StartAuthSession: sessionHandle 0x2000000
Trying to read 616 bytes of public key part from NV
Successfully read public key part from NV

Trying to read size marker of the private key part from NV
Successfully read size marker from NV

Trying to read 222 bytes of private key part from NV
Successfully read private key part from NV

Extraction of key from NVRAM at index 0x1800202 succeeded
Loading SRK: Storage 0x81000200 (282 bytes)
Trying to load the key extracted from NVRAM
Loaded key to 0x80000001
```

「読み取り」のサンプルプログラムでは、鍵の公開部分と秘密部分の両方が NVRAM に格納されている場合、抽出された鍵をロードしようとします。「-aes」スイッチは、パラメーター暗号化の使用をトリガーします。

このサンプルプログラムでは、部分的なキー マテリアル (プライベートまたはパブリック) を使用できます。これは、「-priv」および「-pub」オプションを使用して実現されます。

RSA 非対称キー ペアの秘密キーのみを NVRAM に格納し、パラメータ暗号化を有効にしない場合の典型的な出力

```
$ ./examples/nvram/store -priv
```

```
Parameter Encryption: Not enabled (try -aes or -xor).

Reading 506 bytes from keyblob.bin
Reading the private part of the key
Storing key at TPM NV index 0x1800202 with password protection

Private part = 222 bytes
Stored 2-byte size marker before the private part
NV write of private part succeeded

$ ./examples/nvram/read -priv
Parameter Encryption: Not enabled (try -aes or -xor).

Trying to read size marker of the private key part from NV
Successfully read size marker from NV

Trying to read 222 bytes of private key part from NV
Successfully read private key part from NV

Extraction of key from NVRAM at index 0x1800202 succeeded
```

「読み取り」を使用して鍵の抽出が成功した後、NV インデックスは破棄されます。したがって、“read” を再度使用するには、“store” のサンプルプログラムも再度実行する必要があります。

3.1.12 シール・アンシール

TPM 2.0 は、標準のシール/アンシール手順を使用してシークレットを保護できます。シールは、TPM 2.0 鍵を使用して、または一連の PCR 値に対して作成できます。注: 鍵にシールされるシークレット データは、最大 128 バイトのサイズに制限されています。

使用可能なサンプルプログラムは、`seal/seal` と `seal/unseal` の 2 つです。

パラメーターなしで、デモの使用が可能です。

3.1.12.1 TPM 2.0 鍵へのデータのシール `seal` の例を使用して、新しく生成された TPM 2.0 鍵にデータを安全に保存します。この鍵が TPM にロードされた場合にのみ、シークレット データを読み取ることができます。

シークレット メッセージのシールとアンシールの出力例を示します:

```
$ ./examples/seal/seal keyblob.bin mySecretMessage
TPM2.0 Simple Seal example
    Key Blob: keyblob.bin
    Use Parameter Encryption: NULL
Loading SRK: Storage 0x81000200 (282 bytes)
Sealing the user secret into a new TPM key
Created new TPM seal key (pub 46, priv 141 bytes)
Wrote 193 bytes to keyblob.bin
Key Public Blob 46
Key Private Blob 141

$ ./examples/keygen/keyload -persistent
TPM2.0 Key load example
    Key Blob: keyblob.bin
    Use Parameter Encryption: NULL
Loading SRK: Storage 0x81000200 (282 bytes)
```

```
Reading 193 bytes from keyblob.bin
Reading the private part of the key
Loaded key to 0x80000001
Key was made persistent at 0x81000202

$ ./examples/seal/unseal message.raw
Example how to unseal data using TPM2.0
wolfTPM2_Init: success
Unsealing succeeded
Stored unsealed data to file = message.raw

$ cat message.raw
mySecretMessage
```

アンシールが成功すると、データは新しいファイルに保存されます。ファイル名が指定されていない場合、unseal ツールはデータを unseal.bin に保存します。

3.1.13 GPIO コントロール

一部の TPM 2.0 モジュールには、開発者が使用できる追加の I/O 機能と追加の GPIO があります。この余分な GPIO を使用して、セキュリティ イベントまたはシステム状態について他のサブシステムに通知できます。

現在、GPIO 制御の例は、ST33 および NPCT75x TPM 2.0 モジュールをサポートしています。

利用可能なサンプルプログラムは 4 つあります。gpio/gpio_config を使用した構成。

すべてのサンプルプログラムには、ヘルプ オプション -h があります。各種 GPIO モードについては gpio_config -h でご確認ください。

設定が完了すると、GPIO は gpio/gpio_set と gpio/gpio_read を使用して制御できます。

パラメータが指定されていない場合は、デモを使用できます。GPIO は物理的な世界と相互作用するため、慎重に選択したオプションを使用することをお勧めします。

3.1.13.1 GPIO コンフィグレーション

ST33 は、以下の gpio/gpio_config からの 6 つのモードをサポートします：

```
$ ./examples/gpio/gpio_config -h
Expected usage:
./examples/gpio/gpio_config [num] [mode]
* num is a GPIO number between 0-3 (default 0)
* mode is a number selecting the GPIO mode between 0-6 (default 3):
    0. standard - reset to the GPIO's default mode
    1. floating - input in floating configuration.
    2. pullup   - input with pull up enabled
    3. pulldown - input with pull down enabled
    4. opendrain - output in open drain configuration
    5. pushpull  - output in push pull configuration
    6. unconfigure - delete the NV index for the selected GPIO
Example usage, without parameters, configures GPIO0 as input with a pull down.
```

GPIO を出力に設定するための使用例を以下に示します：

```
$ ./examples/gpio/gpio_config 0 5
GPIO num is: 0
GPIO mode is: 5
```

```
Example how to use extra GPIO on a TPM 2.0 modules
Trying to configure GPIO0...
TPM2_GPIO_Config success
NV Index for GPIO access created
```

ST33 のプルアップを使用して GPIO を入力として構成するための使用例を以下に示します。

```
$ ./examples/gpio/gpio_config 0 3
GPIO num is: 0
GPIO mode is: 3
Demo how to use extra GPIO on a TPM 2.0 modules
Trying to configure GPIO0...
TPM2_GPIO_Config success
NV Index for GPIO access created
```

3.1.13.2 GPIO Config (NPCT75xx) NPCT75x は 3 つの出力モード (入力モードなし) をサポートします。以下の gpio/gpio_config からの情報:

```
$ ./examples/gpio/gpio_config -h
Expected usage:
./examples/gpio/gpio_config [num] [mode]
* num is a GPIO number between 3 and 4 (default 3)
* mode is either push-pull, open-drain or open-drain with pull-up
    1. pushpull  - output in push pull configuration
    2. opendrain - output in open drain configuration
    3. pullup - output in open drain with pull-up enabled
    4. unconfig - delete NV index for GPIO access
Example usage, without parameters, configures GPIO3 as push-pull output.
```

NPCT75x の GPIO 番号付けは GPIO3 から始まり、ST33 は GPIO0 から始まることに注意してください。

```
$ ./examples/gpio/gpio_nuvoton 4 1
Example for GPIO configuration of a NPTC7xx TPM 2.0 module
GPIO number: 4
GPIO mode: 1
Successfully read the current configuration
Successfully wrote new configuration
NV Index for GPIO access created
```

3.1.13.3 GPIO 使用方法 GPIO 構成の切り替えはシームレスです。* ST33 の場合、gpio/gpio_config は既存の NV インデックスの削除を処理するため、新しい GPIO 構成を選択できます。* NPCT75xx の場合、gpio/gpio_config は、作成された NV インデックスを削除せずに任意の GPIO を再構成できます。

```
$ ./examples/gpio/gpio_set 0 -high
GPIO0 set to high level

$ ./examples/gpio/gpio_set 0 -low
GPIO0 set to low level

$ ./examples/gpio/gpio_read 0
GPIO0 is Low
```

3.2 ベンチマーク

wolfTPM ベンチマーク アプリケーションには、サンプル アプリケーションと同じセットアップが必要です。

注: 鍵生成は、階層シードからの既存のテンプレートを使用しています。

Infineon OPTIGA SLB9670 43MHz 上で実行:

```
./examples/bench/bench
TPM2 Benchmark using Wrapper API's
RNG                  16 KB took 1.140 seconds,   14.033 KB/s
Benchmark symmetric AES-128-CBC-enc not supported!
Benchmark symmetric AES-128-CBC-dec not supported!
Benchmark symmetric AES-256-CBC-enc not supported!
Benchmark symmetric AES-256-CBC-dec not supported!
Benchmark symmetric AES-128-CTR-enc not supported!
Benchmark symmetric AES-128-CTR-dec not supported!
Benchmark symmetric AES-256-CTR-enc not supported!
Benchmark symmetric AES-256-CTR-dec not supported!
Benchmark symmetric AES-256-CFB-enc not supported!
Benchmark symmetric AES-256-CFB-dec not supported!
SHA1                138 KB took 1.009 seconds,  136.783 KB/s
SHA256              138 KB took 1.009 seconds,  136.763 KB/s
RSA     2048 key gen        5 ops took 10.981 sec, avg 2196.230 ms, 0.455 ops/
   sec
RSA     2048 Public       113 ops took 1.005 sec, avg 8.893 ms,   112.449 ops/
   sec
RSA     2048 Private        7 ops took 1.142 sec, avg 163.207 ms,   6.127 ops/
   sec
RSA     2048 Pub  OAEP     73 ops took 1.011 sec, avg 13.848 ms,   72.211 ops/
   sec
RSA     2048 Priv OAEP      6 ops took 1.004 sec, avg 167.399 ms,   5.974 ops/
   sec
ECC      256 key gen        5 ops took 1.157 sec, avg 231.350 ms,   4.322 ops/
   sec
ECDSA    256 sign          15 ops took 1.033 sec, avg 68.865 ms,   14.521 ops/
   sec
ECDSA    256 verify         9 ops took 1.022 sec, avg 113.539 ms,   8.808 ops/
   sec
ECDHE    256 agree          5 ops took 1.161 sec, avg 232.144 ms,   4.308 ops/
   sec
```

ST ST33TP SPI 33MHz 上で実行:

```
./examples/bench/bench
TPM2 Benchmark using Wrapper API's
RNG                  14 KB took 1.017 seconds,   13.763 KB/s
AES-128-CBC-enc      40 KB took 1.008 seconds,   39.666 KB/s
AES-128-CBC-dec      42 KB took 1.032 seconds,   40.711 KB/s
AES-256-CBC-enc      40 KB took 1.013 seconds,   39.496 KB/s
AES-256-CBC-dec      40 KB took 1.011 seconds,   39.563 KB/s
AES-128-CTR-enc      26 KB took 1.055 seconds,   24.646 KB/s
AES-128-CTR-dec      26 KB took 1.035 seconds,   25.117 KB/s
AES-256-CTR-enc      26 KB took 1.028 seconds,   25.302 KB/s
AES-256-CTR-dec      26 KB took 1.030 seconds,   25.252 KB/s
AES-128-CFB-enc      42 KB took 1.045 seconds,   40.201 KB/s
```

```
AES-128-CFB-dec     40 KB took 1.008 seconds,   39.699 KB/s
AES-256-CFB-enc     40 KB took 1.022 seconds,   39.151 KB/s
AES-256-CFB-dec     42 KB took 1.041 seconds,   40.362 KB/s
SHA1                86 KB took 1.005 seconds,   85.559 KB/s
SHA256              84 KB took 1.019 seconds,   82.467 KB/s
RSA     2048 key gen        1 ops took 7.455 sec, avg 7455.036 ms, 0.134 ops/
   sec
RSA     2048 Public       110 ops took 1.003 sec, avg 9.122 ms,  109.624 ops/
   sec
RSA     2048 Private        5 ops took 1.239 sec, avg 247.752 ms,  4.036 ops/
   sec
RSA     2048 Pub  OAEP     81 ops took 1.001 sec, avg 12.364 ms,  80.880 ops/
   sec
RSA     2048 Priv OAEP      4 ops took 1.007 sec, avg 251.780 ms,  3.972 ops/
   sec
ECC      256 key gen        5 ops took 1.099 sec, avg 219.770 ms,  4.550 ops/
   sec
ECDSA    256 sign          24 ops took 1.016 sec, avg 42.338 ms,  23.619 ops/
   sec
ECDSA    256 verify        14 ops took 1.036 sec, avg 74.026 ms,  13.509 ops/
   sec
ECDHE    256 agree          5 ops took 1.235 sec, avg 247.085 ms,  4.047 ops/
   sec
```

Microchip ATTPM20 33MHz 上で実行:

```
./examples/bench/bench
TPM2 Benchmark using Wrapper API's
RNG                  2 KB took 1.867 seconds,    1.071 KB/s
Benchmark symmetric AES-128-CBC-enc not supported!
Benchmark symmetric AES-128-CBC-dec not supported!
Benchmark symmetric AES-256-CBC-enc not supported!
Benchmark symmetric AES-256-CBC-dec not supported!
Benchmark symmetric AES-128-CTR-enc not supported!
Benchmark symmetric AES-128-CTR-dec not supported!
Benchmark symmetric AES-256-CTR-enc not supported!
Benchmark symmetric AES-256-CTR-dec not supported!
AES-128-CFB-enc     16 KB took 1.112 seconds,   14.383 KB/s
AES-128-CFB-dec     16 KB took 1.129 seconds,   14.166 KB/s
AES-256-CFB-enc     12 KB took 1.013 seconds,   11.845 KB/s
AES-256-CFB-dec     12 KB took 1.008 seconds,   11.909 KB/s
SHA1                22 KB took 1.009 seconds,   21.797 KB/s
SHA256              22 KB took 1.034 seconds,   21.270 KB/s
RSA     2048 key gen        3 ops took 15.828 sec, avg 5275.861 ms, 0.190 ops/
   sec
RSA     2048 Public        22 ops took 1.034 sec, avg 47.021 ms, 21.267 ops/
   sec
RSA     2048 Private        9 ops took 1.059 sec, avg 117.677 ms, 8.498 ops/
   sec
RSA     2048 Pub  OAEP     21 ops took 1.007 sec, avg 47.959 ms, 20.851 ops/
   sec
RSA     2048 Priv OAEP      9 ops took 1.066 sec, avg 118.423 ms, 8.444 ops/
   sec
ECC      256 key gen        7 ops took 1.072 sec, avg 153.140 ms, 6.530 ops/
   sec
```

```
ECDSA    256 sign          18 ops took 1.056 sec, avg 58.674 ms, 17.043 ops/
   sec
ECDSA    256 verify        24 ops took 1.031 sec, avg 42.970 ms, 23.272 ops/
   sec
ECDHE    256 agree         16 ops took 1.023 sec, avg 63.934 ms, 15.641 ops/
   sec
```

Nations Technologies Inc. TPM 2.0 モジュール 33MHz 上で実行:

```
./examples/bench/bench
TPM2 Benchmark using Wrapper API's
RNG                 12 KB took 1.065 seconds,   11.270 KB/s
AES-128-CBC-enc     48 KB took 1.026 seconds,   46.780 KB/s
AES-128-CBC-dec     48 KB took 1.039 seconds,   46.212 KB/s
AES-256-CBC-enc     48 KB took 1.035 seconds,   46.370 KB/s
AES-256-CBC-dec     48 KB took 1.025 seconds,   46.852 KB/s
Benchmark symmetric AES-128-CTR-enc not supported!
Benchmark symmetric AES-128-CTR-dec not supported!
Benchmark symmetric AES-256-CTR-enc not supported!
Benchmark symmetric AES-256-CTR-dec not supported!
AES-128-CFB-enc     50 KB took 1.029 seconds,   48.591 KB/s
AES-128-CFB-dec     50 KB took 1.035 seconds,   48.294 KB/s
AES-256-CFB-enc     48 KB took 1.000 seconds,   47.982 KB/s
AES-256-CFB-dec     48 KB took 1.003 seconds,   47.855 KB/s
SHA1                80 KB took 1.009 seconds,   79.248 KB/s
SHA256              80 KB took 1.004 seconds,   79.702 KB/s
SHA384              78 KB took 1.018 seconds,   76.639 KB/s
RSA     2048 key gen        8 ops took 17.471 sec, avg 2183.823 ms, 0.458 ops/
   sec
RSA     2048 Public        52 ops took 1.004 sec, avg 19.303 ms, 51.805 ops/
   sec
RSA     2048 Private        8 ops took 1.066 sec, avg 133.243 ms, 7.505 ops/
   sec
RSA     2048 Pub  OAEP     51 ops took 1.001 sec, avg 19.621 ms, 50.966 ops/
   sec
RSA     2048 Priv OAEP      8 ops took 1.073 sec, avg 134.182 ms, 7.453 ops/
   sec
ECC      256 key gen       20 ops took 1.037 sec, avg 51.871 ms, 19.279 ops/
   sec
ECDSA    256 sign          43 ops took 1.006 sec, avg 23.399 ms, 42.736 ops/
   sec
ECDSA    256 verify        28 ops took 1.030 sec, avg 36.785 ms, 27.185 ops/
   sec
ECDHE    256 agree         26 ops took 1.010 sec, avg 38.847 ms, 25.742 ops/
   sec
```

Nuvoton NPCT650 で実行:

```
./examples/bench/bench
TPM2 Benchmark using Wrapper API's
RNG                  8 KB took 1.291 seconds,    6.197 KB/s
Benchmark symmetric AES-128-CBC-enc not supported!
Benchmark symmetric AES-128-CBC-dec not supported!
Benchmark symmetric AES-256-CBC-enc not supported!
Benchmark symmetric AES-256-CBC-dec not supported!
```

```
Benchmark symmetric AES-256-CTR-enc not supported!
Benchmark symmetric AES-256-CTR-dec not supported!
Benchmark symmetric AES-256-CFB-enc not supported!
Benchmark symmetric AES-256-CFB-dec not supported!
SHA1                90 KB took 1.005 seconds,   89.530 KB/s
SHA256              90 KB took 1.010 seconds,   89.139 KB/s
RSA     2048 key gen        8 ops took 35.833 sec, avg 4479.152 ms, 0.223 ops/
   sec
RSA     2048 Public        77 ops took 1.007 sec, avg 13.078 ms, 76.463 ops/
   sec
RSA     2048 Private        2 ops took 1.082 sec, avg 540.926 ms, 1.849 ops/
   sec
RSA     2048 Pub  OAEP     53 ops took 1.005 sec, avg 18.961 ms, 52.739 ops/
   sec
RSA     2048 Priv OAEP      2 ops took 1.088 sec, avg 544.075 ms, 1.838 ops/
   sec
ECC      256 key gen        7 ops took 1.033 sec, avg 147.608 ms, 6.775 ops/
   sec
ECDSA    256 sign           6 ops took 1.141 sec, avg 190.149 ms, 5.259 ops/
   sec
ECDSA    256 verify         4 ops took 1.061 sec, avg 265.216 ms, 3.771 ops/
   sec
ECDHE    256 agree          6 ops took 1.055 sec, avg 175.915 ms, 5.685 ops/
   sec
```

Nuvoton NPCT750 43MHz 上で実行:

```
RNG                 16 KB took 1.114 seconds,   14.368 KB/s
Benchmark symmetric AES-128-CBC-enc not supported!
Benchmark symmetric AES-128-CBC-dec not supported!
Benchmark symmetric AES-256-CBC-enc not supported!
Benchmark symmetric AES-256-CBC-dec not supported!
SHA1               120 KB took 1.012 seconds,  118.618 KB/s
SHA256             122 KB took 1.012 seconds,  120.551 KB/s
SHA384             120 KB took 1.003 seconds,  119.608 KB/s
RSA     2048 key gen        5 ops took 17.043 sec, avg 3408.678 ms, 0.293 ops/
   sec
RSA     2048 Public       134 ops took 1.004 sec, avg 7.490 ms, 133.517 ops/
   sec
RSA     2048 Private       15 ops took 1.054 sec, avg 70.261 ms, 14.233 ops/
   sec
RSA     2048 Pub  OAEP    116 ops took 1.002 sec, avg 8.636 ms, 115.797 ops/
   sec
RSA     2048 Priv OAEP     15 ops took 1.061 sec, avg 70.716 ms, 14.141 ops/
   sec
ECC      256 key gen       12 ops took 1.008 sec, avg 84.020 ms, 11.902 ops/
   sec
ECDSA    256 sign          18 ops took 1.015 sec, avg 56.399 ms, 17.731 ops/
   sec
ECDSA    256 verify        26 ops took 1.018 sec, avg 39.164 ms, 25.533 ops/
   sec
ECDHE    256 agree         35 ops took 1.029 sec, avg 29.402 ms, 34.011 ops/
   sec
```

4 wolfTPM ライブラリデザイン

4.1 ライブラリヘッダーファイル

wolfTPM のヘッダーファイルは以下の場所に格納されています：

wolfTPM : wolftpm/

wolfSSL : wolfssl/

wolfCrypt : wolfssl/wolfcrypt

wolfTPM がインクルードすべき一般的なヘッダーファイルは次のファイルです：

```
#include <wolftpm/tpm2.h>
```

4.2 サンプルプログラムの設計

wolfTPM に含まれるすべてのサンプル アプリケーションには、wolfTPM/examples にある tpm_io.h ヘッダ ファイルをインクルードします。tpm_io.c ファイルは、サンプル アプリケーションを Linux カーネル、STM32 CubeMX HAL、または Atmel/Microchip ASF でテストおよび実行するために必要なサンプル HAL IO コールバックをセットアップします。カスタム IO コールバックまたは別のコールバックを必要に応じて追加または削除できるように、参照は簡単に変更できます。

4.3 # API Reference

4.4 TPM2 固有

このモジュールでは、wolfTPM のみに固有の TPM2 コマンドについて説明します。通常、これらのコマンドには、TPM 2.0 データ構造を処理するためのヘルパーが含まれています。開発時のデバッグやテストに役立つ機能もあります。

4.4.1 関数

戻り値	関数および機能概要
WOLFTPM_API TPM_RC	TPM2_Init(TPM2_CTX* ctx, TPM2HalIoCb ioCb, void* userCtx) HAL IO コールバックとユーザー指定のコンテキストで TPM を初期化します。–enable-devtpm または –enable-swtpm 構成で wolfTPM を使用する場合、ioCb と userCtx は使用されません。
WOLFTPM_API TPM_RC	TPM2_Init_ex(TPM2_CTX* ctx, TPM2HalIoCb ioCb, void* userCtx,int timeoutTries) timeoutTries、HAL IO コールバック、およびユーザー指定のコンテキストで TPM を初期化します。
WOLFTPM_API TPM_RC	TPM2_Init_minimal(TPM2_CTX* ctx) TPM を初期化し、使用される wolfTPM2 コンテキストを設定します。この関数は通常、Windows などのリッチ オペレーティング システムで使用されます。

戻り値	関数および機能概要
WOLFTPM_API TPM_RC	TPM2_Cleanup(TPM2_CTX* ctx) TPM と wolfcrypt を初期化解除します (初期化されている場合)
WOLFTPM_API TPM_RC	TPM2_ChipStartup(TPM2_CTX* ctx, int timeoutTries) TPM2 の起動が完了したことを確認し、TPM デバイス情報を抽出します。
WOLFTPM_API TPM_RC	TPM2_SetHalIoCb(TPM2_CTX* ctx, TPM2HalIoCb ioCb, void* userCtx) TPM 通信に必要なユーザーのコンテキストと IO コールバックを設定します。
WOLFTPM_API TPM_RC	TPM2_SetSessionAuth(TPM2_AUTH_SESSION* session) TPM 承認を保持する構造体を設定します。
WOLFTPM_API int	TPM2_GetSessionAuthCount(TPM2_CTX* ctx) 現在設定されている TPM 承認の数を特定します。
WOLFTPM_API void	TPM2_SetActiveCtx(TPM2_CTX* ctx) 使用する新しい TPM2 コンテキストを設定します。
WOLFTPM_API TPM2_CTX*	TPM2_GetActiveCtx(void) 使用中の TPM2 コンテキストへのポインターを提供します。
WOLFTPM_API int	TPM2_GetHashDigestSize(TPMI_ALG_HASH hashAlg) TPM 2.0 ハッシュ ダイジェストのサイズをバイト単位で決定します。
WOLFTPM_API int	TPM2_GetHashType(TPMI_ALG_HASH hashAlg) TPM2 ハッシュ タイプを対応する wolfcrypt ハッシュ タイプに変換します。
WOLFTPM_API int	TPM2_GetNonce(byte* nonceBuf, int nonceSz) 乱数の新しいナンスを生成します。
WOLFTPM_API void	TPM2_SetupPCRSel(TPML_PCR_SELECTION* pcr, TPM_ALG_ID alg,int pcrIndex) TPM2_Quote を作成する準備をする場合などに、正しい PCR 選択を準備するためのヘルパー関数
const WOLFTPM_API char *	TPM2_GetRCString(int rc) TPM 2.0 リターン コードの人間が判読できる文字列を取得します。
const WOLFTPM_API char *	TPM2_GetAlgName alg)(TPM_ALG_ID alg) 任意の TPM 2.0 アルゴリズムについて、人間が判読できる文字列を取得します。
WOLFTPM_API int	TPM2_GetCurveSize(TPM_ECC_CURVE curveID) TPM ECC 曲線のサイズをバイト単位で決定します。
WOLFTPM_API int	TPM2_GetTpmCurve(int curveID) wolfcrypt 曲線タイプを対応する TPM 曲線タイプに変換します。
WOLFTPM_API int	TPM2_GetWolfCurve(int curve_id) TPM 曲線タイプを対応する wolfcrypt 曲線タイプに変換します。
WOLFTPM_API int	TPM2_ParseAttest(const TPM2B_ATTEST* in, TPMS_ATTEST* out) TPM2B_ATTEST を解析し、TPMS_ATTEST 構造にデータを入力します。
WOLFTPM_API int	TPM2_HashNvPublic(TPMS_NV_PUBLIC* nvPublic, byte* buffer, UINT16* size) nvPublic 構造に基づいて新しい NV インデックス名を計算します。

戻り値	関数および機能概要
WOLFTPM_API int	TPM2_AppendPublic(byte* buf, word32 size, int* sizeUsed, TPM2B_PUBLIC* pub) ユーザー提供のバッファーに基づいて TPM2B_PUBLIC 構造体を設定します。
WOLFTPM_API int	TPM2_ParsePublic(TPM2B_PUBLIC* pub, byte* buf, word32 size, int* sizeUsed) TPM2B_PUBLIC 構造を解析し、ユーザー提供のバッファーに格納します。
WOLFTPM_LOCAL int	TPM2_GetName(TPM2_CTX* ctx, UINT32 handleValue, int handleCnt, int idx, TPM2B_NAME* name) TPM オブジェクトの名前を提供します
WOLFTPM_API UINT16	TPM2_GetVendorID(void) アクティブな TPM2 コンテキストの vendorID を提供します
WOLFTPM_API void	TPM2_PrintBin(const byte* buffer, word32 length) フォーマットされた方法でバイナリ バッファを出力するヘルパー関数
WOLFTPM_API void	TPM2_PrintAuth(const TPMS_AUTH_COMMAND* authCmd) 人間が読める方法で TPMS_AUTH_COMMAND 型の構造を出力するヘルパー関数
WOLFTPM_API void	TPM2_PrintPublicArea(const TPM2B_PUBLIC* pub) 人間が読める方法で TPM2B_PUBLIC 型の構造を出力するヘルパー関数

4.4.2 詳細な説明

このモジュールでは、wolfTPM のみに固有の TPM2 コマンドについて説明します。

通常、これらのコマンドには、TPM 2.0 データ構造を処理するためのヘルパーが含まれています。

開発時のデバッグやテストに役立つ機能もあります。

4.4.3 関数のドキュメント

TPM2_Init

```
WOLFTPM_API TPM_RC TPM2_Init(
    TPM2_CTX * ctx,
    TPM2HalIoCb ioCb,
    void * userCtx
)
```

HAL IO コールバックとユーザー指定のコンテキストで TPM を初期化します。–enable-devtpm または –enable-swtpm 構成で wolfTPM を使用する場合、ioCb と userCtx は使用されません。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター
- **ioCb** TPM2HalIoCb (HAL IO) コールバック関数ポインタ
- **userCtx** ユーザーコンテキストへのポインター

参考:

- TPM2_Startup

- TPM2_GetRCString
- TPM2_Init_minimal
- TPM2_Init_ex
- wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー（IO 関係のエラー）
- BAD_FUNC_ARG：不正な引数

ノート:

TPM2_Init_minimal() ioCb と userCtx の両方を NULL にセットします。その他のモードでは、TIS を使用するために ioCb を設定する必要があります。ベアメタルおよび RTOS アプリケーションのサンプル ioCB は、examples/tpm_io.c で提供されています。

使用例

```
int rc;
TPM2_CTX tpm2Ctx;

rc = TPM2_Init(&tpm2Ctx, TPM2_IoCb, userCtx);
if (rc != TPM_RC_SUCCESS) {
    // TPM2_Init failed
}
```

TPM2_Init_ex

```
WOLFTPM_API TPM_RC TPM2_Init_ex(
    TPM2_CTX * ctx,
    TPM2HalIoCb ioCb,
    void * userCtx,
    int timeoutTries
)
```

timeoutTries、HAL IO コールバック、およびユーザー指定のコンテキストで TPM を初期化します。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター

- **ioCb** TPM2HalIoCb (HAL IO) コールバック関数ポインター
- **userCtx** ユーザーコンテキストへのポインター
- **timeoutTries** TPM2 の起動が完了したことを確認するための試行回数を指定します

参考:

- TPM2_GetRCString
- TPM2_Init_minimal
- TPM2_Init
- wolfTPM2_Init_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: general error (possibly IO)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM2_Init_minimal を直接使用する代わりに、TPM2_Init を使用することをお勧めします。

TPM2_Init_minimal

```
WOLFTPM_API TPM_RC TPM2_Init_minimal(
    TPM2_CTX * ctx
)
```

TPM を初期化し、使用される wolfTPM2 コンテキストを設定します。この関数は通常、Windows などのリッチ オペレーティング システムで使用されます。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター

参考:

- TPM2_GetRCString
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: general error (possibly IO)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM2_Init_minimal を直接使用する代わりに、TPM2_Init を使用することをお勧めします。

TPM2_Cleanup

```
WOLFTPM_API TPM_RC TPM2_Cleanup(
    TPM2_CTX * ctx
)
```

TPM と wolfcrypt を初期化解除します (初期化されている場合)

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター

参考:

- TPM2_GetRCString
- TPM2_Init
- wolfTPM2_Cleanup

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: could not acquire the lock on the wolfTPM2 context
- BAD_FUNC_ARG: 不正な引数（NULL を渡した）

使用例

```
int rc;
TPM2_CTX tpm2Ctx;

rc = TPM2_Cleanup(&tpm2Ctx->dev);
if (rc != TPM_RC_SUCCESS) {
    // TPM2_Cleanup failed
}
```

TPM2_ChipStartup

```
WOLFTPM_API TPM_RC TPM2_ChipStartup(
    TPM2_CTX * ctx,
    int timeoutTries
)
```

TPM2 の起動が完了していることを確認し、TPM デバイス情報を抽出します。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター
- **timeoutTries** TPM2 のスタートアップが完了したかをチェックする回数

参考:

- TPM2_GetRCString
- TPM2_TIS_StartupWait
- TPM2_TIS_RequestLocality
- TPM2_TIS_GetInfo
- TPM2_Init_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (IO 関係のエラー？)
- BAD_FUNC_ARG: 不正な引数
- TPM_RC_TIMEOUT: タイムアウト

ノート:

この関数は TPM2_Init_ex で使用されます。

TPM2_SetHalIoCb

```
WOLFTPM_API TPM_RC TPM2_SetHalIoCb(
    TPM2_CTX * ctx,
    TPM2HalIoCb ioCb,
    void * userCtx
)
```

TPM 通信に必要なユーザーのコンテキストと IO コールバックを設定します。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター

- **ioCb** TPM2HalIoCb (HAL IO) コールバック関数
- **userCtx** ユーザーコンテキストへのポインター

参考:

- TPM2_GetRCString
- TPM2_Init
- wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: TPM2_CTX 構造体の排他ロックを取得できなかった
- BAD_FUNC_ARG: 不正な引数（ctx に NULL ポインターを渡した）

ノート:

コールバックが TPM に使用されないため、devtpm または swtpm でビルドされた場合、SetHalIoCb は失敗します。他の構成ビルドでは、ioCb を NULL 以外の関数ポインターに設定する必要があり、userCtx はオプションです。

通常、TPM2_Init または wolfTPM2_Init を使用して HAL IO を設定します。

TPM2_SetSessionAuth

```
WOLFTPM_API TPM_RC TPM2_SetSessionAuth(
    TPM2_AUTH_SESSION * session
)
```

TPM 承認を保持する構造体を設定します。

パラメータ:

- **session** TPM2_AUTH_SESSION 配列へのポインター

参考:

- TPM2_GetRCString
- TPM2_Init
- wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: TPM2_CTX 構造体の排他ロックを取得できなかった
- BAD_FUNC_ARG: 不正な引数（ctx に NULL ポインターを渡した）

TPM2_Init 関数と wolfTPM2_Init もこの初期化を実行するため、この関数を明示的に呼び出すことはまれです。TPM 2.0 コマンドは最大 3 つの認証スロットを持つことができるため、サイズ MAX_SESSION_NUM の配列を TPM2_SetSessionAuth に提供することをお勧めします (以下の例を参照)。

使用例

```
int rc;
TPM2_AUTH_SESSION session[MAX_SESSION_NUM];

XMEMSET(session, 0, sizeof(session));
session[0].sessionHandle = TPM_RS_PW;

rc = TPM2_SetSessionAuth(session);
if (rc != TPM_RC_SUCCESS) {
    // TPM2_SetSessionAuth failed
}
```

TPM2_GetSessionAuthCount

```
WOLFTPM_API int TPM2_GetSessionAuthCount(
    TPM2_CTX * ctx
)
```

現在設定されている TPM 承認の数を確認します。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター

参考:

- TPM2_CTX
- TPM2_AUTH_SESSION

戻り値:

- アクティブな TPM 認証数 (1～3)
- BAD_FUNC_ARG: 不正な引数

使用例

```
int authCount;
TPM2_CTX tpm2Ctx;

authCount = TPM2_GetSessionAuthCount(tpm2ctx);
if (authCount == BAD_FUNC_ARG) {
    // TPM2_GetSessionAuthCount failed
}
```

TPM2_SetActiveCtx

```
WOLFTPM_API void TPM2_SetActiveCtx(
    TPM2_CTX * ctx
)
```

新たな TPM2_CTX を設定する。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター

参考:

- TPM2_CTX
- TPM2_AUTH_SESSION

使用例

```
TPM2_CTX tpm2Ctx;

TPM2_SetActiveCtx(tpm2ctx);
```

TPM2_GetActiveCtx

```
WOLFTPM_API TPM2_CTX * TPM2_GetActiveCtx(
    void
)
```

TPM2_CTX へのポインターを得る。

参考:

- TPM2_CTX
- TPM2_AUTH_SESSION

戻り値:

TPM2_CTX 構造体へのポインター

使用例

```
TPM2_CTX *tpm2Ctx;

tpm2Ctx = TPM2_GetActiveCtx();
```

TPM2_GetHashDigestSize

```
WOLFTPM_API int TPM2_GetHashDigestSize(
    TPMI_ALG_HASH hashAlg
)
```

TPM 2.0 ハッシュダイジェストのサイズ（バイト数）を得る。

パラメータ:

- **hashAlg** TPM 2.0 ハッシュタイプ

戻り値:

- TPM 2.0 ハッシュダイジェストのサイズ（バイト数）
- ハッシュタイプが不正の場合は 0 を返す

使用例

```
int digestSize = 0;
TPMI_ALG_HASH hashAlg = TPM_ALG_SHA256;

digestSize = TPM2_GetHashDigestSize(hashAlg);
if (digestSize > 0) {
    //digestSize contains a valid value
}
```

TPM2_GetHashType

```
WOLFTPM_API int TPM2_GetHashType(
    TPMI_ALG_HASH hashAlg
)
```

TPM2 ハッシュタイプは対応する wolfcrypt のハッシュタイプに変換する。

パラメータ:

- **hashAlg** TPM 2.0 ハッシュタイプ

戻り値:

- wolfcrypt での定義によるハッシュタイプの値
- ハッシュタイプが不正の場合は 0 を返す

使用例

```
int wc_hashType;
TPMI_ALG_HASH hashAlg = TPM_ALG_SHA256;

wc_hashType = TPM2_GetHashDigestSize(hashAlg);
if (wc_hashType > 0) {
    //wc_hashType contains a valid wolfcrypt hash type
}
```

TPM2_GetNonce

```
WOLFTPM_API int TPM2_GetNonce(
    byte * nonceBuf,
    int nonceSz
)
```

乱数の新しいナンスを生成します。

パラメータ:

- **nonceBuf** BYTE バッファーへのポインター
- **nonceSz** ナンスのサイズ（バイト数）

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO での問題あるいは wolfcrypt のコンフィギュレーション上の問題)
- BAD_FUNC_ARG: 不正な引数

ノート:

WOLFTPM2_USE_HW_RNG が定義されている場合、TPM 乱数ジェネレーターを使用できます。

使用例

```
int rc, nonceSize = 32;
BYTE freshNonce[32];

rc = TPM2_GetNonce(&freshNonce, nonceSize);
if (rc != TPM_RC_SUCCESS) {
    //TPM2_GetNonce failed
}
```

TPM2_SetupPCRSel

```
WOLFTPM_API void TPM2_SetupPCRSel(
    TPML_PCR_SELECTION * pcr,
    TPM_ALG_ID alg,
    int pcrIndex
)
```

正しい PCR 選択を準備するためのヘルパー関数 たとえば、TPM2_Quote を作成する準備をする場合。

パラメータ:

- **pcr** TPML_PCR_SELECTION 構造体へのポインター
- **alg** ハッシュアルゴリズムを示す TPM_ALG_ID
- **pcrIndex** 使用する PCR レジスターを示す 0 から 23 までの値

参考:

- TPM2_PCR_Read
- TPM2_PCR_Extend
- TPM2_PCR_Reset
- TPM2_Quote

使用例

```
int pcrIndex = 16; // This is a PCR register for DEBUG & testing purposes
PCR_Read_In pcrRead;

TPM2_SetupPCRSel(&pcrRead.pcrSelectionIn, TPM_ALG_SHA256, pcrIndex);
```

TPM2_GetRCString

```
const WOLFTPM_API char * TPM2_GetRCString(
    int rc
)
```

TPM 2.0 リターン コードの人間が判読できる文字列を取得します。

パラメータ:

- **rc** TPM リターンコード

戻り値:

文字列定数へのポインター

使用例

```
int rc;

rc = wolfTPM2_Init(&dev, TPM2_IoCb, userCtx);
if (rc != TPM_RC_SUCCESS) {
    printf("wolfTPM2_Init failed 0x%x: %s\n", rc, TPM2_GetRCString(rc));
    return rc;
}
```

TPM2_GetAlgName

```
const WOLFTPM_API char * TPM2_GetAlgName(
    TPM_ALG_ID alg
)
```

TPM 2.0 アルゴリズムの人間が判読できる文字列を取得します。

パラメータ:

- **alg** TPM 2.0 アルゴリズムを表す TPM_ALG_ID 型の値

戻り値:

文字列定数へのポインター

使用例

```
int paramEncAlg = TPM_ALG_CFB;

printf("\tUse Parameter Encryption: %s\n", TPM2_GetAlgName(paramEncAlg));
```

TPM2_GetCurveSize

```
WOLFTPM_API int TPM2_GetCurveSize(
    TPM_ECC_CURVE curveID
)
```

TPM ECC 曲線のサイズ（バイト数）を返す。

パラメータ:

- **curveID** TPM ECC 曲線のタイプ

戻り値:

- 不正な曲線タイプの場合は 0 を返す。
- バイト数を示す整数値

使用例

```
int bytes;
TPM_ECC_CURVE curve = TPM_ECC_NIST_P256;

bytes = TPM2_GetCurveSize(curve);
if (bytes == 0) {
    //TPM2_GetCurveSize failed
}
```

TPM2_GetTpmCurve

```
WOLFTPM_API int TPM2_GetTpmCurve(
    int curveID
)
```

wolfcrypt 曲線タイプを対応する TPM 曲線タイプに変換します。

パラメータ:

- **curveID** BYTE バッファーへのポインター

参考: TPM2_GetWolfCurve

戻り値:

- wolfcrypt 曲線タイプを表す整数値
- 無効な曲線タイプの場合は ECC_CURVE_OID_E を返す

使用例

```
int tpmCurve;
int wc_curve = ECC_SECP256R1;

tpmCurve = TPM2_GetTpmCurve(curve);
\in this case tpmCurve will be TPM_ECC_NIST_P256
if (tpmCurve = ECC_CURVE_OID_E) {
    //TPM2_GetTpmCurve failed
}
```

TPM2_GetWolfCurve

```
WOLFTPM_API int TPM2_GetWolfCurve(
    int curve_id
)
```

TPM 曲線タイプを対応する wolfcrypt 曲線タイプに変換します。

パラメータ:

- **curve_id** BYTE バッファーへのポインター

参考: TPM2_GetTpmCurve

戻り値:

- TPM 曲線タイプを表す整数値
- 無効な曲線タイプの場合は -1 あるいは ECC_CURVE_OID_E を返す

使用例

```
int tpmCurve = TPM_ECC_NIST_P256;
int wc_curve;

wc_curve = TPM2_GetWolfCurve(tpmCurve);
\in this case tpmCurve will be ECC_SECP256R1
if (wc_curve = ECC_CURVE_OID_E || wc_curve == -1) {
    //TPM2_GetWolfCurve failed
}
```

TPM2_ParseAttest

```
WOLFTPM_API int TPM2_ParseAttest(
    const TPM2B_ATTEST * in,
    TPMS_ATTEST * out
)
```

TPM2B_ATTEST 構造体をパースする。

パラメータ:

- **in** TPM2B_ATTEST 型の構造体へのポインター
- **out** TPMS_ATTEST 型の構造体へのポインター

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

これは、ヘルパー関数 TPM2_Packet_ParseAttest のパブリック API です。

使用例

```
TPM2B_ATTEST in; //for example, as part of a TPM2_Quote
TPMS_ATTEST out

rc = TPM2_GetNonce(&in, &out);
if (rc != TPM_RC_SUCCESS) {
    //TPM2_ParseAttest failed
}
```

TPM2_HashNvPublic

```
WOLFTPM_API int TPM2_HashNvPublic(
    TPMS_NV_PUBLIC * nvPublic,
    byte * buffer,
    UINT16 * size
)
```

nvPublic 構造に基づいて新しい NV インデックス名を計算します。

パラメータ:

- **nvPublic**
- **buffer** TPMS_ATTEST 型の構造体へのポインター
- **size** nvIndex のサイズを格納する UINT16 型の変数へのポインタ

戻り値:

- TPM_RC_SUCCESS: 成功
- エラーの場合は負の値
- BAD_FUNC_ARG: 不正な引数
- NOT_COMPILED_IN: wolfcrypt が有効になっているか要確認

使用例

```
TPMS_NV_PUBLIC nvPublic;
BYTE buffer[TPM_MAX_DIGEST_SIZE];
UINT16 size;

rc = TPM2_HashNvPublic(&nvPublic, &buffer, &size);
if (rc != TPM_RC_SUCCESS) {
```

```
    //TPM2_HashNvPublic failed
}
```

TPM2_AppendPublic

```
WOLFTPM_API int TPM2_AppendPublic(
    byte * buf,
    word32 size,
    int * sizeUsed,
    TPM2B_PUBLIC * pub
)
```

ユーザー提供のバッファに基づいて TPM2B_PUBLIC 構造体を設定します。

パラメータ:

- **buf** ユーザーバッファへのポインター
- **size** ユーザーバッファのサイズを指定する word32 型の整数値
- **sizeUsed** pub->buffer の使用サイズを格納する整数変数へのポインター
- **pub** TPM2B_PUBLIC 型の空の構造体へのポインタ

参考: TPM2_ParsePublic

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 不十分なバッファ サイズ
- BAD_FUNC_ARG: 不正な引数

ノート:

ヘルパー関数 TPM2_Packet_AppendPublic の公開 API

使用例

```
TPM2B_PUBLIC pub; //empty
int sizeUsed, rc;
BYTE buffer[sizeof(TPM2B_PUBLIC)];
word32 size = sizeof(buffer);

rc = TPM2_AppendPublic(&buffer, size, &sizeUsed, &pub);
if (rc != TPM_RC_SUCCESS) {
    //TPM2_AppendPublic failed
}
```

TPM2_ParsePublic

```
WOLFTPM_API int TPM2_ParsePublic(
    TPM2B_PUBLIC * pub,
    byte * buf,
    word32 size,
    int * sizeUsed
)
```

TPM2B_PUBLIC 構造体を解析し、ユーザー指定のバッファーに格納します。

パラメータ:

- **pub** TPM2B_PUBLIC 型のデータが取り込まれた構造体へのポインター
- **buf** 空のユーザー バッファへのポインター
- **size** ユーザー バッファの使用可能なサイズを指定する word32 型の整数値
- **sizeUsed** ユーザー バッファーの使用サイズを格納する整数変数へのポインター

参考: TPM2_AppendPublic

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: バッファサイズが不十分
- BAD_FUNC_ARG: 不正な引数

ノート:

ヘルパー関数 TPM2_Packet_ParsePublic の公開 API

使用例

```
TPM2B_PUBLIC pub; //populated
int sizeUsed, rc;
BYTE buffer[sizeof(TPM2B_PUBLIC)];
word32 size = sizeof(buffer);

rc = TPM2_ParsePublic(&pub, buffer, size, &sizeUsed);
if (rc != TPM_RC_SUCCESS) {
    //TPM2_ParsePublic failed
}
```

TPM2_GetName

```
WOLFTPM_LOCAL int TPM2_GetName(
    TPM2_CTX * ctx,
    UINT32 handleValue,
    int handleCnt,
    int idx,
    TPM2B_NAME * name
)
```

TPM オブジェクトの名前を提供します。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター

- **handleValue** 有効な TPM ハンドルを指定する UINT32 型の値
- **handleCnt** 現在の TPM コマンド/セッションで使用されているハンドルの総数
- **idx** 有効な TPM 認証セッションを指定する 1 ～3 のインデックス値
- **name** TPM2B_NAME 型の空の構造体へのポインター

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

オブジェクトは、TPM ハンドルとセッション インデックスによって参照されます。

使用例

```
int rc;
UINT32 handleValue = TRANSIENT_FIRST;
handleCount = 1;
sessionIdx = 0;
TPM2B_NAME name;
```

```
rc = TPM2_GetName(ctx, handleValue, handleCount, sessionIdx, &name);
if (rc != TPM_RC_SUCCESS) {
    //TPM2_GetName failed
}
```

TPM2_GetVendorID

```
WOLFTPM_API UINT16 TPM2_GetVendorID(
    void
)
```

アクティブな TPM2 コンテキストの vendorID を提供します。

参考:

- TPM2_GetCapabilities
- TPM2_GetTpmDevId
- TPM2_Init

戻り値:

- ベンダー ID を指定する UINT16 型の整数値
- TPM2 コンテキストが無効または NULL の場合は 0

ノート:

TPM 初期化中に TPM デバイス情報が正しく読み取られるかどうかに依存します。

使用例

```
TPM2_CTX *tpm2Ctx;

tpm2Ctx = TPM2_GetActiveCtx();
```

TPM2_PrintBin

```
WOLFTPM_API void TPM2_PrintBin(
    const byte * buffer,
    word32 length
)
```

フォーマットされた方法でバイナリ バッファを出力するヘルパー関数。

パラメータ:

- **buffer** BYTE 型のバッファへのポインタ
- **length** バッファのサイズを含む word32 型の整数値

参考:

- TPM2_PrintAuth
- TPM2_PrintPublicArea

ノート:

DEBUG_WOLFTPM を定義する必要があります

使用例

```
BYTE buffer[] = {0x01,0x02,0x03,0x04};
length = sizeof(buffer);

TPM2_PrintBin(&buffer, length);
```

TPM2_PrintAuth

```
WOLFTPM_API void TPM2_PrintAuth(
    const TPMS_AUTH_COMMAND * authCmd
)
```

TPMS_AUTH_COMMAND 型の構造体を人間が読める形式で出力するヘルパー関数。

パラメータ:

- **authCmd** TPMS_AUTH_COMMAND 型のデータが取り込まれた構造体へのポインター

参考:

- TPM2_PrintBin
- TPM2_PrintPublicArea

ノート:

DEBUG_WOLFTPM を定義する必要があります

使用例

```
TPMS_AUTH_COMMAND authCmd; //for example, part of a TPM Authorization session

TPM2_PrintAuthCmd(&authCmd);
```

TPM2_PrintPublicArea

```
WOLFTPM_API void TPM2_PrintPublicArea(
    const TPM2B_PUBLIC * pub
)
```

TPM2B_PUBLIC 型の構造を人間が読める形式で出力するヘルパー関数。

パラメータ:

- **pub** TPM2B_PUBLIC 型のデータが取り込まれた構造体へのポインター

参考:

- TPM2_PrintBin
- TPM2_PrintAuth
- TPM2_Create
- TPM2_ReadPublic

ノート:

DEBUG_WOLFTPM を定義する必要があります

使用例

```
TPM2B_PUBLIC pub; //for example, part of the output of a successful TPM2_Create

TPM2_PrintPublicArea(&pub);
```

4.5 wolftpm/tpm2.h

4.5.1 クラス/構造体

	Name
struct	**TPMS_ALGORITHM_DESCRIPTION**
union	**TPMU_HA**
struct	**TPMT_HA**
struct	**TPM2B_DIGEST**
struct	**TPM2B_DATA**
struct	**TPM2B_EVENT**
struct	**TPM2B_MAX_BUFFER**
struct	**TPM2B_MAX_NV_BUFFER**
struct	**TPM2B_IV**
union	**TPMU_NAME**
struct	**TPM2B_NAME**
struct	**TPMS_PCR_SELECT**
struct	**TPMS_PCR_SELECTION**
struct	**TPMT_TK_CREATION**
struct	**TPMT_TK_VERIFIED**
struct	**TPMT_TK_AUTH**
struct	**TPMT_TK_HASHCHECK**
struct	**TPMS_ALG_PROPERTY**
struct	**TPMS_TAGGED_PROPERTY**
struct	**TPMS_TAGGED_PCR_SELECT**
struct	**TPMS_TAGGED_POLICY**
struct	**TPML_CC**
struct	**TPML_CCA**
struct	**TPML_ALG**
struct	**TPML_HANDLE**
struct	**TPML_DIGEST**
struct	**TPML_DIGEST_VALUES**
struct	**TPML_PCR_SELECTION**
struct	**TPML_ALG_PROPERTY**
struct	**TPML_TAGGED_TPM_PROPERTY**
struct	**TPML_TAGGED_PCR_PROPERTY**
struct	**TPML_ECC_CURVE**
struct	**TPML_TAGGED_POLICY**
union	**TPMU_CAPABILITIES**
struct	**TPMS_CAPABILITY_DATA**
struct	**TPMS_CLOCK_INFO**
struct	**TPMS_TIME_INFO**
struct	**TPMS_TIME_ATTEST_INFO**
struct	**TPMS_CERTIFY_INFO**
struct	**TPMS_QUOTE_INFO**
struct	**TPMS_COMMAND_AUDIT_INFO**
struct	**TPMS_SESSION_AUDIT_INFO**
struct	**TPMS_CREATION_INFO**
struct	**TPMS_NV_CERTIFY_INFO**
union	**TPMU_ATTEST**
struct	**TPMS_ATTEST**
struct	**TPM2B_ATTEST**
union	**TPMU_SYM_KEY_BITS**
union	**TPMU_SYM_MODE**
struct	**TPMT_SYM_DEF**
struct	**TPM2B_SYM_KEY**
struct	**TPMS_SYMCIPHER_PARMS**

	Name
struct	**TPM2B_LABEL**
struct	**TPMS_DERIVE**
struct	**TPM2B_DERIVE**
union	**TPMU_SENSITIVE_CREATE**
struct	**TPM2B_SENSITIVE_DATA**
struct	**TPMS_SENSITIVE_CREATE**
struct	**TPM2B_SENSITIVE_CREATE**
struct	**TPMS_SCHEME_HASH**
struct	**TPMS_SCHEME_ECDAA**
union	**TPMU_SCHEME_KEYEDHASH**
struct	**TPMT_KEYEDHASH_SCHEME**
union	**TPMU_SIG_SCHEME**
struct	**TPMT_SIG_SCHEME**
union	**TPMU_KDF_SCHEME**
struct	**TPMT_KDF_SCHEME**
union	**TPMU_ASYM_SCHEME**
struct	**TPMT_ASYM_SCHEME**
struct	**TPMT_RSA_SCHEME**
struct	**TPMT_RSA_DECRYPT**
struct	**TPM2B_PUBLIC_KEY_RSA**
struct	**TPM2B_PRIVATE_KEY_RSA**
struct	**TPM2B_ECC_PARAMETER**
struct	**TPMS_ECC_POINT**
struct	**TPM2B_ECC_POINT**
struct	**TPMS_ALGORITHM_DETAIL_ECC**
struct	**TPMS_SIGNATURE_RSA**
struct	**TPMS_SIGNATURE_ECC**
union	**TPMU_SIGNATURE**
struct	**TPMT_SIGNATURE**
union	**TPMU_ENCRYPTED_SECRET**
struct	**TPM2B_ENCRYPTED_SECRET**
union	**TPMU_PUBLIC_ID**
struct	**TPMS_KEYEDHASH_PARMS**
struct	**TPMS_ASYM_PARMS**
struct	**TPMS_RSA_PARMS**
struct	**TPMS_ECC_PARMS**
union	**TPMU_PUBLIC_PARMS**
struct	**TPMT_PUBLIC_PARMS**
struct	**TPMT_PUBLIC**
struct	**TPM2B_PUBLIC**
struct	**TPM2B_TEMPLATE**
struct	**TPM2B_PRIVATE_VENDOR_SPECIFIC**
union	**TPMU_SENSITIVE_COMPOSITE**
struct	**TPMT_SENSITIVE**
struct	**TPM2B_SENSITIVE**
struct	**TPMT_PRIVATE**
struct	**TPM2B_PRIVATE**
struct	**TPMS_ID_OBJECT**
struct	**TPM2B_ID_OBJECT**
struct	**TPMS_NV_PIN_COUNTER_PARAMETERS**
struct	**TPMS_NV_PUBLIC**
struct	**TPM2B_NV_PUBLIC**

	Name
struct	**TPM2B_CONTEXT_SENSITIVE**
struct	**TPMS_CONTEXT_DATA**
struct	**TPM2B_CONTEXT_DATA**
struct	**TPMS_CONTEXT**
struct	**TPMS_CREATION_DATA**
struct	**TPM2B_CREATION_DATA**
struct	**TPMS_AUTH_COMMAND**
struct	**TPMS_AUTH_RESPONSE**
struct	**TPM2_AUTH_SESSION**
struct	**wolfTPM_tcpContext**
struct	**wolfTPM_winContext**
struct	**TPM2_CTX**
struct	**Startup_In**
struct	**Shutdown_In**
struct	**GetCapability_In**
struct	**GetCapability_Out**
struct	**SelfTest_In**
struct	**IncrementalSelfTest_In**
struct	**IncrementalSelfTest_Out**
struct	**GetTestResult_Out**
struct	**GetRandom_In**
struct	**GetRandom_Out**
struct	**StirRandom_In**
struct	**PCR_Read_In**
struct	**PCR_Read_Out**
struct	**PCR_Extend_In**
struct	**Create_In**
struct	**Create_Out**
struct	**CreateLoaded_In**
struct	**CreateLoaded_Out**
struct	**CreatePrimary_In**
struct	**CreatePrimary_Out**
struct	**Load_In**
struct	**Load_Out**
struct	**FlushContext_In**
struct	**Unseal_In**
struct	**Unseal_Out**
struct	**StartAuthSession_In**
struct	**StartAuthSession_Out**
struct	**PolicyRestart_In**
struct	**LoadExternal_In**
struct	**LoadExternal_Out**
struct	**ReadPublic_In**
struct	**ReadPublic_Out**
struct	**ActivateCredential_In**
struct	**ActivateCredential_Out**
struct	**MakeCredential_In**
struct	**MakeCredential_Out**
struct	**ObjectChangeAuth_In**
struct	**ObjectChangeAuth_Out**
struct	**Duplicate_In**
struct	**Duplicate_Out**

	Name
struct	**Rewrap_In**
struct	**Rewrap_Out**
struct	**Import_In**
struct	**Import_Out**
struct	**RSA_Encrypt_In**
struct	**RSA_Encrypt_Out**
struct	**RSA_Decrypt_In**
struct	**RSA_Decrypt_Out**
struct	**ECDH_KeyGen_In**
struct	**ECDH_KeyGen_Out**
struct	**ECDH_ZGen_In**
struct	**ECDH_ZGen_Out**
struct	**ECC_Parameters_In**
struct	**ECC_Parameters_Out**
struct	**ZGen_2Phase_In**
struct	**ZGen_2Phase_Out**
struct	**EncryptDecrypt_In**
struct	**EncryptDecrypt_Out**
struct	**EncryptDecrypt2_In**
struct	**EncryptDecrypt2_Out**
struct	**Hash_In**
struct	**Hash_Out**
struct	**HMAC_In**
struct	**HMAC_Out**
struct	**HMAC_Start_In**
struct	**HMAC_Start_Out**
struct	**HashSequenceStart_In**
struct	**HashSequenceStart_Out**
struct	**SequenceUpdate_In**
struct	**SequenceComplete_In**
struct	**SequenceComplete_Out**
struct	**EventSequenceComplete_In**
struct	**EventSequenceComplete_Out**
struct	**Certify_In**
struct	**Certify_Out**
struct	**CertifyCreation_In**
struct	**CertifyCreation_Out**
struct	**Quote_In**
struct	**Quote_Out**
struct	**GetSessionAuditDigest_In**
struct	**GetSessionAuditDigest_Out**
struct	**GetCommandAuditDigest_In**
struct	**GetCommandAuditDigest_Out**
struct	**GetTime_In**
struct	**GetTime_Out**
struct	**Commit_In**
struct	**Commit_Out**
struct	**EC_Ephemeral_In**
struct	**EC_Ephemeral_Out**
struct	**VerifySignature_In**
struct	**VerifySignature_Out**
struct	**Sign_In**

	Name
struct	**Sign_Out**
struct	**SetCommandCodeAuditStatus_In**
struct	**PCR_Event_In**
struct	**PCR_Event_Out**
struct	**PCR_Allocate_In**
struct	**PCR_Allocate_Out**
struct	**PCR_SetAuthPolicy_In**
struct	**PCR_SetAuthValue_In**
struct	**PCR_Reset_In**
struct	**PolicySigned_In**
struct	**PolicySigned_Out**
struct	**PolicySecret_In**
struct	**PolicySecret_Out**
struct	**PolicyTicket_In**
struct	**PolicyOR_In**
struct	**PolicyPCR_In**
struct	**PolicyLocality_In**
struct	**PolicyNV_In**
struct	**PolicyCounterTimer_In**
struct	**PolicyCommandCode_In**
struct	**PolicyPhysicalPresence_In**
struct	**PolicyCpHash_In**
struct	**PolicyNameHash_In**
struct	**PolicyDuplicationSelect_In**
struct	**PolicyAuthorize_In**
struct	**PolicyAuthValue_In**
struct	**PolicyPassword_In**
struct	**PolicyGetDigest_In**
struct	**PolicyGetDigest_Out**
struct	**PolicyNvWritten_In**
struct	**PolicyTemplate_In**
struct	**PolicyAuthorizeNV_In**
struct	**HierarchyControl_In**
struct	**SetPrimaryPolicy_In**
struct	**ChangeSeed_In**
struct	**Clear_In**
struct	**ClearControl_In**
struct	**HierarchyChangeAuth_In**
struct	**DictionaryAttackLockReset_In**
struct	**DictionaryAttackParameters_In**
struct	**PP_Commands_In**
struct	**SetAlgorithmSet_In**
struct	**FieldUpgradeStart_In**
struct	**FieldUpgradeData_In**
struct	**FieldUpgradeData_Out**
struct	**FirmwareRead_In**
struct	**FirmwareRead_Out**
struct	**ContextSave_In**
struct	**ContextSave_Out**
struct	**ContextLoad_In**
struct	**ContextLoad_Out**
struct	**EvictControl_In**

	Name
struct	**ReadClock_Out**
struct	**ClockSet_In**
struct	**ClockRateAdjust_In**
struct	**TestParms_In**
struct	**NV_DefineSpace_In**
struct	**NV_UndefineSpace_In**
struct	**NV_UndefineSpaceSpecial_In**
struct	**NV_ReadPublic_In**
struct	**NV_ReadPublic_Out**
struct	**NV_Write_In**
struct	**NV_Increment_In**
struct	**NV_Extend_In**
struct	**NV_SetBits_In**
struct	**NV_WriteLock_In**
struct	**NV_GlobalWriteLock_In**
struct	**NV_Read_In**
struct	**NV_Read_Out**
struct	**NV_ReadLock_In**
struct	**NV_ChangeAuth_In**
struct	**NV_Certify_In**
struct	**NV_Certify_Out**
struct	**SetCommandSet_In**
struct	**TPM_MODE_SET**
struct	**SetMode_In**
struct	**GetRandom2_Out**
struct	**TPMS_GPIO_CONFIG**
struct	**TPML_GPIO_CONFIG**
struct	**GpioConfig_In**
struct	**CFG_STRUCT**
struct	**NTC2_PreConfig_In**
struct	**NTC2_GetConfig_Out**

4.5.2 型

	Name
enum	**[@0](#enum-@0)** { TPM_SPEC_FAMILY = 0x322E3000, TPM_SPEC_LEVEL = 0, TPM_SPEC_VERSION = 138, TPM_SPEC_YEAR = 2016, TPM_SPEC_DAY_OF_YEAR = 273, TPM_GENERATED_VALUE = 0xff544347}

	Name
enum	**TPM_ALG_ID_T** { TPM_ALG_ERROR = 0x0000, TPM_ALG_RSA = 0x0001, TPM_ALG_SHA = 0x0004, TPM_ALG_SHA1 = TPM_ALG_SHA, TPM_ALG_HMAC = 0x0005, TPM_ALG_AES = 0x0006, TPM_ALG_MGF1 = 0x0007, TPM_ALG_KEYEDHASH = 0x0008, TPM_ALG_XOR = 0x000A, TPM_ALG_SHA256 = 0x000B, TPM_ALG_SHA384 = 0x000C, TPM_ALG_SHA512 = 0x000D, TPM_ALG_NULL = 0x0010, TPM_ALG_SM3_256 = 0x0012, TPM_ALG_SM4 = 0x0013, TPM_ALG_RSASSA = 0x0014, TPM_ALG_RSAES = 0x0015, TPM_ALG_RSAPSS = 0x0016, TPM_ALG_OAEP = 0x0017, TPM_ALG_ECDSA = 0x0018, TPM_ALG_ECDH = 0x0019, TPM_ALG_ECDAA = 0x001A, TPM_ALG_SM2 = 0x001B, TPM_ALG_ECSCHNORR = 0x001C, TPM_ALG_ECMQV = 0x001D, TPM_ALG_KDF1_SP800_56A = 0x0020, TPM_ALG_KDF2 = 0x0021, TPM_ALG_KDF1_SP800_108 = 0x0022, TPM_ALG_ECC = 0x0023, TPM_ALG_SYMCIPHER = 0x0025, TPM_ALG_CAMELLIA = 0x0026, TPM_ALG_CTR = 0x0040, TPM_ALG_OFB = 0x0041, TPM_ALG_CBC = 0x0042, TPM_ALG_CFB = 0x0043, TPM_ALG_ECB = 0x0044}
enum	**TPM_ECC_CURVE_T** { TPM_ECC_NONE = 0x0000, TPM_ECC_NIST_P192 = 0x0001, TPM_ECC_NIST_P224 = 0x0002, TPM_ECC_NIST_P256 = 0x0003, TPM_ECC_NIST_P384 = 0x0004, TPM_ECC_NIST_P521 = 0x0005, TPM_ECC_BN_P256 = 0x0010, TPM_ECC_BN_P638 = 0x0011, TPM_ECC_SM2_P256 = 0x0020}

	Name
enum	**TPM_CC_T** { TPM_CC_FIRST = 0x0000011F, TPM_CC_NV_UndefineSpaceSpecial = TPM_CC_FIRST, TPM_CC_EvictControl = 0x00000120, TPM_CC_HierarchyControl = 0x00000121, TPM_CC_NV_UndefineSpace = 0x00000122, TPM_CC_ChangeEPS = 0x00000124, TPM_CC_ChangePPS = 0x00000125, TPM_CC_Clear = 0x00000126, TPM_CC_ClearControl = 0x00000127, TPM_CC_ClockSet = 0x00000128, TPM_CC_HierarchyChangeAuth = 0x00000129, TPM_CC_NV_DefineSpace = 0x0000012A, TPM_CC_PCR_Allocate = 0x0000012B, TPM_CC_PCR_SetAuthPolicy = 0x0000012C, TPM_CC_PP_Commands = 0x0000012D, TPM_CC_SetPrimaryPolicy = 0x0000012E, TPM_CC_FieldUpgradeStart = 0x0000012F, TPM_CC_ClockRateAdjust = 0x00000130, TPM_CC_CreatePrimary = 0x00000131, TPM_CC_NV_GlobalWriteLock = 0x00000132, TPM_CC_GetCommandAuditDigest = 0x00000133, TPM_CC_NV_Increment = 0x00000134, TPM_CC_NV_SetBits = 0x00000135, TPM_CC_NV_Extend = 0x00000136, TPM_CC_NV_Write = 0x00000137, TPM_CC_NV_WriteLock = 0x00000138, TPM_CC_DictionaryAttackLockReset = 0x00000139, TPM_CC_DictionaryAttackParameters = 0x0000013A, TPM_CC_NV_ChangeAuth = 0x0000013B, TPM_CC_PCR_Event = 0x0000013C, TPM_CC_PCR_Reset = 0x0000013D, TPM_CC_SequenceComplete = 0x0000013E, TPM_CC_SetAlgorithmSet = 0x0000013F, TPM_CC_SetCommandCodeAuditStatus = 0x00000140, TPM_CC_FieldUpgradeData = 0x00000141, TPM_CC_IncrementalSelfTest = 0x00000142, TPM_CC_SelfTest = 0x00000143, TPM_CC_Startup = 0x00000144, TPM_CC_Shutdown = 0x00000145, TPM_CC_StirRandom = 0x00000146, TPM_CC_ActivateCredential = 0x00000147, TPM_CC_Certify = 0x00000148, TPM_CC_PolicyNV = 0x00000149, TPM_CC_CertifyCreation = 0x0000014A, TPM_CC_Duplicate = 0x0000014B, TPM_CC_GetTime = 0x0000014C, TPM_CC_GetSessionAuditDigest = 0x0000014D, TPM_CC_NV_Read = 0x0000014E, TPM_CC_NV_ReadLock = 0x0000014F, TPM_CC_ObjectChangeAuth = 0x00000150, TPM_CC_PolicySecret = 0x00000151, TPM_CC_Rewrap = 0x00000152, TPM_CC_Create = 0x00000153, TPM_CC_ECDH_ZGen = 0x00000154, TPM_CC_HMAC = 0x00000155, TPM_CC_Import = 0x00000156, TPM_CC_Load = 0x00000157, TPM_CC_Quote = 0x00000158, TPM_CC_RSA_Decrypt = 0x00000159,

	Name
enum	**TPM_RC_T** { TPM_RC_SUCCESS = 0x000, TPM_RC_BAD_TAG = 0x01E, RC_VER1 = 0x100, TPM_RC_INITIALIZE = RC_VER1 + 0x000, TPM_RC_FAILURE = RC_VER1 + 0x001, TPM_RC_SEQUENCE = RC_VER1 + 0x003, TPM_RC_PRIVATE = RC_VER1 + 0x00B, TPM_RC_HMAC = RC_VER1 + 0x019, TPM_RC_DISABLED = RC_VER1 + 0x020, TPM_RC_EXCLUSIVE = RC_VER1 + 0x021, TPM_RC_AUTH_TYPE = RC_VER1 + 0x024, TPM_RC_AUTH_MISSING = RC_VER1 + 0x025, TPM_RC_POLICY = RC_VER1 + 0x026, TPM_RC_PCR = RC_VER1 + 0x027, TPM_RC_PCR_CHANGED = RC_VER1 + 0x028, TPM_RC_UPGRADE = RC_VER1 + 0x02D, TPM_RC_TOO_MANY_CONTEXTS = RC_VER1 + 0x02E, TPM_RC_AUTH_UNAVAILABLE = RC_VER1 + 0x02F, TPM_RC_REBOOT = RC_VER1 + 0x030, TPM_RC_UNBALANCED = RC_VER1 + 0x031, TPM_RC_COMMAND_SIZE = RC_VER1 + 0x042, TPM_RC_COMMAND_CODE = RC_VER1 + 0x043, TPM_RC_AUTHSIZE = RC_VER1 + 0x044, TPM_RC_AUTH_CONTEXT = RC_VER1 + 0x045, TPM_RC_NV_RANGE = RC_VER1 + 0x046, TPM_RC_NV_SIZE = RC_VER1 + 0x047, TPM_RC_NV_LOCKED = RC_VER1 + 0x048, TPM_RC_NV_AUTHORIZATION = RC_VER1 + 0x049, TPM_RC_NV_UNINITIALIZED = RC_VER1 + 0x04A, TPM_RC_NV_SPACE = RC_VER1 + 0x04B, TPM_RC_NV_DEFINED = RC_VER1 + 0x04C, TPM_RC_BAD_CONTEXT = RC_VER1 + 0x050, TPM_RC_CPHASH = RC_VER1 + 0x051, TPM_RC_PARENT = RC_VER1 + 0x052, TPM_RC_NEEDS_TEST = RC_VER1 + 0x053, TPM_RC_NO_RESULT = RC_VER1 + 0x054, TPM_RC_SENSITIVE = RC_VER1 + 0x055, RC_MAX_FM0 = RC_VER1 + 0x07F, RC_FMT1 = 0x080, TPM_RC_ASYMMETRIC = RC_FMT1 + 0x001, TPM_RC_ATTRIBUTES = RC_FMT1 + 0x002, TPM_RC_HASH = RC_FMT1 + 0x003, TPM_RC_VALUE = RC_FMT1 + 0x004, TPM_RC_HIERARCHY = RC_FMT1 + 0x005, TPM_RC_KEY_SIZE = RC_FMT1 + 0x007, TPM_RC_MGF = RC_FMT1 + 0x008, TPM_RC_MODE = RC_FMT1 + 0x009, TPM_RC_TYPE = RC_FMT1 + 0x00A, TPM_RC_HANDLE = RC_FMT1 + 0x00B, TPM_RC_KDF = RC_FMT1 + 0x00C, TPM_RC_RANGE = RC_FMT1 + 0x00D, TPM_RC_AUTH_FAIL = RC_FMT1 + 0x00E, TPM_RC_NONCE = RC_FMT1 + 0x00F, TPM_RC_PP = RC_FMT1 + 0x010, TPM_RC_SCHEME = RC_FMT1 + 0x012, TPM_RC_SIZE = RC_FMT1 + 0x015, TPM_RC_SYMMETRIC = RC_FMT1 + 0x016, TPM_RC_TAG = RC_FMT1 + 0x017, TPM_RC_SELECTOR = RC_FMT1 + 0x018, TPM_RC_INSUFFICIENT = RC_FMT1 + 0x01A, TPM_RC_SIGNATURE = RC_FMT1 + 0x01B

	Name
enum	**TPM_CLOCK_ADJUST_T** { TPM_CLOCK_COARSE_SLOWER = -3, TPM_CLOCK_MEDIUM_SLOWER = -2, TPM_CLOCK_FINE_SLOWER = -1, TPM_CLOCK_NO_CHANGE = 0, TPM_CLOCK_FINE_FASTER = 1, TPM_CLOCK_MEDIUM_FASTER = 2, TPM_CLOCK_COARSE_FASTER = 3}
enum	**TPM_EO_T** { TPM_EO_EQ = 0x0000, TPM_EO_NEQ = 0x0001, TPM_EO_SIGNED_GT = 0x0002, TPM_EO_UNSIGNED_GT = 0x0003, TPM_EO_SIGNED_LT = 0x0004, TPM_EO_UNSIGNED_LT = 0x0005, TPM_EO_SIGNED_GE = 0x0006, TPM_EO_UNSIGNED_GE = 0x0007, TPM_EO_SIGNED_LE = 0x0008, TPM_EO_UNSIGNED_LE = 0x0009, TPM_EO_BITSET = 0x000A, TPM_EO_BITCLEAR = 0x000B}
enum	**TPM_ST_T** { TPM_ST_RSP_COMMAND = 0x00C4, TPM_ST_NULL = 0X8000, TPM_ST_NO_SESSIONS = 0x8001, TPM_ST_SESSIONS = 0x8002, TPM_ST_ATTEST_NV = 0x8014, TPM_ST_ATTEST_COMMAND_AUDIT = 0x8015, TPM_ST_ATTEST_SESSION_AUDIT = 0x8016, TPM_ST_ATTEST_CERTIFY = 0x8017, TPM_ST_ATTEST_QUOTE = 0x8018, TPM_ST_ATTEST_TIME = 0x8019, TPM_ST_ATTEST_CREATION = 0x801A, TPM_ST_CREATION = 0x8021, TPM_ST_VERIFIED = 0x8022, TPM_ST_AUTH_SECRET = 0x8023, TPM_ST_HASHCHECK = 0x8024, TPM_ST_AUTH_SIGNED = 0x8025, TPM_ST_FU_MANIFEST = 0x8029}
enum	**TPM_SE_T** { TPM_SE_HMAC = 0x00, TPM_SE_POLICY = 0x01, TPM_SE_TRIAL = 0x03}
enum	**TPM_SU_T** { TPM_SU_CLEAR = 0x0000, TPM_SU_STATE = 0x0001}
enum	**TPM_CAP_T** { TPM_CAP_FIRST = 0x00000000, TPM_CAP_ALGS = TPM_CAP_FIRST, TPM_CAP_HANDLES = 0x00000001, TPM_CAP_COMMANDS = 0x00000002, TPM_CAP_PP_COMMANDS = 0x00000003, TPM_CAP_AUDIT_COMMANDS = 0x00000004, TPM_CAP_PCRS = 0x00000005, TPM_CAP_TPM_PROPERTIES = 0x00000006, TPM_CAP_PCR_PROPERTIES = 0x00000007, TPM_CAP_ECC_CURVES = 0x00000008, TPM_CAP_LAST = TPM_CAP_ECC_CURVES, TPM_CAP_VENDOR_PROPERTY = 0x00000100}

	Name
enum	**TPM_PT_T** { TPM_PT_NONE = 0x00000000, PT_GROUP = 0x00000100, PT_FIXED = PT_GROUP * 1, TPM_PT_FAMILY_INDICATOR = PT_FIXED + 0, TPM_PT_LEVEL = PT_FIXED + 1, TPM_PT_REVISION = PT_FIXED + 2, TPM_PT_DAY_OF_YEAR = PT_FIXED + 3, TPM_PT_YEAR = PT_FIXED + 4, TPM_PT_MANUFACTURER = PT_FIXED + 5, TPM_PT_VENDOR_STRING_1 = PT_FIXED + 6, TPM_PT_VENDOR_STRING_2 = PT_FIXED + 7, TPM_PT_VENDOR_STRING_3 = PT_FIXED + 8, TPM_PT_VENDOR_STRING_4 = PT_FIXED + 9, TPM_PT_VENDOR_TPM_TYPE = PT_FIXED + 10, TPM_PT_FIRMWARE_VERSION_1 = PT_FIXED + 11, TPM_PT_FIRMWARE_VERSION_2 = PT_FIXED + 12, TPM_PT_INPUT_BUFFER = PT_FIXED + 13, TPM_PT_HR_TRANSIENT_MIN = PT_FIXED + 14, TPM_PT_HR_PERSISTENT_MIN = PT_FIXED + 15, TPM_PT_HR_LOADED_MIN = PT_FIXED + 16, TPM_PT_ACTIVE_SESSIONS_MAX = PT_FIXED + 17, TPM_PT_PCR_COUNT = PT_FIXED + 18, TPM_PT_PCR_SELECT_MIN = PT_FIXED + 19, TPM_PT_CONTEXT_GAP_MAX = PT_FIXED + 20, TPM_PT_NV_COUNTERS_MAX = PT_FIXED + 22, TPM_PT_NV_INDEX_MAX = PT_FIXED + 23, TPM_PT_MEMORY = PT_FIXED + 24, TPM_PT_CLOCK_UPDATE = PT_FIXED + 25, TPM_PT_CONTEXT_HASH = PT_FIXED + 26, TPM_PT_CONTEXT_SYM = PT_FIXED + 27, TPM_PT_CONTEXT_SYM_SIZE = PT_FIXED + 28, TPM_PT_ORDERLY_COUNT = PT_FIXED + 29, TPM_PT_MAX_COMMAND_SIZE = PT_FIXED + 30, TPM_PT_MAX_RESPONSE_SIZE = PT_FIXED + 31, TPM_PT_MAX_DIGEST = PT_FIXED + 32, TPM_PT_MAX_OBJECT_CONTEXT = PT_FIXED + 33, TPM_PT_MAX_SESSION_CONTEXT = PT_FIXED + 34, TPM_PT_PS_FAMILY_INDICATOR = PT_FIXED + 35, TPM_PT_PS_LEVEL = PT_FIXED + 36, TPM_PT_PS_REVISION = PT_FIXED + 37, TPM_PT_PS_DAY_OF_YEAR = PT_FIXED + 38, TPM_PT_PS_YEAR = PT_FIXED + 39, TPM_PT_SPLIT_MAX = PT_FIXED + 40, TPM_PT_TOTAL_COMMANDS = PT_FIXED + 41, TPM_PT_LIBRARY_COMMANDS = PT_FIXED + 42, TPM_PT_VENDOR_COMMANDS = PT_FIXED + 43, TPM_PT_NV_BUFFER_MAX = PT_FIXED + 44, TPM_PT_MODES = PT_FIXED + 45, TPM_PT_MAX_CAP_BUFFER = PT_FIXED + 46, PT_VAR = PT_GROUP * 2, TPM_PT_PERMANENT = PT_VAR + 0, TPM_PT_STARTUP_CLEAR = PT_VAR + 1, TPM_PT_HR_NV_INDEX = PT_VAR + 2, TPM_PT_HR_LOADED = PT_VAR + 3, TPM_PT_HR_LOADED_AVAIL = PT_VAR + 4, TPM_PT_HR_ACTIVE = PT_VAR + 5, TPM_PT_HR_ACTIVE_AVAIL = PT_VAR + 6, TPM_PT_HR_TRANSIENT_AVAIL = PT_VAR + 7, TPM_PT_HR_PERSISTENT = PT_VAR + 8, TPM_PT_HR_PERSISTENT_AVAIL = PT_VAR + 9, TPM_PT_NV_COUNTERS = PT_VAR + 10

	Name
enum	**TPM_PT_PCR_T** { TPM_PT_PCR_FIRST = 0x00000000, TPM_PT_PCR_SAVE = TPM_PT_PCR_FIRST, TPM_PT_PCR_EXTEND_L0 = 0x00000001, TPM_PT_PCR_RESET_L0 = 0x00000002, TPM_PT_PCR_EXTEND_L1 = 0x00000003, TPM_PT_PCR_RESET_L1 = 0x00000004, TPM_PT_PCR_EXTEND_L2 = 0x00000005, TPM_PT_PCR_RESET_L2 = 0x00000006, TPM_PT_PCR_EXTEND_L3 = 0x00000007, TPM_PT_PCR_RESET_L3 = 0x00000008, TPM_PT_PCR_EXTEND_L4 = 0x00000009, TPM_PT_PCR_RESET_L4 = 0x0000000A, TPM_PT_PCR_NO_INCREMENT = 0x00000011, TPM_PT_PCR_DRTM_RESET = 0x00000012, TPM_PT_PCR_POLICY = 0x00000013, TPM_PT_PCR_AUTH = 0x00000014, TPM_PT_PCR_LAST = TPM_PT_PCR_AUTH}
enum	**TPM_PS_T** { TPM_PS_MAIN = 0x00000000, TPM_PS_PC = 0x00000001, TPM_PS_PDA = 0x00000002, TPM_PS_CELL_PHONE = 0x00000003, TPM_PS_SERVER = 0x00000004, TPM_PS_PERIPHERAL = 0x00000005, TPM_PS_TSS = 0x00000006, TPM_PS_STORAGE = 0x00000007, TPM_PS_AUTHENTICATION = 0x00000008, TPM_PS_EMBEDDED = 0x00000009, TPM_PS_HARDCOPY = 0x0000000A, TPM_PS_INFRASTRUCTURE = 0x0000000B, TPM_PS_VIRTUALIZATION = 0x0000000C, TPM_PS_TNC = 0x0000000D, TPM_PS_MULTI_TENANT = 0x0000000E, TPM_PS_TC = 0x0000000F}
enum	**TPM_HT_T** { TPM_HT_PCR = 0x00, TPM_HT_NV_INDEX = 0x01, TPM_HT_HMAC_SESSION = 0x02, TPM_HT_LOADED_SESSION = 0x02, TPM_HT_POLICY_SESSION = 0x03, TPM_HT_ACTIVE_SESSION = 0x03, TPM_HT_PERMANENT = 0x40, TPM_HT_TRANSIENT = 0x80, TPM_HT_PERSISTENT = 0x81}

	Name
enum	**TPM_RH_T** { TPM_RH_FIRST = 0x40000000, TPM_RH_SRK = TPM_RH_FIRST, TPM_RH_OWNER = 0x40000001, TPM_RH_REVOKE = 0x40000002, TPM_RH_TRANSPORT = 0x40000003, TPM_RH_OPERATOR = 0x40000004, TPM_RH_ADMIN = 0x40000005, TPM_RH_EK = 0x40000006, TPM_RH_NULL = 0x40000007, TPM_RH_UNASSIGNED = 0x40000008, TPM_RS_PW = 0x40000009, TPM_RH_LOCKOUT = 0x4000000A, TPM_RH_ENDORSEMENT = 0x4000000B, TPM_RH_PLATFORM = 0x4000000C, TPM_RH_PLATFORM_NV = 0x4000000D, TPM_RH_AUTH_00 = 0x40000010, TPM_RH_AUTH_FF = 0x4000010F, TPM_RH_LAST = TPM_RH_AUTH_FF}

	Name
enum	**TPM_HC_T** { HR_HANDLE_MASK = 0x00FFFFFF, HR_RANGE_MASK = 0xFF000000, HR_SHIFT = 24, HR_PCR = ((UINT32)TPM_HT_PCR « HR_SHIFT), HR_HMAC_SESSION = ((UINT32)TPM_HT_HMAC_SESSION « HR_SHIFT), HR_POLICY_SESSION = ((UINT32)TPM_HT_POLICY_SESSION « HR_SHIFT), HR_TRANSIENT = ((UINT32)TPM_HT_TRANSIENT « HR_SHIFT), HR_PERSISTENT = ((UINT32)TPM_HT_PERSISTENT « HR_SHIFT), HR_NV_INDEX = ((UINT32)TPM_HT_NV_INDEX « HR_SHIFT), HR_PERMANENT = ((UINT32)TPM_HT_PERMANENT « HR_SHIFT), PCR_FIRST = (HR_PCR + 0), PCR_LAST = (PCR_FIRST + IMPLEMENTATION_PCR-1), HMAC_SESSION_FIRST = (HR_HMAC_SESSION + 0), HMAC_SESSION_LAST = (HMAC_SESSION_FIRST+MAX_ACTIVE_SESSIONS-1), LOADED_SESSION_FIRST = HMAC_SESSION_FIRST, LOADED_SESSION_LAST = HMAC_SESSION_LAST, POLICY_SESSION_FIRST = (HR_POLICY_SESSION + 0), POLICY_SESSION_LAST = (POLICY_SESSION_FIRST+MAX_ACTIVE_SESSIONS-1), TRANSIENT_FIRST = (HR_TRANSIENT + 0), ACTIVE_SESSION_FIRST = POLICY_SESSION_FIRST, ACTIVE_SESSION_LAST = POLICY_SESSION_LAST, TRANSIENT_LAST = (TRANSIENT_FIRST+MAX_LOADED_OBJECTS-1), PERSISTENT_FIRST = (HR_PERSISTENT + 0), PERSISTENT_LAST = (PERSISTENT_FIRST + 0x00FFFFFF), PLATFORM_PERSISTENT = (PERSISTENT_FIRST + 0x00800000), NV_INDEX_FIRST = (HR_NV_INDEX + 0), NV_INDEX_LAST = (NV_INDEX_FIRST + 0x00FFFFFF), PERMANENT_FIRST = TPM_RH_FIRST, PERMANENT_LAST = TPM_RH_LAST}
enum	**TPMA_ALGORITHM_mask** { TPMA_ALGORITHM_asymmetric = 0x00000001, TPMA_ALGORITHM_symmetric = 0x00000002, TPMA_ALGORITHM_hash = 0x00000004, TPMA_ALGORITHM_object = 0x00000008, TPMA_ALGORITHM_signing = 0x00000010, TPMA_ALGORITHM_encrypting = 0x00000020, TPMA_ALGORITHM_method = 0x00000040}

	Name
enum	**TPMA_OBJECT_mask** { TPMA_OBJECT_fixedTPM = 0x00000002, TPMA_OBJECT_stClear = 0x00000004, TPMA_OBJECT_fixedParent = 0x00000010, TPMA_OBJECT_sensitiveDataOrigin = 0x00000020, TPMA_OBJECT_userWithAuth = 0x00000040, TPMA_OBJECT_adminWithPolicy = 0x00000080, TPMA_OBJECT_derivedDataOrigin = 0x00000200, TPMA_OBJECT_noDA = 0x00000400, TPMA_OBJECT_encryptedDuplication = 0x00000800, TPMA_OBJECT_restricted = 0x00010000, TPMA_OBJECT_decrypt = 0x00020000, TPMA_OBJECT_sign = 0x00040000}
enum	**TPMA_SESSION_mask** { TPMA_SESSION_continueSession = 0x01, TPMA_SESSION_auditExclusive = 0x02, TPMA_SESSION_auditReset = 0x04, TPMA_SESSION_decrypt = 0x20, TPMA_SESSION_encrypt = 0x40, TPMA_SESSION_audit = 0x80}
enum	**TPMA_LOCALITY_mask** { TPM_LOC_ZERO = 0x01, TPM_LOC_ONE = 0x02, TPM_LOC_TWO = 0x04, TPM_LOC_THREE = 0x08, TPM_LOC_FOUR = 0x10}
enum	**TPMA_PERMANENT_mask** { TPMA_PERMANENT_ownerAuthSet = 0x00000001, TPMA_PERMANENT_endorsementAuthSet = 0x00000002, TPMA_PERMANENT_lockoutAuthSet = 0x00000004, TPMA_PERMANENT_disableClear = 0x00000100, TPMA_PERMANENT_inLockout = 0x00000200, TPMA_PERMANENT_tpmGeneratedEPS = 0x00000400}
enum	**TPMA_STARTUP_CLEAR_mask** { TPMA_STARTUP_CLEAR_phEnable = 0x00000001, TPMA_STARTUP_CLEAR_shEnable = 0x00000002, TPMA_STARTUP_CLEAR_ehEnable = 0x00000004, TPMA_STARTUP_CLEAR_phEnableNV = 0x00000008, TPMA_STARTUP_CLEAR_orderly = 0x80000000}
enum	**TPMA_MEMORY_mask** { TPMA_MEMORY_sharedRAM = 0x00000001, TPMA_MEMORY_sharedNV = 0x00000002, TPMA_MEMORY_objectCopiedToRam = 0x00000004}

	Name
enum	**TPMA_CC_mask** { TPMA_CC_commandIndex = 0x0000FFFF, TPMA_CC_nv = 0x00400000, TPMA_CC_extensive = 0x00800000, TPMA_CC_flushed = 0x01000000, TPMA_CC_cHandles = 0x0E000000, TPMA_CC_rHandle = 0x10000000, TPMA_CC_V = 0x20000000}
enum	**TPM_NV_INDEX_mask** { TPM_NV_INDEX_index = 0x00FFFFFF, TPM_NV_INDEX_RH_NV = 0xFF000000}
enum	**TPM_NT** { TPM_NT_ORDINARY = 0x0, TPM_NT_COUNTER = 0x1, TPM_NT_BITS = 0x2, TPM_NT_EXTEND = 0x4, TPM_NT_PIN_FAIL = 0x8, TPM_NT_PIN_PASS = 0x9}
enum	**TPMA_NV_mask** { TPMA_NV_PPWRITE = 0x00000001, TPMA_NV_OWNERWRITE = 0x00000002, TPMA_NV_AUTHWRITE = 0x00000004, TPMA_NV_POLICYWRITE = 0x00000008, TPMA_NV_TPM_NT = 0x000000F0, TPMA_NV_POLICY_DELETE = 0x00000400, TPMA_NV_WRITELOCKED = 0x00000800, TPMA_NV_WRITEALL = 0x00001000, TPMA_NV_WRITEDEFINE = 0x00002000, TPMA_NV_WRITE_STCLEAR = 0x00004000, TPMA_NV_GLOBALLOCK = 0x00008000, TPMA_NV_PPREAD = 0x00010000, TPMA_NV_OWNERREAD = 0x00020000, TPMA_NV_AUTHREAD = 0x00040000, TPMA_NV_POLICYREAD = 0x00080000, TPMA_NV_NO_DA = 0x02000000, TPMA_NV_ORDERLY = 0x04000000, TPMA_NV_CLEAR_STCLEAR = 0x08000000, TPMA_NV_READLOCKED = 0x10000000, TPMA_NV_WRITTEN = 0x20000000, TPMA_NV_PLATFORMCREATE = 0x40000000, TPMA_NV_READ_STCLEAR = 0x80000000}
enum	**[@1](#enum-@1)** { TPMLib_2 = 0x01, TPMFips = 0x02, TPMLowPowerOff = 0x00, TPMLowPowerByRegister = 0x04, TPMLowPowerByGpio = 0x08, TPMLowPowerAuto = 0x0C}
enum	**TPMI_GPIO_NAME_T** { TPM_GPIO_PP = 0x00000000, TPM_GPIO_LP = 0x00000001, TPM_GPIO_C = 0x00000002, TPM_GPIO_D = 0x00000003}

	Name
enum	**TPMI_GPIO_MODE_T** { TPM_GPIO_MODE_STANDARD = 0x00000000, TPM_GPIO_MODE_FLOATING = 0x00000001, TPM_GPIO_MODE_PULLUP = 0x00000002, TPM_GPIO_MODE_PULLDOWN = 0x00000003, TPM_GPIO_MODE_OPENDRAIN = 0x00000004, TPM_GPIO_MODE_PUSHPULL = 0x00000005, TPM_GPIO_MODE_UNCONFIG = 0x00000006, TPM_GPIO_MODE_DEFAULT = TPM_GPIO_MODE_PULLDOWN, TPM_GPIO_MODE_MAX = TPM_GPIO_MODE_UNCONFIG, TPM_GPIO_MODE_INPUT_MIN = TPM_GPIO_MODE_FLOATING, TPM_GPIO_MODE_INPUT_MAX = TPM_GPIO_MODE_PULLDOWN, TPM_GPIO_MODE_PUSHPULL = 0x00000005, TPM_GPIO_MODE_OPENDRAIN = 0x00000004, TPM_GPIO_MODE_PULLUP = 0x00000002, TPM_GPIO_MODE_UNCONFIG = 0x00000006, TPM_GPIO_MODE_DEFAULT = TPM_GPIO_MODE_PULLDOWN, TPM_GPIO_MODE_MAX = TPM_GPIO_MODE_UNCONFIG, TPM_GPIO_MODE_INPUT_MIN = TPM_GPIO_MODE_FLOATING, TPM_GPIO_MODE_INPUT_MAX = TPM_GPIO_MODE_PULLDOWN}

	Name
enum	**TPMI_GPIO_MODE_T** { TPM_GPIO_MODE_STANDARD = 0x00000000, TPM_GPIO_MODE_FLOATING = 0x00000001, TPM_GPIO_MODE_PULLUP = 0x00000002, TPM_GPIO_MODE_PULLDOWN = 0x00000003, TPM_GPIO_MODE_OPENDRAIN = 0x00000004, TPM_GPIO_MODE_PUSHPULL = 0x00000005, TPM_GPIO_MODE_UNCONFIG = 0x00000006, TPM_GPIO_MODE_DEFAULT = TPM_GPIO_MODE_PULLDOWN, TPM_GPIO_MODE_MAX = TPM_GPIO_MODE_UNCONFIG, TPM_GPIO_MODE_INPUT_MIN = TPM_GPIO_MODE_FLOATING, TPM_GPIO_MODE_INPUT_MAX = TPM_GPIO_MODE_PULLDOWN, TPM_GPIO_MODE_PUSHPULL = 0x00000005, TPM_GPIO_MODE_OPENDRAIN = 0x00000004, TPM_GPIO_MODE_PULLUP = 0x00000002, TPM_GPIO_MODE_UNCONFIG = 0x00000006, TPM_GPIO_MODE_DEFAULT = TPM_GPIO_MODE_PULLDOWN, TPM_GPIO_MODE_MAX = TPM_GPIO_MODE_UNCONFIG, TPM_GPIO_MODE_INPUT_MIN = TPM_GPIO_MODE_FLOATING, TPM_GPIO_MODE_INPUT_MAX = TPM_GPIO_MODE_PULLDOWN}
enum	**TPM_Vendor_t** { TPM_VENDOR_UNKNOWN = 0, TPM_VENDOR_INFINEON = 0x15d1, TPM_VENDOR_STM = 0x104a, TPM_VENDOR_MCHP = 0x1114, TPM_VENDOR_NUVOTON = 0x1050, TPM_VENDOR_NATIONTECH = 0x1B4E}
typedef UINT32	**TPM_MODIFIER_INDICATOR**
typedef UINT32	**TPM_AUTHORIZATION_SIZE**
typedef UINT32	**TPM_PARAMETER_SIZE**
typedef UINT16	**TPM_KEY_SIZE**
typedef UINT16	**TPM_KEY_BITS**
typedef UINT32	**TPM_GENERATED**
typedef UINT16	**TPM_ALG_ID**
typedef UINT16	**TPM_ECC_CURVE**
typedef UINT32	**TPM_CC**
typedef INT32	**TPM_RC**
typedef UINT8	**TPM_CLOCK_ADJUST**
typedef UINT16	**TPM_EO**
typedef UINT16	**TPM_ST**
typedef UINT8	**TPM_SE**
typedef UINT16	**TPM_SU**
typedef UINT32	**TPM_CAP**
typedef UINT32	**TPM_PT**
typedef UINT32	**TPM_PT_PCR**

	Name
typedef UINT32	**TPM_PS**
typedef UINT32	**TPM_HANDLE**
typedef UINT8	**TPM_HT**
typedef UINT32	**TPM_RH**
typedef UINT32	**TPM_HC**
typedef UINT32	**TPMA_ALGORITHM**
typedef UINT32	**TPMA_OBJECT**
typedef BYTE	**TPMA_SESSION**
typedef BYTE	**TPMA_LOCALITY**
typedef UINT32	**TPMA_PERMANENT**
typedef UINT32	**TPMA_STARTUP_CLEAR**
typedef UINT32	**TPMA_MEMORY**
typedef UINT32	**TPMA_CC**
typedef BYTE	**TPMI_YES_NO**
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_HANDLE**	
typedef TPM_ALG_ID**	
typedef TPM_ALG_ID**	
typedef TPM_ALG_ID**	
typedef TPM_ALG_ID**	
typedef TPM_ALG_ID**	
typedef TPM_ALG_ID**	
typedef TPM_ALG_ID**	
typedef TPM_ALG_ID**	
typedef TPM_ST**	
typedef struct TPMS_ALGORITHM_DESCRIPTION**	
typedef union TPMU_HA**	
typedef struct TPMT_HA**	
typedef struct TPM2B_DIGEST**	
typedef struct TPM2B_DATA**	
typedef TPM2B_DIGEST**	
typedef TPM2B_DIGEST**	
typedef TPM2B_DIGEST**	

Name
typedef struct TPM2B_EVENT**
typedef struct TPM2B_MAX_BUFFER**
typedef struct TPM2B_MAX_NV_BUFFER**
typedef TPM2B_DIGEST**
typedef struct TPM2B_IV**
typedef union TPMU_NAME**
typedef struct TPM2B_NAME**
typedef struct TPMS_PCR_SELECT**
typedef struct TPMS_PCR_SELECTION**
typedef struct TPMT_TK_CREATION**
typedef struct TPMT_TK_VERIFIED**
typedef struct TPMT_TK_AUTH**
typedef struct TPMT_TK_HASHCHECK**
typedef struct TPMS_ALG_PROPERTY**
typedef struct TPMS_TAGGED_PROPERTY**
typedef struct TPMS_TAGGED_PCR_SELECT**
typedef struct TPMS_TAGGED_POLICY**
typedef struct TPML_CC**
typedef struct TPML_CCA**
typedef struct TPML_ALG**
typedef struct TPML_HANDLE**
typedef struct TPML_DIGEST**
typedef struct TPML_DIGEST_VALUES**
typedef struct TPML_PCR_SELECTION**
typedef struct TPML_ALG_PROPERTY**
typedef struct TPML_TAGGED_TPM_PROPERTY**
typedef struct TPML_TAGGED_PCR_PROPERTY**
typedef struct TPML_ECC_CURVE**
typedef struct TPML_TAGGED_POLICY**
typedef union TPMU_CAPABILITIES**
typedef struct TPMS_CAPABILITY_DATA**
typedef struct TPMS_CLOCK_INFO**
typedef struct TPMS_TIME_INFO**
typedef struct TPMS_TIME_ATTEST_INFO**
typedef struct TPMS_CERTIFY_INFO**
typedef struct TPMS_QUOTE_INFO**
typedef struct TPMS_COMMAND_AUDIT_INFO**
typedef struct TPMS_SESSION_AUDIT_INFO**
typedef struct TPMS_CREATION_INFO**
typedef struct TPMS_NV_CERTIFY_INFO**
typedef TPM_ST**
typedef union TPMU_ATTEST**
typedef struct TPMS_ATTEST**
typedef struct TPM2B_ATTEST**
typedef TPM_KEY_BITS**
typedef union TPMU_SYM_KEY_BITS**
typedef union TPMU_SYM_MODE**
typedef struct TPMT_SYM_DEF**
typedef TPMT_SYM_DEF**

Name
typedef struct TPM2B_SYM_KEY**
typedef struct TPMS_SYMCIPHER_PARMS**
typedef struct TPM2B_LABEL**
typedef struct TPMS_DERIVE**
typedef struct TPM2B_DERIVE**
typedef union TPMU_SENSITIVE_CREATE**
typedef struct TPM2B_SENSITIVE_DATA**
typedef struct TPMS_SENSITIVE_CREATE**
typedef struct TPM2B_SENSITIVE_CREATE**
typedef struct TPMS_SCHEME_HASH**
typedef struct TPMS_SCHEME_ECDAA**
typedef TPM_ALG_ID**
typedef TPMS_SCHEME_HASH**
typedef union TPMU_SCHEME_KEYEDHASH**
typedef struct TPMT_KEYEDHASH_SCHEME**
typedef TPMS_SCHEME_HASH**
typedef TPMS_SCHEME_HASH**
typedef TPMS_SCHEME_HASH**
typedef TPMS_SCHEME_ECDAA**
typedef union TPMU_SIG_SCHEME**
typedef struct TPMT_SIG_SCHEME**
typedef TPMS_SCHEME_HASH**
typedef TPMS_SCHEME_HASH**
typedef TPMS_SCHEME_HASH**
typedef TPMS_SCHEME_HASH**
typedef TPMS_SCHEME_HASH**
typedef TPMS_SCHEME_HASH**
typedef TPMS_SCHEME_HASH**
typedef union TPMU_KDF_SCHEME**
typedef struct TPMT_KDF_SCHEME**
typedef TPM_ALG_ID**
typedef union TPMU_ASYM_SCHEME**
typedef struct TPMT_ASYM_SCHEME**
typedef TPM_ALG_ID**
typedef struct TPMT_RSA_SCHEME**
typedef TPM_ALG_ID**
typedef struct TPMT_RSA_DECRYPT**
typedef struct TPM2B_PUBLIC_KEY_RSA**
typedef TPM_KEY_BITS**
typedef struct TPM2B_PRIVATE_KEY_RSA**
typedef struct TPM2B_ECC_PARAMETER**
typedef struct TPMS_ECC_POINT**
typedef struct TPM2B_ECC_POINT**
typedef TPM_ALG_ID**
typedef TPM_ECC_CURVE**
typedef TPMT_SIG_SCHEME**
typedef struct TPMS_ALGORITHM_DETAIL_ECC**
typedef struct TPMS_SIGNATURE_RSA**
typedef TPMS_SIGNATURE_RSA**
typedef TPMS_SIGNATURE_RSA**
typedef struct TPMS_SIGNATURE_ECC**

Name
typedef TPMS_SIGNATURE_ECC**
typedef TPMS_SIGNATURE_ECC**
typedef union TPMU_SIGNATURE**
typedef struct TPMT_SIGNATURE**
typedef union TPMU_ENCRYPTED_SECRET**
typedef struct TPM2B_ENCRYPTED_SECRET**
typedef TPM_ALG_ID**
typedef union TPMU_PUBLIC_ID**
typedef struct TPMS_KEYEDHASH_PARMS**
typedef struct TPMS_ASYM_PARMS**
typedef struct TPMS_RSA_PARMS**
typedef struct TPMS_ECC_PARMS**
typedef union TPMU_PUBLIC_PARMS**
typedef struct TPMT_PUBLIC_PARMS**
typedef struct TPMT_PUBLIC**
typedef struct TPM2B_PUBLIC**
typedef struct TPM2B_TEMPLATE**
typedef struct TPM2B_PRIVATE_VENDOR_SPECIFIC**
typedef union TPMU_SENSITIVE_COMPOSITE**
typedef struct TPMT_SENSITIVE**
typedef struct TPM2B_SENSITIVE**
typedef struct TPMT_PRIVATE**
typedef struct TPM2B_PRIVATE**
typedef struct TPMS_ID_OBJECT**
typedef struct TPM2B_ID_OBJECT**
typedef UINT32 **TPM_NV_INDEX**
typedef enum TPM_NT**
typedef struct TPMS_NV_PIN_COUNTER_PARAMETERS**
typedef UINT32 **TPMA_NV**
typedef struct TPMS_NV_PUBLIC**
typedef struct TPM2B_NV_PUBLIC**
typedef struct TPM2B_CONTEXT_SENSITIVE**
typedef struct TPMS_CONTEXT_DATA**
typedef struct TPM2B_CONTEXT_DATA**
typedef struct TPMS_CONTEXT**
typedef struct TPMS_CREATION_DATA**
typedef struct TPM2B_CREATION_DATA**
typedef struct TPMS_AUTH_COMMAND**
typedef struct TPMS_AUTH_RESPONSE**
typedef struct TPM2_AUTH_SESSION**
typedef int()*(struct TPM2_CTX* , INT32 isRead, UINT32 addr, BYTE *xferBuf, UINT16 xferSz, void* userCtx) **TPM2HalIoCb**
typedef struct TPM2_CTX**
typedef ChangeSeed_In**
typedef ChangeSeed_In**
typedef struct TPM_MODE_SET**
typedef GetRandom_In**
typedef UINT32 **TPMI_GPIO_NAME**
typedef UINT32 **TPMI_GPIO_MODE**

Name
typedef struct TPMS_GPIO_CONFIG**
typedef struct TPML_GPIO_CONFIG**

4.5.3 関数

戻り値	関数名
WOLFTPM_API TPM_RC	**TPM2_Startup**
WOLFTPM_API TPM_RC	**TPM2_Shutdown**
WOLFTPM_API TPM_RC	**TPM2_GetCapability**
WOLFTPM_API TPM_RC	**TPM2_SelfTest**
WOLFTPM_API TPM_RC	**TPM2_IncrementalSelfTest**
WOLFTPM_API TPM_RC	**TPM2_GetTestResult**
WOLFTPM_API TPM_RC	**TPM2_GetRandom**
WOLFTPM_API TPM_RC	**TPM2_StirRandom**
WOLFTPM_API TPM_RC	**TPM2_PCR_Read**
WOLFTPM_API TPM_RC	**TPM2_PCR_Extend**
WOLFTPM_API TPM_RC	**TPM2_Create**
WOLFTPM_API TPM_RC	**TPM2_CreateLoaded**
WOLFTPM_API TPM_RC	**TPM2_CreatePrimary**
WOLFTPM_API TPM_RC	**TPM2_Load**
WOLFTPM_API TPM_RC	**TPM2_FlushContext**
WOLFTPM_API TPM_RC	**TPM2_Unseal**
WOLFTPM_API TPM_RC	**TPM2_StartAuthSession**
WOLFTPM_API TPM_RC	**TPM2_PolicyRestart**
WOLFTPM_API TPM_RC	**TPM2_LoadExternal**
WOLFTPM_API TPM_RC	**TPM2_ReadPublic**
WOLFTPM_API TPM_RC	**TPM2_ActivateCredential**
WOLFTPM_API TPM_RC	**TPM2_MakeCredential**
WOLFTPM_API TPM_RC	**TPM2_ObjectChangeAuth**
WOLFTPM_API TPM_RC	**TPM2_Duplicate**
WOLFTPM_API TPM_RC	**TPM2_Rewrap**
WOLFTPM_API TPM_RC	**TPM2_Import**
WOLFTPM_API TPM_RC	**TPM2_RSA_Encrypt**
WOLFTPM_API TPM_RC	**TPM2_RSA_Decrypt**
WOLFTPM_API TPM_RC	**TPM2_ECDH_KeyGen**
WOLFTPM_API TPM_RC	**TPM2_ECDH_ZGen**
WOLFTPM_API TPM_RC	**TPM2_ECC_Parameters**
WOLFTPM_API TPM_RC	**TPM2_ZGen_2Phase**
WOLFTPM_API TPM_RC	**TPM2_EncryptDecrypt**
WOLFTPM_API TPM_RC	**TPM2_EncryptDecrypt2**
WOLFTPM_API TPM_RC	**TPM2_Hash**
WOLFTPM_API TPM_RC	**TPM2_HMAC**
WOLFTPM_API TPM_RC	**TPM2_HMAC_Start**
WOLFTPM_API TPM_RC	**TPM2_HashSequenceStart**
WOLFTPM_API TPM_RC	**TPM2_SequenceUpdate**
WOLFTPM_API TPM_RC	**TPM2_SequenceComplete**
WOLFTPM_API TPM_RC	**TPM2_EventSequenceComplete**
WOLFTPM_API TPM_RC	**TPM2_Certify**
WOLFTPM_API TPM_RC	**TPM2_CertifyCreation**
WOLFTPM_API TPM_RC	**TPM2_Quote**

戻り値	関数名
WOLFTPM_API TPM_RC	**TPM2_GetSessionAuditDigest**
WOLFTPM_API TPM_RC	**TPM2_GetCommandAuditDigest**
WOLFTPM_API TPM_RC	**TPM2_GetTime**
WOLFTPM_API TPM_RC	**TPM2_Commit**
WOLFTPM_API TPM_RC	**TPM2_EC_Ephemeral**
WOLFTPM_API TPM_RC	**TPM2_VerifySignature**
WOLFTPM_API TPM_RC	**TPM2_Sign**
WOLFTPM_API TPM_RC	**TPM2_SetCommandCodeAuditStatus**
WOLFTPM_API TPM_RC	**TPM2_PCR_Event**
WOLFTPM_API TPM_RC	**TPM2_PCR_Allocate**
WOLFTPM_API TPM_RC	**TPM2_PCR_SetAuthPolicy**
WOLFTPM_API TPM_RC	**TPM2_PCR_SetAuthValue**
WOLFTPM_API TPM_RC	**TPM2_PCR_Reset**
WOLFTPM_API TPM_RC	**TPM2_PolicySigned**
WOLFTPM_API TPM_RC	**TPM2_PolicySecret**
WOLFTPM_API TPM_RC	**TPM2_PolicyTicket**
WOLFTPM_API TPM_RC	**TPM2_PolicyOR**
WOLFTPM_API TPM_RC	**TPM2_PolicyPCR**
WOLFTPM_API TPM_RC	**TPM2_PolicyLocality**
WOLFTPM_API TPM_RC	**TPM2_PolicyNV**
WOLFTPM_API TPM_RC	**TPM2_PolicyCounterTimer**
WOLFTPM_API TPM_RC	**TPM2_PolicyCommandCode**
WOLFTPM_API TPM_RC	**TPM2_PolicyPhysicalPresence**
WOLFTPM_API TPM_RC	**TPM2_PolicyCpHash**
WOLFTPM_API TPM_RC	**TPM2_PolicyNameHash**
WOLFTPM_API TPM_RC	**TPM2_PolicyDuplicationSelect**
WOLFTPM_API TPM_RC	**TPM2_PolicyAuthorize**
WOLFTPM_API TPM_RC	**TPM2_PolicyAuthValue**
WOLFTPM_API TPM_RC	**TPM2_PolicyPassword**
WOLFTPM_API TPM_RC	**TPM2_PolicyGetDigest**
WOLFTPM_API TPM_RC	**TPM2_PolicyNvWritten**
WOLFTPM_API TPM_RC	**TPM2_PolicyTemplate**
WOLFTPM_API TPM_RC	**TPM2_PolicyAuthorizeNV**
WOLFTPM_API void	**_TPM_Hash_Start**
WOLFTPM_API void	**_TPM_Hash_Data**
WOLFTPM_API void	**_TPM_Hash_End**
WOLFTPM_API TPM_RC	**TPM2_HierarchyControl**
WOLFTPM_API TPM_RC	**TPM2_SetPrimaryPolicy**
WOLFTPM_API TPM_RC	**TPM2_ChangePPS**
WOLFTPM_API TPM_RC	**TPM2_ChangeEPS**
WOLFTPM_API TPM_RC	**TPM2_Clear**
WOLFTPM_API TPM_RC	**TPM2_ClearControl**
WOLFTPM_API TPM_RC	**TPM2_HierarchyChangeAuth**
WOLFTPM_API TPM_RC	**TPM2_DictionaryAttackLockReset**
WOLFTPM_API TPM_RC	**TPM2_DictionaryAttackParameters**
WOLFTPM_API TPM_RC	**TPM2_PP_Commands**
WOLFTPM_API TPM_RC	**TPM2_SetAlgorithmSet**
WOLFTPM_API TPM_RC	**TPM2_FieldUpgradeStart**
WOLFTPM_API TPM_RC	**TPM2_FieldUpgradeData**
WOLFTPM_API TPM_RC	**TPM2_FirmwareRead**
WOLFTPM_API TPM_RC	**TPM2_ContextSave**
WOLFTPM_API TPM_RC	**TPM2_ContextLoad**

戻り値	関数名
WOLFTPM_API TPM_RC	**TPM2_EvictControl**
WOLFTPM_API TPM_RC	**TPM2_ReadClock**
WOLFTPM_API TPM_RC	**TPM2_ClockSet**
WOLFTPM_API TPM_RC	**TPM2_ClockRateAdjust**
WOLFTPM_API TPM_RC	**TPM2_TestParms**
WOLFTPM_API TPM_RC	**TPM2_NV_DefineSpace**
WOLFTPM_API TPM_RC	**TPM2_NV_UndefineSpace**
WOLFTPM_API TPM_RC	**TPM2_NV_UndefineSpaceSpecial**
WOLFTPM_API TPM_RC	**TPM2_NV_ReadPublic**
WOLFTPM_API TPM_RC	**TPM2_NV_Write**
WOLFTPM_API TPM_RC	**TPM2_NV_Increment**
WOLFTPM_API TPM_RC	**TPM2_NV_Extend**
WOLFTPM_API TPM_RC	**TPM2_NV_SetBits**
WOLFTPM_API TPM_RC	**TPM2_NV_WriteLock**
WOLFTPM_API TPM_RC	**TPM2_NV_GlobalWriteLock**
WOLFTPM_API TPM_RC	**TPM2_NV_Read**
WOLFTPM_API TPM_RC	**TPM2_NV_ReadLock**
WOLFTPM_API TPM_RC	**TPM2_NV_ChangeAuth**
WOLFTPM_API TPM_RC	**TPM2_NV_Certify**
WOLFTPM_API int	**TPM2_SetCommandSet**
WOLFTPM_API int	**TPM2_SetMode**
WOLFTPM_API TPM_RC	**TPM2_GetRandom2**
WOLFTPM_API int	**TPM2_GPIO_Config**
WOLFTPM_API int	**TPM2_NTC2_PreConfig**
WOLFTPM_API int	**TPM2_NTC2_GetConfig**
WOLFTPM_API TPM_RC	**TPM2_Init** HAL IO コールバックとユーザー指定のコンテキストで TPM を初期化します。–enable-devtpm または –enable-swtpm 構成で wolfTPM を使用する場合、ioCb と userCtx は使用されません。
WOLFTPM_API TPM_RC	**TPM2_Init_ex** timeoutTries、HAL IO コールバック、およびユーザー指定のコンテキストで TPM を初期化します。
WOLFTPM_API TPM_RC	**TPM2_Init_minimal** TPM を初期化し、使用される wolfTPM2 コンテキストを設定します。この関数は通常、Windows などのリッチ オペレーティング システムで使用されます。
WOLFTPM_API TPM_RC	**TPM2_Cleanup** TPM と wolfcrypt を初期化解除します (初期化されている場合)
WOLFTPM_API TPM_RC	**TPM2_ChipStartup** TPM2 の起動が完了していることを確認し、TPM デバイス情報を抽出します。
WOLFTPM_API TPM_RC	**TPM2_SetHalloCb** TPM 通信に必要なユーザーのコンテキストと IO コールバックを設定します。
WOLFTPM_API TPM_RC	**TPM2_SetSessionAuth** TPM 承認を保持する構造体を設定します。
WOLFTPM_API int	**TPM2_GetSessionAuthCount** 現在設定されている TPM 承認の数を確認します。
WOLFTPM_API void	**TPM2_SetActiveCtx** 使用する新しい TPM2 コンテキストを設定します。
WOLFTPM_API TPM2_CTX	**TPM2_GetActiveCtx** 使用中の TPM2 コンテキストへのポインターを提供します。

戻り値	関数名
WOLFTPM_API int	**TPM2_GetHashDigestSize** TPM 2.0 ハッシュ ダイジェストのサイズをバイト単位で決定します。
WOLFTPM_API int	**TPM2_GetHashType**TPM2 ハッシュ タイプを対応する wolfcrypt ハッシュ タイプに変換します。
WOLFTPM_API int	**TPM2_GetNonce** 乱数の新しいナンスを生成します。
WOLFTPM_API void	**TPM2_SetupPCRSel** 正しい PCR 選択を準備するためのヘルパー関数 たとえば、TPM2_Quote を作成する準備をする場合。
const WOLFTPM_API char *	**TPM2_GetRCString**TPM 2.0 リターン コードの人間が判読できる文字列を取得します。
const WOLFTPM_API char *	**TPM2_GetAlgName**任意の TPM 2.0 アルゴリズムについて、人間が判読できる文字列を取得します。
WOLFTPM_API int	**TPM2_GetCurveSize** TPM ECC 曲線のサイズをバイト単位で決定します。
WOLFTPM_API int	**TPM2_GetTpmCurve** wolfcrypt 曲線タイプを対応する TPM 曲線タイプに変換します。
WOLFTPM_API int	**TPM2_GetWolfCurve**TPM 曲線タイプを対応する wolfcrypt 曲線タイプに変換します。
WOLFTPM_API int	**TPM2_ParseAttest**
WOLFTPM_API int	**TPM2_HashNvPublic**nvPublic 構造に基づいて新しい NV インデックス名を計算します。
WOLFTPM_API int	**TPM2_AppendPublic** ユーザー提供のバッファーに基づいて TPM2B_PUBLIC 構造体を設定します。
WOLFTPM_API int	**TPM2_ParsePublic** TPM2B_PUBLIC 構造を解析し、ユーザー指定のバッファーに格納します。
WOLFTPM_LOCAL int	**TPM2_GetName**TPM オブジェクトの名前を提供します。
WOLFTPM_API int	**TPM2_GetWolfRng**
WOLFTPM_API UINT16	**TPM2_GetVendorID**アクティブな TPM2 コンテキストの vendorID を提供します。
WOLFTPM_API void	**TPM2_PrintBin**フォーマットされた方法でバイナリ バッファを出力するヘルパー関数。
WOLFTPM_API void	**TPM2_PrintAuth** 人間が読める方法で TPMS_AUTH_COMMAND 型の構造を出力するヘルパー関数。
WOLFTPM_API void	**TPM2_PrintPublicArea** 人間が判読できる方法で TPM2B_PUBLIC 型の構造を出力するヘルパー関数。

4.5.4 属性

	Name
	C
const BYTE	**TPM_20_EK_AUTH_POLICY**

4.5.5 型の詳細

4.5.5.1 enum @0

Enumerator	Value	Description
TPM_SPEC_FAMILY	0x322E3000	
TPM_SPEC_LEVEL	0	
TPM_SPEC_VERSION	138	
TPM_SPEC_YEAR	2016	
TPM_SPEC_DAY_OF_YEAR	273	
TPM_GENERATED_VALUE	0xff544347	

4.5.5.2 enum TPM_ALG_ID_T

Enumerator	Value	Description
TPM_ALG_ERROR	0x0000	
TPM_ALG_RSA	0x0001	
TPM_ALG_SHA	0x0004	
TPM_ALG_SHA1	TPM_ALG_SHA	
TPM_ALG_HMAC	0x0005	
TPM_ALG_AES	0x0006	
TPM_ALG_MGF1	0x0007	
TPM_ALG_KEYEDHASH	0x0008	
TPM_ALG_XOR	0x000A	
TPM_ALG_SHA256	0x000B	
TPM_ALG_SHA384	0x000C	
TPM_ALG_SHA512	0x000D	
TPM_ALG_NULL	0x0010	
TPM_ALG_SM3_256	0x0012	
TPM_ALG_SM4	0x0013	
TPM_ALG_RSASSA	0x0014	
TPM_ALG_RSAES	0x0015	
TPM_ALG_RSAPSS	0x0016	
TPM_ALG_OAEP	0x0017	
TPM_ALG_ECDSA	0x0018	
TPM_ALG_ECDH	0x0019	
TPM_ALG_ECDAA	0x001A	
TPM_ALG_SM2	0x001B	
TPM_ALG_ECSCHNORR	0x001C	
TPM_ALG_ECMQV	0x001D	
TPM_ALG_KDF1_SP800_56A	0x0020	
TPM_ALG_KDF2	0x0021	
TPM_ALG_KDF1_SP800_108	0x0022	
TPM_ALG_ECC	0x0023	
TPM_ALG_SYMCIPHER	0x0025	
TPM_ALG_CAMELLIA	0x0026	
TPM_ALG_CTR	0x0040	
TPM_ALG_OFB	0x0041	
TPM_ALG_CBC	0x0042	
TPM_ALG_CFB	0x0043	
TPM_ALG_ECB	0x0044	

4.5.5.3 enum TPM_ECC_CURVE_T

Enumerator	Value	Description
TPM_ECC_NONE	0x0000	
TPM_ECC_NIST_P192	0x0001	
TPM_ECC_NIST_P224	0x0002	
TPM_ECC_NIST_P256	0x0003	
TPM_ECC_NIST_P384	0x0004	
TPM_ECC_NIST_P521	0x0005	
TPM_ECC_BN_P256	0x0010	
TPM_ECC_BN_P638	0x0011	
TPM_ECC_SM2_P256	0x0020	

4.5.5.4 enum TPM_CC_T

Enumerator	Value	Description
TPM_CC_FIRST	0x0000011F	
TPM_CC_NV_UndefineSpaceSpecial	TPM_CC_FIRST	
TPM_CC_EvictControl	0x00000120	
TPM_CC_HierarchyControl	0x00000121	
TPM_CC_NV_UndefineSpace	0x00000122	
TPM_CC_ChangeEPS	0x00000124	
TPM_CC_ChangePPS	0x00000125	
TPM_CC_Clear	0x00000126	
TPM_CC_ClearControl	0x00000127	
TPM_CC_ClockSet	0x00000128	
TPM_CC_HierarchyChangeAuth	0x00000129	
TPM_CC_NV_DefineSpace	0x0000012A	
TPM_CC_PCR_Allocate	0x0000012B	
TPM_CC_PCR_SetAuthPolicy	0x0000012C	
TPM_CC_PP_Commands	0x0000012D	
TPM_CC_SetPrimaryPolicy	0x0000012E	
TPM_CC_FieldUpgradeStart	0x0000012F	
TPM_CC_ClockRateAdjust	0x00000130	
TPM_CC_CreatePrimary	0x00000131	
TPM_CC_NV_GlobalWriteLock	0x00000132	
TPM_CC_GetCommandAuditDigest	0x00000133	
TPM_CC_NV_Increment	0x00000134	
TPM_CC_NV_SetBits	0x00000135	
TPM_CC_NV_Extend	0x00000136	
TPM_CC_NV_Write	0x00000137	
TPM_CC_NV_WriteLock	0x00000138	
TPM_CC_DictionaryAttackLockReset	0x00000139	
TPM_CC_DictionaryAttackParameters	0x0000013A	
TPM_CC_NV_ChangeAuth	0x0000013B	
TPM_CC_PCR_Event	0x0000013C	
TPM_CC_PCR_Reset	0x0000013D	
TPM_CC_SequenceComplete	0x0000013E	
TPM_CC_SetAlgorithmSet	0x0000013F	
TPM_CC_SetCommandCodeAuditStatus	0x00000140	
TPM_CC_FieldUpgradeData	0x00000141	
TPM_CC_IncrementalSelfTest	0x00000142	
TPM_CC_SelfTest	0x00000143	

Enumerator	Value	Description
TPM_CC_Startup	0x00000144	
TPM_CC_Shutdown	0x00000145	
TPM_CC_StirRandom	0x00000146	
TPM_CC_ActivateCredential	0x00000147	
TPM_CC_Certify	0x00000148	
TPM_CC_PolicyNV	0x00000149	
TPM_CC_CertifyCreation	0x0000014A	
TPM_CC_Duplicate	0x0000014B	
TPM_CC_GetTime	0x0000014C	
TPM_CC_GetSessionAuditDigest	0x0000014D	
TPM_CC_NV_Read	0x0000014E	
TPM_CC_NV_ReadLock	0x0000014F	
TPM_CC_ObjectChangeAuth	0x00000150	
TPM_CC_PolicySecret	0x00000151	
TPM_CC_Rewrap	0x00000152	
TPM_CC_Create	0x00000153	
TPM_CC_ECDH_ZGen	0x00000154	
TPM_CC_HMAC	0x00000155	
TPM_CC_Import	0x00000156	
TPM_CC_Load	0x00000157	
TPM_CC_Quote	0x00000158	
TPM_CC_RSA_Decrypt	0x00000159	
TPM_CC_HMAC_Start	0x0000015B	
TPM_CC_SequenceUpdate	0x0000015C	
TPM_CC_Sign	0x0000015D	
TPM_CC_Unseal	0x0000015E	
TPM_CC_PolicySigned	0x00000160	
TPM_CC_ContextLoad	0x00000161	
TPM_CC_ContextSave	0x00000162	
TPM_CC_ECDH_KeyGen	0x00000163	
TPM_CC_EncryptDecrypt	0x00000164	
TPM_CC_FlushContext	0x00000165	
TPM_CC_LoadExternal	0x00000167	
TPM_CC_MakeCredential	0x00000168	
TPM_CC_NV_ReadPublic	0x00000169	
TPM_CC_PolicyAuthorize	0x0000016A	
TPM_CC_PolicyAuthValue	0x0000016B	
TPM_CC_PolicyCommandCode	0x0000016C	
TPM_CC_PolicyCounterTimer	0x0000016D	
TPM_CC_PolicyCpHash	0x0000016E	
TPM_CC_PolicyLocality	0x0000016F	
TPM_CC_PolicyNameHash	0x00000170	
TPM_CC_PolicyOR	0x00000171	
TPM_CC_PolicyTicket	0x00000172	
TPM_CC_ReadPublic	0x00000173	
TPM_CC_RSA_Encrypt	0x00000174	
TPM_CC_StartAuthSession	0x00000176	
TPM_CC_VerifySignature	0x00000177	
TPM_CC_ECC_Parameters	0x00000178	
TPM_CC_FirmwareRead	0x00000179	
TPM_CC_GetCapability	0x0000017A	
TPM_CC_GetRandom	0x0000017B	

Enumerator	Value	Description
TPM_CC_GetTestResult	0x0000017C	
TPM_CC_Hash	0x0000017D	
TPM_CC_PCR_Read	0x0000017E	
TPM_CC_PolicyPCR	0x0000017F	
TPM_CC_PolicyRestart	0x00000180	
TPM_CC_ReadClock	0x00000181	
TPM_CC_PCR_Extend	0x00000182	
TPM_CC_PCR_SetAuthValue	0x00000183	
TPM_CC_NV_Certify	0x00000184	
TPM_CC_EventSequenceComplete	0x00000185	
TPM_CC_HashSequenceStart	0x00000186	
TPM_CC_PolicyPhysicalPresence	0x00000187	
TPM_CC_PolicyDuplicationSelect	0x00000188	
TPM_CC_PolicyGetDigest	0x00000189	
TPM_CC_TestParms	0x0000018A	
TPM_CC_Commit	0x0000018B	
TPM_CC_PolicyPassword	0x0000018C	
TPM_CC_ZGen_2Phase	0x0000018D	
TPM_CC_EC_Ephemeral	0x0000018E	
TPM_CC_PolicyNvWritten	0x0000018F	
TPM_CC_PolicyTemplate	0x00000190	
TPM_CC_CreateLoaded	0x00000191	
TPM_CC_PolicyAuthorizeNV	0x00000192	
TPM_CC_EncryptDecrypt2	0x00000193	
TPM_CC_LAST	TPM_CC_EncryptDecrypt2	
CC_VEND	0x20000000	
TPM_CC_Vendor_TCG_Test	CC_VEND + 0x0000	
TPM_CC_SetMode	CC_VEND + 0x0307	
TPM_CC_SetCommandSet	CC_VEND + 0x0309	
TPM_CC_GetRandom2	CC_VEND + 0x030E	
TPM_CC_RestoreEK	CC_VEND + 0x030A	
TPM_CC_SetCommandSetLock	CC_VEND + 0x030B	
TPM_CC_GPIO_Config	CC_VEND + 0x030F	
TPM_CC_NTC2_PreConfig	CC_VEND + 0x0211	
TPM_CC_NTC2_GetConfig	CC_VEND + 0x0213	

4.5.5.5 enum TPM_RC_T

Enumerator	Value	Description
TPM_RC_SUCCESS	0x000	
TPM_RC_BAD_TAG	0x01E	
RC_VER1	0x100	
TPM_RC_INITIALIZE	RC_VER1 + 0x000	
TPM_RC_FAILURE	RC_VER1 + 0x001	
TPM_RC_SEQUENCE	RC_VER1 + 0x003	
TPM_RC_PRIVATE	RC_VER1 + 0x00B	
TPM_RC_HMAC	RC_VER1 + 0x019	
TPM_RC_DISABLED	RC_VER1 + 0x020	
TPM_RC_EXCLUSIVE	RC_VER1 + 0x021	
TPM_RC_AUTH_TYPE	RC_VER1 + 0x024	

Enumerator	Value	Description
TPM_RC_AUTH_MISSING	RC_VER1 + 0x025	
TPM_RC_POLICY	RC_VER1 + 0x026	
TPM_RC_PCR	RC_VER1 + 0x027	
TPM_RC_PCR_CHANGED	RC_VER1 + 0x028	
TPM_RC_UPGRADE	RC_VER1 + 0x02D	
TPM_RC_TOO_MANY_CONTEXTS	RC_VER1 + 0x02E	
TPM_RC_AUTH_UNAVAILABLE	RC_VER1 + 0x02F	
TPM_RC_REBOOT	RC_VER1 + 0x030	
TPM_RC_UNBALANCED	RC_VER1 + 0x031	
TPM_RC_COMMAND_SIZE	RC_VER1 + 0x042	
TPM_RC_COMMAND_CODE	RC_VER1 + 0x043	
TPM_RC_AUTHSIZE	RC_VER1 + 0x044	
TPM_RC_AUTH_CONTEXT	RC_VER1 + 0x045	
TPM_RC_NV_RANGE	RC_VER1 + 0x046	
TPM_RC_NV_SIZE	RC_VER1 + 0x047	
TPM_RC_NV_LOCKED	RC_VER1 + 0x048	
TPM_RC_NV_AUTHORIZATION	RC_VER1 + 0x049	
TPM_RC_NV_UNINITIALIZED	RC_VER1 + 0x04A	
TPM_RC_NV_SPACE	RC_VER1 + 0x04B	
TPM_RC_NV_DEFINED	RC_VER1 + 0x04C	
TPM_RC_BAD_CONTEXT	RC_VER1 + 0x050	
TPM_RC_CPHASH	RC_VER1 + 0x051	
TPM_RC_PARENT	RC_VER1 + 0x052	
TPM_RC_NEEDS_TEST	RC_VER1 + 0x053	
TPM_RC_NO_RESULT	RC_VER1 + 0x054	
TPM_RC_SENSITIVE	RC_VER1 + 0x055	
RC_MAX_FM0	RC_VER1 + 0x07F	
RC_FMT1	0x080	
TPM_RC_ASYMMETRIC	RC_FMT1 + 0x001	
TPM_RC_ATTRIBUTES	RC_FMT1 + 0x002	
TPM_RC_HASH	RC_FMT1 + 0x003	
TPM_RC_VALUE	RC_FMT1 + 0x004	
TPM_RC_HIERARCHY	RC_FMT1 + 0x005	
TPM_RC_KEY_SIZE	RC_FMT1 + 0x007	
TPM_RC_MGF	RC_FMT1 + 0x008	
TPM_RC_MODE	RC_FMT1 + 0x009	
TPM_RC_TYPE	RC_FMT1 + 0x00A	
TPM_RC_HANDLE	RC_FMT1 + 0x00B	
TPM_RC_KDF	RC_FMT1 + 0x00C	
TPM_RC_RANGE	RC_FMT1 + 0x00D	
TPM_RC_AUTH_FAIL	RC_FMT1 + 0x00E	
TPM_RC_NONCE	RC_FMT1 + 0x00F	
TPM_RC_PP	RC_FMT1 + 0x010	
TPM_RC_SCHEME	RC_FMT1 + 0x012	
TPM_RC_SIZE	RC_FMT1 + 0x015	
TPM_RC_SYMMETRIC	RC_FMT1 + 0x016	
TPM_RC_TAG	RC_FMT1 + 0x017	
TPM_RC_SELECTOR	RC_FMT1 + 0x018	
TPM_RC_INSUFFICIENT	RC_FMT1 + 0x01A	
TPM_RC_SIGNATURE	RC_FMT1 + 0x01B	
TPM_RC_KEY	RC_FMT1 + 0x01C	
TPM_RC_POLICY_FAIL	RC_FMT1 + 0x01D	

Enumerator	Value	Description
TPM_RC_INTEGRITY	RC_FMT1 + 0x01F	
TPM_RC_TICKET	RC_FMT1 + 0x020	
TPM_RC_RESERVED_BITS	RC_FMT1 + 0x021	
TPM_RC_BAD_AUTH	RC_FMT1 + 0x022	
TPM_RC_EXPIRED	RC_FMT1 + 0x023	
TPM_RC_POLICY_CC	RC_FMT1 + 0x024	
TPM_RC_BINDING	RC_FMT1 + 0x025	
TPM_RC_CURVE	RC_FMT1 + 0x026	
TPM_RC_ECC_POINT	RC_FMT1 + 0x027	
RC_MAX_FMT1	RC_FMT1 + 0x03F	
RC_WARN	0x900	
TPM_RC_CONTEXT_GAP	RC_WARN + 0x001	
TPM_RC_OBJECT_MEMORY	RC_WARN + 0x002	
TPM_RC_SESSION_MEMORY	RC_WARN + 0x003	
TPM_RC_MEMORY	RC_WARN + 0x004	
TPM_RC_SESSION_HANDLES	RC_WARN + 0x005	
TPM_RC_OBJECT_HANDLES	RC_WARN + 0x006	
TPM_RC_LOCALITY	RC_WARN + 0x007	
TPM_RC_YIELDED	RC_WARN + 0x008	
TPM_RC_CANCELED	RC_WARN + 0x009	
TPM_RC_TESTING	RC_WARN + 0x00A	
TPM_RC_REFERENCE_H0	RC_WARN + 0x010	
TPM_RC_REFERENCE_H1	RC_WARN + 0x011	
TPM_RC_REFERENCE_H2	RC_WARN + 0x012	
TPM_RC_REFERENCE_H3	RC_WARN + 0x013	
TPM_RC_REFERENCE_H4	RC_WARN + 0x014	
TPM_RC_REFERENCE_H5	RC_WARN + 0x015	
TPM_RC_REFERENCE_H6	RC_WARN + 0x016	
TPM_RC_REFERENCE_S0	RC_WARN + 0x018	
TPM_RC_REFERENCE_S1	RC_WARN + 0x019	
TPM_RC_REFERENCE_S2	RC_WARN + 0x01A	
TPM_RC_REFERENCE_S3	RC_WARN + 0x01B	
TPM_RC_REFERENCE_S4	RC_WARN + 0x01C	
TPM_RC_REFERENCE_S5	RC_WARN + 0x01D	
TPM_RC_REFERENCE_S6	RC_WARN + 0x01E	
TPM_RC_NV_RATE	RC_WARN + 0x020	
TPM_RC_LOCKOUT	RC_WARN + 0x021	
TPM_RC_RETRY	RC_WARN + 0x022	
TPM_RC_NV_UNAVAILABLE	RC_WARN + 0x023	
RC_MAX_WARN	RC_WARN + 0x03F	
TPM_RC_NOT_USED	RC_WARN + 0x07F	
TPM_RC_H	0x000	
TPM_RC_P	0x040	
TPM_RC_S	0x800	
TPM_RC_1	0x100	
TPM_RC_2	0x200	
TPM_RC_3	0x300	
TPM_RC_4	0x400	
TPM_RC_5	0x500	
TPM_RC_6	0x600	
TPM_RC_7	0x700	
TPM_RC_8	0x800	

Enumerator	Value	Description
TPM_RC_9	0x900	
TPM_RC_A	0xA00	
TPM_RC_B	0xB00	
TPM_RC_C	0xC00	
TPM_RC_D	0xD00	
TPM_RC_E	0xE00	
TPM_RC_F	0xF00	
TPM_RC_N_MASK	0xF00	
TPM_RC_TIMEOUT	-100	

4.5.5.6 enum TPM_CLOCK_ADJUST_T

Enumerator	Value	Description
TPM_CLOCK_COARSE_SLOWER	-3	
TPM_CLOCK_MEDIUM_SLOWER	-2	
TPM_CLOCK_FINE_SLOWER	-1	
TPM_CLOCK_NO_CHANGE	0	
TPM_CLOCK_FINE_FASTER	1	
TPM_CLOCK_MEDIUM_FASTER	2	
TPM_CLOCK_COARSE_FASTER	3	

4.5.5.7 enum TPM_EO_T

Enumerator	Value	Description
TPM_EO_EQ	0x0000	
TPM_EO_NEQ	0x0001	
TPM_EO_SIGNED_GT	0x0002	
TPM_EO_UNSIGNED_GT	0x0003	
TPM_EO_SIGNED_LT	0x0004	
TPM_EO_UNSIGNED_LT	0x0005	
TPM_EO_SIGNED_GE	0x0006	
TPM_EO_UNSIGNED_GE	0x0007	
TPM_EO_SIGNED_LE	0x0008	
TPM_EO_UNSIGNED_LE	0x0009	
TPM_EO_BITSET	0x000A	
TPM_EO_BITCLEAR	0x000B	

4.5.5.8 enum TPM_ST_T

Enumerator	Value	Description
TPM_ST_RSP_COMMAND	0x00C4	
TPM_ST_NULL	0X8000	
TPM_ST_NO_SESSIONS	0x8001	
TPM_ST_SESSIONS	0x8002	
TPM_ST_ATTEST_NV	0x8014	
TPM_ST_ATTEST_COMMAND_AUDIT	0x8015	
TPM_ST_ATTEST_SESSION_AUDIT	0x8016	

Enumerator	Value	Description
TPM_ST_ATTEST_CERTIFY	0x8017	
TPM_ST_ATTEST_QUOTE	0x8018	
TPM_ST_ATTEST_TIME	0x8019	
TPM_ST_ATTEST_CREATION	0x801A	
TPM_ST_CREATION	0x8021	
TPM_ST_VERIFIED	0x8022	
TPM_ST_AUTH_SECRET	0x8023	
TPM_ST_HASHCHECK	0x8024	
TPM_ST_AUTH_SIGNED	0x8025	
TPM_ST_FU_MANIFEST	0x8029	

4.5.5.9 enum TPM_SE_T

Enumerator	Value	Description
TPM_SE_HMAC	0x00	
TPM_SE_POLICY	0x01	
TPM_SE_TRIAL	0x03	

4.5.5.10 enum TPM_SU_T

Enumerator	Value	Description
TPM_SU_CLEAR	0x0000	
TPM_SU_STATE	0x0001	

4.5.5.11 enum TPM_CAP_T

Enumerator	Value	Description
TPM_CAP_FIRST	0x00000000	
TPM_CAP_ALGS	TPM_CAP_FIRST	
TPM_CAP_HANDLES	0x00000001	
TPM_CAP_COMMANDS	0x00000002	
TPM_CAP_PP_COMMANDS	0x00000003	
TPM_CAP_AUDIT_COMMANDS	0x00000004	
TPM_CAP_PCRS	0x00000005	
TPM_CAP_TPM_PROPERTIES	0x00000006	
TPM_CAP_PCR_PROPERTIES	0x00000007	
TPM_CAP_ECC_CURVES	0x00000008	
TPM_CAP_LAST	TPM_CAP_ECC_CURVES	
TPM_CAP_VENDOR_PROPERTY	0x00000100	

4.5.5.12 enum TPM_PT_T

Enumerator	Value	Description
TPM_PT_NONE	0x00000000	
PT_GROUP	0x00000100	

Enumerator	Value	Description
PT_FIXED	PT_GROUP * 1	
TPM_PT_FAMILY_INDICATOR	PT_FIXED + 0	
TPM_PT_LEVEL	PT_FIXED + 1	
TPM_PT_REVISION	PT_FIXED + 2	
TPM_PT_DAY_OF_YEAR	PT_FIXED + 3	
TPM_PT_YEAR	PT_FIXED + 4	
TPM_PT_MANUFACTURER	PT_FIXED + 5	
TPM_PT_VENDOR_STRING_1	PT_FIXED + 6	
TPM_PT_VENDOR_STRING_2	PT_FIXED + 7	
TPM_PT_VENDOR_STRING_3	PT_FIXED + 8	
TPM_PT_VENDOR_STRING_4	PT_FIXED + 9	
TPM_PT_VENDOR_TPM_TYPE	PT_FIXED + 10	
TPM_PT_FIRMWARE_VERSION_1	PT_FIXED + 11	
TPM_PT_FIRMWARE_VERSION_2	PT_FIXED + 12	
TPM_PT_INPUT_BUFFER	PT_FIXED + 13	
TPM_PT_HR_TRANSIENT_MIN	PT_FIXED + 14	
TPM_PT_HR_PERSISTENT_MIN	PT_FIXED + 15	
TPM_PT_HR_LOADED_MIN	PT_FIXED + 16	
TPM_PT_ACTIVE_SESSIONS_MAX	PT_FIXED + 17	
TPM_PT_PCR_COUNT	PT_FIXED + 18	
TPM_PT_PCR_SELECT_MIN	PT_FIXED + 19	
TPM_PT_CONTEXT_GAP_MAX	PT_FIXED + 20	
TPM_PT_NV_COUNTERS_MAX	PT_FIXED + 22	
TPM_PT_NV_INDEX_MAX	PT_FIXED + 23	
TPM_PT_MEMORY	PT_FIXED + 24	
TPM_PT_CLOCK_UPDATE	PT_FIXED + 25	
TPM_PT_CONTEXT_HASH	PT_FIXED + 26	
TPM_PT_CONTEXT_SYM	PT_FIXED + 27	
TPM_PT_CONTEXT_SYM_SIZE	PT_FIXED + 28	
TPM_PT_ORDERLY_COUNT	PT_FIXED + 29	
TPM_PT_MAX_COMMAND_SIZE	PT_FIXED + 30	
TPM_PT_MAX_RESPONSE_SIZE	PT_FIXED + 31	
TPM_PT_MAX_DIGEST	PT_FIXED + 32	
TPM_PT_MAX_OBJECT_CONTEXT	PT_FIXED + 33	
TPM_PT_MAX_SESSION_CONTEXT	PT_FIXED + 34	
TPM_PT_PS_FAMILY_INDICATOR	PT_FIXED + 35	
TPM_PT_PS_LEVEL	PT_FIXED + 36	
TPM_PT_PS_REVISION	PT_FIXED + 37	
TPM_PT_PS_DAY_OF_YEAR	PT_FIXED + 38	
TPM_PT_PS_YEAR	PT_FIXED + 39	
TPM_PT_SPLIT_MAX	PT_FIXED + 40	
TPM_PT_TOTAL_COMMANDS	PT_FIXED + 41	
TPM_PT_LIBRARY_COMMANDS	PT_FIXED + 42	
TPM_PT_VENDOR_COMMANDS	PT_FIXED + 43	
TPM_PT_NV_BUFFER_MAX	PT_FIXED + 44	
TPM_PT_MODES	PT_FIXED + 45	
TPM_PT_MAX_CAP_BUFFER	PT_FIXED + 46	
PT_VAR	PT_GROUP * 2	
TPM_PT_PERMANENT	PT_VAR + 0	
TPM_PT_STARTUP_CLEAR	PT_VAR + 1	
TPM_PT_HR_NV_INDEX	PT_VAR + 2	
TPM_PT_HR_LOADED	PT_VAR + 3	

Enumerator	Value	Description
TPM_PT_HR_LOADED_AVAIL	PT_VAR + 4	
TPM_PT_HR_ACTIVE	PT_VAR + 5	
TPM_PT_HR_ACTIVE_AVAIL	PT_VAR + 6	
TPM_PT_HR_TRANSIENT_AVAIL	PT_VAR + 7	
TPM_PT_HR_PERSISTENT	PT_VAR + 8	
TPM_PT_HR_PERSISTENT_AVAIL	PT_VAR + 9	
TPM_PT_NV_COUNTERS	PT_VAR + 10	
TPM_PT_NV_COUNTERS_AVAIL	PT_VAR + 11	
TPM_PT_ALGORITHM_SET	PT_VAR + 12	
TPM_PT_LOADED_CURVES	PT_VAR + 13	
TPM_PT_LOCKOUT_COUNTER	PT_VAR + 14	
TPM_PT_MAX_AUTH_FAIL	PT_VAR + 15	
TPM_PT_LOCKOUT_INTERVAL	PT_VAR + 16	
TPM_PT_LOCKOUT_RECOVERY	PT_VAR + 17	
TPM_PT_NV_WRITE_RECOVERY	PT_VAR + 18	
TPM_PT_AUDIT_COUNTER_0	PT_VAR + 19	
TPM_PT_AUDIT_COUNTER_1	PT_VAR + 20	

4.5.5.13 enum TPM_PT_PCR_T

Enumerator	Value	Description
TPM_PT_PCR_FIRST	0x00000000	
TPM_PT_PCR_SAVE	TPM_PT_PCR_FIRST	
TPM_PT_PCR_EXTEND_L0	0x00000001	
TPM_PT_PCR_RESET_L0	0x00000002	
TPM_PT_PCR_EXTEND_L1	0x00000003	
TPM_PT_PCR_RESET_L1	0x00000004	
TPM_PT_PCR_EXTEND_L2	0x00000005	
TPM_PT_PCR_RESET_L2	0x00000006	
TPM_PT_PCR_EXTEND_L3	0x00000007	
TPM_PT_PCR_RESET_L3	0x00000008	
TPM_PT_PCR_EXTEND_L4	0x00000009	
TPM_PT_PCR_RESET_L4	0x0000000A	
TPM_PT_PCR_NO_INCREMENT	0x00000011	
TPM_PT_PCR_DRTM_RESET	0x00000012	
TPM_PT_PCR_POLICY	0x00000013	
TPM_PT_PCR_AUTH	0x00000014	
TPM_PT_PCR_LAST	TPM_PT_PCR_AUTH	

4.5.5.14 enum TPM_PS_T

Enumerator	Value	Description
TPM_PS_MAIN	0x00000000	
TPM_PS_PC	0x00000001	
TPM_PS_PDA	0x00000002	
TPM_PS_CELL_PHONE	0x00000003	
TPM_PS_SERVER	0x00000004	
TPM_PS_PERIPHERAL	0x00000005	

Enumerator	Value	Description
TPM_PS_TSS	0x00000006	
TPM_PS_STORAGE	0x00000007	
TPM_PS_AUTHENTICATION	0x00000008	
TPM_PS_EMBEDDED	0x00000009	
TPM_PS_HARDCOPY	0x0000000A	
TPM_PS_INFRASTRUCTURE	0x0000000B	
TPM_PS_VIRTUALIZATION	0x0000000C	
TPM_PS_TNC	0x0000000D	
TPM_PS_MULTI_TENANT	0x0000000E	
TPM_PS_TC	0x0000000F	

4.5.5.15 enum TPM_HT_T

Enumerator	Value	Description
TPM_HT_PCR	0x00	
TPM_HT_NV_INDEX	0x01	
TPM_HT_HMAC_SESSION	0x02	
TPM_HT_LOADED_SESSION	0x02	
TPM_HT_POLICY_SESSION	0x03	
TPM_HT_ACTIVE_SESSION	0x03	
TPM_HT_PERMANENT	0x40	
TPM_HT_TRANSIENT	0x80	
TPM_HT_PERSISTENT	0x81	

4.5.5.16 enum TPM_RH_T

Enumerator	Value	Description
TPM_RH_FIRST	0x40000000	
TPM_RH_SRK	TPM_RH_FIRST	
TPM_RH_OWNER	0x40000001	
TPM_RH_REVOKE	0x40000002	
TPM_RH_TRANSPORT	0x40000003	
TPM_RH_OPERATOR	0x40000004	
TPM_RH_ADMIN	0x40000005	
TPM_RH_EK	0x40000006	
TPM_RH_NULL	0x40000007	
TPM_RH_UNASSIGNED	0x40000008	
TPM_RS_PW	0x40000009	
TPM_RH_LOCKOUT	0x4000000A	
TPM_RH_ENDORSEMENT	0x4000000B	
TPM_RH_PLATFORM	0x4000000C	
TPM_RH_PLATFORM_NV	0x4000000D	
TPM_RH_AUTH_00	0x40000010	
TPM_RH_AUTH_FF	0x4000010F	
TPM_RH_LAST	TPM_RH_AUTH_FF	

4.5.5.17 enum TPM_HC_T

Enumerator	Value	Description
HR_HANDLE_MASK	0x00FFFFFF	
HR_RANGE_MASK	0xFF000000	
HR_SHIFT	24	
HR_PCR	((UINT32)TPM_HT_PCR « HR_SHIFT)	
HR_HMAC_SESSION	((UINT32)TPM_HT_HMAC_SESSION « HR_SHIFT)	
HR_POLICY_SESSION	((UINT32)TPM_HT_POLICY_SESSION « HR_SHIFT)	
HR_TRANSIENT	((UINT32)TPM_HT_TRANSIENT « HR_SHIFT)	
HR_PERSISTENT	((UINT32)TPM_HT_PERSISTENT « HR_SHIFT)	
HR_NV_INDEX	((UINT32)TPM_HT_NV_INDEX « HR_SHIFT)	
HR_PERMANENT	((UINT32)TPM_HT_PERMANENT « HR_SHIFT)	
PCR_FIRST	(HR_PCR + 0)	
PCR_LAST	(PCR_FIRST + IMPLEMENTATION_PCR-1)	
HMAC_SESSION_FIRST	(HR_HMAC_SESSION + 0)	
HMAC_SESSION_LAST	(HMAC_SESSION_FIRST+MAX_ACTIVE_SESSIONS-1)	
LOADED_SESSION_FIRST	HMAC_SESSION_FIRST	
LOADED_SESSION_LAST	HMAC_SESSION_LAST	
POLICY_SESSION_FIRST	(HR_POLICY_SESSION + 0)	
POLICY_SESSION_LAST	(POLICY_SESSION_FIRST+MAX_ACTIVE_SESSIONS-1)	
TRANSIENT_FIRST	(HR_TRANSIENT + 0)	
ACTIVE_SESSION_FIRST	POLICY_SESSION_FIRST	
ACTIVE_SESSION_LAST	POLICY_SESSION_LAST	
TRANSIENT_LAST	(TRANSIENT_FIRST+MAX_LOADED_OBJECTS-1)	
PERSISTENT_FIRST	(HR_PERSISTENT + 0)	
PERSISTENT_LAST	(PERSISTENT_FIRST + 0x00FFFFFF)	
PLATFORM_PERSISTENT	(PERSISTENT_FIRST + 0x00800000)	
NV_INDEX_FIRST	(HR_NV_INDEX + 0)	
NV_INDEX_LAST	(NV_INDEX_FIRST + 0x00FFFFFF)	
PERMANENT_FIRST	TPM_RH_FIRST	
PERMANENT_LAST	TPM_RH_LAST	

4.5.5.18 enum TPMA_ALGORITHM_mask

Enumerator	Value	Description
TPMA_ALGORITHM_asymmetric	0x00000001	
TPMA_ALGORITHM_symmetric	0x00000002	
TPMA_ALGORITHM_hash	0x00000004	
TPMA_ALGORITHM_object	0x00000008	
TPMA_ALGORITHM_signing	0x00000010	
TPMA_ALGORITHM_encrypting	0x00000020	
TPMA_ALGORITHM_method	0x00000040	

4.5.5.19 enum TPMA_OBJECT_mask

Enumerator	Value	Description
TPMA_OBJECT_fixedTPM	0x00000002	
TPMA_OBJECT_stClear	0x00000004	
TPMA_OBJECT_fixedParent	0x00000010	
TPMA_OBJECT_sensitiveDataOrigin	0x00000020	

Enumerator	Value	Description
TPMA_OBJECT_userWithAuth	0x00000040	
TPMA_OBJECT_adminWithPolicy	0x00000080	
TPMA_OBJECT_derivedDataOrigin	0x00000200	
TPMA_OBJECT_noDA	0x00000400	
TPMA_OBJECT_encryptedDuplication	0x00000800	
TPMA_OBJECT_restricted	0x00010000	
TPMA_OBJECT_decrypt	0x00020000	
TPMA_OBJECT_sign	0x00040000	

4.5.5.20 enum TPMA_SESSION_mask

Enumerator	Value	Description
TPMA_SESSION_continueSession	0x01	
TPMA_SESSION_auditExclusive	0x02	
TPMA_SESSION_auditReset	0x04	
TPMA_SESSION_decrypt	0x20	
TPMA_SESSION_encrypt	0x40	
TPMA_SESSION_audit	0x80	

4.5.5.21 enum TPMA_LOCALITY_mask

Enumerator	Value	Description
TPM_LOC_ZERO	0x01	
TPM_LOC_ONE	0x02	
TPM_LOC_TWO	0x04	
TPM_LOC_THREE	0x08	
TPM_LOC_FOUR	0x10	

4.5.5.22 enum TPMA_PERMANENT_mask

Enumerator	Value	Description
TPMA_PERMANENT_ownerAuthSet	0x00000001	
TPMA_PERMANENT_endorsementAuthSet	0x00000002	
TPMA_PERMANENT_lockoutAuthSet	0x00000004	
TPMA_PERMANENT_disableClear	0x00000100	
TPMA_PERMANENT_inLockout	0x00000200	
TPMA_PERMANENT_tpmGeneratedEPS	0x00000400	

4.5.5.23 enum TPMA_STARTUP_CLEAR_mask

Enumerator	Value	Description
TPMA_STARTUP_CLEAR_phEnable	0x00000001	
TPMA_STARTUP_CLEAR_shEnable	0x00000002	
TPMA_STARTUP_CLEAR_ehEnable	0x00000004	
TPMA_STARTUP_CLEAR_phEnableNV	0x00000008	

Enumerator	Value	Description
TPMA_STARTUP_CLEAR_orderly	0x80000000	

4.5.5.24 enum TPMA_MEMORY_mask

Enumerator	Value	Description
TPMA_MEMORY_sharedRAM	0x00000001	
TPMA_MEMORY_sharedNV	0x00000002	
TPMA_MEMORY_objectCopiedToRam	0x00000004	

4.5.5.25 enum TPMA_CC_mask

Enumerator	Value	Description
TPMA_CC_commandIndex	0x0000FFFF	
TPMA_CC_nv	0x00400000	
TPMA_CC_extensive	0x00800000	
TPMA_CC_flushed	0x01000000	
TPMA_CC_cHandles	0x0E000000	
TPMA_CC_rHandle	0x10000000	
TPMA_CC_V	0x20000000	

4.5.5.26 enum TPM_NV_INDEX_mask

Enumerator	Value	Description
TPM_NV_INDEX_index	0x00FFFFFF	
TPM_NV_INDEX_RH_NV	0xFF000000	

4.5.5.27 enum TPM_NT

Enumerator	Value	Description
TPM_NT_ORDINARY	0x0	
TPM_NT_COUNTER	0x1	
TPM_NT_BITS	0x2	
TPM_NT_EXTEND	0x4	
TPM_NT_PIN_FAIL	0x8	
TPM_NT_PIN_PASS	0x9	

4.5.5.28 enum TPMA_NV_mask

Enumerator	Value	Description
TPMA_NV_PPWRITE	0x00000001	
TPMA_NV_OWNERWRITE	0x00000002	
TPMA_NV_AUTHWRITE	0x00000004	
TPMA_NV_POLICYWRITE	0x00000008	

Enumerator	Value	Description
TPMA_NV_TPM_NT	0x000000F0	
TPMA_NV_POLICY_DELETE	0x00000400	
TPMA_NV_WRITELOCKED	0x00000800	
TPMA_NV_WRITEALL	0x00001000	
TPMA_NV_WRITEDEFINE	0x00002000	
TPMA_NV_WRITE_STCLEAR	0x00004000	
TPMA_NV_GLOBALLOCK	0x00008000	
TPMA_NV_PPREAD	0x00010000	
TPMA_NV_OWNERREAD	0x00020000	
TPMA_NV_AUTHREAD	0x00040000	
TPMA_NV_POLICYREAD	0x00080000	
TPMA_NV_NO_DA	0x02000000	
TPMA_NV_ORDERLY	0x04000000	
TPMA_NV_CLEAR_STCLEAR	0x08000000	
TPMA_NV_READLOCKED	0x10000000	
TPMA_NV_WRITTEN	0x20000000	
TPMA_NV_PLATFORMCREATE	0x40000000	
TPMA_NV_READ_STCLEAR	0x80000000	

4.5.5.29 enum @1

Enumerator	Value	Description
TPMLib_2	0x01	
TPMFips	0x02	
TPMLowPowerOff	0x00	
TPMLowPowerByRegister	0x04	
TPMLowPowerByGpio	0x08	
TPMLowPowerAuto	0x0C	

4.5.5.30 enum TPMI_GPIO_NAME_T

Enumerator	Value	Description
TPM_GPIO_PP	0x00000000	
TPM_GPIO_LP	0x00000001	
TPM_GPIO_C	0x00000002	
TPM_GPIO_D	0x00000003	

4.5.5.31 enum TPMI_GPIO_MODE_T

Enumerator	Value	Description
TPM_GPIO_MODE_STANDARD	0x00000000	
TPM_GPIO_MODE_FLOATING	0x00000001	
TPM_GPIO_MODE_PULLUP	0x00000002	
TPM_GPIO_MODE_PULLDOWN	0x00000003	
TPM_GPIO_MODE_OPENDRAIN	0x00000004	
TPM_GPIO_MODE_PUSHPULL	0x00000005	
TPM_GPIO_MODE_UNCONFIG	0x00000006	

Enumerator	Value	Description
TPM_GPIO_MODE_DEFAULT	TPM_GPIO_MODE_PULLDOWN	
TPM_GPIO_MODE_MAX	TPM_GPIO_MODE_UNCONFIG	
TPM_GPIO_MODE_INPUT_MIN	TPM_GPIO_MODE_FLOATING	
TPM_GPIO_MODE_INPUT_MAX	TPM_GPIO_MODE_PULLDOWN	
TPM_GPIO_MODE_PUSHPULL	0x00000005	
TPM_GPIO_MODE_OPENDRAIN	0x00000004	
TPM_GPIO_MODE_PULLUP	0x00000002	
TPM_GPIO_MODE_UNCONFIG	0x00000006	
TPM_GPIO_MODE_DEFAULT	TPM_GPIO_MODE_PULLDOWN	
TPM_GPIO_MODE_MAX	TPM_GPIO_MODE_UNCONFIG	
TPM_GPIO_MODE_INPUT_MIN	TPM_GPIO_MODE_FLOATING	
TPM_GPIO_MODE_INPUT_MAX	TPM_GPIO_MODE_PULLDOWN	

4.5.5.32 enum TPMI_GPIO_MODE_T

Enumerator	Value	Description
TPM_GPIO_MODE_STANDARD	0x00000000	
TPM_GPIO_MODE_FLOATING	0x00000001	
TPM_GPIO_MODE_PULLUP	0x00000002	
TPM_GPIO_MODE_PULLDOWN	0x00000003	
TPM_GPIO_MODE_OPENDRAIN	0x00000004	
TPM_GPIO_MODE_PUSHPULL	0x00000005	
TPM_GPIO_MODE_UNCONFIG	0x00000006	
TPM_GPIO_MODE_DEFAULT	TPM_GPIO_MODE_PULLDOWN	
TPM_GPIO_MODE_MAX	TPM_GPIO_MODE_UNCONFIG	
TPM_GPIO_MODE_INPUT_MIN	TPM_GPIO_MODE_FLOATING	
TPM_GPIO_MODE_INPUT_MAX	TPM_GPIO_MODE_PULLDOWN	
TPM_GPIO_MODE_PUSHPULL	0x00000005	
TPM_GPIO_MODE_OPENDRAIN	0x00000004	
TPM_GPIO_MODE_PULLUP	0x00000002	
TPM_GPIO_MODE_UNCONFIG	0x00000006	
TPM_GPIO_MODE_DEFAULT	TPM_GPIO_MODE_PULLDOWN	
TPM_GPIO_MODE_MAX	TPM_GPIO_MODE_UNCONFIG	
TPM_GPIO_MODE_INPUT_MIN	TPM_GPIO_MODE_FLOATING	
TPM_GPIO_MODE_INPUT_MAX	TPM_GPIO_MODE_PULLDOWN	

4.5.5.33 enum TPM_Vendor_t

Enumerator	Value	Description
TPM_VENDOR_UNKNOWN	0	
TPM_VENDOR_INFINEON	0x15d1	
TPM_VENDOR_STM	0x104a	
TPM_VENDOR_MCHP	0x1114	
TPM_VENDOR_NUVOTON	0x1050	
TPM_VENDOR_NATIONTECH	0x1B4E	

4.5.5.34 typedef TPM_MODIFIER_INDICATOR

```
typedef UINT32 TPM_MODIFIER_INDICATOR;
```

4.5.5.35 typedef TPM_AUTHORIZATION_SIZE

```
typedef UINT32 TPM_AUTHORIZATION_SIZE;
```

4.5.5.36 typedef TPM_PARAMETER_SIZE

```
typedef UINT32 TPM_PARAMETER_SIZE;
```

4.5.5.37 typedef TPM_KEY_SIZE

```
typedef UINT16 TPM_KEY_SIZE;
```

4.5.5.38 typedef TPM_KEY_BITS

```
typedef UINT16 TPM_KEY_BITS;
```

4.5.5.39 typedef TPM_GENERATED

```
typedef UINT32 TPM_GENERATED;
```

4.5.5.40 typedef TPM_ALG_ID

```
typedef UINT16 TPM_ALG_ID;
```

4.5.5.41 typedef TPM_ECC_CURVE

```
typedef UINT16 TPM_ECC_CURVE;
```

4.5.5.42 typedef TPM_CC

```
typedef UINT32 TPM_CC;
```

4.5.5.43 typedef TPM_RC

```
typedef INT32 TPM_RC;
```

4.5.5.44 typedef TPM_CLOCK_ADJUST

```
typedef UINT8 TPM_CLOCK_ADJUST;
```

4.5.5.45 typedef TPM_EO

```
typedef UINT16 TPM_EO;
```

4.5.5.46 typedef TPM_ST

```
typedef UINT16 TPM_ST;
```

4.5.5.47 typedef TPM_SE

```
typedef UINT8 TPM_SE;
```

4.5.5.48 typedef TPM_SU

```
typedef UINT16 TPM_SU;
```

4.5.5.49 typedef TPM_CAP

```
typedef UINT32 TPM_CAP;
```

4.5.5.50 typedef TPM_PT

```
typedef UINT32 TPM_PT;
```

4.5.5.51 typedef TPM_PT_PCR

```
typedef UINT32 TPM_PT_PCR;
```

4.5.5.52 typedef TPM_PS

```
typedef UINT32 TPM_PS;
```

4.5.5.53 typedef TPM_HANDLE

```
typedef UINT32 TPM_HANDLE;
```

4.5.5.54 typedef TPM_HT

```
typedef UINT8 TPM_HT;
```

4.5.5.55 typedef TPM_RH

```
typedef UINT32 TPM_RH;
```

4.5.5.56 typedef TPM_HC

```
typedef UINT32 TPM_HC;
```

4.5.5.57 typedef TPMA_ALGORITHM

```
typedef UINT32 TPMA_ALGORITHM;
```

4.5.5.58 typedef TPMA_OBJECT

```
typedef UINT32 TPMA_OBJECT;
```

4.5.5.59 typedef TPMA_SESSION

```
typedef BYTE TPMA_SESSION;
```

4.5.5.60 typedef TPMA_LOCALITY

```
typedef BYTE TPMA_LOCALITY;
```

4.5.5.61 typedef TPMA_PERMANENT

```
typedef UINT32 TPMA_PERMANENT;
```

4.5.5.62 typedef TPMA_STARTUP_CLEAR

```
typedef UINT32 TPMA_STARTUP_CLEAR;
```

4.5.5.63 typedef TPMA_MEMORY

```
typedef UINT32 TPMA_MEMORY;
```

4.5.5.64 typedef TPMA_CC

```
typedef UINT32 TPMA_CC;
```

4.5.5.65 typedef TPMI_YES_NO

```
typedef BYTE TPMI_YES_NO;
```

4.5.5.66 typedef TPMI_DH_OBJECT

```
typedef TPM_HANDLE TPMI_DH_OBJECT;
```

4.5.5.67 typedef TPMI_DH_PARENT

```
typedef TPM_HANDLE TPMI_DH_PARENT;
```

4.5.5.68 typedef TPMI_DH_PERSISTENT

```
typedef TPM_HANDLE TPMI_DH_PERSISTENT;
```

4.5.5.69 typedef TPMI_DH_ENTITY

```
typedef TPM_HANDLE TPMI_DH_ENTITY;
```

4.5.5.70 typedef TPMI_DH_PCR

```
typedef TPM_HANDLE TPMI_DH_PCR;
```

4.5.5.71 typedef TPMI_SH_AUTH_SESSION

```
typedef TPM_HANDLE TPMI_SH_AUTH_SESSION;
```

4.5.5.72 typedef TPMI_SH_HMAC

```
typedef TPM_HANDLE TPMI_SH_HMAC;
```

4.5.5.73 typedef TPMI_SH_POLICY

```
typedef TPM_HANDLE TPMI_SH_POLICY;
```

4.5.5.74 typedef TPMI_DH_CONTEXT

```
typedef TPM_HANDLE TPMI_DH_CONTEXT;
```

4.5.5.75 typedef TPMI_RH_HIERARCHY

```
typedef TPM_HANDLE TPMI_RH_HIERARCHY;
```

4.5.5.76 typedef TPMI_RH_ENABLES

```
typedef TPM_HANDLE TPMI_RH_ENABLES;
```

4.5.5.77 typedef TPMI_RH_HIERARCHY_AUTH

```
typedef TPM_HANDLE TPMI_RH_HIERARCHY_AUTH;
```

4.5.5.78 typedef TPMI_RH_PLATFORM

```
typedef TPM_HANDLE TPMI_RH_PLATFORM;
```

4.5.5.79 typedef TPMI_RH_OWNER

```
typedef TPM_HANDLE TPMI_RH_OWNER;
```

4.5.5.80 typedef TPMI_RH_ENDORSEMENT

```
typedef TPM_HANDLE TPMI_RH_ENDORSEMENT;
```

4.5.5.81 typedef TPMI_RH_PROVISION

```
typedef TPM_HANDLE TPMI_RH_PROVISION;
```

4.5.5.82 typedef TPMI_RH_CLEAR

```
typedef TPM_HANDLE TPMI_RH_CLEAR;
```

4.5.5.83 typedef TPMI_RH_NV_AUTH

```
typedef TPM_HANDLE TPMI_RH_NV_AUTH;
```

4.5.5.84 typedef TPMI_RH_LOCKOUT

```
typedef TPM_HANDLE TPMI_RH_LOCKOUT;
```

4.5.5.85 typedef TPMI_RH_NV_INDEX

```
typedef TPM_HANDLE TPMI_RH_NV_INDEX;
```

4.5.5.86 typedef TPMI_ALG_HASH

```
typedef TPM_ALG_ID TPMI_ALG_HASH;
```

4.5.5.87 typedef TPMI_ALG_ASYM

```
typedef TPM_ALG_ID TPMI_ALG_ASYM;
```

4.5.5.88 typedef TPMI_ALG_SYM

```
typedef TPM_ALG_ID TPMI_ALG_SYM;
```

4.5.5.89 typedef TPMI_ALG_SYM_OBJECT

```
typedef TPM_ALG_ID TPMI_ALG_SYM_OBJECT;
```

4.5.5.90 typedef TPMI_ALG_SYM_MODE

```
typedef TPM_ALG_ID TPMI_ALG_SYM_MODE;
```

4.5.5.91 typedef TPMI_ALG_KDF

```
typedef TPM_ALG_ID TPMI_ALG_KDF;
```

4.5.5.92 typedef TPMI_ALG_SIG_SCHEME

```
typedef TPM_ALG_ID TPMI_ALG_SIG_SCHEME;
```

4.5.5.93 typedef TPMI_ECC_KEY_EXCHANGE

```
typedef TPM_ALG_ID TPMI_ECC_KEY_EXCHANGE;
```

4.5.5.94 typedef TPMI_ST_COMMAND_TAG

```
typedef TPM_ST TPMI_ST_COMMAND_TAG;
```

4.5.5.95 typedef TPMS_ALGORITHM_DESCRIPTION

```
typedef struct TPMS_ALGORITHM_DESCRIPTION TPMS_ALGORITHM_DESCRIPTION;
```

4.5.5.96 typedef TPMU_HA

```
typedef union TPMU_HA TPMU_HA;
```

4.5.5.97 typedef TPMT_HA

```
typedef struct TPMT_HA TPMT_HA;
```

4.5.5.98 typedef TPM2B_DIGEST

```
typedef struct TPM2B_DIGEST TPM2B_DIGEST;
```

4.5.5.99 typedef TPM2B_DATA

```
typedef struct TPM2B_DATA TPM2B_DATA;
```

4.5.5.100 typedef TPM2B_NONCE

```
typedef TPM2B_DIGEST TPM2B_NONCE;
```

4.5.5.101 typedef TPM2B_AUTH

```
typedef TPM2B_DIGEST TPM2B_AUTH;
```

4.5.5.102 typedef TPM2B_OPERAND

```
typedef TPM2B_DIGEST TPM2B_OPERAND;
```

4.5.5.103 typedef TPM2B_EVENT

```
typedef struct TPM2B_EVENT TPM2B_EVENT;
```

4.5.5.104 typedef TPM2B_MAX_BUFFER

```
typedef struct TPM2B_MAX_BUFFER TPM2B_MAX_BUFFER;
```

4.5.5.105 typedef TPM2B_MAX_NV_BUFFER

```
typedef struct TPM2B_MAX_NV_BUFFER TPM2B_MAX_NV_BUFFER;
```

4.5.5.106 typedef TPM2B_TIMEOUT

```
typedef TPM2B_DIGEST TPM2B_TIMEOUT;
```

4.5.5.107 typedef TPM2B_IV

```
typedef struct TPM2B_IV TPM2B_IV;
```

4.5.5.108 typedef TPMU_NAME

```
typedef union TPMU_NAME TPMU_NAME;
```

4.5.5.109 typedef TPM2B_NAME

```
typedef struct TPM2B_NAME TPM2B_NAME;
```

4.5.5.110 typedef TPMS_PCR_SELECT

```
typedef struct TPMS_PCR_SELECT TPMS_PCR_SELECT;
```

4.5.5.111 typedef TPMS_PCR_SELECTION

```
typedef struct TPMS_PCR_SELECTION TPMS_PCR_SELECTION;
```

4.5.5.112 typedef TPMT_TK_CREATION

```
typedef struct TPMT_TK_CREATION TPMT_TK_CREATION;
```

4.5.5.113 typedef TPMT_TK_VERIFIED

```
typedef struct TPMT_TK_VERIFIED TPMT_TK_VERIFIED;
```

4.5.5.114 typedef TPMT_TK_AUTH

```
typedef struct TPMT_TK_AUTH TPMT_TK_AUTH;
```

4.5.5.115 typedef TPMT_TK_HASHCHECK

```
typedef struct TPMT_TK_HASHCHECK TPMT_TK_HASHCHECK;
```

4.5.5.116 typedef TPMS_ALG_PROPERTY

```
typedef struct TPMS_ALG_PROPERTY TPMS_ALG_PROPERTY;
```

4.5.5.117 typedef TPMS_TAGGED_PROPERTY

```
typedef struct TPMS_TAGGED_PROPERTY TPMS_TAGGED_PROPERTY;
```

4.5.5.118 typedef TPMS_TAGGED_PCR_SELECT

```
typedef struct TPMS_TAGGED_PCR_SELECT TPMS_TAGGED_PCR_SELECT;
```

4.5.5.119 typedef TPMS_TAGGED_POLICY

```
typedef struct TPMS_TAGGED_POLICY TPMS_TAGGED_POLICY;
```

4.5.5.120 typedef TPML_CC

```
typedef struct TPML_CC TPML_CC;
```

4.5.5.121 typedef TPML_CCA

```
typedef struct TPML_CCA TPML_CCA;
```

4.5.5.122 typedef TPML_ALG

```
typedef struct TPML_ALG TPML_ALG;
```

4.5.5.123 typedef TPML_HANDLE

```
typedef struct TPML_HANDLE TPML_HANDLE;
```

4.5.5.124 typedef TPML_DIGEST

```
typedef struct TPML_DIGEST TPML_DIGEST;
```

4.5.5.125 typedef TPML_DIGEST_VALUES

```
typedef struct TPML_DIGEST_VALUES TPML_DIGEST_VALUES;
```

4.5.5.126 typedef TPML_PCR_SELECTION

```
typedef struct TPML_PCR_SELECTION TPML_PCR_SELECTION;
```

4.5.5.127 typedef TPML_ALG_PROPERTY

```
typedef struct TPML_ALG_PROPERTY TPML_ALG_PROPERTY;
```

4.5.5.128 typedef TPML_TAGGED_TPM_PROPERTY

```
typedef struct TPML_TAGGED_TPM_PROPERTY TPML_TAGGED_TPM_PROPERTY;
```

4.5.5.129 typedef TPML_TAGGED_PCR_PROPERTY

```
typedef struct TPML_TAGGED_PCR_PROPERTY TPML_TAGGED_PCR_PROPERTY;
```

4.5.5.130 typedef TPML_ECC_CURVE

```
typedef struct TPML_ECC_CURVE TPML_ECC_CURVE;
```

4.5.5.131 typedef TPML_TAGGED_POLICY

```
typedef struct TPML_TAGGED_POLICY TPML_TAGGED_POLICY;
```

4.5.5.132 typedef TPMU_CAPABILITIES

```
typedef union TPMU_CAPABILITIES TPMU_CAPABILITIES;
```

4.5.5.133 typedef TPMS_CAPABILITY_DATA

```
typedef struct TPMS_CAPABILITY_DATA TPMS_CAPABILITY_DATA;
```

4.5.5.134 typedef TPMS_CLOCK_INFO

```
typedef struct TPMS_CLOCK_INFO TPMS_CLOCK_INFO;
```

4.5.5.135 typedef TPMS_TIME_INFO

```
typedef struct TPMS_TIME_INFO TPMS_TIME_INFO;
```

4.5.5.136 typedef TPMS_TIME_ATTEST_INFO

```
typedef struct TPMS_TIME_ATTEST_INFO TPMS_TIME_ATTEST_INFO;
```

4.5.5.137 typedef TPMS_CERTIFY_INFO

```
typedef struct TPMS_CERTIFY_INFO TPMS_CERTIFY_INFO;
```

4.5.5.138 typedef TPMS_QUOTE_INFO

```
typedef struct TPMS_QUOTE_INFO TPMS_QUOTE_INFO;
```

4.5.5.139 typedef TPMS_COMMAND_AUDIT_INFO

```
typedef struct TPMS_COMMAND_AUDIT_INFO TPMS_COMMAND_AUDIT_INFO;
```

4.5.5.140 typedef TPMS_SESSION_AUDIT_INFO

```
typedef struct TPMS_SESSION_AUDIT_INFO TPMS_SESSION_AUDIT_INFO;
```

4.5.5.141 typedef TPMS_CREATION_INFO

```
typedef struct TPMS_CREATION_INFO TPMS_CREATION_INFO;
```

4.5.5.142 typedef TPMS_NV_CERTIFY_INFO

```
typedef struct TPMS_NV_CERTIFY_INFO TPMS_NV_CERTIFY_INFO;
```

4.5.5.143 typedef TPMI_ST_ATTEST

```
typedef TPM_ST TPMI_ST_ATTEST;
```

4.5.5.144 typedef TPMU_ATTEST

```
typedef union TPMU_ATTEST TPMU_ATTEST;
```

4.5.5.145 typedef TPMS_ATTEST

```
typedef struct TPMS_ATTEST TPMS_ATTEST;
```

4.5.5.146 typedef TPM2B_ATTEST

```
typedef struct TPM2B_ATTEST TPM2B_ATTEST;
```

4.5.5.147 typedef TPMI_AES_KEY_BITS

```
typedef TPM_KEY_BITS TPMI_AES_KEY_BITS;
```

4.5.5.148 typedef TPMU_SYM_KEY_BITS

```
typedef union TPMU_SYM_KEY_BITS TPMU_SYM_KEY_BITS;
```

4.5.5.149 typedef TPMU_SYM_MODE

```
typedef union TPMU_SYM_MODE TPMU_SYM_MODE;
```

4.5.5.150 typedef TPMT_SYM_DEF

```
typedef struct TPMT_SYM_DEF TPMT_SYM_DEF;
```

4.5.5.151 typedef TPMT_SYM_DEF_OBJECT

```
typedef TPMT_SYM_DEF TPMT_SYM_DEF_OBJECT;
```

4.5.5.152 typedef TPM2B_SYM_KEY

```
typedef struct TPM2B_SYM_KEY TPM2B_SYM_KEY;
```

4.5.5.153 typedef TPMS_SYMCIPHER_PARMS

```
typedef struct TPMS_SYMCIPHER_PARMS TPMS_SYMCIPHER_PARMS;
```

4.5.5.154 typedef TPM2B_LABEL

```
typedef struct TPM2B_LABEL TPM2B_LABEL;
```

4.5.5.155 typedef TPMS_DERIVE

```
typedef struct TPMS_DERIVE TPMS_DERIVE;
```

4.5.5.156 typedef TPM2B_DERIVE

```
typedef struct TPM2B_DERIVE TPM2B_DERIVE;
```

4.5.5.157 typedef TPMU_SENSITIVE_CREATE

```
typedef union TPMU_SENSITIVE_CREATE TPMU_SENSITIVE_CREATE;
```

4.5.5.158 typedef TPM2B_SENSITIVE_DATA

```
typedef struct TPM2B_SENSITIVE_DATA TPM2B_SENSITIVE_DATA;
```

4.5.5.159 typedef TPMS_SENSITIVE_CREATE

```
typedef struct TPMS_SENSITIVE_CREATE TPMS_SENSITIVE_CREATE;
```

4.5.5.160 typedef TPM2B_SENSITIVE_CREATE

```
typedef struct TPM2B_SENSITIVE_CREATE TPM2B_SENSITIVE_CREATE;
```

4.5.5.161 typedef TPMS_SCHEME_HASH

```
typedef struct TPMS_SCHEME_HASH TPMS_SCHEME_HASH;
```

4.5.5.162 typedef TPMS_SCHEME_ECDAA

```
typedef struct TPMS_SCHEME_ECDAA TPMS_SCHEME_ECDAA;
```

4.5.5.163 typedef TPMI_ALG_KEYEDHASH_SCHEME

```
typedef TPM_ALG_ID TPMI_ALG_KEYEDHASH_SCHEME;
```

4.5.5.164 typedef TPMS_SCHEME_HMAC

```
typedef TPMS_SCHEME_HASH TPMS_SCHEME_HMAC;
```

4.5.5.165 typedef TPMU_SCHEME_KEYEDHASH

```
typedef union TPMU_SCHEME_KEYEDHASH TPMU_SCHEME_KEYEDHASH;
```

4.5.5.166 typedef TPMT_KEYEDHASH_SCHEME

```
typedef struct TPMT_KEYEDHASH_SCHEME TPMT_KEYEDHASH_SCHEME;
```

4.5.5.167 typedef TPMS_SIG_SCHEME_RSASSA

```
typedef TPMS_SCHEME_HASH TPMS_SIG_SCHEME_RSASSA;
```

4.5.5.168 typedef TPMS_SIG_SCHEME_RSAPSS

```
typedef TPMS_SCHEME_HASH TPMS_SIG_SCHEME_RSAPSS;
```

4.5.5.169 typedef TPMS_SIG_SCHEME_ECDSA

```
typedef TPMS_SCHEME_HASH TPMS_SIG_SCHEME_ECDSA;
```

4.5.5.170 typedef TPMS_SIG_SCHEME_ECDAA

```
typedef TPMS_SCHEME_ECDAA TPMS_SIG_SCHEME_ECDAA;
```

4.5.5.171 typedef TPMU_SIG_SCHEME

```
typedef union TPMU_SIG_SCHEME TPMU_SIG_SCHEME;
```

4.5.5.172 typedef TPMT_SIG_SCHEME

```
typedef struct TPMT_SIG_SCHEME TPMT_SIG_SCHEME;
```

4.5.5.173 typedef TPMS_ENC_SCHEME_OAEP

```
typedef TPMS_SCHEME_HASH TPMS_ENC_SCHEME_OAEP;
```

4.5.5.174 typedef TPMS_KEY_SCHEME_ECDH

```
typedef TPMS_SCHEME_HASH TPMS_KEY_SCHEME_ECDH;
```

4.5.5.175 typedef TPMS_KEY_SCHEME_ECMQV

```
typedef TPMS_SCHEME_HASH TPMS_KEY_SCHEME_ECMQV;
```

4.5.5.176 typedef TPMS_SCHEME_MGF1

```
typedef TPMS_SCHEME_HASH TPMS_SCHEME_MGF1;
```

4.5.5.177 typedef TPMS_SCHEME_KDF1_SP800_56A

```
typedef TPMS_SCHEME_HASH TPMS_SCHEME_KDF1_SP800_56A;
```

4.5.5.178 typedef TPMS_SCHEME_KDF2

```
typedef TPMS_SCHEME_HASH TPMS_SCHEME_KDF2;
```

4.5.5.179 typedef TPMS_SCHEME_KDF1_SP800_108

```
typedef TPMS_SCHEME_HASH TPMS_SCHEME_KDF1_SP800_108;
```

4.5.5.180 typedef TPMU_KDF_SCHEME

```
typedef union TPMU_KDF_SCHEME TPMU_KDF_SCHEME;
```

4.5.5.181 typedef TPMT_KDF_SCHEME

```
typedef struct TPMT_KDF_SCHEME TPMT_KDF_SCHEME;
```

4.5.5.182 typedef TPMI_ALG_ASYM_SCHEME

```
typedef TPM_ALG_ID TPMI_ALG_ASYM_SCHEME;
```

4.5.5.183 typedef TPMU_ASYM_SCHEME

```
typedef union TPMU_ASYM_SCHEME TPMU_ASYM_SCHEME;
```

4.5.5.184 typedef TPMT_ASYM_SCHEME

```
typedef struct TPMT_ASYM_SCHEME TPMT_ASYM_SCHEME;
```

4.5.5.185 typedef TPMI_ALG_RSA_SCHEME

```
typedef TPM_ALG_ID TPMI_ALG_RSA_SCHEME;
```

4.5.5.186 typedef TPMT_RSA_SCHEME

```
typedef struct TPMT_RSA_SCHEME TPMT_RSA_SCHEME;
```

4.5.5.187 typedef TPMI_ALG_RSA_DECRYPT

```
typedef TPM_ALG_ID TPMI_ALG_RSA_DECRYPT;
```

4.5.5.188 typedef TPMT_RSA_DECRYPT

```
typedef struct TPMT_RSA_DECRYPT TPMT_RSA_DECRYPT;
```

4.5.5.189 typedef TPM2B_PUBLIC_KEY_RSA

```
typedef struct TPM2B_PUBLIC_KEY_RSA TPM2B_PUBLIC_KEY_RSA;
```

4.5.5.190 typedef TPMI_RSA_KEY_BITS

```
typedef TPM_KEY_BITS TPMI_RSA_KEY_BITS;
```

4.5.5.191 typedef TPM2B_PRIVATE_KEY_RSA

```
typedef struct TPM2B_PRIVATE_KEY_RSA TPM2B_PRIVATE_KEY_RSA;
```

4.5.5.192 typedef TPM2B_ECC_PARAMETER

```
typedef struct TPM2B_ECC_PARAMETER TPM2B_ECC_PARAMETER;
```

4.5.5.193 typedef TPMS_ECC_POINT

```
typedef struct TPMS_ECC_POINT TPMS_ECC_POINT;
```

4.5.5.194 typedef TPM2B_ECC_POINT

```
typedef struct TPM2B_ECC_POINT TPM2B_ECC_POINT;
```

4.5.5.195 typedef TPMI_ALG_ECC_SCHEME

```
typedef TPM_ALG_ID TPMI_ALG_ECC_SCHEME;
```

4.5.5.196 typedef TPMI_ECC_CURVE

```
typedef TPM_ECC_CURVE TPMI_ECC_CURVE;
```

4.5.5.197 typedef TPMT_ECC_SCHEME

```
typedef TPMT_SIG_SCHEME TPMT_ECC_SCHEME;
```

4.5.5.198 typedef TPMS_ALGORITHM_DETAIL_ECC

```
typedef struct TPMS_ALGORITHM_DETAIL_ECC TPMS_ALGORITHM_DETAIL_ECC;
```

4.5.5.199 typedef TPMS_SIGNATURE_RSA

```
typedef struct TPMS_SIGNATURE_RSA TPMS_SIGNATURE_RSA;
```

4.5.5.200 typedef TPMS_SIGNATURE_RSASSA

```
typedef TPMS_SIGNATURE_RSA TPMS_SIGNATURE_RSASSA;
```

4.5.5.201 typedef TPMS_SIGNATURE_RSAPSS

```
typedef TPMS_SIGNATURE_RSA TPMS_SIGNATURE_RSAPSS;
```

4.5.5.202 typedef TPMS_SIGNATURE_ECC

```
typedef struct TPMS_SIGNATURE_ECC TPMS_SIGNATURE_ECC;
```

4.5.5.203 typedef TPMS_SIGNATURE_ECDSA

```
typedef TPMS_SIGNATURE_ECC TPMS_SIGNATURE_ECDSA;
```

4.5.5.204 typedef TPMS_SIGNATURE_ECDAA

```
typedef TPMS_SIGNATURE_ECC TPMS_SIGNATURE_ECDAA;
```

4.5.5.205 typedef TPMU_SIGNATURE

```
typedef union TPMU_SIGNATURE TPMU_SIGNATURE;
```

4.5.5.206 typedef TPMT_SIGNATURE

```
typedef struct TPMT_SIGNATURE TPMT_SIGNATURE;
```

4.5.5.207 typedef TPMU_ENCRYPTED_SECRET

```
typedef union TPMU_ENCRYPTED_SECRET TPMU_ENCRYPTED_SECRET;
```

4.5.5.208 typedef TPM2B_ENCRYPTED_SECRET

```
typedef struct TPM2B_ENCRYPTED_SECRET TPM2B_ENCRYPTED_SECRET;
```

4.5.5.209 typedef TPMI_ALG_PUBLIC

```
typedef TPM_ALG_ID TPMI_ALG_PUBLIC;
```

4.5.5.210 typedef TPMU_PUBLIC_ID

```
typedef union TPMU_PUBLIC_ID TPMU_PUBLIC_ID;
```

4.5.5.211 typedef TPMS_KEYEDHASH_PARMS

```
typedef struct TPMS_KEYEDHASH_PARMS TPMS_KEYEDHASH_PARMS;
```

4.5.5.212 typedef TPMS_ASYM_PARMS

```
typedef struct TPMS_ASYM_PARMS TPMS_ASYM_PARMS;
```

4.5.5.213 typedef TPMS_RSA_PARMS

```
typedef struct TPMS_RSA_PARMS TPMS_RSA_PARMS;
```

4.5.5.214 typedef TPMS_ECC_PARMS

```
typedef struct TPMS_ECC_PARMS TPMS_ECC_PARMS;
```

4.5.5.215 typedef TPMU_PUBLIC_PARMS

```
typedef union TPMU_PUBLIC_PARMS TPMU_PUBLIC_PARMS;
```

4.5.5.216 typedef TPMT_PUBLIC_PARMS

```
typedef struct TPMT_PUBLIC_PARMS TPMT_PUBLIC_PARMS;
```

4.5.5.217 typedef TPMT_PUBLIC

```
typedef struct TPMT_PUBLIC TPMT_PUBLIC;
```

4.5.5.218 typedef TPM2B_PUBLIC

```
typedef struct TPM2B_PUBLIC TPM2B_PUBLIC;
```

4.5.5.219 typedef TPM2B_TEMPLATE

```
typedef struct TPM2B_TEMPLATE TPM2B_TEMPLATE;
```

4.5.5.220 typedef TPM2B_PRIVATE_VENDOR_SPECIFIC

```
typedef struct TPM2B_PRIVATE_VENDOR_SPECIFIC TPM2B_PRIVATE_VENDOR_SPECIFIC;
```

4.5.5.221 typedef TPMU_SENSITIVE_COMPOSITE

```
typedef union TPMU_SENSITIVE_COMPOSITE TPMU_SENSITIVE_COMPOSITE;
```

4.5.5.222 typedef TPMT_SENSITIVE

```
typedef struct TPMT_SENSITIVE TPMT_SENSITIVE;
```

4.5.5.223 typedef TPM2B_SENSITIVE

```
typedef struct TPM2B_SENSITIVE TPM2B_SENSITIVE;
```

4.5.5.224 typedef TPMT_PRIVATE

```
typedef struct TPMT_PRIVATE TPMT_PRIVATE;
```

4.5.5.225 typedef TPM2B_PRIVATE

```
typedef struct TPM2B_PRIVATE TPM2B_PRIVATE;
```

4.5.5.226 typedef TPMS_ID_OBJECT

```
typedef struct TPMS_ID_OBJECT TPMS_ID_OBJECT;
```

4.5.5.227 typedef TPM2B_ID_OBJECT

```
typedef struct TPM2B_ID_OBJECT TPM2B_ID_OBJECT;
```

4.5.5.228 typedef TPM_NV_INDEX

```
typedef UINT32 TPM_NV_INDEX;
```

4.5.5.229 typedef TPM_NT

```
typedef enum TPM_NT TPM_NT;
```

4.5.5.230 typedef TPMS_NV_PIN_COUNTER_PARAMETERS

```
typedef struct TPMS_NV_PIN_COUNTER_PARAMETERS TPMS_NV_PIN_COUNTER_PARAMETERS;
```

4.5.5.231 typedef TPMA_NV

```
typedef UINT32 TPMA_NV;
```

4.5.5.232 typedef TPMS_NV_PUBLIC

```
typedef struct TPMS_NV_PUBLIC TPMS_NV_PUBLIC;
```

4.5.5.233 typedef TPM2B_NV_PUBLIC

```
typedef struct TPM2B_NV_PUBLIC TPM2B_NV_PUBLIC;
```

4.5.5.234 typedef TPM2B_CONTEXT_SENSITIVE

```
typedef struct TPM2B_CONTEXT_SENSITIVE TPM2B_CONTEXT_SENSITIVE;
```

4.5.5.235 typedef TPMS_CONTEXT_DATA

```
typedef struct TPMS_CONTEXT_DATA TPMS_CONTEXT_DATA;
```

4.5.5.236 typedef TPM2B_CONTEXT_DATA

```
typedef struct TPM2B_CONTEXT_DATA TPM2B_CONTEXT_DATA;
```

4.5.5.237 typedef TPMS_CONTEXT

```
typedef struct TPMS_CONTEXT TPMS_CONTEXT;
```

4.5.5.238 typedef TPMS_CREATION_DATA

```
typedef struct TPMS_CREATION_DATA TPMS_CREATION_DATA;
```

4.5.5.239 typedef TPM2B_CREATION_DATA

```
typedef struct TPM2B_CREATION_DATA TPM2B_CREATION_DATA;
```

4.5.5.240 typedef TPMS_AUTH_COMMAND

```
typedef struct TPMS_AUTH_COMMAND TPMS_AUTH_COMMAND;
```

4.5.5.241 typedef TPMS_AUTH_RESPONSE

```
typedef struct TPMS_AUTH_RESPONSE TPMS_AUTH_RESPONSE;
```

4.5.5.242 typedef TPM2_AUTH_SESSION

```
typedef struct TPM2_AUTH_SESSION TPM2_AUTH_SESSION;
```

4.5.5.243 typedef TPM2HalIoCb

```
typedef int(* TPM2HalIoCb)(struct TPM2_CTX *, const BYTE *txBuf, BYTE *rxBuf,
 ↪  UINT16 xferSz, void *userCtx);
```

4.5.5.244 typedef TPM2_CTX

```
typedef struct TPM2_CTX TPM2_CTX;
```

4.5.5.245 typedef ChangePPS_In

```
typedef ChangeSeed_In ChangePPS_In;
```

4.5.5.246 typedef ChangeEPS_In

```
typedef ChangeSeed_In ChangeEPS_In;
```

4.5.5.247 typedef TPM_MODE_SET

```
typedef struct TPM_MODE_SET TPM_MODE_SET;
```

4.5.5.248 typedef GetRandom2_In

```
typedef GetRandom_In GetRandom2_In;
```

4.5.5.249 typedef TPMI_GPIO_NAME

```
typedef UINT32 TPMI_GPIO_NAME;
```

4.5.5.250 typedef TPMI_GPIO_MODE

```
typedef UINT32 TPMI_GPIO_MODE;
```

4.5.5.251 typedef TPMS_GPIO_CONFIG

```
typedef struct TPMS_GPIO_CONFIG TPMS_GPIO_CONFIG;
```

4.5.5.252 typedef TPML_GPIO_CONFIG

```
typedef struct TPML_GPIO_CONFIG TPML_GPIO_CONFIG;
```

4.5.6 関数の詳細

function TPM2_Startup

```
WOLFTPM_API TPM_RC TPM2_Startup(
    Startup_In * in
)
```

function TPM2_Shutdown

```
WOLFTPM_API TPM_RC TPM2_Shutdown(
    Shutdown_In * in
)
```

function TPM2_GetCapability

```
WOLFTPM_API TPM_RC TPM2_GetCapability(
    GetCapability_In * in,
    GetCapability_Out * out
)
```

function TPM2_SelfTest

```
WOLFTPM_API TPM_RC TPM2_SelfTest(
    SelfTest_In * in
)
```

function TPM2_IncrementalSelfTest

```
WOLFTPM_API TPM_RC TPM2_IncrementalSelfTest(
    IncrementalSelfTest_In * in,
    IncrementalSelfTest_Out * out
)
```

function TPM2_GetTestResult

```
WOLFTPM_API TPM_RC TPM2_GetTestResult(
    GetTestResult_Out * out
)
```

function TPM2_GetRandom

```
WOLFTPM_API TPM_RC TPM2_GetRandom(
    GetRandom_In * in,
    GetRandom_Out * out
)
```

function TPM2_StirRandom

```
WOLFTPM_API TPM_RC TPM2_StirRandom(
    StirRandom_In * in
)
```

function TPM2_PCR_Read

```
WOLFTPM_API TPM_RC TPM2_PCR_Read(
    PCR_Read_In * in,
    PCR_Read_Out * out
)
```

function TPM2_PCR_Extend

```
WOLFTPM_API TPM_RC TPM2_PCR_Extend(
    PCR_Extend_In * in
)
```

function TPM2_Create

```
WOLFTPM_API TPM_RC TPM2_Create(
    Create_In * in,
    Create_Out * out
)
```

function TPM2_CreateLoaded

```
WOLFTPM_API TPM_RC TPM2_CreateLoaded(
    CreateLoaded_In * in,
    CreateLoaded_Out * out
)
```

function TPM2_CreatePrimary

```
WOLFTPM_API TPM_RC TPM2_CreatePrimary(
    CreatePrimary_In * in,
    CreatePrimary_Out * out
)
```

function TPM2_Load

```
WOLFTPM_API TPM_RC TPM2_Load(
    Load_In * in,
    Load_Out * out
)
```

function TPM2_FlushContext

```
WOLFTPM_API TPM_RC TPM2_FlushContext(
    FlushContext_In * in
)
```

function TPM2_Unseal

```
WOLFTPM_API TPM_RC TPM2_Unseal(
    Unseal_In * in,
    Unseal_Out * out
)
```

function TPM2_StartAuthSession

```
WOLFTPM_API TPM_RC TPM2_StartAuthSession(
    StartAuthSession_In * in,
    StartAuthSession_Out * out
)
```

function TPM2_PolicyRestart

```
WOLFTPM_API TPM_RC TPM2_PolicyRestart(
    PolicyRestart_In * in
)
```

function TPM2_LoadExternal

```
WOLFTPM_API TPM_RC TPM2_LoadExternal(
    LoadExternal_In * in,
    LoadExternal_Out * out
)
```

function TPM2_ReadPublic

```
WOLFTPM_API TPM_RC TPM2_ReadPublic(
    ReadPublic_In * in,
    ReadPublic_Out * out
)
```

function TPM2_ActivateCredential

```
WOLFTPM_API TPM_RC TPM2_ActivateCredential(
    ActivateCredential_In * in,
    ActivateCredential_Out * out
)
```

function TPM2_MakeCredential

```
WOLFTPM_API TPM_RC TPM2_MakeCredential(
    MakeCredential_In * in,
    MakeCredential_Out * out
)
```

function TPM2_ObjectChangeAuth

```
WOLFTPM_API TPM_RC TPM2_ObjectChangeAuth(
    ObjectChangeAuth_In * in,
    ObjectChangeAuth_Out * out
)
```

function TPM2_Duplicate

```
WOLFTPM_API TPM_RC TPM2_Duplicate(
    Duplicate_In * in,
    Duplicate_Out * out
)
```

function TPM2_Rewrap

```
WOLFTPM_API TPM_RC TPM2_Rewrap(
    Rewrap_In * in,
    Rewrap_Out * out
)
```

function TPM2_Import

```
WOLFTPM_API TPM_RC TPM2_Import(
    Import_In * in,
    Import_Out * out
)
```

function TPM2_RSA_Encrypt

```
WOLFTPM_API TPM_RC TPM2_RSA_Encrypt(
    RSA_Encrypt_In * in,
    RSA_Encrypt_Out * out
)
```

function TPM2_RSA_Decrypt

```
WOLFTPM_API TPM_RC TPM2_RSA_Decrypt(
    RSA_Decrypt_In * in,
    RSA_Decrypt_Out * out
)
```

function TPM2_ECDH_KeyGen

```
WOLFTPM_API TPM_RC TPM2_ECDH_KeyGen(
    ECDH_KeyGen_In * in,
    ECDH_KeyGen_Out * out
)
```

function TPM2_ECDH_ZGen

```
WOLFTPM_API TPM_RC TPM2_ECDH_ZGen(
    ECDH_ZGen_In * in,
    ECDH_ZGen_Out * out
)
```

function TPM2_ECC_Parameters

```
WOLFTPM_API TPM_RC TPM2_ECC_Parameters(
    ECC_Parameters_In * in,
    ECC_Parameters_Out * out
)
```

function TPM2_ZGen_2Phase

```
WOLFTPM_API TPM_RC TPM2_ZGen_2Phase(
    ZGen_2Phase_In * in,
    ZGen_2Phase_Out * out
)
```

function TPM2_EncryptDecrypt

```
WOLFTPM_API TPM_RC TPM2_EncryptDecrypt(
    EncryptDecrypt_In * in,
    EncryptDecrypt_Out * out
)
```

function TPM2_EncryptDecrypt2

```
WOLFTPM_API TPM_RC TPM2_EncryptDecrypt2(
    EncryptDecrypt2_In * in,
    EncryptDecrypt2_Out * out
)
```

function TPM2_Hash

```
WOLFTPM_API TPM_RC TPM2_Hash(
    Hash_In * in,
    Hash_Out * out
)
```

function TPM2_HMAC

```
WOLFTPM_API TPM_RC TPM2_HMAC(
    HMAC_In * in,
    HMAC_Out * out
)
```

function TPM2_HMAC_Start

```
WOLFTPM_API TPM_RC TPM2_HMAC_Start(
    HMAC_Start_In * in,
    HMAC_Start_Out * out
)
```

function TPM2_HashSequenceStart

```
WOLFTPM_API TPM_RC TPM2_HashSequenceStart(
    HashSequenceStart_In * in,
    HashSequenceStart_Out * out
)
```

function TPM2_SequenceUpdate

```
WOLFTPM_API TPM_RC TPM2_SequenceUpdate(
    SequenceUpdate_In * in
)
```

function TPM2_SequenceComplete

```
WOLFTPM_API TPM_RC TPM2_SequenceComplete(
    SequenceComplete_In * in,
    SequenceComplete_Out * out
)
```

function TPM2_EventSequenceComplete

```
WOLFTPM_API TPM_RC TPM2_EventSequenceComplete(
    EventSequenceComplete_In * in,
    EventSequenceComplete_Out * out
)
```

function TPM2_Certify

```
WOLFTPM_API TPM_RC TPM2_Certify(
    Certify_In * in,
    Certify_Out * out
)
```

function TPM2_CertifyCreation

```
WOLFTPM_API TPM_RC TPM2_CertifyCreation(
    CertifyCreation_In * in,
    CertifyCreation_Out * out
)
```

function TPM2_Quote

```
WOLFTPM_API TPM_RC TPM2_Quote(
    Quote_In * in,
    Quote_Out * out
)
```

function TPM2_GetSessionAuditDigest

```
WOLFTPM_API TPM_RC TPM2_GetSessionAuditDigest(
    GetSessionAuditDigest_In * in,
    GetSessionAuditDigest_Out * out
)
```

function TPM2_GetCommandAuditDigest

```
WOLFTPM_API TPM_RC TPM2_GetCommandAuditDigest(
    GetCommandAuditDigest_In * in,
    GetCommandAuditDigest_Out * out
)
```

function TPM2_GetTime

```
WOLFTPM_API TPM_RC TPM2_GetTime(
    GetTime_In * in,
    GetTime_Out * out
)
```

function TPM2_Commit

```
WOLFTPM_API TPM_RC TPM2_Commit(
    Commit_In * in,
    Commit_Out * out
)
```

function TPM2_EC_Ephemeral

```
WOLFTPM_API TPM_RC TPM2_EC_Ephemeral(
    EC_Ephemeral_In * in,
    EC_Ephemeral_Out * out
)
```

function TPM2_VerifySignature

```
WOLFTPM_API TPM_RC TPM2_VerifySignature(
    VerifySignature_In * in,
    VerifySignature_Out * out
)
```

function TPM2_Sign

```
WOLFTPM_API TPM_RC TPM2_Sign(
    Sign_In * in,
    Sign_Out * out
)
```

function TPM2_SetCommandCodeAuditStatus

```
WOLFTPM_API TPM_RC TPM2_SetCommandCodeAuditStatus(
    SetCommandCodeAuditStatus_In * in
)
```

function TPM2_PCR_Event

```
WOLFTPM_API TPM_RC TPM2_PCR_Event(
    PCR_Event_In * in,
    PCR_Event_Out * out
)
```

function TPM2_PCR_Allocate

```
WOLFTPM_API TPM_RC TPM2_PCR_Allocate(
    PCR_Allocate_In * in,
    PCR_Allocate_Out * out
)
```

function TPM2_PCR_SetAuthPolicy

```
WOLFTPM_API TPM_RC TPM2_PCR_SetAuthPolicy(
    PCR_SetAuthPolicy_In * in
)
```

function TPM2_PCR_SetAuthValue

```
WOLFTPM_API TPM_RC TPM2_PCR_SetAuthValue(
    PCR_SetAuthValue_In * in
)
```

function TPM2_PCR_Reset

```
WOLFTPM_API TPM_RC TPM2_PCR_Reset(
    PCR_Reset_In * in
)
```

function TPM2_PolicySigned

```
WOLFTPM_API TPM_RC TPM2_PolicySigned(
    PolicySigned_In * in,
    PolicySigned_Out * out
)
```

function TPM2_PolicySecret

```
WOLFTPM_API TPM_RC TPM2_PolicySecret(
    PolicySecret_In * in,
    PolicySecret_Out * out
)
```

function TPM2_PolicyTicket

```
WOLFTPM_API TPM_RC TPM2_PolicyTicket(
    PolicyTicket_In * in
)
```

function TPM2_PolicyOR

```
WOLFTPM_API TPM_RC TPM2_PolicyOR(
    PolicyOR_In * in
)
```

function TPM2_PolicyPCR

```
WOLFTPM_API TPM_RC TPM2_PolicyPCR(
    PolicyPCR_In * in
)
```

function TPM2_PolicyLocality

```
WOLFTPM_API TPM_RC TPM2_PolicyLocality(
    PolicyLocality_In * in
)
```

function TPM2_PolicyNV

```
WOLFTPM_API TPM_RC TPM2_PolicyNV(
    PolicyNV_In * in
)
```

function TPM2_PolicyCounterTimer

```
WOLFTPM_API TPM_RC TPM2_PolicyCounterTimer(
    PolicyCounterTimer_In * in
)
```

function TPM2_PolicyCommandCode

```
WOLFTPM_API TPM_RC TPM2_PolicyCommandCode(
    PolicyCommandCode_In * in
)
```

function TPM2_PolicyPhysicalPresence

```
WOLFTPM_API TPM_RC TPM2_PolicyPhysicalPresence(
    PolicyPhysicalPresence_In * in
)
```

function TPM2_PolicyCpHash

```
WOLFTPM_API TPM_RC TPM2_PolicyCpHash(
    PolicyCpHash_In * in
)
```

function TPM2_PolicyNameHash

```
WOLFTPM_API TPM_RC TPM2_PolicyNameHash(
    PolicyNameHash_In * in
)
```

function TPM2_PolicyDuplicationSelect

```
WOLFTPM_API TPM_RC TPM2_PolicyDuplicationSelect(
    PolicyDuplicationSelect_In * in
)
```

function TPM2_PolicyAuthorize

```
WOLFTPM_API TPM_RC TPM2_PolicyAuthorize(
    PolicyAuthorize_In * in
)
```

function TPM2_PolicyAuthValue

```
WOLFTPM_API TPM_RC TPM2_PolicyAuthValue(
    PolicyAuthValue_In * in
)
```

function TPM2_PolicyPassword

```
WOLFTPM_API TPM_RC TPM2_PolicyPassword(
    PolicyPassword_In * in
)
```

function TPM2_PolicyGetDigest

```
WOLFTPM_API TPM_RC TPM2_PolicyGetDigest(
    PolicyGetDigest_In * in,
    PolicyGetDigest_Out * out
)
```

function TPM2_PolicyNvWritten

```
WOLFTPM_API TPM_RC TPM2_PolicyNvWritten(
    PolicyNvWritten_In * in
)
```

function TPM2_PolicyTemplate

```
WOLFTPM_API TPM_RC TPM2_PolicyTemplate(
    PolicyTemplate_In * in
)
```

function TPM2_PolicyAuthorizeNV

```
WOLFTPM_API TPM_RC TPM2_PolicyAuthorizeNV(
    PolicyAuthorizeNV_In * in
)
```

function _TPM_Hash_Start

```
WOLFTPM_API void _TPM_Hash_Start(
    void
)
```

function _TPM_Hash_Data

```
WOLFTPM_API void _TPM_Hash_Data(
    UINT32 dataSize,
    BYTE * data
)
```

function _TPM_Hash_End

```
WOLFTPM_API void _TPM_Hash_End(
    void
)
```

function TPM2_HierarchyControl

```
WOLFTPM_API TPM_RC TPM2_HierarchyControl(
    HierarchyControl_In * in
)
```

function TPM2_SetPrimaryPolicy

```
WOLFTPM_API TPM_RC TPM2_SetPrimaryPolicy(
    SetPrimaryPolicy_In * in
)
```

function TPM2_ChangePPS

```
WOLFTPM_API TPM_RC TPM2_ChangePPS(
    ChangePPS_In * in
)
```

function TPM2_ChangeEPS

```
WOLFTPM_API TPM_RC TPM2_ChangeEPS(
    ChangeEPS_In * in
)
```

function TPM2_Clear

```
WOLFTPM_API TPM_RC TPM2_Clear(
    Clear_In * in
)
```

function TPM2_ClearControl

```
WOLFTPM_API TPM_RC TPM2_ClearControl(
    ClearControl_In * in
)
```

function TPM2_HierarchyChangeAuth

```
WOLFTPM_API TPM_RC TPM2_HierarchyChangeAuth(
    HierarchyChangeAuth_In * in
)
```

function TPM2_DictionaryAttackLockReset

```
WOLFTPM_API TPM_RC TPM2_DictionaryAttackLockReset(
    DictionaryAttackLockReset_In * in
)
```

function TPM2_DictionaryAttackParameters

```
WOLFTPM_API TPM_RC TPM2_DictionaryAttackParameters(
    DictionaryAttackParameters_In * in
)
```

function TPM2_PP_Commands

```
WOLFTPM_API TPM_RC TPM2_PP_Commands(
    PP_Commands_In * in
)
```

function TPM2_SetAlgorithmSet

```
WOLFTPM_API TPM_RC TPM2_SetAlgorithmSet(
    SetAlgorithmSet_In * in
)
```

function TPM2_FieldUpgradeStart

```
WOLFTPM_API TPM_RC TPM2_FieldUpgradeStart(
    FieldUpgradeStart_In * in
)
```

function TPM2_FieldUpgradeData

```
WOLFTPM_API TPM_RC TPM2_FieldUpgradeData(
    FieldUpgradeData_In * in,
    FieldUpgradeData_Out * out
)
```

function TPM2_FirmwareRead

```
WOLFTPM_API TPM_RC TPM2_FirmwareRead(
    FirmwareRead_In * in,
    FirmwareRead_Out * out
)
```

function TPM2_ContextSave

```
WOLFTPM_API TPM_RC TPM2_ContextSave(
    ContextSave_In * in,
    ContextSave_Out * out
)
```

function TPM2_ContextLoad

```
WOLFTPM_API TPM_RC TPM2_ContextLoad(
    ContextLoad_In * in,
    ContextLoad_Out * out
)
```

function TPM2_EvictControl

```
WOLFTPM_API TPM_RC TPM2_EvictControl(
    EvictControl_In * in
)
```

function TPM2_ReadClock

```
WOLFTPM_API TPM_RC TPM2_ReadClock(
    ReadClock_Out * out
)
```

function TPM2_ClockSet

```
WOLFTPM_API TPM_RC TPM2_ClockSet(
    ClockSet_In * in
)
```

function TPM2_ClockRateAdjust

```
WOLFTPM_API TPM_RC TPM2_ClockRateAdjust(
    ClockRateAdjust_In * in
)
```

function TPM2_TestParms

```
WOLFTPM_API TPM_RC TPM2_TestParms(
    TestParms_In * in
)
```

function TPM2_NV_DefineSpace

```
WOLFTPM_API TPM_RC TPM2_NV_DefineSpace(
    NV_DefineSpace_In * in
)
```

function TPM2_NV_UndefineSpace

```
WOLFTPM_API TPM_RC TPM2_NV_UndefineSpace(
    NV_UndefineSpace_In * in
)
```

function TPM2_NV_UndefineSpaceSpecial

```
WOLFTPM_API TPM_RC TPM2_NV_UndefineSpaceSpecial(
    NV_UndefineSpaceSpecial_In * in
)
```

function TPM2_NV_ReadPublic

```
WOLFTPM_API TPM_RC TPM2_NV_ReadPublic(
    NV_ReadPublic_In * in,
    NV_ReadPublic_Out * out
)
```

function TPM2_NV_Write

```
WOLFTPM_API TPM_RC TPM2_NV_Write(
    NV_Write_In * in
)
```

function TPM2_NV_Increment

```
WOLFTPM_API TPM_RC TPM2_NV_Increment(
    NV_Increment_In * in
)
```

function TPM2_NV_Extend

```
WOLFTPM_API TPM_RC TPM2_NV_Extend(
    NV_Extend_In * in
)
```

function TPM2_NV_SetBits

```
WOLFTPM_API TPM_RC TPM2_NV_SetBits(
    NV_SetBits_In * in
)
```

function TPM2_NV_WriteLock

```
WOLFTPM_API TPM_RC TPM2_NV_WriteLock(
    NV_WriteLock_In * in
)
```

function TPM2_NV_GlobalWriteLock

```
WOLFTPM_API TPM_RC TPM2_NV_GlobalWriteLock(
    NV_GlobalWriteLock_In * in
)
```

function TPM2_NV_Read

```
WOLFTPM_API TPM_RC TPM2_NV_Read(
    NV_Read_In * in,
```

```
    NV_Read_Out * out
)
```

function TPM2_NV_ReadLock

```
WOLFTPM_API TPM_RC TPM2_NV_ReadLock(
    NV_ReadLock_In * in
)
```

function TPM2_NV_ChangeAuth

```
WOLFTPM_API TPM_RC TPM2_NV_ChangeAuth(
    NV_ChangeAuth_In * in
)
```

function TPM2_NV_Certify

```
WOLFTPM_API TPM_RC TPM2_NV_Certify(
    NV_Certify_In * in,
    NV_Certify_Out * out
)
```

function TPM2_SetCommandSet

```
WOLFTPM_API int TPM2_SetCommandSet(
    SetCommandSet_In * in
)
```

function TPM2_SetMode

```
WOLFTPM_API int TPM2_SetMode(
    SetMode_In * in
)
```

function TPM2_GetRandom2

```
WOLFTPM_API TPM_RC TPM2_GetRandom2(
    GetRandom2_In * in,
    GetRandom2_Out * out
)
```

function TPM2_GPIO_Config

```
WOLFTPM_API int TPM2_GPIO_Config(
    GpioConfig_In * in
)
```

function TPM2_NTC2_PreConfig

```
WOLFTPM_API int TPM2_NTC2_PreConfig(
    NTC2_PreConfig_In * in
)
```

function TPM2_NTC2_GetConfig

```
WOLFTPM_API int TPM2_NTC2_GetConfig(
    NTC2_GetConfig_Out * out
)
```

function TPM2_Init

```
WOLFTPM_API TPM_RC TPM2_Init(
    TPM2_CTX * ctx,
    TPM2HalIoCb ioCb,
```

```
    void * userCtx
)
```

HAL IO コールバックとユーザー指定のコンテキストで TPM を初期化します。–enable-devtpm または –enable-swtpm 構成で wolfTPM を使用する場合、ioCb と userCtx は使用されません。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター
- **ioCb** TPM2HalIoCb (HAL IO) コールバック関数へのポインター
- **userCtx** ctx 構造体のメンバーとして格納されるユーザーのコンテキストへのポインター

参照:

- TPM2_Startup
- TPM2_GetRCString
- TPM2_Init_minimal
- TPM2_Init_ex
- wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: s 成功
- TPM_RC_FAILURE: 一般的なエラー (おそらく IO)
- BAD_FUNC_ARG 指定された引数を確認してください

ノート: * TPM2_Init_minimal() ioCb と userCtx の両方を NULL に設定します。その他のモードでは、TIS を使用するために ioCb を設定する必要があります。ベアメタルおよび RTOS アプリケーションの ioCB の例は、hal/tpm_io.c で提供されています。

使用例

```
int rc;
TPM2_CTX tpm2Ctx;

rc = TPM2_Init(&tpm2Ctx, TPM2_IoCb, userCtx);
if (rc != TPM_RC_SUCCESS) {
    // TPM2_Init failed
}
```

function TPM2_Init_ex

```
WOLFTPM_API TPM_RC TPM2_Init_ex(
    TPM2_CTX * ctx,
    TPM2HalIoCb ioCb,
    void * userCtx,
    int timeoutTries
)
```

timeoutTries、HAL IO コールバック、およびユーザー指定のコンテキストで TPM を初期化します。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター
- **ioCb** TPM2HalIoCb (HAL IO) コールバック関数へのポインター
- **userCtx** ctx 構造体のメンバーとして格納されるユーザーのコンテキストへのポインター
- **timeoutTries** TPM2 の起動が完了したことを確認するための試行回数を指定します

参照:

- TPM2_GetRCString
- TPM2_Init_minimal
- TPM2_Init

- wolfTPM2_Init_ex

戻り値:

- TPM_RC_SUCCESS: s 成功
- TPM_RC_FAILURE: 一般的なエラー (おそらく IO)
- BAD_FUNC_ARG 指定された引数を確認してください

ノート: * TPM2_Init_ex を直接使用する代わりに、TPM2_Init を使用することをお勧めします。

function TPM2_Init_minimal

```
WOLFTPM_API TPM_RC TPM2_Init_minimal(
    TPM2_CTX * ctx
)
```

TPM を初期化し、使用される wolfTPM2 コンテキストを設定します。この関数は通常、Windows などのリッチ オペレーティング システムで使用されます。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター

参照:

- TPM2_GetRCString
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: s 成功
- TPM_RC_FAILURE: 一般的なエラー (おそらく IO)
- BAD_FUNC_ARG: 指定された引数を確認してください

ノート: * TPM2_Init_minimal を直接使用する代わりに、TPM2_Init を使用することをお勧めします。

function TPM2_Cleanup

```
WOLFTPM_API TPM_RC TPM2_Cleanup(
    TPM2_CTX * ctx
)
```

TPM と wolfcrypt を初期化解除します (初期化されている場合)

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター

参照:

- TPM2_GetRCString
- TPM2_Init
- wolfTPM2_Cleanup

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: wolfTPM2 コンテキストでロックを取得できませんでした
- BAD_FUNC_ARG: TPM2 デバイス構造は NULL ポインターです

使用例

```
int rc;
TPM2_CTX tpm2Ctx;

rc = TPM2_Cleanup(&tpm2Ctx->dev);
```

```
if (rc != TPM_RC_SUCCESS) {
    // TPM2_Cleanup failed
}
```

function TPM2_ChipStartup

```
WOLFTPM_API TPM_RC TPM2_ChipStartup(
    TPM2_CTX * ctx,
    int timeoutTries
)
```

TPM2 の起動が完了していることを確認し、TPM デバイス情報を抽出します。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター
- **timeoutTries** TPM2 の起動が完了したかどうかを確認する試行回数を指定します

参照:

- TPM2_GetRCString
- TPM2_TIS_StartupWait
- TPM2_TIS_RequestLocality
- TPM2_TIS_GetInfo
- TPM2_Init_ex

戻り値:

- TPM_RC_SUCCESS: s 成功
- TPM_RC_FAILURE: 一般的なエラー (おそらく IO)
- BAD_FUNC_ARG: 指定された引数を確認してください
- TPM_RC_TIMEOUT: タイムアウトが発生しました

ノート: This function is used in TPM2_Init_ex

function TPM2_SetHalIoCb

```
WOLFTPM_API TPM_RC TPM2_SetHalIoCb(
    TPM2_CTX * ctx,
    TPM2HalIoCb ioCb,
    void * userCtx
)
```

TPM 通信に必要なユーザーのコンテキストと IO コールバックを設定します。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター
- **ioCb** TPM2HalIoCb (HAL IO) コールバック関数へのポインター
- **userCtx** ctx 構造体のメンバーとして格納されるユーザーのコンテキストへのポインター

参照:

- TPM2_GetRCString
- TPM2_Init
- wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: wolfTPM2 コンテキストでロックを取得できませんでした
- BAD_FUNC_ARG: TPM2 デバイス構造は NULL ポインターです

ノート:

- コールバックが TPM に使用されないため、devtpm または swtpm でビルドされた場合、SetHalIoCb は失敗します。他の構成ビルドでは、ioCb を NULL 以外の関数ポインターに設定する必要があり、userCtx はオプションです。通常、TPM2_Init または wolfTPM2_Init を使用して HAL IO を設定します。

function TPM2_SetSessionAuth

```
WOLFTPM_API TPM_RC TPM2_SetSessionAuth(
    TPM2_AUTH_SESSION * session
)
```

TPM 承認を保持する構造体を設定します。

パラメータ:

- **session** タイプ TPM2_AUTH_SESSION の配列へのポインター

参照:

- TPM2_GetRCString
- TPM2_Init
- wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: s 成功
- TPM_RC_FAILURE: wolfTPM2 コンテキストでロックを取得できませんでした
- BAD_FUNC_ARG: TPM2 コンテキスト構造は NULL ポインターです

TPM2_Init 関数と wolfTPM2_Init もこの初期化を実行するため、めったに使用されません。TPM 2.0 コマンドは最大 3 つの認証スロットを持つことができるため、サイズ MAX_SESSION_NUM の配列を TPM2_SetSessionAuth に提供することをお勧めします (以下の例を参照)。

使用例

```
int rc;
TPM2_AUTH_SESSION session[MAX_SESSION_NUM];

XMEMSET(session, 0, sizeof(session));
session[0].sessionHandle = TPM_RS_PW;

rc = TPM2_SetSessionAuth(session);
if (rc != TPM_RC_SUCCESS) {
    // TPM2_SetSessionAuth failed
}
```

function TPM2_GetSessionAuthCount

```
WOLFTPM_API int TPM2_GetSessionAuthCount(
    TPM2_CTX * ctx
)
```

現在設定されている TPM 承認の数を確認します。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター

参照:

- TPM2_CTX
- TPM2_AUTH_SESSION

戻り値:

- アクティブな TPM 認証の数 (1 〜3)
- BAD_FUNC_ARG: NULL ポインタに指定された引数を確認してください

使用例

```
int authCount;
TPM2_CTX tpm2Ctx;

authCount = TPM2_GetSessionAuthCount(tpm2ctx);
if (authCount == BAD_FUNC_ARG) {
    // TPM2_GetSessionAuthCount failed
}
```

function TPM2_SetActiveCtx

```
WOLFTPM_API void TPM2_SetActiveCtx(
    TPM2_CTX * ctx
)
```

使用する新しい TPM2 コンテキストを設定します。

パラメータ:

- **ctx** TPM2_CTX 構造体へのポインター

参照:

- TPM2_CTX
- TPM2_AUTH_SESSION

使用例

```
TPM2_CTX tpm2Ctx;

TPM2_SetActiveCtx(tpm2ctx);
```

function TPM2_GetActiveCtx

```
WOLFTPM_API TPM2_CTX * TPM2_GetActiveCtx(
    void
)
```

使用中の TPM2 コンテキストへのポインターを提供します。

参照:

- TPM2_CTX
- TPM2_AUTH_SESSION

戻り値: ctx TPM2_CTX 構造体へのポインター

使用例

```
TPM2_CTX *tpm2Ctx;

tpm2Ctx = TPM2_GetActiveCtx();
```

function TPM2_GetHashDigestSize

```
WOLFTPM_API int TPM2_GetHashDigestSize(
    TPMI_ALG_HASH hashAlg
)
```

TPM 2.0 ハッシュ ダイジェストのサイズをバイト単位で決定します。

パラメータ:

- **hashAlg** 有効な TPM 2.0 ハッシュ タイプ

戻り値:

- the size of a TPM 2.0 hash digest as number of bytes
- ハッシュ タイプが無効な場合は 0

使用例

```
int digestSize = 0;
TPMI_ALG_HASH hashAlg = TPM_ALG_SHA256;

digestSize = TPM2_GetHashDigestSize(hashAlg);
if (digestSize > 0) {
    //digestSize contains a valid value
}
```

function TPM2_GetHashType

```
WOLFTPM_API int TPM2_GetHashType(
    TPMI_ALG_HASH hashAlg
)
```

TPM2 ハッシュ タイプを対応する wolfcrypt ハッシュ タイプに変換します。

パラメータ:

- **hashAlg** 有効な TPM 2.0 ハッシュ タイプ

戻り値:

- wolfcrypt で使用するハッシュ タイプを指定する値
- ハッシュ タイプが無効な場合は 0

使用例

```
int wc_hashType;
TPMI_ALG_HASH hashAlg = TPM_ALG_SHA256;

wc_hashType = TPM2_GetHashDigestSize(hashAlg);
if (wc_hashType > 0) {
    //wc_hashType contains a valid wolfcrypt hash type
}
```

function TPM2_GetNonce

```
WOLFTPM_API int TPM2_GetNonce(
    byte * nonceBuf,
    int nonceSz
)
```

乱数の新しいナンスを生成します。

パラメータ:

- **nonceBuf** BYTE バッファへのポインター
- **nonceSz** nonce のバイト単位のサイズ

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的な障害 (TPM IO の問題または wolfcrypt 構成)
- BAD_FUNC_ARG: 指定された引数を確認してください

ノート:

- WOLFTPM2_USE_HW_RNG が定義されている場合、TPM 乱数ジェネレーターを使用できます

使用例

```
int rc, nonceSize = 32;
BYTE freshNonce[32];

rc = TPM2_GetNonce(&freshNonce, nonceSize);
if (rc != TPM_RC_SUCCESS) {
    //TPM2_GetNonce failed
}
```

function TPM2_SetupPCRSel

```
WOLFTPM_API void TPM2_SetupPCRSel(
    TPML_PCR_SELECTION * pcr,
    TPM_ALG_ID alg,
    int pcrIndex
)
```

正しい PCR 選択を準備するためのヘルパー関数 たとえば、TPM2_Quote を作成する準備をする場合。

パラメータ:

- **pcr** TPML_PCR_SELECTION 型の構造体へのポインター
- **alg** 使用するハッシュ アルゴリズムのタイプを指定するタイプ TPM_ALG_ID の値
- **pcrIndex** 使用する PCR レジスタを指定する 0 ～23 の値

参照:

- TPM2_PCR_Read
- TPM2_PCR_Extend
- TPM2_PCR_Reset
- TPM2_Quote

使用例

```
int pcrIndex = 16; // This is a PCR register for DEBUG & testing purposes
PCR_Read_In pcrRead;

TPM2_SetupPCRSel(&pcrRead.pcrSelectionIn, TPM_ALG_SHA256, pcrIndex);
```

function TPM2_GetRCString

```
const WOLFTPM_API char * TPM2_GetRCString(
    int rc
)
```

TPM 2.0 リターン コードの人間が判読できる文字列を取得します。

パラメータ:

- **rc** TPM リターン コードを表す整数値

戻り値:

- 文字列定数へのポインター

使用例

```
int rc;

rc = wolfTPM2_Init(&dev, TPM2_IoCb, userCtx);
if (rc != TPM_RC_SUCCESS) {
```

```
    printf("wolfTPM2_Init failed 0x%x: %s\n", rc, TPM2_GetRCString(rc));
    return rc;
}
```

function TPM2_GetAlgName

```
const WOLFTPM_API char * TPM2_GetAlgName(
    TPM_ALG_ID alg
)
```

任意の TPM 2.0 アルゴリズムについて、人間が判読できる文字列を取得します。

パラメータ:

- **alg** 有効な TPM 2.0 アルゴリズムを指定する TPM_ALG_ID 型の値

戻り値:

- 文字列定数へのポインター

使用例

```
int paramEncAlg = TPM_ALG_CFB;

printf("\tUse Parameter Encryption: %s\n", TPM2_GetAlgName(paramEncAlg));
```

function TPM2_GetCurveSize

```
WOLFTPM_API int TPM2_GetCurveSize(
    TPM_ECC_CURVE curveID
)
```

TPM ECC 曲線のサイズをバイト単位で決定します。

パラメータ:

- **curveID** TPM_ECC_CURVE 型の値

戻り値:

- 無効な曲線タイプの場合は 0
- バイト数を表す整数値

使用例

```
int bytes;
TPM_ECC_CURVE curve = TPM_ECC_NIST_P256;

bytes = TPM2_GetCurveSize(curve);
if (bytes == 0) {
    //TPM2_GetCurveSize failed
}
```

function TPM2_GetTpmCurve

```
WOLFTPM_API int TPM2_GetTpmCurve(
    int curveID
)
```

wolfcrypt 曲線タイプを対応する TPM 曲線タイプに変換します。

パラメータ:

- **curveID** BYTE バッファへのポインター

参照:

- TPM2_GetWolfCurve

戻り値:

- wolfcrypt 曲線タイプを表す整数値
- 無効な曲線タイプの場合は ECC_CURVE_OID_E

使用例

```
int tpmCurve;
int wc_curve = ECC_SECP256R1;

tpmCurve = TPM2_GetTpmCurve(curve);
\in this case tpmCurve will be TPM_ECC_NIST_P256
if (tpmCurve = ECC_CURVE_OID_E) {
    //TPM2_GetTpmCurve failed
}
```

function TPM2_GetWolfCurve

```
WOLFTPM_API int TPM2_GetWolfCurve(
    int curve_id
)
```

TPM 曲線タイプを対応する wolfcrypt 曲線タイプに変換します。

パラメータ:

- **curve_id** BYTE バッファへのポインター

参照:

- TPM2_GetTpmCurve

戻り値:

- PM 曲線タイプを表す整数値
- 無効な曲線タイプの場合は -1 または ECC_CURVE_OID_E

使用例

```
int tpmCurve = TPM_ECC_NIST_P256;
int wc_curve;

wc_curve = TPM2_GetWolfCurve(tpmCurve);
\in this case tpmCurve will be ECC_SECP256R1
if (wc_curve = ECC_CURVE_OID_E || wc_curve == -1) {
    //TPM2_GetWolfCurve failed
}
```

function TPM2_ParseAttest

```
WOLFTPM_API int TPM2_ParseAttest(
    const TPM2B_ATTEST * in,
    TPMS_ATTEST * out
)
```

TPM2B_ATTEST 構造を解析します。

パラメータ:

- **in** TPM2B_ATTEST 型の構造体へのポインター
- **out** TPMS_ATTEST 型の構造体へのポインター

戻り値:

- TPM_RC_SUCCESS: s 成功
- BAD_FUNC_ARG: 指定された引数を確認してください

ノート:

- これは、ヘルパー関数 TPM2_Packet_ParseAttest のパブリック API です。

使用例

```
TPM2B_ATTEST in; //for example, as part of a TPM2_Quote
TPMS_ATTEST out

rc = TPM2_GetNonce(&in, &out);
if (rc != TPM_RC_SUCCESS) {
    //TPM2_ParseAttest failed
}
```

function TPM2_HashNvPublic

```
WOLFTPM_API int TPM2_HashNvPublic(
    TPMS_NV_PUBLIC * nvPublic,
    byte * buffer,
    UINT16 * size
)
```

nvPublic 構造に基づいて新しい NV インデックス名を計算します。

パラメータ:

- **nvPublic**
- **buffer** TPMS_ATTEST 型の構造体へのポインター
- **size** nvIndex のサイズを格納する UINT16 型の変数へのポインター

戻り値:

- TPM_RC_SUCCESS: 成功
- negative integer value in case of an error
- BAD_FUNC_ARG: 指定された引数を確認してください
- NOT_COMPILED_IN: check if wolfcrypt is enabled

使用例

```
TPMS_NV_PUBLIC nvPublic;
BYTE buffer[TPM_MAX_DIGEST_SIZE];
UINT16 size;

rc = TPM2_HashNvPublic(&nvPublic, &buffer, &size);
if (rc != TPM_RC_SUCCESS) {
    //TPM2_HashNvPublic failed
}
```

function TPM2_AppendPublic

```
WOLFTPM_API int TPM2_AppendPublic(
    byte * buf,
    word32 size,
    int * sizeUsed,
    TPM2B_PUBLIC * pub
)
```

Populates TPM2B_PUBLIC structure based on a user provided buffer.

パラメータ:

- **buf** ユーザーバッファへのポインター
- **size** ユーザーバッファのサイズを指定する word32 型の整数値
- **sizeUsed** 整数変数へのポインタ、pub->buffer の使用サイズを格納
- **pub** TPM2B_PUBLIC 型の空の構造体へのポインター

参照: * TPM2_ParsePublic

戻り値:

- TPM_RC_SUCCESS: s 成功
- TPM_RC_FAILURE: 不十分なバッファ サイズ
- BAD_FUNC_ARG: 指定された引数を確認してください

ノート:

** ヘルパー関数 TPM2_Packet_AppendPublic のパブリック API

使用例

```
TPM2B_PUBLIC pub; //empty
int sizeUsed, rc;
BYTE buffer[sizeof(TPM2B_PUBLIC)];
word32 size = sizeof(buffer);

rc = TPM2_AppendPublic(&buffer, size, &sizeUsed, &pub);
if (rc != TPM_RC_SUCCESS) {
    //TPM2_AppendPublic failed
}
```

function TPM2_ParsePublic

```
WOLFTPM_API int TPM2_ParsePublic(
    TPM2B_PUBLIC * pub,
    byte * buf,
    word32 size,
    int * sizeUsed
)
```

TPM2B_PUBLIC 構造体を解析し、ユーザー指定のバッファーに格納します。

パラメータ:

- **pub** TPM2B_PUBLIC 型のデータが取り込まれた構造体へのポインター
- **buf** 空のユーザー バッファへのポインタ
- **size** ユーザー バッファの使用可能なサイズを指定する word32 型の整数値
- **sizeUsed** 整数変数へのポインター。ユーザー バッファーの使用サイズを格納します。

参照: TPM2_AppendPublic

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 不十分なバッファ サイズ
- BAD_FUNC_ARG: 指定された引数を確認してください

ノート:

- ヘルパー関数 TPM2_Packet_ParsePublic の公開 API

使用例

```
TPM2B_PUBLIC pub; //populated
int sizeUsed, rc;
BYTE buffer[sizeof(TPM2B_PUBLIC)];
word32 size = sizeof(buffer);

rc = TPM2_ParsePublic(&pub, buffer, size, &sizeUsed);
if (rc != TPM_RC_SUCCESS) {
    //TPM2_ParsePublic failed
}
```

function TPM2_GetName

```
WOLFTPM_LOCAL int TPM2_GetName(
    TPM2_CTX * ctx,
    UINT32 handleValue,
    int handleCnt,
    int idx,
    TPM2B_NAME * name
)
```

Provides the Name of a TPM object.

パラメータ:

- **ctx** TPM2 コンテキストへのポインター
- **handleValue** 有効な TPM ハンドルを指定する UINT32 型の値
- **handleCnt** 現在の TPM コマンド/セッションで使用されているハンドルの総数
- **idx** 有効な TPM 認証セッションを指定する 1 ～3 のインデックス値
- **name** TPM2B_NAME 型の空の構造体へのポインター

戻り値:

- TPM_RC_SUCCESS: s 成功
- BAD_FUNC_ARG: 指定された引数を確認してください

ノート:

- オブジェクトは、TPM ハンドルとセッション インデックスによって参照されます

使用例

```
int rc;
UINT32 handleValue = TRANSIENT_FIRST;
handleCount = 1;
sessionIdx = 0;
TPM2B_NAME name;

rc = TPM2_GetName(ctx, handleValue, handleCount, sessionIdx, &name);
if (rc != TPM_RC_SUCCESS) {
    //TPM2_GetName failed
}
```

function TPM2_GetWolfRng

```
WOLFTPM_API int TPM2_GetWolfRng(
    WC_RNG ** rng
)
```

function TPM2_GetVendorID

```
WOLFTPM_API UINT16 TPM2_GetVendorID(
    void
)
```

アクティブな TPM2 コンテキストの vendorID を提供します。

参照:

- TPM2_GetCapabilities
- TPM2_GetTpmDevId
- TPM2_Init

戻り値:

- ベンダー ID を指定する UINT16 型の整数値
- TPM2 コンテキストが無効または NULL の場合は 0

ノート:

- TPM 初期化中に TPM デバイス情報が正しく読み取られるかどうかに依存します

使用例

```
TPM2_CTX *tpm2Ctx;

tpm2Ctx = TPM2_GetActiveCtx();
```

function TPM2_PrintBin

```
WOLFTPM_API void TPM2_PrintBin(
    const byte * buffer,
    word32 length
)
```

フォーマットされた方法でバイナリ バッファを出力するヘルパー関数。

パラメータ:

- **buffer** BYTE 型のバッファへのポインター
- **length** バッファのサイズを含む word32 型の整数値

参照:

- TPM2_PrintAuth
- TPM2_PrintPublicArea

ノート: DEBUG_WOLFTPM を定義する必要があります

使用例

```
BYTE buffer[] = {0x01,0x02,0x03,0x04};
length = sizeof(buffer);

TPM2_PrintBin(&buffer, length);
```

function TPM2_PrintAuth

```
WOLFTPM_API void TPM2_PrintAuth(
    const TPMS_AUTH_COMMAND * authCmd
)
```

TPMS_AUTH_COMMAND 型の構造体を人間が読める形式で出力するヘルパー関数。

パラメータ:

- **authCmd** TPMS_AUTH_COMMAND 型のデータが取り込まれた構造体へのポインター

参照:

- TPM2_PrintBin
- TPM2_PrintPublicArea

ノート:

- DEBUG_WOLFTPM を定義する必要があります

使用例

```
TPMS_AUTH_COMMAND authCmd; //for example, part of a TPM Authorization session

TPM2_PrintAuthCmd(&authCmd);
```

function TPM2_PrintPublicArea

```
WOLFTPM_API void TPM2_PrintPublicArea(
    const TPM2B_PUBLIC * pub
)
```

TPM2B_PUBLIC 型の構造を人間が読める形式で出力するヘルパー関数。

パラメータ:

- **pub** TPM2B_PUBLIC 型のデータが取り込まれた構造体へのポインター

参照:

- TPM2_PrintBin
- TPM2_PrintAuth
- TPM2_Create
- TPM2_ReadPublic

ノート:

- DEBUG_WOLFTPM を定義する必要があります

使用例

```
TPM2B_PUBLIC pub; //for example, part of the output of a successful TPM2_Create

TPM2_PrintPublicArea(&pub);
```

4.5.7 属性の詳細

4.5.7.1 variable C

```
C {
#endif
```

```
typedef UINT32 TPM_ALGORITHM_ID;
```

4.5.7.2 variable TPM_20_EK_AUTH_POLICY

```
static const BYTE TPM_20_EK_AUTH_POLICY = {
    0x83, 0x71, 0x97, 0x67, 0x44, 0x84, 0xb3, 0xf8, 0x1a, 0x90, 0xcc,
    0x8d, 0x46, 0xa5, 0xd7, 0x24, 0xfd, 0x52, 0xd7, 0x6e, 0x06, 0x52,
    0x0b, 0x64, 0xf2, 0xa1, 0xda, 0x1b, 0x33, 0x14, 0x69, 0xaa,
};
```

4.5.8 ソースコード

```
/* tpm2.h
 *
 * Copyright (C) 2006-2022 wolfSSL Inc.
 *
 * This file is part of wolfTPM.
 *
 * wolfTPM is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * wolfTPM is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
 */

#ifndef __TPM2_H__
#define __TPM2_H__

#include <wolftpm/tpm2_types.h>

#ifdef __cplusplus
    extern "C" {
#endif


/* ---------------------------------------------------------------------------
 ↪  */
/* TYPES */
/* ---------------------------------------------------------------------------
 ↪  */

typedef UINT32 TPM_ALGORITHM_ID;
typedef UINT32 TPM_MODIFIER_INDICATOR;
typedef UINT32 TPM_AUTHORIZATION_SIZE;
typedef UINT32 TPM_PARAMETER_SIZE;
typedef UINT16 TPM_KEY_SIZE;
typedef UINT16 TPM_KEY_BITS;
typedef UINT32 TPM_GENERATED;
```

```
/* ---------------------------------------------------------------------------
 ↪  */
/* ENUMERATIONS */
/* ---------------------------------------------------------------------------
 ↪  */

enum {
    TPM_SPEC_FAMILY = 0x322E3000,
    TPM_SPEC_LEVEL = 0,
    TPM_SPEC_VERSION = 138,
    TPM_SPEC_YEAR = 2016,
    TPM_SPEC_DAY_OF_YEAR = 273,

    TPM_GENERATED_VALUE = 0xff544347,
};


typedef enum {
    TPM_ALG_ERROR           = 0x0000,
    TPM_ALG_RSA             = 0x0001,
    TPM_ALG_SHA             = 0x0004,
    TPM_ALG_SHA1            = TPM_ALG_SHA,
    TPM_ALG_HMAC            = 0x0005,
    TPM_ALG_AES             = 0x0006,
    TPM_ALG_MGF1            = 0x0007,
    TPM_ALG_KEYEDHASH       = 0x0008,
    TPM_ALG_XOR             = 0x000A,
    TPM_ALG_SHA256          = 0x000B,
    TPM_ALG_SHA384          = 0x000C,
    TPM_ALG_SHA512          = 0x000D,
    TPM_ALG_NULL            = 0x0010,
    TPM_ALG_SM3_256         = 0x0012,
    TPM_ALG_SM4             = 0x0013,
    TPM_ALG_RSASSA          = 0x0014,
    TPM_ALG_RSAES           = 0x0015,
    TPM_ALG_RSAPSS          = 0x0016,
    TPM_ALG_OAEP            = 0x0017,
    TPM_ALG_ECDSA           = 0x0018,
    TPM_ALG_ECDH            = 0x0019,
    TPM_ALG_ECDAA           = 0x001A,
    TPM_ALG_SM2             = 0x001B,
    TPM_ALG_ECSCHNORR       = 0x001C,
    TPM_ALG_ECMQV           = 0x001D,
    TPM_ALG_KDF1_SP800_56A  = 0x0020,
    TPM_ALG_KDF2            = 0x0021,
    TPM_ALG_KDF1_SP800_108  = 0x0022,
    TPM_ALG_ECC             = 0x0023,
    TPM_ALG_SYMCIPHER       = 0x0025,
    TPM_ALG_CAMELLIA        = 0x0026,
    TPM_ALG_CTR             = 0x0040,
    TPM_ALG_OFB             = 0x0041,
    TPM_ALG_CBC             = 0x0042,
    TPM_ALG_CFB             = 0x0043,
    TPM_ALG_ECB             = 0x0044,
```

```
} TPM_ALG_ID_T;
typedef UINT16 TPM_ALG_ID;

typedef enum {
    TPM_ECC_NONE        = 0x0000,
    TPM_ECC_NIST_P192   = 0x0001,
    TPM_ECC_NIST_P224   = 0x0002,
    TPM_ECC_NIST_P256   = 0x0003,
    TPM_ECC_NIST_P384   = 0x0004,
    TPM_ECC_NIST_P521   = 0x0005,
    TPM_ECC_BN_P256     = 0x0010,
    TPM_ECC_BN_P638     = 0x0011,
    TPM_ECC_SM2_P256    = 0x0020,
} TPM_ECC_CURVE_T;
typedef UINT16 TPM_ECC_CURVE;

/* Command Codes */
typedef enum {
    TPM_CC_FIRST                    = 0x0000011F,
    TPM_CC_NV_UndefineSpaceSpecial  = TPM_CC_FIRST,
    TPM_CC_EvictControl             = 0x00000120,
    TPM_CC_HierarchyControl         = 0x00000121,
    TPM_CC_NV_UndefineSpace         = 0x00000122,
    TPM_CC_ChangeEPS                = 0x00000124,
    TPM_CC_ChangePPS                = 0x00000125,
    TPM_CC_Clear                    = 0x00000126,
    TPM_CC_ClearControl             = 0x00000127,
    TPM_CC_ClockSet                 = 0x00000128,
    TPM_CC_HierarchyChangeAuth      = 0x00000129,
    TPM_CC_NV_DefineSpace           = 0x0000012A,
    TPM_CC_PCR_Allocate             = 0x0000012B,
    TPM_CC_PCR_SetAuthPolicy        = 0x0000012C,
    TPM_CC_PP_Commands              = 0x0000012D,
    TPM_CC_SetPrimaryPolicy         = 0x0000012E,
    TPM_CC_FieldUpgradeStart        = 0x0000012F,
    TPM_CC_ClockRateAdjust          = 0x00000130,
    TPM_CC_CreatePrimary            = 0x00000131,
    TPM_CC_NV_GlobalWriteLock       = 0x00000132,
    TPM_CC_GetCommandAuditDigest    = 0x00000133,
    TPM_CC_NV_Increment             = 0x00000134,
    TPM_CC_NV_SetBits               = 0x00000135,
    TPM_CC_NV_Extend                = 0x00000136,
    TPM_CC_NV_Write                 = 0x00000137,
    TPM_CC_NV_WriteLock             = 0x00000138,
    TPM_CC_DictionaryAttackLockReset = 0x00000139,
    TPM_CC_DictionaryAttackParameters = 0x0000013A,
    TPM_CC_NV_ChangeAuth            = 0x0000013B,
    TPM_CC_PCR_Event                = 0x0000013C,
    TPM_CC_PCR_Reset                = 0x0000013D,
    TPM_CC_SequenceComplete         = 0x0000013E,
    TPM_CC_SetAlgorithmSet          = 0x0000013F,
    TPM_CC_SetCommandCodeAuditStatus = 0x00000140,
    TPM_CC_FieldUpgradeData         = 0x00000141,
    TPM_CC_IncrementalSelfTest      = 0x00000142,
```

```
    TPM_CC_SelfTest                 = 0x00000143,
    TPM_CC_Startup                  = 0x00000144,
    TPM_CC_Shutdown                 = 0x00000145,
    TPM_CC_StirRandom               = 0x00000146,
    TPM_CC_ActivateCredential       = 0x00000147,
    TPM_CC_Certify                  = 0x00000148,
    TPM_CC_PolicyNV                 = 0x00000149,
    TPM_CC_CertifyCreation          = 0x0000014A,
    TPM_CC_Duplicate                = 0x0000014B,
    TPM_CC_GetTime                  = 0x0000014C,
    TPM_CC_GetSessionAuditDigest    = 0x0000014D,
    TPM_CC_NV_Read                  = 0x0000014E,
    TPM_CC_NV_ReadLock              = 0x0000014F,
    TPM_CC_ObjectChangeAuth         = 0x00000150,
    TPM_CC_PolicySecret             = 0x00000151,
    TPM_CC_Rewrap                   = 0x00000152,
    TPM_CC_Create                   = 0x00000153,
    TPM_CC_ECDH_ZGen                = 0x00000154,
    TPM_CC_HMAC                     = 0x00000155,
    TPM_CC_Import                   = 0x00000156,
    TPM_CC_Load                     = 0x00000157,
    TPM_CC_Quote                    = 0x00000158,
    TPM_CC_RSA_Decrypt              = 0x00000159,
    TPM_CC_HMAC_Start               = 0x0000015B,
    TPM_CC_SequenceUpdate           = 0x0000015C,
    TPM_CC_Sign                     = 0x0000015D,
    TPM_CC_Unseal                   = 0x0000015E,
    TPM_CC_PolicySigned             = 0x00000160,
    TPM_CC_ContextLoad              = 0x00000161,
    TPM_CC_ContextSave              = 0x00000162,
    TPM_CC_ECDH_KeyGen              = 0x00000163,
    TPM_CC_EncryptDecrypt           = 0x00000164,
    TPM_CC_FlushContext             = 0x00000165,
    TPM_CC_LoadExternal             = 0x00000167,
    TPM_CC_MakeCredential           = 0x00000168,
    TPM_CC_NV_ReadPublic            = 0x00000169,
    TPM_CC_PolicyAuthorize          = 0x0000016A,
    TPM_CC_PolicyAuthValue          = 0x0000016B,
    TPM_CC_PolicyCommandCode        = 0x0000016C,
    TPM_CC_PolicyCounterTimer       = 0x0000016D,
    TPM_CC_PolicyCpHash             = 0x0000016E,
    TPM_CC_PolicyLocality           = 0x0000016F,
    TPM_CC_PolicyNameHash           = 0x00000170,
    TPM_CC_PolicyOR                 = 0x00000171,
    TPM_CC_PolicyTicket             = 0x00000172,
    TPM_CC_ReadPublic               = 0x00000173,
    TPM_CC_RSA_Encrypt              = 0x00000174,
    TPM_CC_StartAuthSession         = 0x00000176,
    TPM_CC_VerifySignature          = 0x00000177,
    TPM_CC_ECC_Parameters           = 0x00000178,
    TPM_CC_FirmwareRead             = 0x00000179,
    TPM_CC_GetCapability            = 0x0000017A,
    TPM_CC_GetRandom                = 0x0000017B,
    TPM_CC_GetTestResult            = 0x0000017C,
```

```
    TPM_CC_Hash                     = 0x0000017D,
    TPM_CC_PCR_Read                 = 0x0000017E,
    TPM_CC_PolicyPCR                = 0x0000017F,
    TPM_CC_PolicyRestart            = 0x00000180,
    TPM_CC_ReadClock                = 0x00000181,
    TPM_CC_PCR_Extend               = 0x00000182,
    TPM_CC_PCR_SetAuthValue         = 0x00000183,
    TPM_CC_NV_Certify               = 0x00000184,
    TPM_CC_EventSequenceComplete    = 0x00000185,
    TPM_CC_HashSequenceStart        = 0x00000186,
    TPM_CC_PolicyPhysicalPresence   = 0x00000187,
    TPM_CC_PolicyDuplicationSelect  = 0x00000188,
    TPM_CC_PolicyGetDigest          = 0x00000189,
    TPM_CC_TestParms                = 0x0000018A,
    TPM_CC_Commit                   = 0x0000018B,
    TPM_CC_PolicyPassword           = 0x0000018C,
    TPM_CC_ZGen_2Phase              = 0x0000018D,
    TPM_CC_EC_Ephemeral             = 0x0000018E,
    TPM_CC_PolicyNvWritten          = 0x0000018F,
    TPM_CC_PolicyTemplate           = 0x00000190,
    TPM_CC_CreateLoaded             = 0x00000191,
    TPM_CC_PolicyAuthorizeNV        = 0x00000192,
    TPM_CC_EncryptDecrypt2          = 0x00000193,
    TPM_CC_LAST                     = TPM_CC_EncryptDecrypt2,

    CC_VEND                         = 0x20000000,
    TPM_CC_Vendor_TCG_Test          = CC_VEND + 0x0000,
#if defined(WOLFTPM_ST33) || defined(WOLFTPM_AUTODETECT)
    TPM_CC_SetMode                  = CC_VEND + 0x0307,
    TPM_CC_SetCommandSet            = CC_VEND + 0x0309,
    TPM_CC_GetRandom2               = CC_VEND + 0x030E,
#endif
#ifdef WOLFTPM_ST33
    TPM_CC_RestoreEK                = CC_VEND + 0x030A,
    TPM_CC_SetCommandSetLock        = CC_VEND + 0x030B,
    TPM_CC_GPIO_Config              = CC_VEND + 0x030F,
#endif
#ifdef WOLFTPM_NUVOTON
    TPM_CC_NTC2_PreConfig           = CC_VEND + 0x0211,
    TPM_CC_NTC2_GetConfig           = CC_VEND + 0x0213,
#endif
} TPM_CC_T;
typedef UINT32 TPM_CC;

/* Response Codes */
typedef enum {
    TPM_RC_SUCCESS  = 0x000,
    TPM_RC_BAD_TAG  = 0x01E,

    RC_VER1 = 0x100,
    TPM_RC_INITIALIZE           = RC_VER1 + 0x000,
    TPM_RC_FAILURE              = RC_VER1 + 0x001,
    TPM_RC_SEQUENCE             = RC_VER1 + 0x003,
    TPM_RC_PRIVATE              = RC_VER1 + 0x00B,
```

```
    TPM_RC_HMAC                 = RC_VER1 + 0x019,
    TPM_RC_DISABLED             = RC_VER1 + 0x020,
    TPM_RC_EXCLUSIVE            = RC_VER1 + 0x021,
    TPM_RC_AUTH_TYPE            = RC_VER1 + 0x024,
    TPM_RC_AUTH_MISSING         = RC_VER1 + 0x025,
    TPM_RC_POLICY               = RC_VER1 + 0x026,
    TPM_RC_PCR                  = RC_VER1 + 0x027,
    TPM_RC_PCR_CHANGED          = RC_VER1 + 0x028,
    TPM_RC_UPGRADE              = RC_VER1 + 0x02D,
    TPM_RC_TOO_MANY_CONTEXTS    = RC_VER1 + 0x02E,
    TPM_RC_AUTH_UNAVAILABLE     = RC_VER1 + 0x02F,
    TPM_RC_REBOOT               = RC_VER1 + 0x030,
    TPM_RC_UNBALANCED           = RC_VER1 + 0x031,
    TPM_RC_COMMAND_SIZE         = RC_VER1 + 0x042,
    TPM_RC_COMMAND_CODE         = RC_VER1 + 0x043,
    TPM_RC_AUTHSIZE             = RC_VER1 + 0x044,
    TPM_RC_AUTH_CONTEXT         = RC_VER1 + 0x045,
    TPM_RC_NV_RANGE             = RC_VER1 + 0x046,
    TPM_RC_NV_SIZE              = RC_VER1 + 0x047,
    TPM_RC_NV_LOCKED            = RC_VER1 + 0x048,
    TPM_RC_NV_AUTHORIZATION     = RC_VER1 + 0x049,
    TPM_RC_NV_UNINITIALIZED     = RC_VER1 + 0x04A,
    TPM_RC_NV_SPACE             = RC_VER1 + 0x04B,
    TPM_RC_NV_DEFINED           = RC_VER1 + 0x04C,
    TPM_RC_BAD_CONTEXT          = RC_VER1 + 0x050,
    TPM_RC_CPHASH               = RC_VER1 + 0x051,
    TPM_RC_PARENT               = RC_VER1 + 0x052,
    TPM_RC_NEEDS_TEST           = RC_VER1 + 0x053,
    TPM_RC_NO_RESULT            = RC_VER1 + 0x054,
    TPM_RC_SENSITIVE            = RC_VER1 + 0x055,
    RC_MAX_FM0                  = RC_VER1 + 0x07F,

    RC_FMT1 = 0x080,
    TPM_RC_ASYMMETRIC       = RC_FMT1 + 0x001,
    TPM_RC_ATTRIBUTES       = RC_FMT1 + 0x002,
    TPM_RC_HASH             = RC_FMT1 + 0x003,
    TPM_RC_VALUE            = RC_FMT1 + 0x004,
    TPM_RC_HIERARCHY        = RC_FMT1 + 0x005,
    TPM_RC_KEY_SIZE         = RC_FMT1 + 0x007,
    TPM_RC_MGF              = RC_FMT1 + 0x008,
    TPM_RC_MODE             = RC_FMT1 + 0x009,
    TPM_RC_TYPE             = RC_FMT1 + 0x00A,
    TPM_RC_HANDLE           = RC_FMT1 + 0x00B,
    TPM_RC_KDF              = RC_FMT1 + 0x00C,
    TPM_RC_RANGE            = RC_FMT1 + 0x00D,
    TPM_RC_AUTH_FAIL        = RC_FMT1 + 0x00E,
    TPM_RC_NONCE            = RC_FMT1 + 0x00F,
    TPM_RC_PP               = RC_FMT1 + 0x010,
    TPM_RC_SCHEME           = RC_FMT1 + 0x012,
    TPM_RC_SIZE             = RC_FMT1 + 0x015,
    TPM_RC_SYMMETRIC        = RC_FMT1 + 0x016,
    TPM_RC_TAG              = RC_FMT1 + 0x017,
    TPM_RC_SELECTOR         = RC_FMT1 + 0x018,
    TPM_RC_INSUFFICIENT     = RC_FMT1 + 0x01A,
```

```
    TPM_RC_SIGNATURE        = RC_FMT1 + 0x01B,
    TPM_RC_KEY              = RC_FMT1 + 0x01C,
    TPM_RC_POLICY_FAIL      = RC_FMT1 + 0x01D,
    TPM_RC_INTEGRITY        = RC_FMT1 + 0x01F,
    TPM_RC_TICKET           = RC_FMT1 + 0x020,
    TPM_RC_RESERVED_BITS    = RC_FMT1 + 0x021,
    TPM_RC_BAD_AUTH         = RC_FMT1 + 0x022,
    TPM_RC_EXPIRED          = RC_FMT1 + 0x023,
    TPM_RC_POLICY_CC        = RC_FMT1 + 0x024,
    TPM_RC_BINDING          = RC_FMT1 + 0x025,
    TPM_RC_CURVE            = RC_FMT1 + 0x026,
    TPM_RC_ECC_POINT        = RC_FMT1 + 0x027,
    RC_MAX_FMT1             = RC_FMT1 + 0x03F,

    RC_WARN = 0x900,
    TPM_RC_CONTEXT_GAP      = RC_WARN + 0x001,
    TPM_RC_OBJECT_MEMORY    = RC_WARN + 0x002,
    TPM_RC_SESSION_MEMORY   = RC_WARN + 0x003,
    TPM_RC_MEMORY           = RC_WARN + 0x004,
    TPM_RC_SESSION_HANDLES  = RC_WARN + 0x005,
    TPM_RC_OBJECT_HANDLES   = RC_WARN + 0x006,
    TPM_RC_LOCALITY         = RC_WARN + 0x007,
    TPM_RC_YIELDED          = RC_WARN + 0x008,
    TPM_RC_CANCELED         = RC_WARN + 0x009,
    TPM_RC_TESTING          = RC_WARN + 0x00A,
    TPM_RC_REFERENCE_H0     = RC_WARN + 0x010,
    TPM_RC_REFERENCE_H1     = RC_WARN + 0x011,
    TPM_RC_REFERENCE_H2     = RC_WARN + 0x012,
    TPM_RC_REFERENCE_H3     = RC_WARN + 0x013,
    TPM_RC_REFERENCE_H4     = RC_WARN + 0x014,
    TPM_RC_REFERENCE_H5     = RC_WARN + 0x015,
    TPM_RC_REFERENCE_H6     = RC_WARN + 0x016,
    TPM_RC_REFERENCE_S0     = RC_WARN + 0x018,
    TPM_RC_REFERENCE_S1     = RC_WARN + 0x019,
    TPM_RC_REFERENCE_S2     = RC_WARN + 0x01A,
    TPM_RC_REFERENCE_S3     = RC_WARN + 0x01B,
    TPM_RC_REFERENCE_S4     = RC_WARN + 0x01C,
    TPM_RC_REFERENCE_S5     = RC_WARN + 0x01D,
    TPM_RC_REFERENCE_S6     = RC_WARN + 0x01E,
    TPM_RC_NV_RATE          = RC_WARN + 0x020,
    TPM_RC_LOCKOUT          = RC_WARN + 0x021,
    TPM_RC_RETRY            = RC_WARN + 0x022,
    TPM_RC_NV_UNAVAILABLE   = RC_WARN + 0x023,
    RC_MAX_WARN             = RC_WARN + 0x03F,

    TPM_RC_NOT_USED         = RC_WARN + 0x07F,

    TPM_RC_H        = 0x000,
    TPM_RC_P        = 0x040,
    TPM_RC_S        = 0x800,
    TPM_RC_1        = 0x100,
    TPM_RC_2        = 0x200,
    TPM_RC_3        = 0x300,
    TPM_RC_4        = 0x400,
```

```
    TPM_RC_5        = 0x500,
    TPM_RC_6        = 0x600,
    TPM_RC_7        = 0x700,
    TPM_RC_8        = 0x800,
    TPM_RC_9        = 0x900,
    TPM_RC_A        = 0xA00,
    TPM_RC_B        = 0xB00,
    TPM_RC_C        = 0xC00,
    TPM_RC_D        = 0xD00,
    TPM_RC_E        = 0xE00,
    TPM_RC_F        = 0xF00,
    TPM_RC_N_MASK   = 0xF00,

    /* use negative codes for internal errors */
    TPM_RC_TIMEOUT = -100,
} TPM_RC_T;
typedef INT32 TPM_RC; /* type is unsigned 16-bits, but internally use signed
 ↪  32-bit */

typedef enum {
    TPM_CLOCK_COARSE_SLOWER = -3,
    TPM_CLOCK_MEDIUM_SLOWER = -2,
    TPM_CLOCK_FINE_SLOWER   = -1,
    TPM_CLOCK_NO_CHANGE     = 0,
    TPM_CLOCK_FINE_FASTER   = 1,
    TPM_CLOCK_MEDIUM_FASTER = 2,
    TPM_CLOCK_COARSE_FASTER = 3,
} TPM_CLOCK_ADJUST_T;
typedef UINT8 TPM_CLOCK_ADJUST;

/* EA Arithmetic Operands */
typedef enum {
    TPM_EO_EQ           = 0x0000,
    TPM_EO_NEQ          = 0x0001,
    TPM_EO_SIGNED_GT    = 0x0002,
    TPM_EO_UNSIGNED_GT  = 0x0003,
    TPM_EO_SIGNED_LT    = 0x0004,
    TPM_EO_UNSIGNED_LT  = 0x0005,
    TPM_EO_SIGNED_GE    = 0x0006,
    TPM_EO_UNSIGNED_GE  = 0x0007,
    TPM_EO_SIGNED_LE    = 0x0008,
    TPM_EO_UNSIGNED_LE  = 0x0009,
    TPM_EO_BITSET       = 0x000A,
    TPM_EO_BITCLEAR     = 0x000B,
} TPM_EO_T;
typedef UINT16 TPM_EO;

/* Structure Tags */
typedef enum {
    TPM_ST_RSP_COMMAND          = 0x00C4,
    TPM_ST_NULL                 = 0X8000,
    TPM_ST_NO_SESSIONS          = 0x8001,
    TPM_ST_SESSIONS             = 0x8002,
    TPM_ST_ATTEST_NV            = 0x8014,
```

```
    TPM_ST_ATTEST_COMMAND_AUDIT = 0x8015,
    TPM_ST_ATTEST_SESSION_AUDIT = 0x8016,
    TPM_ST_ATTEST_CERTIFY       = 0x8017,
    TPM_ST_ATTEST_QUOTE         = 0x8018,
    TPM_ST_ATTEST_TIME          = 0x8019,
    TPM_ST_ATTEST_CREATION      = 0x801A,
    TPM_ST_CREATION             = 0x8021,
    TPM_ST_VERIFIED             = 0x8022,
    TPM_ST_AUTH_SECRET          = 0x8023,
    TPM_ST_HASHCHECK            = 0x8024,
    TPM_ST_AUTH_SIGNED          = 0x8025,
    TPM_ST_FU_MANIFEST          = 0x8029,
} TPM_ST_T;
typedef UINT16 TPM_ST;

/* Session Type */
typedef enum {
    TPM_SE_HMAC     = 0x00,
    TPM_SE_POLICY   = 0x01,
    TPM_SE_TRIAL    = 0x03,
} TPM_SE_T;
typedef UINT8 TPM_SE;


/* Startup Type */
typedef enum {
    TPM_SU_CLEAR = 0x0000,
    TPM_SU_STATE = 0x0001,
} TPM_SU_T;
typedef UINT16 TPM_SU;

/* Capabilities */
typedef enum {
    TPM_CAP_FIRST           = 0x00000000,
    TPM_CAP_ALGS            = TPM_CAP_FIRST,
    TPM_CAP_HANDLES         = 0x00000001,
    TPM_CAP_COMMANDS        = 0x00000002,
    TPM_CAP_PP_COMMANDS     = 0x00000003,
    TPM_CAP_AUDIT_COMMANDS  = 0x00000004,
    TPM_CAP_PCRS            = 0x00000005,
    TPM_CAP_TPM_PROPERTIES  = 0x00000006,
    TPM_CAP_PCR_PROPERTIES  = 0x00000007,
    TPM_CAP_ECC_CURVES      = 0x00000008,
    TPM_CAP_LAST            = TPM_CAP_ECC_CURVES,

    TPM_CAP_VENDOR_PROPERTY = 0x00000100,
} TPM_CAP_T;
typedef UINT32 TPM_CAP;

/* Property Tag */
typedef enum {
    TPM_PT_NONE    = 0x00000000,
    PT_GROUP       = 0x00000100,
```

```
    PT_FIXED = PT_GROUP * 1,
    TPM_PT_FAMILY_INDICATOR     = PT_FIXED + 0,
    TPM_PT_LEVEL                = PT_FIXED + 1,
    TPM_PT_REVISION             = PT_FIXED + 2,
    TPM_PT_DAY_OF_YEAR          = PT_FIXED + 3,
    TPM_PT_YEAR                 = PT_FIXED + 4,
    TPM_PT_MANUFACTURER         = PT_FIXED + 5,
    TPM_PT_VENDOR_STRING_1      = PT_FIXED + 6,
    TPM_PT_VENDOR_STRING_2      = PT_FIXED + 7,
    TPM_PT_VENDOR_STRING_3      = PT_FIXED + 8,
    TPM_PT_VENDOR_STRING_4      = PT_FIXED + 9,
    TPM_PT_VENDOR_TPM_TYPE      = PT_FIXED + 10,
    TPM_PT_FIRMWARE_VERSION_1   = PT_FIXED + 11,
    TPM_PT_FIRMWARE_VERSION_2   = PT_FIXED + 12,
    TPM_PT_INPUT_BUFFER         = PT_FIXED + 13,
    TPM_PT_HR_TRANSIENT_MIN     = PT_FIXED + 14,
    TPM_PT_HR_PERSISTENT_MIN    = PT_FIXED + 15,
    TPM_PT_HR_LOADED_MIN        = PT_FIXED + 16,
    TPM_PT_ACTIVE_SESSIONS_MAX  = PT_FIXED + 17,
    TPM_PT_PCR_COUNT            = PT_FIXED + 18,
    TPM_PT_PCR_SELECT_MIN       = PT_FIXED + 19,
    TPM_PT_CONTEXT_GAP_MAX      = PT_FIXED + 20,
    TPM_PT_NV_COUNTERS_MAX      = PT_FIXED + 22,
    TPM_PT_NV_INDEX_MAX         = PT_FIXED + 23,
    TPM_PT_MEMORY               = PT_FIXED + 24,
    TPM_PT_CLOCK_UPDATE         = PT_FIXED + 25,
    TPM_PT_CONTEXT_HASH         = PT_FIXED + 26,
    TPM_PT_CONTEXT_SYM          = PT_FIXED + 27,
    TPM_PT_CONTEXT_SYM_SIZE     = PT_FIXED + 28,
    TPM_PT_ORDERLY_COUNT        = PT_FIXED + 29,
    TPM_PT_MAX_COMMAND_SIZE     = PT_FIXED + 30,
    TPM_PT_MAX_RESPONSE_SIZE    = PT_FIXED + 31,
    TPM_PT_MAX_DIGEST           = PT_FIXED + 32,
    TPM_PT_MAX_OBJECT_CONTEXT   = PT_FIXED + 33,
    TPM_PT_MAX_SESSION_CONTEXT  = PT_FIXED + 34,
    TPM_PT_PS_FAMILY_INDICATOR  = PT_FIXED + 35,
    TPM_PT_PS_LEVEL             = PT_FIXED + 36,
    TPM_PT_PS_REVISION          = PT_FIXED + 37,
    TPM_PT_PS_DAY_OF_YEAR       = PT_FIXED + 38,
    TPM_PT_PS_YEAR              = PT_FIXED + 39,
    TPM_PT_SPLIT_MAX            = PT_FIXED + 40,
    TPM_PT_TOTAL_COMMANDS       = PT_FIXED + 41,
    TPM_PT_LIBRARY_COMMANDS     = PT_FIXED + 42,
    TPM_PT_VENDOR_COMMANDS      = PT_FIXED + 43,
    TPM_PT_NV_BUFFER_MAX        = PT_FIXED + 44,
    TPM_PT_MODES                = PT_FIXED + 45,
    TPM_PT_MAX_CAP_BUFFER       = PT_FIXED + 46,

    PT_VAR = PT_GROUP * 2,
    TPM_PT_PERMANENT            = PT_VAR + 0,
    TPM_PT_STARTUP_CLEAR        = PT_VAR + 1,
    TPM_PT_HR_NV_INDEX          = PT_VAR + 2,
    TPM_PT_HR_LOADED            = PT_VAR + 3,
    TPM_PT_HR_LOADED_AVAIL      = PT_VAR + 4,
```

```
    TPM_PT_HR_ACTIVE            = PT_VAR + 5,
    TPM_PT_HR_ACTIVE_AVAIL      = PT_VAR + 6,
    TPM_PT_HR_TRANSIENT_AVAIL   = PT_VAR + 7,
    TPM_PT_HR_PERSISTENT        = PT_VAR + 8,
    TPM_PT_HR_PERSISTENT_AVAIL  = PT_VAR + 9,
    TPM_PT_NV_COUNTERS          = PT_VAR + 10,
    TPM_PT_NV_COUNTERS_AVAIL    = PT_VAR + 11,
    TPM_PT_ALGORITHM_SET        = PT_VAR + 12,
    TPM_PT_LOADED_CURVES        = PT_VAR + 13,
    TPM_PT_LOCKOUT_COUNTER      = PT_VAR + 14,
    TPM_PT_MAX_AUTH_FAIL        = PT_VAR + 15,
    TPM_PT_LOCKOUT_INTERVAL     = PT_VAR + 16,
    TPM_PT_LOCKOUT_RECOVERY     = PT_VAR + 17,
    TPM_PT_NV_WRITE_RECOVERY    = PT_VAR + 18,
    TPM_PT_AUDIT_COUNTER_0      = PT_VAR + 19,
    TPM_PT_AUDIT_COUNTER_1      = PT_VAR + 20,
} TPM_PT_T;
typedef UINT32 TPM_PT;

/* PCR Property Tag */
typedef enum {
    TPM_PT_PCR_FIRST        = 0x00000000,
    TPM_PT_PCR_SAVE         = TPM_PT_PCR_FIRST,
    TPM_PT_PCR_EXTEND_L0    = 0x00000001,
    TPM_PT_PCR_RESET_L0     = 0x00000002,
    TPM_PT_PCR_EXTEND_L1    = 0x00000003,
    TPM_PT_PCR_RESET_L1     = 0x00000004,
    TPM_PT_PCR_EXTEND_L2    = 0x00000005,
    TPM_PT_PCR_RESET_L2     = 0x00000006,
    TPM_PT_PCR_EXTEND_L3    = 0x00000007,
    TPM_PT_PCR_RESET_L3     = 0x00000008,
    TPM_PT_PCR_EXTEND_L4    = 0x00000009,
    TPM_PT_PCR_RESET_L4     = 0x0000000A,
    TPM_PT_PCR_NO_INCREMENT = 0x00000011,
    TPM_PT_PCR_DRTM_RESET   = 0x00000012,
    TPM_PT_PCR_POLICY       = 0x00000013,
    TPM_PT_PCR_AUTH         = 0x00000014,
    TPM_PT_PCR_LAST         = TPM_PT_PCR_AUTH,
} TPM_PT_PCR_T;
typedef UINT32 TPM_PT_PCR;

/* Platform Specific */
typedef enum {
    TPM_PS_MAIN             = 0x00000000,
    TPM_PS_PC               = 0x00000001,
    TPM_PS_PDA              = 0x00000002,
    TPM_PS_CELL_PHONE       = 0x00000003,
    TPM_PS_SERVER           = 0x00000004,
    TPM_PS_PERIPHERAL       = 0x00000005,
    TPM_PS_TSS              = 0x00000006,
    TPM_PS_STORAGE          = 0x00000007,
    TPM_PS_AUTHENTICATION   = 0x00000008,
    TPM_PS_EMBEDDED         = 0x00000009,
    TPM_PS_HARDCOPY         = 0x0000000A,
```

```
    TPM_PS_INFRASTRUCTURE   = 0x0000000B,
    TPM_PS_VIRTUALIZATION   = 0x0000000C,
    TPM_PS_TNC              = 0x0000000D,
    TPM_PS_MULTI_TENANT     = 0x0000000E,
    TPM_PS_TC               = 0x0000000F,
} TPM_PS_T;
typedef UINT32 TPM_PS;


/* HANDLES */
typedef UINT32 TPM_HANDLE;

/* Handle Types */
typedef enum {
    TPM_HT_PCR              = 0x00,
    TPM_HT_NV_INDEX         = 0x01,
    TPM_HT_HMAC_SESSION     = 0x02,
    TPM_HT_LOADED_SESSION   = 0x02,
    TPM_HT_POLICY_SESSION   = 0x03,
    TPM_HT_ACTIVE_SESSION   = 0x03,
    TPM_HT_PERMANENT        = 0x40,
    TPM_HT_TRANSIENT        = 0x80,
    TPM_HT_PERSISTENT       = 0x81,
} TPM_HT_T;
typedef UINT8 TPM_HT;

/* Permanent Handles */
typedef enum {
    TPM_RH_FIRST        = 0x40000000,
    TPM_RH_SRK          = TPM_RH_FIRST,
    TPM_RH_OWNER        = 0x40000001,
    TPM_RH_REVOKE       = 0x40000002,
    TPM_RH_TRANSPORT    = 0x40000003,
    TPM_RH_OPERATOR     = 0x40000004,
    TPM_RH_ADMIN        = 0x40000005,
    TPM_RH_EK           = 0x40000006,
    TPM_RH_NULL         = 0x40000007,
    TPM_RH_UNASSIGNED   = 0x40000008,
    TPM_RS_PW           = 0x40000009,
    TPM_RH_LOCKOUT      = 0x4000000A,
    TPM_RH_ENDORSEMENT  = 0x4000000B,
    TPM_RH_PLATFORM     = 0x4000000C,
    TPM_RH_PLATFORM_NV  = 0x4000000D,
    TPM_RH_AUTH_00      = 0x40000010,
    TPM_RH_AUTH_FF      = 0x4000010F,
    TPM_RH_LAST         = TPM_RH_AUTH_FF,
} TPM_RH_T;
typedef UINT32 TPM_RH;

/* Handle Value Constants */
typedef enum {
    HR_HANDLE_MASK          = 0x00FFFFFF,
    HR_RANGE_MASK           = 0xFF000000,
    HR_SHIFT                = 24,
```

```
    HR_PCR                  = ((UINT32)TPM_HT_PCR << HR_SHIFT),
    HR_HMAC_SESSION         = ((UINT32)TPM_HT_HMAC_SESSION << HR_SHIFT),
    HR_POLICY_SESSION       = ((UINT32)TPM_HT_POLICY_SESSION << HR_SHIFT),
    HR_TRANSIENT            = ((UINT32)TPM_HT_TRANSIENT << HR_SHIFT),
    HR_PERSISTENT           = ((UINT32)TPM_HT_PERSISTENT << HR_SHIFT),
    HR_NV_INDEX             = ((UINT32)TPM_HT_NV_INDEX << HR_SHIFT),
    HR_PERMANENT            = ((UINT32)TPM_HT_PERMANENT << HR_SHIFT),
    PCR_FIRST               = (HR_PCR + 0),
    PCR_LAST                = (PCR_FIRST + IMPLEMENTATION_PCR-1),
    HMAC_SESSION_FIRST      = (HR_HMAC_SESSION + 0),
    HMAC_SESSION_LAST       = (HMAC_SESSION_FIRST+MAX_ACTIVE_SESSIONS-1),
    LOADED_SESSION_FIRST    = HMAC_SESSION_FIRST,
    LOADED_SESSION_LAST     = HMAC_SESSION_LAST,
    POLICY_SESSION_FIRST    = (HR_POLICY_SESSION + 0),
    POLICY_SESSION_LAST     = (POLICY_SESSION_FIRST+MAX_ACTIVE_SESSIONS-1),
    TRANSIENT_FIRST         = (HR_TRANSIENT + 0),
    ACTIVE_SESSION_FIRST    = POLICY_SESSION_FIRST,
    ACTIVE_SESSION_LAST     = POLICY_SESSION_LAST,
    TRANSIENT_LAST          = (TRANSIENT_FIRST+MAX_LOADED_OBJECTS-1),
    PERSISTENT_FIRST        = (HR_PERSISTENT + 0),
    PERSISTENT_LAST         = (PERSISTENT_FIRST + 0x00FFFFFF),
    PLATFORM_PERSISTENT     = (PERSISTENT_FIRST + 0x00800000),
    NV_INDEX_FIRST          = (HR_NV_INDEX + 0),
    NV_INDEX_LAST           = (NV_INDEX_FIRST + 0x00FFFFFF),
    PERMANENT_FIRST         = TPM_RH_FIRST,
    PERMANENT_LAST          = TPM_RH_LAST,
} TPM_HC_T;
typedef UINT32 TPM_HC;


/* Attributes */
typedef UINT32 TPMA_ALGORITHM;
enum TPMA_ALGORITHM_mask {
    TPMA_ALGORITHM_asymmetric = 0x00000001,
    TPMA_ALGORITHM_symmetric  = 0x00000002,
    TPMA_ALGORITHM_hash       = 0x00000004,
    TPMA_ALGORITHM_object     = 0x00000008,
    TPMA_ALGORITHM_signing    = 0x00000010,
    TPMA_ALGORITHM_encrypting = 0x00000020,
    TPMA_ALGORITHM_method     = 0x00000040,
};

typedef UINT32 TPMA_OBJECT;
enum TPMA_OBJECT_mask {
    TPMA_OBJECT_fixedTPM            = 0x00000002,
    TPMA_OBJECT_stClear             = 0x00000004,
    TPMA_OBJECT_fixedParent         = 0x00000010,
    TPMA_OBJECT_sensitiveDataOrigin = 0x00000020,
    TPMA_OBJECT_userWithAuth        = 0x00000040,
    TPMA_OBJECT_adminWithPolicy     = 0x00000080,
    TPMA_OBJECT_derivedDataOrigin   = 0x00000200,
    TPMA_OBJECT_noDA                = 0x00000400,
    TPMA_OBJECT_encryptedDuplication= 0x00000800,
    TPMA_OBJECT_restricted          = 0x00010000,
```

```
    TPMA_OBJECT_decrypt             = 0x00020000,
    TPMA_OBJECT_sign                = 0x00040000,
};

typedef BYTE TPMA_SESSION;
enum TPMA_SESSION_mask {
    TPMA_SESSION_continueSession    = 0x01,
    TPMA_SESSION_auditExclusive     = 0x02,
    TPMA_SESSION_auditReset         = 0x04,
    TPMA_SESSION_decrypt            = 0x20,
    TPMA_SESSION_encrypt            = 0x40,
    TPMA_SESSION_audit              = 0x80,
};

typedef BYTE TPMA_LOCALITY;
enum TPMA_LOCALITY_mask {
    TPM_LOC_ZERO = 0x01,
    TPM_LOC_ONE = 0x02,
    TPM_LOC_TWO = 0x04,
    TPM_LOC_THREE = 0x08,
    TPM_LOC_FOUR = 0x10,
};

typedef UINT32 TPMA_PERMANENT;
enum TPMA_PERMANENT_mask {
    TPMA_PERMANENT_ownerAuthSet         = 0x00000001,
    TPMA_PERMANENT_endorsementAuthSet   = 0x00000002,
    TPMA_PERMANENT_lockoutAuthSet       = 0x00000004,
    TPMA_PERMANENT_disableClear         = 0x00000100,
    TPMA_PERMANENT_inLockout            = 0x00000200,
    TPMA_PERMANENT_tpmGeneratedEPS      = 0x00000400,
};

typedef UINT32 TPMA_STARTUP_CLEAR;
enum TPMA_STARTUP_CLEAR_mask {
    TPMA_STARTUP_CLEAR_phEnable     = 0x00000001,
    TPMA_STARTUP_CLEAR_shEnable     = 0x00000002,
    TPMA_STARTUP_CLEAR_ehEnable     = 0x00000004,
    TPMA_STARTUP_CLEAR_phEnableNV   = 0x00000008,
    TPMA_STARTUP_CLEAR_orderly      = 0x80000000,
};

typedef UINT32 TPMA_MEMORY;
enum TPMA_MEMORY_mask {
    TPMA_MEMORY_sharedRAM           = 0x00000001,
    TPMA_MEMORY_sharedNV            = 0x00000002,
    TPMA_MEMORY_objectCopiedToRam   = 0x00000004,
};

typedef UINT32 TPMA_CC;
enum TPMA_CC_mask {
    TPMA_CC_commandIndex = 0x0000FFFF,
    TPMA_CC_nv           = 0x00400000,
    TPMA_CC_extensive    = 0x00800000,
```

```
    TPMA_CC_flushed      = 0x01000000,
    TPMA_CC_cHandles     = 0x0E000000,
    TPMA_CC_rHandle      = 0x10000000,
    TPMA_CC_V            = 0x20000000,
};


/* Interface Types */

typedef BYTE TPMI_YES_NO;
typedef TPM_HANDLE TPMI_DH_OBJECT;
typedef TPM_HANDLE TPMI_DH_PARENT;
typedef TPM_HANDLE TPMI_DH_PERSISTENT;
typedef TPM_HANDLE TPMI_DH_ENTITY;
typedef TPM_HANDLE TPMI_DH_PCR;
typedef TPM_HANDLE TPMI_SH_AUTH_SESSION;
typedef TPM_HANDLE TPMI_SH_HMAC;
typedef TPM_HANDLE TPMI_SH_POLICY;
typedef TPM_HANDLE TPMI_DH_CONTEXT;
typedef TPM_HANDLE TPMI_RH_HIERARCHY;
typedef TPM_HANDLE TPMI_RH_ENABLES;
typedef TPM_HANDLE TPMI_RH_HIERARCHY_AUTH;
typedef TPM_HANDLE TPMI_RH_PLATFORM;
typedef TPM_HANDLE TPMI_RH_OWNER;
typedef TPM_HANDLE TPMI_RH_ENDORSEMENT;
typedef TPM_HANDLE TPMI_RH_PROVISION;
typedef TPM_HANDLE TPMI_RH_CLEAR;
typedef TPM_HANDLE TPMI_RH_NV_AUTH;
typedef TPM_HANDLE TPMI_RH_LOCKOUT;
typedef TPM_HANDLE TPMI_RH_NV_INDEX;

typedef TPM_ALG_ID TPMI_ALG_HASH;
typedef TPM_ALG_ID TPMI_ALG_ASYM;
typedef TPM_ALG_ID TPMI_ALG_SYM;
typedef TPM_ALG_ID TPMI_ALG_SYM_OBJECT;
typedef TPM_ALG_ID TPMI_ALG_SYM_MODE;
typedef TPM_ALG_ID TPMI_ALG_KDF;
typedef TPM_ALG_ID TPMI_ALG_SIG_SCHEME;
typedef TPM_ALG_ID TPMI_ECC_KEY_EXCHANGE;

typedef TPM_ST TPMI_ST_COMMAND_TAG;


/* Structures */

typedef struct TPMS_ALGORITHM_DESCRIPTION {
    TPM_ALG_ID alg;
    TPMA_ALGORITHM attributes;
} TPMS_ALGORITHM_DESCRIPTION;


typedef union TPMU_HA {
    BYTE sha512[TPM_SHA512_DIGEST_SIZE];
```

```
    BYTE sha384[TPM_SHA384_DIGEST_SIZE];
    BYTE sha256[TPM_SHA256_DIGEST_SIZE];
    BYTE sha224[TPM_SHA224_DIGEST_SIZE];
    BYTE sha[TPM_SHA_DIGEST_SIZE];
    BYTE md5[TPM_MD5_DIGEST_SIZE];
    BYTE H[TPM_MAX_DIGEST_SIZE];
} TPMU_HA;

typedef struct TPMT_HA {
    TPMI_ALG_HASH hashAlg;
    TPMU_HA digest;
} TPMT_HA;

typedef struct TPM2B_DIGEST {
    UINT16 size;
    BYTE buffer[sizeof(TPMU_HA)];
} TPM2B_DIGEST;

typedef struct TPM2B_DATA {
    UINT16 size;
    BYTE buffer[sizeof(TPMT_HA)];
} TPM2B_DATA;

typedef TPM2B_DIGEST TPM2B_NONCE;
typedef TPM2B_DIGEST TPM2B_AUTH;
typedef TPM2B_DIGEST TPM2B_OPERAND;

typedef struct TPM2B_EVENT {
    UINT16 size;
    BYTE buffer[1024];
} TPM2B_EVENT;

typedef struct TPM2B_MAX_BUFFER {
    UINT16 size;
    BYTE buffer[MAX_DIGEST_BUFFER];
} TPM2B_MAX_BUFFER;

typedef struct TPM2B_MAX_NV_BUFFER {
    UINT16 size;
    BYTE buffer[MAX_NV_BUFFER_SIZE];
} TPM2B_MAX_NV_BUFFER;


typedef TPM2B_DIGEST TPM2B_TIMEOUT;

typedef struct TPM2B_IV {
    UINT16 size;
    BYTE buffer[MAX_SYM_BLOCK_SIZE];
} TPM2B_IV;


/* Names */
typedef union TPMU_NAME {
    TPMT_HA digest;
```

```
    TPM_HANDLE handle;
} TPMU_NAME;

typedef struct TPM2B_NAME {
    UINT16 size;
    BYTE name[sizeof(TPMU_NAME)];
} TPM2B_NAME;


/* PCR */

typedef struct TPMS_PCR_SELECT {
    BYTE sizeofSelect;
    BYTE pcrSelect[PCR_SELECT_MIN];
} TPMS_PCR_SELECT;


typedef struct TPMS_PCR_SELECTION {
    TPMI_ALG_HASH hash;
    BYTE sizeofSelect;
    BYTE pcrSelect[PCR_SELECT_MIN];
} TPMS_PCR_SELECTION;


/* Tickets */

typedef struct TPMT_TK_CREATION {
    TPM_ST tag;
    TPMI_RH_HIERARCHY hierarchy;
    TPM2B_DIGEST digest;
} TPMT_TK_CREATION;

typedef struct TPMT_TK_VERIFIED {
    TPM_ST tag;
    TPMI_RH_HIERARCHY hierarchy;
    TPM2B_DIGEST digest;
} TPMT_TK_VERIFIED;

typedef struct TPMT_TK_AUTH {
    TPM_ST tag;
    TPMI_RH_HIERARCHY hierarchy;
    TPM2B_DIGEST digest;
} TPMT_TK_AUTH;

typedef struct TPMT_TK_HASHCHECK {
    TPM_ST tag;
    TPMI_RH_HIERARCHY hierarchy;
    TPM2B_DIGEST digest;
} TPMT_TK_HASHCHECK;

typedef struct TPMS_ALG_PROPERTY {
    TPM_ALG_ID alg;
    TPMA_ALGORITHM algProperties;
} TPMS_ALG_PROPERTY;
```

```
typedef struct TPMS_TAGGED_PROPERTY {
    TPM_PT property;
    UINT32 value;
} TPMS_TAGGED_PROPERTY;

typedef struct TPMS_TAGGED_PCR_SELECT {
    TPM_PT_PCR tag;
    BYTE sizeofSelect;
    BYTE pcrSelect[PCR_SELECT_MAX];
} TPMS_TAGGED_PCR_SELECT;

typedef struct TPMS_TAGGED_POLICY {
    TPM_HANDLE handle;
    TPMT_HA policyHash;
} TPMS_TAGGED_POLICY;


/* Lists */

typedef struct TPML_CC {
    UINT32 count;
    TPM_CC commandCodes[MAX_CAP_CC];
} TPML_CC;

typedef struct TPML_CCA {
    UINT32 count;
    TPMA_CC commandAttributes[MAX_CAP_CC];
} TPML_CCA;

typedef struct TPML_ALG {
    UINT32 count;
    TPM_ALG_ID algorithms[MAX_ALG_LIST_SIZE];
} TPML_ALG;

typedef struct TPML_HANDLE {
    UINT32 count;
    TPM_HANDLE handle[MAX_CAP_HANDLES];
} TPML_HANDLE;

typedef struct TPML_DIGEST {
    UINT32 count;
    TPM2B_DIGEST digests[8];
} TPML_DIGEST;

typedef struct TPML_DIGEST_VALUES {
    UINT32 count;
    TPMT_HA digests[HASH_COUNT];
} TPML_DIGEST_VALUES;

typedef struct TPML_PCR_SELECTION {
    UINT32 count;
    TPMS_PCR_SELECTION pcrSelections[HASH_COUNT];
} TPML_PCR_SELECTION;
```

```
typedef struct TPML_ALG_PROPERTY {
    UINT32 count;
    TPMS_ALG_PROPERTY algProperties[MAX_CAP_ALGS];
} TPML_ALG_PROPERTY;

typedef struct TPML_TAGGED_TPM_PROPERTY {
    UINT32 count;
    TPMS_TAGGED_PROPERTY tpmProperty[MAX_TPM_PROPERTIES];
} TPML_TAGGED_TPM_PROPERTY;

typedef struct TPML_TAGGED_PCR_PROPERTY {
    UINT32 count;
    TPMS_TAGGED_PCR_SELECT pcrProperty[MAX_PCR_PROPERTIES];
} TPML_TAGGED_PCR_PROPERTY;

typedef struct TPML_ECC_CURVE {
    UINT32 count;
    TPM_ECC_CURVE eccCurves[MAX_ECC_CURVES];
} TPML_ECC_CURVE;

typedef struct TPML_TAGGED_POLICY {
    UINT32 count;
    TPMS_TAGGED_POLICY policies[MAX_TAGGED_POLICIES];
} TPML_TAGGED_POLICY;


/* Capabilities Structures */

typedef union TPMU_CAPABILITIES {
    TPML_ALG_PROPERTY algorithms; /* TPM_CAP_ALGS */
    TPML_HANDLE handles; /* TPM_CAP_HANDLES */
    TPML_CCA command; /* TPM_CAP_COMMANDS */
    TPML_CC ppCommands; /* TPM_CAP_PP_COMMANDS */
    TPML_CC auditCommands; /* TPM_CAP_AUDIT_COMMANDS */
    TPML_PCR_SELECTION assignedPCR; /* TPM_CAP_PCRS */
    TPML_TAGGED_TPM_PROPERTY tpmProperties; /* TPM_CAP_TPM_PROPERTIES */
    TPML_TAGGED_PCR_PROPERTY pcrProperties; /* TPM_CAP_PCR_PROPERTIES */
    TPML_ECC_CURVE eccCurves; /* TPM_CAP_ECC_CURVES */
    TPML_TAGGED_POLICY authPolicies; /* TPM_CAP_AUTH_POLICIES */
} TPMU_CAPABILITIES;

typedef struct TPMS_CAPABILITY_DATA {
    TPM_CAP capability;
    TPMU_CAPABILITIES data;
} TPMS_CAPABILITY_DATA;

typedef struct TPMS_CLOCK_INFO {
    UINT64 clock;
    UINT32 resetCount;
    UINT32 restartCount;
    TPMI_YES_NO safe;
} TPMS_CLOCK_INFO;
```

```
typedef struct TPMS_TIME_INFO {
    UINT64 time;
    TPMS_CLOCK_INFO clockInfo;
} TPMS_TIME_INFO;

typedef struct TPMS_TIME_ATTEST_INFO {
    TPMS_TIME_INFO time;
    UINT64 firmwareVersion;
} TPMS_TIME_ATTEST_INFO;

typedef struct TPMS_CERTIFY_INFO {
    TPM2B_NAME name;
    TPM2B_NAME qualifiedName;
} TPMS_CERTIFY_INFO;

typedef struct TPMS_QUOTE_INFO {
    TPML_PCR_SELECTION pcrSelect;
    TPM2B_DIGEST pcrDigest;
} TPMS_QUOTE_INFO;

typedef struct TPMS_COMMAND_AUDIT_INFO {
    UINT64 auditCounter;
    TPM_ALG_ID digestAlg;
    TPM2B_DIGEST auditDigest;
    TPM2B_DIGEST commandDigest;
} TPMS_COMMAND_AUDIT_INFO;

typedef struct TPMS_SESSION_AUDIT_INFO {
    TPMI_YES_NO exclusiveSession;
    TPM2B_DIGEST sessionDigest;
} TPMS_SESSION_AUDIT_INFO;

typedef struct TPMS_CREATION_INFO {
    TPM2B_NAME objectName;
    TPM2B_DIGEST creationHash;
} TPMS_CREATION_INFO;

typedef struct TPMS_NV_CERTIFY_INFO {
    TPM2B_NAME indexName;
    UINT16 offset;
    TPM2B_MAX_NV_BUFFER nvContents;
} TPMS_NV_CERTIFY_INFO;


typedef TPM_ST TPMI_ST_ATTEST;
typedef union TPMU_ATTEST {
    TPMS_CERTIFY_INFO       certify;        /* TPM_ST_ATTEST_CERTIFY */
    TPMS_CREATION_INFO      creation;       /* TPM_ST_ATTEST_CREATION */
    TPMS_QUOTE_INFO         quote;          /* TPM_ST_ATTEST_QUOTE */
    TPMS_COMMAND_AUDIT_INFO commandAudit;   /* TPM_ST_ATTEST_COMMAND_AUDIT */
    TPMS_SESSION_AUDIT_INFO sessionAudit;   /* TPM_ST_ATTEST_SESSION_AUDIT */
    TPMS_TIME_ATTEST_INFO   time;           /* TPM_ST_ATTEST_TIME */
    TPMS_NV_CERTIFY_INFO    nv;             /* TPM_ST_ATTEST_NV */
} TPMU_ATTEST;
```

```
typedef struct TPMS_ATTEST {
    TPM_GENERATED magic;
    TPMI_ST_ATTEST type;
    TPM2B_NAME qualifiedSigner;
    TPM2B_DATA extraData;
    TPMS_CLOCK_INFO clockInfo;
    UINT64 firmwareVersion;
    TPMU_ATTEST attested;
} TPMS_ATTEST;

typedef struct TPM2B_ATTEST {
    UINT16 size;
    BYTE attestationData[sizeof(TPMS_ATTEST)];
} TPM2B_ATTEST;


/* Algorithm Parameters and Structures */

/* Symmetric */
typedef TPM_KEY_BITS TPMI_AES_KEY_BITS;

typedef union TPMU_SYM_KEY_BITS {
    TPMI_AES_KEY_BITS aes;
    TPM_KEY_BITS sym;
    TPMI_ALG_HASH xorr;
} TPMU_SYM_KEY_BITS;

typedef union TPMU_SYM_MODE {
    TPMI_ALG_SYM_MODE aes;
    TPMI_ALG_SYM_MODE sym;
} TPMU_SYM_MODE;

typedef struct TPMT_SYM_DEF {
    TPMI_ALG_SYM algorithm;
    TPMU_SYM_KEY_BITS keyBits;
    TPMU_SYM_MODE mode;
    //TPMU_SYM_DETAILS details;
} TPMT_SYM_DEF;

typedef TPMT_SYM_DEF TPMT_SYM_DEF_OBJECT;

typedef struct TPM2B_SYM_KEY {
    UINT16 size;
    BYTE buffer[MAX_SYM_KEY_BYTES];
} TPM2B_SYM_KEY;

typedef struct TPMS_SYMCIPHER_PARMS {
    TPMT_SYM_DEF_OBJECT sym;
} TPMS_SYMCIPHER_PARMS;

typedef struct TPM2B_LABEL {
    UINT16 size;
    BYTE buffer[LABEL_MAX_BUFFER];
```

```
} TPM2B_LABEL;

typedef struct TPMS_DERIVE {
    TPM2B_LABEL label;
    TPM2B_LABEL context;
} TPMS_DERIVE;

typedef struct TPM2B_DERIVE {
    UINT16 size;
    BYTE buffer[sizeof(TPMS_DERIVE)];
} TPM2B_DERIVE;

typedef union TPMU_SENSITIVE_CREATE {
    BYTE create[MAX_SYM_DATA];
    TPMS_DERIVE derive;
} TPMU_SENSITIVE_CREATE;

typedef struct TPM2B_SENSITIVE_DATA {
    UINT16 size;
    BYTE buffer[sizeof(TPMU_SENSITIVE_CREATE)];
} TPM2B_SENSITIVE_DATA;

typedef struct TPMS_SENSITIVE_CREATE {
    TPM2B_AUTH userAuth;
    TPM2B_SENSITIVE_DATA data;
} TPMS_SENSITIVE_CREATE;

typedef struct TPM2B_SENSITIVE_CREATE {
    UINT16 size;
    TPMS_SENSITIVE_CREATE sensitive;
} TPM2B_SENSITIVE_CREATE;

typedef struct TPMS_SCHEME_HASH {
    TPMI_ALG_HASH hashAlg;
} TPMS_SCHEME_HASH;

typedef struct TPMS_SCHEME_ECDAA {
    TPMI_ALG_HASH hashAlg;
    UINT16 count;
} TPMS_SCHEME_ECDAA;

typedef TPM_ALG_ID TPMI_ALG_KEYEDHASH_SCHEME;
typedef TPMS_SCHEME_HASH TPMS_SCHEME_HMAC;

typedef union TPMU_SCHEME_KEYEDHASH {
    TPMS_SCHEME_HMAC hmac;
} TPMU_SCHEME_KEYEDHASH;

typedef struct TPMT_KEYEDHASH_SCHEME {
    TPMI_ALG_KEYEDHASH_SCHEME scheme;
    TPMU_SCHEME_KEYEDHASH details;
} TPMT_KEYEDHASH_SCHEME;
```

```
/* Asymmetric */

typedef TPMS_SCHEME_HASH  TPMS_SIG_SCHEME_RSASSA;
typedef TPMS_SCHEME_HASH  TPMS_SIG_SCHEME_RSAPSS;
typedef TPMS_SCHEME_HASH  TPMS_SIG_SCHEME_ECDSA;

typedef TPMS_SCHEME_ECDAA TPMS_SIG_SCHEME_ECDAA;


typedef union TPMU_SIG_SCHEME {
    TPMS_SIG_SCHEME_RSASSA rsassa;
    TPMS_SIG_SCHEME_RSAPSS rsapss;
    TPMS_SIG_SCHEME_ECDSA  ecdsa;
    TPMS_SIG_SCHEME_ECDAA  ecdaa;
    TPMS_SCHEME_HMAC       hmac;
    TPMS_SCHEME_HASH       any;
} TPMU_SIG_SCHEME;

typedef struct TPMT_SIG_SCHEME {
    TPMI_ALG_SIG_SCHEME scheme;
    TPMU_SIG_SCHEME details;
} TPMT_SIG_SCHEME;


/* Encryption / Key Exchange Schemes */
typedef TPMS_SCHEME_HASH TPMS_ENC_SCHEME_OAEP;
typedef TPMS_SCHEME_HASH TPMS_KEY_SCHEME_ECDH;
typedef TPMS_SCHEME_HASH TPMS_KEY_SCHEME_ECMQV;

/* Key Derivation Schemes */
typedef TPMS_SCHEME_HASH TPMS_SCHEME_MGF1;
typedef TPMS_SCHEME_HASH TPMS_SCHEME_KDF1_SP800_56A;
typedef TPMS_SCHEME_HASH TPMS_SCHEME_KDF2;
typedef TPMS_SCHEME_HASH TPMS_SCHEME_KDF1_SP800_108;

typedef union TPMU_KDF_SCHEME {
    TPMS_SCHEME_MGF1            mgf1;
    TPMS_SCHEME_KDF1_SP800_56A  kdf1_sp800_56a;
    TPMS_SCHEME_KDF2            kdf2;
    TPMS_SCHEME_KDF1_SP800_108  kdf1_sp800_108;
    TPMS_SCHEME_HASH            any;
} TPMU_KDF_SCHEME;

typedef struct TPMT_KDF_SCHEME {
    TPMI_ALG_KDF scheme;
    TPMU_KDF_SCHEME details;
} TPMT_KDF_SCHEME;

typedef TPM_ALG_ID TPMI_ALG_ASYM_SCHEME;
typedef union TPMU_ASYM_SCHEME {
    TPMS_KEY_SCHEME_ECDH    ecdh;
    TPMS_SIG_SCHEME_RSASSA  rsassa;
    TPMS_SIG_SCHEME_RSAPSS  rsapss;
    TPMS_SIG_SCHEME_ECDSA   ecdsa;
```

```
    TPMS_ENC_SCHEME_OAEP    oaep;
    TPMS_SCHEME_HASH        anySig;
} TPMU_ASYM_SCHEME;

typedef struct TPMT_ASYM_SCHEME {
    TPMI_ALG_ASYM_SCHEME scheme;
    TPMU_ASYM_SCHEME details;
} TPMT_ASYM_SCHEME;

/* RSA */
typedef TPM_ALG_ID TPMI_ALG_RSA_SCHEME;
typedef struct TPMT_RSA_SCHEME {
    TPMI_ALG_RSA_SCHEME scheme;
    TPMU_ASYM_SCHEME details;
} TPMT_RSA_SCHEME;

typedef TPM_ALG_ID TPMI_ALG_RSA_DECRYPT;
typedef struct TPMT_RSA_DECRYPT {
    TPMI_ALG_RSA_DECRYPT scheme;
    TPMU_ASYM_SCHEME details;
} TPMT_RSA_DECRYPT;

typedef struct TPM2B_PUBLIC_KEY_RSA {
    UINT16 size;
    BYTE buffer[MAX_RSA_KEY_BYTES];
} TPM2B_PUBLIC_KEY_RSA;

typedef TPM_KEY_BITS TPMI_RSA_KEY_BITS;
typedef struct TPM2B_PRIVATE_KEY_RSA {
    UINT16 size;
    BYTE buffer[MAX_RSA_KEY_BYTES/2];
} TPM2B_PRIVATE_KEY_RSA;


/* ECC */
typedef struct TPM2B_ECC_PARAMETER {
    UINT16 size;
    BYTE buffer[MAX_ECC_KEY_BYTES];
} TPM2B_ECC_PARAMETER;

typedef struct TPMS_ECC_POINT {
    TPM2B_ECC_PARAMETER x;
    TPM2B_ECC_PARAMETER y;
} TPMS_ECC_POINT;

typedef struct TPM2B_ECC_POINT {
    UINT16 size;
    TPMS_ECC_POINT point;
} TPM2B_ECC_POINT;


typedef TPM_ALG_ID TPMI_ALG_ECC_SCHEME;
typedef TPM_ECC_CURVE TPMI_ECC_CURVE;
typedef TPMT_SIG_SCHEME TPMT_ECC_SCHEME;
```

```
typedef struct TPMS_ALGORITHM_DETAIL_ECC {
    TPM_ECC_CURVE curveID;
    UINT16 keySize;
    TPMT_KDF_SCHEME kdf;
    TPMT_ECC_SCHEME sign;
    TPM2B_ECC_PARAMETER p;
    TPM2B_ECC_PARAMETER a;
    TPM2B_ECC_PARAMETER b;
    TPM2B_ECC_PARAMETER gX;
    TPM2B_ECC_PARAMETER gY;
    TPM2B_ECC_PARAMETER n;
    TPM2B_ECC_PARAMETER h;
} TPMS_ALGORITHM_DETAIL_ECC;


/* Signatures */

typedef struct TPMS_SIGNATURE_RSA {
    TPMI_ALG_HASH hash;
    TPM2B_PUBLIC_KEY_RSA sig;
} TPMS_SIGNATURE_RSA;

typedef TPMS_SIGNATURE_RSA TPMS_SIGNATURE_RSASSA;
typedef TPMS_SIGNATURE_RSA TPMS_SIGNATURE_RSAPSS;

typedef struct TPMS_SIGNATURE_ECC {
    TPMI_ALG_HASH hash;
    TPM2B_ECC_PARAMETER signatureR;
    TPM2B_ECC_PARAMETER signatureS;
} TPMS_SIGNATURE_ECC;

typedef TPMS_SIGNATURE_ECC TPMS_SIGNATURE_ECDSA;
typedef TPMS_SIGNATURE_ECC TPMS_SIGNATURE_ECDAA;

typedef union TPMU_SIGNATURE {
    TPMS_SIGNATURE_ECDSA ecdsa;
    TPMS_SIGNATURE_ECDAA ecdaa;
    TPMS_SIGNATURE_RSASSA rsassa;
    TPMS_SIGNATURE_RSAPSS rsapss;
    TPMT_HA hmac;
    TPMS_SCHEME_HASH any;
} TPMU_SIGNATURE;

typedef struct TPMT_SIGNATURE {
    TPMI_ALG_SIG_SCHEME sigAlg;
    TPMU_SIGNATURE signature;
} TPMT_SIGNATURE;


/* Key/Secret Exchange */

typedef union TPMU_ENCRYPTED_SECRET {
    BYTE ecc[sizeof(TPMS_ECC_POINT)];      /* TPM_ALG_ECC */
```

```
    BYTE rsa[MAX_RSA_KEY_BYTES];          /* TPM_ALG_RSA */
    BYTE symmetric[sizeof(TPM2B_DIGEST)]; /* TPM_ALG_SYMCIPHER */
    BYTE keyedHash[sizeof(TPM2B_DIGEST)]; /* TPM_ALG_KEYEDHASH */
} TPMU_ENCRYPTED_SECRET;

typedef struct TPM2B_ENCRYPTED_SECRET {
    UINT16 size;
    BYTE secret[sizeof(TPMU_ENCRYPTED_SECRET)];
} TPM2B_ENCRYPTED_SECRET;


/* Key/Object Complex */

typedef TPM_ALG_ID TPMI_ALG_PUBLIC;

typedef union TPMU_PUBLIC_ID {
    TPM2B_DIGEST keyedHash;   /* TPM_ALG_KEYEDHASH */
    TPM2B_DIGEST sym;         /* TPM_ALG_SYMCIPHER */
    TPM2B_PUBLIC_KEY_RSA rsa; /* TPM_ALG_RSA */
    TPMS_ECC_POINT ecc;       /* TPM_ALG_ECC */
    TPMS_DERIVE derive;
} TPMU_PUBLIC_ID;

typedef struct TPMS_KEYEDHASH_PARMS {
    TPMT_KEYEDHASH_SCHEME scheme;
} TPMS_KEYEDHASH_PARMS;


typedef struct TPMS_ASYM_PARMS {
    TPMT_SYM_DEF_OBJECT symmetric;
    TPMT_ASYM_SCHEME scheme;
} TPMS_ASYM_PARMS;

typedef struct TPMS_RSA_PARMS {
    TPMT_SYM_DEF_OBJECT symmetric;
    TPMT_RSA_SCHEME scheme;
    TPMI_RSA_KEY_BITS keyBits;
    UINT32 exponent;
} TPMS_RSA_PARMS;

typedef struct TPMS_ECC_PARMS {
    TPMT_SYM_DEF_OBJECT symmetric;
    TPMT_ECC_SCHEME scheme;
    TPMI_ECC_CURVE curveID;
    TPMT_KDF_SCHEME kdf;
} TPMS_ECC_PARMS;

typedef union TPMU_PUBLIC_PARMS {
    TPMS_KEYEDHASH_PARMS keyedHashDetail;
    TPMS_SYMCIPHER_PARMS symDetail;
    TPMS_RSA_PARMS rsaDetail;
    TPMS_ECC_PARMS eccDetail;
    TPMS_ASYM_PARMS asymDetail;
} TPMU_PUBLIC_PARMS;
```

```
typedef struct TPMT_PUBLIC_PARMS {
    TPMI_ALG_PUBLIC type;
    TPMU_PUBLIC_PARMS parameters;
} TPMT_PUBLIC_PARMS;


typedef struct TPMT_PUBLIC {
    TPMI_ALG_PUBLIC type;
    TPMI_ALG_HASH nameAlg;
    TPMA_OBJECT objectAttributes;
    TPM2B_DIGEST authPolicy;
    TPMU_PUBLIC_PARMS parameters;
    TPMU_PUBLIC_ID unique;
} TPMT_PUBLIC;

typedef struct TPM2B_PUBLIC {
    UINT16 size;
    TPMT_PUBLIC publicArea;
} TPM2B_PUBLIC;

typedef struct TPM2B_TEMPLATE {
    UINT16 size;
    BYTE buffer[sizeof(TPMT_PUBLIC)];
} TPM2B_TEMPLATE;


/* Private Structures */

typedef struct TPM2B_PRIVATE_VENDOR_SPECIFIC {
    UINT16 size;
    BYTE buffer[PRIVATE_VENDOR_SPECIFIC_BYTES];
} TPM2B_PRIVATE_VENDOR_SPECIFIC;

typedef union TPMU_SENSITIVE_COMPOSITE {
    TPM2B_PRIVATE_KEY_RSA rsa;  /* TPM_ALG_RSA */
    TPM2B_ECC_PARAMETER ecc;    /* TPM_ALG_ECC */
    TPM2B_SENSITIVE_DATA bits;  /* TPM_ALG_KEYEDHASH */
    TPM2B_SYM_KEY sym;          /* TPM_ALG_SYMCIPHER */
    TPM2B_PRIVATE_VENDOR_SPECIFIC any;
} TPMU_SENSITIVE_COMPOSITE;


typedef struct TPMT_SENSITIVE {
    TPMI_ALG_PUBLIC sensitiveType;
    TPM2B_AUTH authValue;
    TPM2B_DIGEST seedValue;
    TPMU_SENSITIVE_COMPOSITE sensitive;
} TPMT_SENSITIVE;

typedef struct TPM2B_SENSITIVE {
    UINT16 size;
    TPMT_SENSITIVE sensitiveArea;
} TPM2B_SENSITIVE;
```

```
typedef struct TPMT_PRIVATE {
    TPM2B_DIGEST integrityOuter;
    TPM2B_DIGEST integrityInner;
    TPM2B_SENSITIVE sensitive;
} TPMT_PRIVATE;

typedef struct TPM2B_PRIVATE {
    UINT16 size;
    BYTE buffer[sizeof(TPMT_PRIVATE)];
} TPM2B_PRIVATE;


/* Identity Object */

typedef struct TPMS_ID_OBJECT {
    TPM2B_DIGEST integrityHMAC;
    TPM2B_DIGEST encIdentity;
} TPMS_ID_OBJECT;

typedef struct TPM2B_ID_OBJECT {
    UINT16 size;
    BYTE buffer[sizeof(TPMS_ID_OBJECT)];
} TPM2B_ID_OBJECT;


/* NV Storage Structures */

typedef UINT32 TPM_NV_INDEX;
enum TPM_NV_INDEX_mask {
    TPM_NV_INDEX_index = 0x00FFFFFF,
    TPM_NV_INDEX_RH_NV = 0xFF000000,
};

typedef enum TPM_NT {
    TPM_NT_ORDINARY = 0x0,
    TPM_NT_COUNTER  = 0x1,
    TPM_NT_BITS     = 0x2,
    TPM_NT_EXTEND   = 0x4,
    TPM_NT_PIN_FAIL = 0x8,
    TPM_NT_PIN_PASS = 0x9,
} TPM_NT;


typedef struct TPMS_NV_PIN_COUNTER_PARAMETERS {
    UINT32 pinCount;
    UINT32 pinLimit;
} TPMS_NV_PIN_COUNTER_PARAMETERS;

typedef UINT32 TPMA_NV;
enum TPMA_NV_mask {
    TPMA_NV_PPWRITE         = 0x00000001,
    TPMA_NV_OWNERWRITE      = 0x00000002,
```

```
    TPMA_NV_AUTHWRITE       = 0x00000004,
    TPMA_NV_POLICYWRITE     = 0x00000008,
    TPMA_NV_TPM_NT          = 0x000000F0,
    TPMA_NV_POLICY_DELETE   = 0x00000400,
    TPMA_NV_WRITELOCKED     = 0x00000800,
    TPMA_NV_WRITEALL        = 0x00001000,
    TPMA_NV_WRITEDEFINE     = 0x00002000,
    TPMA_NV_WRITE_STCLEAR   = 0x00004000,
    TPMA_NV_GLOBALLOCK      = 0x00008000,
    TPMA_NV_PPREAD          = 0x00010000,
    TPMA_NV_OWNERREAD       = 0x00020000,
    TPMA_NV_AUTHREAD        = 0x00040000,
    TPMA_NV_POLICYREAD      = 0x00080000,
    TPMA_NV_NO_DA           = 0x02000000,
    TPMA_NV_ORDERLY         = 0x04000000,
    TPMA_NV_CLEAR_STCLEAR   = 0x08000000,
    TPMA_NV_READLOCKED      = 0x10000000,
    TPMA_NV_WRITTEN         = 0x20000000,
    TPMA_NV_PLATFORMCREATE  = 0x40000000,
    TPMA_NV_READ_STCLEAR    = 0x80000000,
};

typedef struct TPMS_NV_PUBLIC {
    TPMI_RH_NV_INDEX nvIndex;
    TPMI_ALG_HASH nameAlg;
    TPMA_NV attributes;
    TPM2B_DIGEST authPolicy;
    UINT16 dataSize;
} TPMS_NV_PUBLIC;

typedef struct TPM2B_NV_PUBLIC {
    UINT16 size;
    TPMS_NV_PUBLIC nvPublic;
} TPM2B_NV_PUBLIC;


/* Context Data */

typedef struct TPM2B_CONTEXT_SENSITIVE {
    UINT16 size;
    BYTE buffer[MAX_CONTEXT_SIZE];
} TPM2B_CONTEXT_SENSITIVE;

typedef struct TPMS_CONTEXT_DATA {
    TPM2B_DIGEST integrity;
    TPM2B_CONTEXT_SENSITIVE encrypted;
} TPMS_CONTEXT_DATA;

typedef struct TPM2B_CONTEXT_DATA {
    UINT16 size;
    BYTE buffer[sizeof(TPMS_CONTEXT_DATA)];
} TPM2B_CONTEXT_DATA;

typedef struct TPMS_CONTEXT {
```

```
    UINT64 sequence;
    TPMI_DH_CONTEXT savedHandle;
    TPMI_RH_HIERARCHY hierarchy;
    TPM2B_CONTEXT_DATA contextBlob;
} TPMS_CONTEXT;


typedef struct TPMS_CREATION_DATA {
    TPML_PCR_SELECTION pcrSelect;
    TPM2B_DIGEST pcrDigest;
    TPMA_LOCALITY locality;
    TPM_ALG_ID parentNameAlg;
    TPM2B_NAME parentName;
    TPM2B_NAME parentQualifiedName;
    TPM2B_DATA outsideInfo;
} TPMS_CREATION_DATA;

typedef struct TPM2B_CREATION_DATA {
    UINT16 size;
    TPMS_CREATION_DATA creationData;
} TPM2B_CREATION_DATA;


/* Authorization Structures */
typedef struct TPMS_AUTH_COMMAND {
    TPMI_SH_AUTH_SESSION sessionHandle;
    TPM2B_NONCE nonce; /* nonceCaller */
    TPMA_SESSION sessionAttributes;
    TPM2B_AUTH hmac;
} TPMS_AUTH_COMMAND;

typedef struct TPMS_AUTH_RESPONSE {
    TPM2B_NONCE nonce;
    TPMA_SESSION sessionAttributes;
    TPM2B_AUTH hmac;
} TPMS_AUTH_RESPONSE;

/* Implementation specific authorization session information */
typedef struct TPM2_AUTH_SESSION {
    /* BEGIN */
    /* This section should match TPMS_AUTH_COMMAND */
    TPMI_SH_AUTH_SESSION sessionHandle;
    TPM2B_NONCE nonceCaller;
    TPMA_SESSION sessionAttributes;
    TPM2B_AUTH auth;
    /* END */

    /* additional auth data required for implementation */
    TPM2B_NONCE nonceTPM;
    TPMT_SYM_DEF symmetric;
    TPMI_ALG_HASH authHash;
    TPM2B_NAME name;
} TPM2_AUTH_SESSION;
```

```
/* Macros to determine TPM 2.0 Session type */
#define TPM2_IS_PWD_SESSION(sessionHandle) ((sessionHandle) == TPM_RS_PW)
#define TPM2_IS_HMAC_SESSION(sessionHandle) ((sessionHandle & 0xFF000000) ==
 ↪  HMAC_SESSION_FIRST)
#define TPM2_IS_POLICY_SESSION(sessionHandle) ((sessionHandle & 0xFF000000) ==
 ↪  POLICY_SESSION_FIRST)

/* Predetermined TPM 2.0 Indexes */
#define TPM_20_TPM_MFG_NV_SPACE        ((TPM_HT_NV_INDEX << 24) | (0x00 << 22))
#define TPM_20_PLATFORM_MFG_NV_SPACE   ((TPM_HT_NV_INDEX << 24) | (0x01 << 22))
#define TPM_20_OWNER_NV_SPACE          ((TPM_HT_NV_INDEX << 24) | (0x02 << 22))
#define TPM_20_TCG_NV_SPACE            ((TPM_HT_NV_INDEX << 24) | (0x03 << 22))

#define TPM_20_NV_INDEX_EK_CERTIFICATE (TPM_20_PLATFORM_MFG_NV_SPACE + 2)
#define TPM_20_NV_INDEX_EK_NONCE       (TPM_20_PLATFORM_MFG_NV_SPACE + 3)
#define TPM_20_NV_INDEX_EK_TEMPLATE    (TPM_20_PLATFORM_MFG_NV_SPACE + 4)

/* Predetermined TPM 2.0 Endorsement policy auth template */
static const BYTE TPM_20_EK_AUTH_POLICY[] = {
    0x83, 0x71, 0x97, 0x67, 0x44, 0x84, 0xb3, 0xf8, 0x1a, 0x90, 0xcc,
    0x8d, 0x46, 0xa5, 0xd7, 0x24, 0xfd, 0x52, 0xd7, 0x6e, 0x06, 0x52,
    0x0b, 0x64, 0xf2, 0xa1, 0xda, 0x1b, 0x33, 0x14, 0x69, 0xaa,
};



/* HAL IO Callbacks */
struct TPM2_CTX;

#ifdef WOLFTPM_SWTPM
struct wolfTPM_tcpContext {
    int fd;
};
#endif /* WOLFTPM_SWTPM */

#ifdef WOLFTPM_WINAPI
#include <tbs.h>
#include <winerror.h>

struct wolfTPM_winContext {
  TBS_HCONTEXT tbs_context;
};
/* may be needed with msys */
#ifndef TPM_E_COMMAND_BLOCKED
#define TPM_E_COMMAND_BLOCKED (0x80284000)
#endif

#define WOLFTPM_IS_COMMAND_UNAVAILABLE(code) ((code) == TPM_RC_COMMAND_CODE ||
 ↪  (code) == TPM_E_COMMAND_BLOCKED)
#else
#define WOLFTPM_IS_COMMAND_UNAVAILABLE(code) (code == TPM_RC_COMMAND_CODE)
#endif /* WOLFTPM_WINAPI */

/* make sure advanced IO is enabled for I2C */
```

```
#ifdef WOLFTPM_I2C
    #undef  WOLFTPM_ADV_IO
    #define WOLFTPM_ADV_IO
#endif

#ifdef WOLFTPM_ADV_IO
typedef int (*TPM2HalIoCb)(struct TPM2_CTX*, INT32 isRead, UINT32 addr,
    BYTE* xferBuf, UINT16 xferSz, void* userCtx);
#else
typedef int (*TPM2HalIoCb)(struct TPM2_CTX*, const BYTE* txBuf, BYTE* rxBuf,
    UINT16 xferSz, void* userCtx);
#endif

#if !defined(WOLFTPM2_NO_WOLFCRYPT) && !defined(WC_NO_RNG) && \
    !defined(WOLFTPM2_USE_HW_RNG)
    #define WOLFTPM2_USE_WOLF_RNG
#endif

typedef struct TPM2_CTX {
    TPM2HalIoCb ioCb;
    void* userCtx;
#ifdef WOLFTPM_SWTPM
    struct wolfTPM_tcpContext tcpCtx;
#endif
#ifdef WOLFTPM_WINAPI
    struct wolfTPM_winContext winCtx;
#endif
#ifndef WOLFTPM2_NO_WOLFCRYPT
#ifndef SINGLE_THREADED
    wolfSSL_Mutex hwLock;
    int lockCount;
#endif
    #ifdef WOLFTPM2_USE_WOLF_RNG
    WC_RNG rng;
    #endif
#endif /* !WOLFTPM2_NO_WOLFCRYPT */

    /* TPM TIS Info */
    int locality;
    word32 caps;
    word32 did_vid;

    /* Pointer to current TPM auth sessions */
    TPM2_AUTH_SESSION* session;

    /* Command / Response Buffer */
    byte cmdBuf[MAX_COMMAND_SIZE];

    byte rid;
    /* Informational Bits - use unsigned int for best compiler compatibility */
#ifndef WOLFTPM2_NO_WOLFCRYPT
    #ifndef SINGLE_THREADED
    unsigned int hwLockInit:1;
    #endif
```

```
    #ifndef WC_NO_RNG
    unsigned int rngInit:1;
    #endif
#endif
} TPM2_CTX;


/* TPM Specification Functions */
typedef struct {
    TPM_SU startupType;
} Startup_In;
WOLFTPM_API TPM_RC TPM2_Startup(Startup_In* in);

typedef struct {
    TPM_SU shutdownType;
} Shutdown_In;
WOLFTPM_API TPM_RC TPM2_Shutdown(Shutdown_In* in);


typedef struct {
    TPM_CAP capability;
    UINT32 property;
    UINT32 propertyCount;
} GetCapability_In;
typedef struct {
    TPMI_YES_NO moreData;
    TPMS_CAPABILITY_DATA capabilityData;
} GetCapability_Out;
WOLFTPM_API TPM_RC TPM2_GetCapability(GetCapability_In* in,
    GetCapability_Out* out);


typedef struct {
    TPMI_YES_NO fullTest;
} SelfTest_In;
WOLFTPM_API TPM_RC TPM2_SelfTest(SelfTest_In* in);

typedef struct {
    TPML_ALG toTest;
} IncrementalSelfTest_In;
typedef struct {
    TPML_ALG toDoList;
} IncrementalSelfTest_Out;
WOLFTPM_API TPM_RC TPM2_IncrementalSelfTest(IncrementalSelfTest_In* in,
    IncrementalSelfTest_Out* out);

typedef struct {
    TPM2B_MAX_BUFFER outData;
    UINT16 testResult; /* TPM_RC */
} GetTestResult_Out;
WOLFTPM_API TPM_RC TPM2_GetTestResult(GetTestResult_Out* out);


typedef struct {
```

```
    UINT16 bytesRequested;
} GetRandom_In;
typedef struct {
    TPM2B_DIGEST randomBytes; /* hardware max is 32-bytes */
} GetRandom_Out;
WOLFTPM_API TPM_RC TPM2_GetRandom(GetRandom_In* in, GetRandom_Out* out);

typedef struct {
    TPM2B_SENSITIVE_DATA inData;
} StirRandom_In;
WOLFTPM_API TPM_RC TPM2_StirRandom(StirRandom_In* in);

typedef struct {
    TPML_PCR_SELECTION pcrSelectionIn;
} PCR_Read_In;
typedef struct {
    UINT32 pcrUpdateCounter;
    TPML_PCR_SELECTION pcrSelectionOut;
    TPML_DIGEST pcrValues;
} PCR_Read_Out;
WOLFTPM_API TPM_RC TPM2_PCR_Read(PCR_Read_In* in, PCR_Read_Out* out);


typedef struct {
    TPMI_DH_PCR pcrHandle;
    TPML_DIGEST_VALUES digests;
} PCR_Extend_In;
WOLFTPM_API TPM_RC TPM2_PCR_Extend(PCR_Extend_In* in);


typedef struct {
    TPMI_DH_OBJECT parentHandle;
    TPM2B_SENSITIVE_CREATE inSensitive;
    TPM2B_PUBLIC inPublic;
    TPM2B_DATA outsideInfo;
    TPML_PCR_SELECTION creationPCR;
} Create_In;
typedef struct {
    TPM2B_PRIVATE outPrivate;
    TPM2B_PUBLIC outPublic;
    TPM2B_CREATION_DATA creationData;
    TPM2B_DIGEST creationHash;
    TPMT_TK_CREATION creationTicket;
} Create_Out;
WOLFTPM_API TPM_RC TPM2_Create(Create_In* in, Create_Out* out);

typedef struct {
    TPMI_DH_OBJECT parentHandle;
    TPM2B_SENSITIVE_CREATE inSensitive;
    TPM2B_PUBLIC inPublic;
} CreateLoaded_In;
typedef struct {
    TPM_HANDLE objectHandle;
    TPM2B_PRIVATE outPrivate;
```

```
    TPM2B_PUBLIC outPublic;
    TPM2B_NAME name;
} CreateLoaded_Out;
WOLFTPM_API TPM_RC TPM2_CreateLoaded(CreateLoaded_In* in,
    CreateLoaded_Out* out);


typedef struct {
    TPMI_RH_HIERARCHY primaryHandle;
    TPM2B_SENSITIVE_CREATE inSensitive;
    TPM2B_PUBLIC inPublic;
    TPM2B_DATA outsideInfo;
    TPML_PCR_SELECTION creationPCR;
} CreatePrimary_In;
typedef struct {
    TPM_HANDLE objectHandle;
    TPM2B_PUBLIC outPublic;
    TPM2B_CREATION_DATA creationData;
    TPM2B_DIGEST creationHash;
    TPMT_TK_CREATION creationTicket;
    TPM2B_NAME name;
} CreatePrimary_Out;
WOLFTPM_API TPM_RC TPM2_CreatePrimary(CreatePrimary_In* in,
    CreatePrimary_Out* out);

typedef struct {
    TPMI_DH_OBJECT parentHandle;
    TPM2B_PRIVATE inPrivate;
    TPM2B_PUBLIC inPublic;
} Load_In;
typedef struct {
    TPM_HANDLE objectHandle;
    TPM2B_NAME name;
} Load_Out;
WOLFTPM_API TPM_RC TPM2_Load(Load_In* in, Load_Out* out);


typedef struct {
    TPMI_DH_CONTEXT flushHandle;
} FlushContext_In;
WOLFTPM_API TPM_RC TPM2_FlushContext(FlushContext_In* in);


typedef struct {
    TPMI_DH_OBJECT itemHandle;
} Unseal_In;
typedef struct {
    TPM2B_SENSITIVE_DATA outData;
} Unseal_Out;
WOLFTPM_API TPM_RC TPM2_Unseal(Unseal_In* in, Unseal_Out* out);


typedef struct {
    TPMI_DH_OBJECT tpmKey;
```

```
    TPMI_DH_ENTITY bind;
    TPM2B_NONCE nonceCaller;
    TPM2B_ENCRYPTED_SECRET encryptedSalt;
    TPM_SE sessionType;
    TPMT_SYM_DEF symmetric;
    TPMI_ALG_HASH authHash;
} StartAuthSession_In;
typedef struct {
    TPMI_SH_AUTH_SESSION sessionHandle;
    TPM2B_NONCE nonceTPM;
} StartAuthSession_Out;
WOLFTPM_API TPM_RC TPM2_StartAuthSession(StartAuthSession_In* in,
    StartAuthSession_Out* out);

typedef struct {
    TPMI_SH_POLICY sessionHandle;
} PolicyRestart_In;
WOLFTPM_API TPM_RC TPM2_PolicyRestart(PolicyRestart_In* in);


typedef struct {
    TPM2B_SENSITIVE inPrivate;
    TPM2B_PUBLIC inPublic;
    TPMI_RH_HIERARCHY hierarchy;
} LoadExternal_In;
typedef struct {
    TPM_HANDLE objectHandle;
    TPM2B_NAME name;
} LoadExternal_Out;
WOLFTPM_API TPM_RC TPM2_LoadExternal(LoadExternal_In* in,
    LoadExternal_Out* out);

typedef struct {
    TPMI_DH_OBJECT objectHandle;
} ReadPublic_In;
typedef struct {
    TPM2B_PUBLIC outPublic;
    TPM2B_NAME name;
    TPM2B_NAME qualifiedName;
} ReadPublic_Out;
WOLFTPM_API TPM_RC TPM2_ReadPublic(ReadPublic_In* in, ReadPublic_Out* out);

typedef struct {
    TPMI_DH_OBJECT activateHandle;
    TPMI_DH_OBJECT keyHandle;
    TPM2B_ID_OBJECT credentialBlob;
    TPM2B_ENCRYPTED_SECRET secret;
} ActivateCredential_In;
typedef struct {
    TPM2B_DIGEST certInfo;
} ActivateCredential_Out;
WOLFTPM_API TPM_RC TPM2_ActivateCredential(ActivateCredential_In* in,
    ActivateCredential_Out* out);
```

```
typedef struct {
    TPMI_DH_OBJECT handle;
    TPM2B_DIGEST credential;
    TPM2B_NAME objectName;
} MakeCredential_In;
typedef struct {
    TPM2B_ID_OBJECT credentialBlob;
    TPM2B_ENCRYPTED_SECRET secret;
} MakeCredential_Out;
WOLFTPM_API TPM_RC TPM2_MakeCredential(MakeCredential_In* in,
    MakeCredential_Out* out);

typedef struct {
    TPMI_DH_OBJECT objectHandle;
    TPMI_DH_OBJECT parentHandle;
    TPM2B_AUTH newAuth;
} ObjectChangeAuth_In;
typedef struct {
    TPM2B_PRIVATE outPrivate;
} ObjectChangeAuth_Out;
WOLFTPM_API TPM_RC TPM2_ObjectChangeAuth(ObjectChangeAuth_In* in,
    ObjectChangeAuth_Out* out);


typedef struct {
    TPMI_DH_OBJECT objectHandle;
    TPMI_DH_OBJECT newParentHandle;
    TPM2B_DATA encryptionKeyIn;
    TPMT_SYM_DEF_OBJECT symmetricAlg;
} Duplicate_In;
typedef struct {
    TPM2B_DATA encryptionKeyOut;
    TPM2B_PRIVATE duplicate;
    TPM2B_ENCRYPTED_SECRET outSymSeed;
} Duplicate_Out;
WOLFTPM_API TPM_RC TPM2_Duplicate(Duplicate_In* in, Duplicate_Out* out);

typedef struct {
    TPMI_DH_OBJECT oldParent;
    TPMI_DH_OBJECT newParent;
    TPM2B_PRIVATE inDuplicate;
    TPM2B_NAME name;
    TPM2B_ENCRYPTED_SECRET inSymSeed;
} Rewrap_In;
typedef struct {
    TPM2B_PRIVATE outDuplicate;
    TPM2B_ENCRYPTED_SECRET outSymSeed;
} Rewrap_Out;
WOLFTPM_API TPM_RC TPM2_Rewrap(Rewrap_In* in, Rewrap_Out* out);

typedef struct {
    TPMI_DH_OBJECT parentHandle;
    TPM2B_DATA encryptionKey;
    TPM2B_PUBLIC objectPublic;
```

```
    TPM2B_PRIVATE duplicate;
    TPM2B_ENCRYPTED_SECRET inSymSeed;
    TPMT_SYM_DEF_OBJECT symmetricAlg;
} Import_In;
typedef struct {
    TPM2B_PRIVATE outPrivate;
} Import_Out;
WOLFTPM_API TPM_RC TPM2_Import(Import_In* in, Import_Out* out);

typedef struct {
    TPMI_DH_OBJECT keyHandle;
    TPM2B_PUBLIC_KEY_RSA message;
    TPMT_RSA_DECRYPT inScheme;
    TPM2B_DATA label;
} RSA_Encrypt_In;
typedef struct {
    TPM2B_PUBLIC_KEY_RSA outData;
} RSA_Encrypt_Out;
WOLFTPM_API TPM_RC TPM2_RSA_Encrypt(RSA_Encrypt_In* in, RSA_Encrypt_Out* out);


typedef struct {
    TPMI_DH_OBJECT keyHandle;
    TPM2B_PUBLIC_KEY_RSA cipherText;
    TPMT_RSA_DECRYPT inScheme;
    TPM2B_DATA label;
} RSA_Decrypt_In;
typedef struct {
    TPM2B_PUBLIC_KEY_RSA message;
} RSA_Decrypt_Out;
WOLFTPM_API TPM_RC TPM2_RSA_Decrypt(RSA_Decrypt_In* in, RSA_Decrypt_Out* out);


typedef struct {
    TPMI_DH_OBJECT keyHandle;
} ECDH_KeyGen_In;
typedef struct {
    TPM2B_ECC_POINT zPoint;
    TPM2B_ECC_POINT pubPoint;
} ECDH_KeyGen_Out;
WOLFTPM_API TPM_RC TPM2_ECDH_KeyGen(ECDH_KeyGen_In* in, ECDH_KeyGen_Out* out);


typedef struct {
    TPMI_DH_OBJECT keyHandle;
    TPM2B_ECC_POINT inPoint;
} ECDH_ZGen_In;
typedef struct {
    TPM2B_ECC_POINT outPoint;
} ECDH_ZGen_Out;
WOLFTPM_API TPM_RC TPM2_ECDH_ZGen(ECDH_ZGen_In* in, ECDH_ZGen_Out* out);

typedef struct {
    TPMI_ECC_CURVE curveID;
```

```
} ECC_Parameters_In;
typedef struct {
    TPMS_ALGORITHM_DETAIL_ECC parameters;
} ECC_Parameters_Out;
WOLFTPM_API TPM_RC TPM2_ECC_Parameters(ECC_Parameters_In* in,
    ECC_Parameters_Out* out);

typedef struct {
    TPMI_DH_OBJECT keyA;
    TPM2B_ECC_POINT inQsB;
    TPM2B_ECC_POINT inQeB;
    TPMI_ECC_KEY_EXCHANGE inScheme;
    UINT16 counter;
} ZGen_2Phase_In;
typedef struct {
    TPM2B_ECC_POINT outZ1;
    TPM2B_ECC_POINT outZ2;
} ZGen_2Phase_Out;
WOLFTPM_API TPM_RC TPM2_ZGen_2Phase(ZGen_2Phase_In* in, ZGen_2Phase_Out* out);


typedef struct {
    TPMI_DH_OBJECT keyHandle;
    TPMI_YES_NO decrypt;
    TPMI_ALG_SYM_MODE mode;
    TPM2B_IV ivIn;
    TPM2B_MAX_BUFFER inData;
} EncryptDecrypt_In;
typedef struct {
    TPM2B_MAX_BUFFER outData;
    TPM2B_IV ivOut;
} EncryptDecrypt_Out;
WOLFTPM_API TPM_RC TPM2_EncryptDecrypt(EncryptDecrypt_In* in,
    EncryptDecrypt_Out* out);

typedef struct {
    TPMI_DH_OBJECT keyHandle;
    TPM2B_MAX_BUFFER inData;
    TPMI_YES_NO decrypt;
    TPMI_ALG_SYM_MODE mode;
    TPM2B_IV ivIn;
} EncryptDecrypt2_In;
typedef struct {
    TPM2B_MAX_BUFFER outData;
    TPM2B_IV ivOut;
} EncryptDecrypt2_Out;
WOLFTPM_API TPM_RC TPM2_EncryptDecrypt2(EncryptDecrypt2_In* in,
    EncryptDecrypt2_Out* out);


typedef struct {
    TPM2B_MAX_BUFFER data;
    TPMI_ALG_HASH hashAlg;
    TPMI_RH_HIERARCHY hierarchy;
```

```
} Hash_In;
typedef struct {
    TPM2B_DIGEST outHash;
    TPMT_TK_HASHCHECK validation;
} Hash_Out;
WOLFTPM_API TPM_RC TPM2_Hash(Hash_In* in, Hash_Out* out);

typedef struct {
    TPMI_DH_OBJECT handle;
    TPM2B_MAX_BUFFER buffer;
    TPMI_ALG_HASH hashAlg;
} HMAC_In;
typedef struct {
    TPM2B_DIGEST outHMAC;
} HMAC_Out;
WOLFTPM_API TPM_RC TPM2_HMAC(HMAC_In* in, HMAC_Out* out);


typedef struct {
    TPMI_DH_OBJECT handle;
    TPM2B_AUTH auth;
    TPMI_ALG_HASH hashAlg;
} HMAC_Start_In;
typedef struct {
    TPMI_DH_OBJECT sequenceHandle;
} HMAC_Start_Out;
WOLFTPM_API TPM_RC TPM2_HMAC_Start(HMAC_Start_In* in, HMAC_Start_Out* out);


typedef struct {
    TPM2B_AUTH auth;
    TPMI_ALG_HASH hashAlg;
} HashSequenceStart_In;
typedef struct {
    TPMI_DH_OBJECT sequenceHandle;
} HashSequenceStart_Out;
WOLFTPM_API TPM_RC TPM2_HashSequenceStart(HashSequenceStart_In* in,
    HashSequenceStart_Out* out);

typedef struct {
    TPMI_DH_OBJECT sequenceHandle;
    TPM2B_MAX_BUFFER buffer;
} SequenceUpdate_In;
WOLFTPM_API TPM_RC TPM2_SequenceUpdate(SequenceUpdate_In* in);

typedef struct {
    TPMI_DH_OBJECT sequenceHandle;
    TPM2B_MAX_BUFFER buffer;
    TPMI_RH_HIERARCHY hierarchy;
} SequenceComplete_In;
typedef struct {
    TPM2B_DIGEST result;
    TPMT_TK_HASHCHECK validation;
} SequenceComplete_Out;
```

```
WOLFTPM_API TPM_RC TPM2_SequenceComplete(SequenceComplete_In* in,
    SequenceComplete_Out* out);


typedef struct {
    TPMI_DH_PCR pcrHandle;
    TPMI_DH_OBJECT sequenceHandle;
    TPM2B_MAX_BUFFER buffer;
} EventSequenceComplete_In;
typedef struct {
    TPML_DIGEST_VALUES results;
} EventSequenceComplete_Out;
WOLFTPM_API TPM_RC TPM2_EventSequenceComplete(EventSequenceComplete_In* in,
    EventSequenceComplete_Out* out);


typedef struct {
    TPMI_DH_OBJECT objectHandle;
    TPMI_DH_OBJECT signHandle;
    TPM2B_DATA qualifyingData;
    TPMT_SIG_SCHEME inScheme;
} Certify_In;
typedef struct {
    TPM2B_ATTEST certifyInfo;
    TPMT_SIGNATURE signature;
} Certify_Out;
WOLFTPM_API TPM_RC TPM2_Certify(Certify_In* in, Certify_Out* out);


typedef struct {
    TPMI_DH_OBJECT signHandle;
    TPMI_DH_OBJECT objectHandle;
    TPM2B_DATA qualifyingData;
    TPM2B_DIGEST creationHash;
    TPMT_SIG_SCHEME inScheme;
    TPMT_TK_CREATION creationTicket;
} CertifyCreation_In;
typedef struct {
    TPM2B_ATTEST certifyInfo;
    TPMT_SIGNATURE signature;
} CertifyCreation_Out;
WOLFTPM_API TPM_RC TPM2_CertifyCreation(CertifyCreation_In* in,
 ↪  CertifyCreation_Out* out);


typedef struct {
    TPMI_DH_OBJECT signHandle;
    TPM2B_DATA qualifyingData;
    TPMT_SIG_SCHEME inScheme;
    TPML_PCR_SELECTION PCRselect;
} Quote_In;
typedef struct {
    TPM2B_ATTEST quoted;
    TPMT_SIGNATURE signature;
```

```
} Quote_Out;
WOLFTPM_API TPM_RC TPM2_Quote(Quote_In* in, Quote_Out* out);

typedef struct {
    TPMI_RH_ENDORSEMENT privacyAdminHandle;
    TPMI_DH_OBJECT signHandle;
    TPMI_SH_HMAC sessionHandle;
    TPM2B_DATA qualifyingData;
    TPMT_SIG_SCHEME inScheme;
} GetSessionAuditDigest_In;
typedef struct {
    TPM2B_ATTEST auditInfo;
    TPMT_SIGNATURE signature;
} GetSessionAuditDigest_Out;
WOLFTPM_API TPM_RC TPM2_GetSessionAuditDigest(GetSessionAuditDigest_In* in,
    GetSessionAuditDigest_Out* out);

typedef struct {
    TPMI_RH_ENDORSEMENT privacyHandle;
    TPMI_DH_OBJECT signHandle;
    TPM2B_DATA qualifyingData;
    TPMT_SIG_SCHEME inScheme;
} GetCommandAuditDigest_In;
typedef struct {
    TPM2B_ATTEST auditInfo;
    TPMT_SIGNATURE signature;
} GetCommandAuditDigest_Out;
WOLFTPM_API TPM_RC TPM2_GetCommandAuditDigest(GetCommandAuditDigest_In* in,
    GetCommandAuditDigest_Out* out);

typedef struct {
    TPMI_RH_ENDORSEMENT privacyAdminHandle;
    TPMI_DH_OBJECT signHandle;
    TPM2B_DATA qualifyingData;
    TPMT_SIG_SCHEME inScheme;
} GetTime_In;
typedef struct {
    TPM2B_ATTEST timeInfo;
    TPMT_SIGNATURE signature;
} GetTime_Out;
WOLFTPM_API TPM_RC TPM2_GetTime(GetTime_In* in, GetTime_Out* out);

typedef struct {
    TPMI_DH_OBJECT signHandle;
    TPM2B_ECC_POINT P1;
    TPM2B_SENSITIVE_DATA s2;
    TPM2B_ECC_PARAMETER y2;
} Commit_In;
typedef struct {
    TPM2B_ECC_POINT K;
    TPM2B_ECC_POINT L;
    TPM2B_ECC_POINT E;
    UINT16 counter;
} Commit_Out;
```

```
WOLFTPM_API TPM_RC TPM2_Commit(Commit_In* in, Commit_Out* out);


typedef struct {
    TPMI_ECC_CURVE curveID;
} EC_Ephemeral_In;
typedef struct {
    TPM2B_ECC_POINT Q;
    UINT16 counter;
} EC_Ephemeral_Out;
WOLFTPM_API TPM_RC TPM2_EC_Ephemeral(EC_Ephemeral_In* in,
    EC_Ephemeral_Out* out);

typedef struct {
    TPMI_DH_OBJECT keyHandle;
    TPM2B_DIGEST digest;
    TPMT_SIGNATURE signature;
} VerifySignature_In;
typedef struct {
    TPMT_TK_VERIFIED validation;
} VerifySignature_Out;
WOLFTPM_API TPM_RC TPM2_VerifySignature(VerifySignature_In* in,
    VerifySignature_Out* out);


typedef struct {
    TPMI_DH_OBJECT keyHandle;
    TPM2B_DIGEST digest;
    TPMT_SIG_SCHEME inScheme;
    TPMT_TK_HASHCHECK validation;
} Sign_In;
typedef struct {
    TPMT_SIGNATURE signature;
} Sign_Out;
WOLFTPM_API TPM_RC TPM2_Sign(Sign_In* in, Sign_Out* out);


typedef struct {
    TPMI_RH_PROVISION auth;
    TPMI_ALG_HASH auditAlg;
    TPML_CC setList;
    TPML_CC clearList;
} SetCommandCodeAuditStatus_In;
WOLFTPM_API TPM_RC TPM2_SetCommandCodeAuditStatus(
    SetCommandCodeAuditStatus_In* in);


typedef struct {
    TPMI_DH_PCR pcrHandle;
    TPM2B_EVENT eventData;
} PCR_Event_In;
typedef struct {
    TPML_DIGEST_VALUES digests;
} PCR_Event_Out;
```

```
WOLFTPM_API TPM_RC TPM2_PCR_Event(PCR_Event_In* in, PCR_Event_Out* out);


typedef struct {
    TPMI_RH_PLATFORM authHandle;
    TPML_PCR_SELECTION pcrAllocation;
} PCR_Allocate_In;
typedef struct {
    TPMI_YES_NO allocationSuccess;
    UINT32 maxPCR;
    UINT32 sizeNeeded;
    UINT32 sizeAvailable;
} PCR_Allocate_Out;
WOLFTPM_API TPM_RC TPM2_PCR_Allocate(PCR_Allocate_In* in,
    PCR_Allocate_Out* out);

typedef struct {
    TPMI_RH_PLATFORM authHandle;
    TPM2B_DIGEST authPolicy;
    TPMI_ALG_HASH hashAlg;
    TPMI_DH_PCR pcrNum;
} PCR_SetAuthPolicy_In;
WOLFTPM_API TPM_RC TPM2_PCR_SetAuthPolicy(PCR_SetAuthPolicy_In* in);

typedef struct {
    TPMI_DH_PCR pcrHandle;
    TPM2B_DIGEST auth;
} PCR_SetAuthValue_In;
WOLFTPM_API TPM_RC TPM2_PCR_SetAuthValue(PCR_SetAuthValue_In* in);

typedef struct {
    TPMI_DH_PCR pcrHandle;
} PCR_Reset_In;
WOLFTPM_API TPM_RC TPM2_PCR_Reset(PCR_Reset_In* in);


typedef struct {
    TPMI_DH_OBJECT authObject;
    TPMI_SH_POLICY policySession;
    TPM2B_NONCE nonceTPM;
    TPM2B_DIGEST cpHashA;
    TPM2B_NONCE policyRef;
    INT32 expiration;
    TPMT_SIGNATURE auth;
} PolicySigned_In;
typedef struct {
    TPM2B_TIMEOUT timeout;
    TPMT_TK_AUTH policyTicket;
} PolicySigned_Out;
WOLFTPM_API TPM_RC TPM2_PolicySigned(PolicySigned_In* in,
    PolicySigned_Out* out);

typedef struct {
    TPMI_DH_ENTITY authHandle;
```

```
    TPMI_SH_POLICY policySession;
    TPM2B_NONCE nonceTPM;
    TPM2B_DIGEST cpHashA;
    TPM2B_NONCE policyRef;
    INT32 expiration;
} PolicySecret_In;
typedef struct {
    TPM2B_TIMEOUT timeout;
    TPMT_TK_AUTH policyTicket;
} PolicySecret_Out;
WOLFTPM_API TPM_RC TPM2_PolicySecret(PolicySecret_In* in,
    PolicySecret_Out* out);

typedef struct {
    TPMI_SH_POLICY policySession;
    TPM2B_TIMEOUT timeout;
    TPM2B_DIGEST cpHashA;
    TPM2B_NONCE policyRef;
    TPM2B_NAME authName;
    TPMT_TK_AUTH ticket;
} PolicyTicket_In;
WOLFTPM_API TPM_RC TPM2_PolicyTicket(PolicyTicket_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
    TPML_DIGEST pHashList;
} PolicyOR_In;
WOLFTPM_API TPM_RC TPM2_PolicyOR(PolicyOR_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
    TPM2B_DIGEST pcrDigest;
    TPML_PCR_SELECTION pcrs;
} PolicyPCR_In;
WOLFTPM_API TPM_RC TPM2_PolicyPCR(PolicyPCR_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
    TPMA_LOCALITY locality;
} PolicyLocality_In;
WOLFTPM_API TPM_RC TPM2_PolicyLocality(PolicyLocality_In* in);

typedef struct {
    TPMI_RH_NV_AUTH authHandle;
    TPMI_RH_NV_INDEX nvIndex;
    TPMI_SH_POLICY policySession;
    TPM2B_OPERAND operandB;
    UINT16 offset;
    TPM_EO operation;
} PolicyNV_In;
WOLFTPM_API TPM_RC TPM2_PolicyNV(PolicyNV_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
```

```
    TPM2B_OPERAND operandB;
    UINT16 offset;
    TPM_EO operation;
} PolicyCounterTimer_In;
WOLFTPM_API TPM_RC TPM2_PolicyCounterTimer(PolicyCounterTimer_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
    TPM_CC code;
} PolicyCommandCode_In;
WOLFTPM_API TPM_RC TPM2_PolicyCommandCode(PolicyCommandCode_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
} PolicyPhysicalPresence_In;
WOLFTPM_API TPM_RC TPM2_PolicyPhysicalPresence(PolicyPhysicalPresence_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
    TPM2B_DIGEST cpHashA;
} PolicyCpHash_In;
WOLFTPM_API TPM_RC TPM2_PolicyCpHash(PolicyCpHash_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
    TPM2B_DIGEST nameHash;
} PolicyNameHash_In;
WOLFTPM_API TPM_RC TPM2_PolicyNameHash(PolicyNameHash_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
    TPM2B_NAME objectName;
    TPM2B_NAME newParentName;
    TPMI_YES_NO includeObject;
} PolicyDuplicationSelect_In;
WOLFTPM_API TPM_RC TPM2_PolicyDuplicationSelect(PolicyDuplicationSelect_In*
 ↪  in);

typedef struct {
    TPMI_SH_POLICY policySession;
    TPM2B_DIGEST approvedPolicy;
    TPM2B_NONCE policyRef;
    TPM2B_NAME keySign;
    TPMT_TK_VERIFIED checkTicket;
} PolicyAuthorize_In;
WOLFTPM_API TPM_RC TPM2_PolicyAuthorize(PolicyAuthorize_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
} PolicyAuthValue_In;
WOLFTPM_API TPM_RC TPM2_PolicyAuthValue(PolicyAuthValue_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
```

```
} PolicyPassword_In;
WOLFTPM_API TPM_RC TPM2_PolicyPassword(PolicyPassword_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
} PolicyGetDigest_In;
typedef struct {
    TPM2B_DIGEST policyDigest;
} PolicyGetDigest_Out;
WOLFTPM_API TPM_RC TPM2_PolicyGetDigest(PolicyGetDigest_In* in,
 ↪  PolicyGetDigest_Out* out);

typedef struct {
    TPMI_SH_POLICY policySession;
    TPMI_YES_NO writtenSet;
} PolicyNvWritten_In;
WOLFTPM_API TPM_RC TPM2_PolicyNvWritten(PolicyNvWritten_In* in);

typedef struct {
    TPMI_SH_POLICY policySession;
    TPM2B_DIGEST templateHash;
} PolicyTemplate_In;
WOLFTPM_API TPM_RC TPM2_PolicyTemplate(PolicyTemplate_In* in);

typedef struct {
    TPMI_RH_NV_AUTH authHandle;
    TPMI_RH_NV_INDEX nvIndex;
    TPMI_SH_POLICY policySession;
} PolicyAuthorizeNV_In;
WOLFTPM_API TPM_RC TPM2_PolicyAuthorizeNV(PolicyAuthorizeNV_In* in);



WOLFTPM_API void _TPM_Hash_Start(void);
WOLFTPM_API void _TPM_Hash_Data(UINT32 dataSize, BYTE *data);
WOLFTPM_API void _TPM_Hash_End(void);


typedef struct {
    TPMI_RH_HIERARCHY authHandle;
    TPMI_RH_ENABLES enable;
    TPMI_YES_NO state;
} HierarchyControl_In;
WOLFTPM_API TPM_RC TPM2_HierarchyControl(HierarchyControl_In* in);

typedef struct {
    TPMI_RH_HIERARCHY_AUTH authHandle;
    TPM2B_DIGEST authPolicy;
    TPMI_ALG_HASH hashAlg;
} SetPrimaryPolicy_In;
WOLFTPM_API TPM_RC TPM2_SetPrimaryPolicy(SetPrimaryPolicy_In* in);

typedef struct {
    TPMI_RH_PLATFORM authHandle;
```

```
} ChangeSeed_In;

typedef ChangeSeed_In ChangePPS_In;
WOLFTPM_API TPM_RC TPM2_ChangePPS(ChangePPS_In* in);

typedef ChangeSeed_In ChangeEPS_In;
WOLFTPM_API TPM_RC TPM2_ChangeEPS(ChangeEPS_In* in);


typedef struct {
    TPMI_RH_CLEAR authHandle;
} Clear_In;
WOLFTPM_API TPM_RC TPM2_Clear(Clear_In* in);

typedef struct {
    TPMI_RH_CLEAR auth;
    TPMI_YES_NO disable;
} ClearControl_In;
WOLFTPM_API TPM_RC TPM2_ClearControl(ClearControl_In* in);

typedef struct {
    TPMI_RH_HIERARCHY_AUTH authHandle;
    TPM2B_AUTH newAuth;
} HierarchyChangeAuth_In;
WOLFTPM_API TPM_RC TPM2_HierarchyChangeAuth(HierarchyChangeAuth_In* in);

typedef struct {
    TPMI_RH_LOCKOUT lockHandle;
} DictionaryAttackLockReset_In;
WOLFTPM_API TPM_RC
 ↪  TPM2_DictionaryAttackLockReset(DictionaryAttackLockReset_In* in);

typedef struct {
    TPMI_RH_LOCKOUT lockHandle;
    UINT32 newMaxTries;
    UINT32 newRecoveryTime;
    UINT32 lockoutRecovery;
} DictionaryAttackParameters_In;
WOLFTPM_API TPM_RC
 ↪  TPM2_DictionaryAttackParameters(DictionaryAttackParameters_In* in);


typedef struct {
    TPMI_RH_PLATFORM auth;
    TPML_CC setList;
    TPML_CC clearList;
} PP_Commands_In;
WOLFTPM_API TPM_RC TPM2_PP_Commands(PP_Commands_In* in);

typedef struct {
    TPMI_RH_PLATFORM authHandle;
    UINT32 algorithmSet;
} SetAlgorithmSet_In;
WOLFTPM_API TPM_RC TPM2_SetAlgorithmSet(SetAlgorithmSet_In* in);
```

```
typedef struct {
    TPMI_RH_PLATFORM authorization;
    TPMI_DH_OBJECT keyHandle;
    TPM2B_DIGEST fuDigest;
    TPMT_SIGNATURE manifestSignature;
} FieldUpgradeStart_In;
WOLFTPM_API TPM_RC TPM2_FieldUpgradeStart(FieldUpgradeStart_In* in);

typedef struct {
    TPM2B_MAX_BUFFER fuData;
} FieldUpgradeData_In;
typedef struct {
    TPMT_HA nextDigest;
    TPMT_HA firstDigest;
} FieldUpgradeData_Out;
WOLFTPM_API TPM_RC TPM2_FieldUpgradeData(FieldUpgradeData_In* in,
    FieldUpgradeData_Out* out);

typedef struct {
    UINT32 sequenceNumber;
} FirmwareRead_In;
typedef struct {
    TPM2B_MAX_BUFFER fuData;
} FirmwareRead_Out;
WOLFTPM_API TPM_RC TPM2_FirmwareRead(FirmwareRead_In* in, FirmwareRead_Out*
 ↪  out);


typedef struct {
    TPMI_DH_CONTEXT saveHandle;
} ContextSave_In;
typedef struct {
    TPMS_CONTEXT context;
} ContextSave_Out;
WOLFTPM_API TPM_RC TPM2_ContextSave(ContextSave_In* in, ContextSave_Out* out);

typedef struct {
    TPMS_CONTEXT context;
} ContextLoad_In;
typedef struct {
    TPMI_DH_CONTEXT loadedHandle;
} ContextLoad_Out;
WOLFTPM_API TPM_RC TPM2_ContextLoad(ContextLoad_In* in, ContextLoad_Out* out);


typedef struct {
    TPMI_RH_PROVISION auth;
    TPMI_DH_OBJECT objectHandle;
    TPMI_DH_PERSISTENT persistentHandle;
} EvictControl_In;
WOLFTPM_API TPM_RC TPM2_EvictControl(EvictControl_In* in);
```

```
typedef struct {
    TPMS_TIME_INFO currentTime;
} ReadClock_Out;
WOLFTPM_API TPM_RC TPM2_ReadClock(ReadClock_Out* out);

typedef struct {
    TPMI_RH_PROVISION auth;
    UINT64 newTime;
} ClockSet_In;
WOLFTPM_API TPM_RC TPM2_ClockSet(ClockSet_In* in);

typedef struct {
    TPMI_RH_PROVISION auth;
    TPM_CLOCK_ADJUST rateAdjust;
} ClockRateAdjust_In;
WOLFTPM_API TPM_RC TPM2_ClockRateAdjust(ClockRateAdjust_In* in);


typedef struct {
    TPMT_PUBLIC_PARMS parameters;
} TestParms_In;
WOLFTPM_API TPM_RC TPM2_TestParms(TestParms_In* in);


typedef struct {
    TPMI_RH_PROVISION authHandle;
    TPM2B_AUTH auth;
    TPM2B_NV_PUBLIC publicInfo;
} NV_DefineSpace_In;
WOLFTPM_API TPM_RC TPM2_NV_DefineSpace(NV_DefineSpace_In* in);

typedef struct {
    TPMI_RH_PROVISION authHandle;
    TPMI_RH_NV_INDEX nvIndex;
} NV_UndefineSpace_In;
WOLFTPM_API TPM_RC TPM2_NV_UndefineSpace(NV_UndefineSpace_In* in);

typedef struct {
    TPMI_RH_NV_INDEX nvIndex;
    TPMI_RH_PLATFORM platform;
} NV_UndefineSpaceSpecial_In;
WOLFTPM_API TPM_RC TPM2_NV_UndefineSpaceSpecial(NV_UndefineSpaceSpecial_In*
 ↪  in);

typedef struct {
    TPMI_RH_NV_INDEX nvIndex;
} NV_ReadPublic_In;
typedef struct {
    TPM2B_NV_PUBLIC nvPublic;
    TPM2B_NAME nvName;
} NV_ReadPublic_Out;
WOLFTPM_API TPM_RC TPM2_NV_ReadPublic(NV_ReadPublic_In* in, NV_ReadPublic_Out*
 ↪  out);
```

```
typedef struct {
    TPMI_RH_NV_AUTH authHandle;
    TPMI_RH_NV_INDEX nvIndex;
    TPM2B_MAX_NV_BUFFER data;
    UINT16 offset;
} NV_Write_In;
WOLFTPM_API TPM_RC TPM2_NV_Write(NV_Write_In* in);

typedef struct {
    TPMI_RH_NV_AUTH authHandle;
    TPMI_RH_NV_INDEX nvIndex;
} NV_Increment_In;
WOLFTPM_API TPM_RC TPM2_NV_Increment(NV_Increment_In* in);

typedef struct {
    TPMI_RH_NV_AUTH authHandle;
    TPMI_RH_NV_INDEX nvIndex;
    TPM2B_MAX_NV_BUFFER data;
} NV_Extend_In;
WOLFTPM_API TPM_RC TPM2_NV_Extend(NV_Extend_In* in);

typedef struct {
    TPMI_RH_NV_AUTH authHandle;
    TPMI_RH_NV_INDEX nvIndex;
    UINT64 bits;
} NV_SetBits_In;
WOLFTPM_API TPM_RC TPM2_NV_SetBits(NV_SetBits_In* in);

typedef struct {
    TPMI_RH_NV_AUTH authHandle;
    TPMI_RH_NV_INDEX nvIndex;
} NV_WriteLock_In;
WOLFTPM_API TPM_RC TPM2_NV_WriteLock(NV_WriteLock_In* in);

typedef struct {
    TPMI_RH_PROVISION authHandle;
} NV_GlobalWriteLock_In;
WOLFTPM_API TPM_RC TPM2_NV_GlobalWriteLock(NV_GlobalWriteLock_In* in);

typedef struct {
    TPMI_RH_NV_AUTH authHandle;
    TPMI_RH_NV_INDEX nvIndex;
    UINT16 size;
    UINT16 offset;
} NV_Read_In;
typedef struct {
    TPM2B_MAX_NV_BUFFER data;
} NV_Read_Out;
WOLFTPM_API TPM_RC TPM2_NV_Read(NV_Read_In* in, NV_Read_Out* out);

typedef struct {
    TPMI_RH_NV_AUTH authHandle;
    TPMI_RH_NV_INDEX nvIndex;
} NV_ReadLock_In;
```

```
WOLFTPM_API TPM_RC TPM2_NV_ReadLock(NV_ReadLock_In* in);

typedef struct {
    TPMI_RH_NV_INDEX nvIndex;
    TPM2B_AUTH newAuth;
} NV_ChangeAuth_In;
WOLFTPM_API TPM_RC TPM2_NV_ChangeAuth(NV_ChangeAuth_In* in);

typedef struct {
    TPMI_DH_OBJECT signHandle;
    TPMI_RH_NV_AUTH authHandle;
    TPMI_RH_NV_INDEX nvIndex;
    TPM2B_DATA qualifyingData;
    TPMT_SIG_SCHEME inScheme;
    UINT16 size;
    UINT16 offset;
} NV_Certify_In;
typedef struct {
    TPM2B_ATTEST certifyInfo;
    TPMT_SIGNATURE signature;
} NV_Certify_Out;
WOLFTPM_API TPM_RC TPM2_NV_Certify(NV_Certify_In* in, NV_Certify_Out* out);


/* Vendor Specific API's */
#if defined(WOLFTPM_ST33) || defined(WOLFTPM_AUTODETECT)
    /* Enable command code vendor API */
    typedef struct {
        TPMI_RH_HIERARCHY authHandle;
        TPM_CC commandCode;
        UINT32 enableFlag;
        UINT32 lockFlag;
    } SetCommandSet_In;
    WOLFTPM_API int TPM2_SetCommandSet(SetCommandSet_In* in);

    enum {
        TPMLib_2 = 0x01,
        TPMFips = 0x02,
        TPMLowPowerOff = 0x00,
        TPMLowPowerByRegister = 0x04,
        TPMLowPowerByGpio = 0x08,
        TPMLowPowerAuto = 0x0C,
    };
    typedef struct TPM_MODE_SET {
        BYTE CmdToLowPower;
        BYTE BootToLowPower;
        BYTE modeLock;
        BYTE mode;
    } TPM_MODE_SET;
    typedef struct {
        TPMI_RH_HIERARCHY authHandle;
        TPM_MODE_SET modeSet;
    } SetMode_In;
    WOLFTPM_API int TPM2_SetMode(SetMode_In* in);
```

```
    /* The TPM2_GetRandom2 command does not require any authorization */
    typedef GetRandom_In GetRandom2_In; /* same input */
    typedef struct {
        TPM2B_MAX_BUFFER randomBytes;
    } GetRandom2_Out;
    /* If bytesRequested is longer than TPM2B_MAX_BUFFER can accommodate, no
     * error is returned, but the TPM returns as much data as a TPM2B_DATA
     * buffer can contain. */
    WOLFTPM_API TPM_RC TPM2_GetRandom2(GetRandom2_In* in, GetRandom2_Out* out);
#endif

/* Vendor Specific GPIO */
#ifdef WOLFTPM_ST33

    #ifdef WOLFTPM_I2C
        #define MAX_GPIO_COUNT 4
    #else /* SPI variant */
        #define MAX_GPIO_COUNT 2
    #endif

    /* ST33 variants can have different count of GPIO available:
     * - SPI variant - 0, 1 or 2
     * - I2C variant - 0, 1, 2, 3 or 4
     * The user can configure this option at build or use default value. */
    #ifndef TPM_GPIO_COUNT
        #define TPM_GPIO_COUNT MAX_GPIO_COUNT
    #endif

    #define TPM_GPIO_NUM_MIN (TPM_GPIO_A)
    #define TPM_GPIO_NUM_MAX (TPM_GPIO_A + TPM_GPIO_COUNT - 1)

    /* GPIO configuration uses specific range of NV space */
    #define TPM_NV_GPIO_SPACE     0x01C40000

    typedef enum {
        TPM_GPIO_PP = 0x00000000, /* GPIO A by default is a Physical Presence
↪   pin */
        TPM_GPIO_LP = 0x00000001, /* GPIO B can only be used as an input */
    #ifdef WOLFTPM_I2C
        /* Only the I2C variant of ST33 has GPIO C and D */
        TPM_GPIO_C  = 0x00000002,
        TPM_GPIO_D  = 0x00000003,
    #endif
    } TPMI_GPIO_NAME_T;
    typedef UINT32 TPMI_GPIO_NAME;

    /* For portability and readability in code */
    #define TPM_GPIO_A TPM_GPIO_PP
    #define TPM_GPIO_B TPM_GPIO_LP

    typedef enum {
        TPM_GPIO_MODE_STANDARD    = 0x00000000,
        TPM_GPIO_MODE_FLOATING    = 0x00000001,
```

```
        TPM_GPIO_MODE_PULLUP     = 0x00000002,
        TPM_GPIO_MODE_PULLDOWN   = 0x00000003,
        TPM_GPIO_MODE_OPENDRAIN  = 0x00000004,
        TPM_GPIO_MODE_PUSHPULL   = 0x00000005,
        TPM_GPIO_MODE_UNCONFIG   = 0x00000006,
        TPM_GPIO_MODE_DEFAULT    = TPM_GPIO_MODE_PULLDOWN,
        TPM_GPIO_MODE_MAX        = TPM_GPIO_MODE_UNCONFIG,
        TPM_GPIO_MODE_INPUT_MIN  = TPM_GPIO_MODE_FLOATING,
        TPM_GPIO_MODE_INPUT_MAX  = TPM_GPIO_MODE_PULLDOWN
    } TPMI_GPIO_MODE_T;
    typedef UINT32 TPMI_GPIO_MODE;

    typedef struct TPMS_GPIO_CONFIG {
        TPMI_GPIO_NAME name;
        TPMI_RH_NV_INDEX index;
        TPMI_GPIO_MODE mode;
    } TPMS_GPIO_CONFIG;

    typedef struct TPML_GPIO_CONFIG {
        UINT32 count;
        TPMS_GPIO_CONFIG gpio[MAX_GPIO_COUNT];
    } TPML_GPIO_CONFIG;

    typedef struct {
        TPMI_RH_PLATFORM authHandle;
        TPML_GPIO_CONFIG config;
    } GpioConfig_In;
    WOLFTPM_API int TPM2_GPIO_Config(GpioConfig_In* in);

#elif defined(WOLFTPM_NUVOTON)

    #define MAX_GPIO_COUNT 2

    /* NPCT7XX supports a maximum of 2 GPIO for user control */
    /* Added in FW-US version 7.2.3.0 or later */
    #ifndef TPM_GPIO_COUNT
    #define TPM_GPIO_COUNT MAX_GPIO_COUNT
    #endif

    /* For portability */
    #undef  TPM_GPIO_A
    #define TPM_GPIO_A 3 /* NPCT75xx GPIO start at number 3 */

    #define TPM_GPIO_NUM_MIN (TPM_GPIO_A)
    #define TPM_GPIO_NUM_MAX (TPM_GPIO_A + TPM_GPIO_COUNT - 1)

    /* GPIO configuration uses specific range of NV space */
    #define TPM_NV_GPIO_SPACE     0x01C40003

    /* Nuvoton GPIO Modes */
    typedef enum {
        TPM_GPIO_MODE_PUSHPULL   = 1,
        TPM_GPIO_MODE_OPENDRAIN  = 2,
        TPM_GPIO_MODE_PULLUP     = 3,
```

```
        TPM_GPIO_MODE_UNCONFIG   = 4,
        TPM_GPIO_MODE_DEFAULT    = TPM_GPIO_MODE_PUSHPULL,
        TPM_GPIO_MODE_MAX        = TPM_GPIO_MODE_UNCONFIG,
        TPM_GPIO_MODE_INPUT_MIN  = TPM_GPIO_MODE_PULLUP,
        TPM_GPIO_MODE_INPUT_MAX  = TPM_GPIO_MODE_PULLUP
    } TPMI_GPIO_MODE_T;
    typedef UINT32 TPMI_GPIO_MODE;

    typedef struct {
        BYTE Base0;
        BYTE Base1;
        BYTE GpioAltCfg;
        BYTE GpioInitValue;
        BYTE GpioPullUp;
        BYTE GpioPushPull;
        BYTE Cfg_A;
        BYTE Cfg_B;
        BYTE Cfg_C;
        BYTE Cfg_D;
        BYTE Cfg_E;
        BYTE Cfg_F;
        BYTE Cfg_G;
        BYTE Cfg_H;
        BYTE Cfg_I;
        BYTE Cfg_J;
        BYTE isValid;
        BYTE isLocked;
    } CFG_STRUCT;

    typedef struct {
        TPMI_RH_PLATFORM authHandle;
        CFG_STRUCT preConfig;
    } NTC2_PreConfig_In;
    WOLFTPM_API int TPM2_NTC2_PreConfig(NTC2_PreConfig_In* in);

    typedef struct {
        CFG_STRUCT preConfig;
    } NTC2_GetConfig_Out;
    WOLFTPM_API int TPM2_NTC2_GetConfig(NTC2_GetConfig_Out* out);

#endif /* WOLFTPM_ST33 || WOLFTPM_AUTODETECT */


/* Non-standard API's */

#define _TPM_Init TPM2_Init

WOLFTPM_API TPM_RC TPM2_Init(TPM2_CTX* ctx, TPM2HalIoCb ioCb, void* userCtx);

WOLFTPM_API TPM_RC TPM2_Init_ex(TPM2_CTX* ctx, TPM2HalIoCb ioCb, void* userCtx,
    int timeoutTries);

WOLFTPM_API TPM_RC TPM2_Init_minimal(TPM2_CTX* ctx);
```

```
WOLFTPM_API TPM_RC TPM2_Cleanup(TPM2_CTX* ctx);

/* Other API's - Not in TPM Specification */

WOLFTPM_API TPM_RC TPM2_ChipStartup(TPM2_CTX* ctx, int timeoutTries);

WOLFTPM_API TPM_RC TPM2_SetHalIoCb(TPM2_CTX* ctx, TPM2HalIoCb ioCb, void*
 ↪  userCtx);

WOLFTPM_API TPM_RC TPM2_SetSessionAuth(TPM2_AUTH_SESSION *session);

WOLFTPM_API int    TPM2_GetSessionAuthCount(TPM2_CTX* ctx);

WOLFTPM_API void      TPM2_SetActiveCtx(TPM2_CTX* ctx);

WOLFTPM_API TPM2_CTX* TPM2_GetActiveCtx(void);

WOLFTPM_API int TPM2_GetHashDigestSize(TPMI_ALG_HASH hashAlg);

WOLFTPM_API int TPM2_GetHashType(TPMI_ALG_HASH hashAlg);

WOLFTPM_API int TPM2_GetNonce(byte* nonceBuf, int nonceSz);

WOLFTPM_API void TPM2_SetupPCRSel(TPML_PCR_SELECTION* pcr, TPM_ALG_ID alg,
    int pcrIndex);

WOLFTPM_API const char* TPM2_GetRCString(int rc);

WOLFTPM_API const char* TPM2_GetAlgName(TPM_ALG_ID alg);

WOLFTPM_API int TPM2_GetCurveSize(TPM_ECC_CURVE curveID);

WOLFTPM_API int TPM2_GetTpmCurve(int curveID);

WOLFTPM_API int TPM2_GetWolfCurve(int curve_id);

WOLFTPM_API int TPM2_ParseAttest(const TPM2B_ATTEST* in, TPMS_ATTEST* out);

WOLFTPM_API int TPM2_HashNvPublic(TPMS_NV_PUBLIC* nvPublic, byte* buffer,
 ↪  UINT16* size);

WOLFTPM_API int TPM2_AppendPublic(byte* buf, word32 size, int* sizeUsed,
 ↪  TPM2B_PUBLIC* pub);

WOLFTPM_API int TPM2_ParsePublic(TPM2B_PUBLIC* pub, byte* buf, word32 size,
 ↪  int* sizeUsed);

WOLFTPM_LOCAL int TPM2_GetName(TPM2_CTX* ctx, UINT32 handleValue, int
 ↪  handleCnt, int idx, TPM2B_NAME* name);

#ifdef WOLFTPM2_USE_WOLF_RNG
WOLFTPM_API int TPM2_GetWolfRng(WC_RNG** rng);
#endif
```

```
typedef enum {
    TPM_VENDOR_UNKNOWN = 0,
    TPM_VENDOR_INFINEON = 0x15d1,
    TPM_VENDOR_STM = 0x104a,
    TPM_VENDOR_MCHP = 0x1114,
    TPM_VENDOR_NUVOTON = 0x1050,
    TPM_VENDOR_NATIONTECH = 0x1B4E,
} TPM_Vendor_t;


WOLFTPM_API UINT16 TPM2_GetVendorID(void);

#ifdef DEBUG_WOLFTPM

WOLFTPM_API void TPM2_PrintBin(const byte* buffer, word32 length);

WOLFTPM_API void TPM2_PrintAuth(const TPMS_AUTH_COMMAND* authCmd);

WOLFTPM_API void TPM2_PrintPublicArea(const TPM2B_PUBLIC* pub);
#else
#define TPM2_PrintBin(b, l)
#define TPM2_PrintAuth(b, l)
#define TPM2_PrintPublicArea(b)
#endif

#ifdef __cplusplus
    }  /* extern "C" */
#endif

#endif /* __TPM2_H__ */
```

4.6 wolftpm/tpm2_wrap.h

4.6.1 クラス/構造体

	Name
struct	**WOLFTPM2_SESSION**
struct	**WOLFTPM2_DEV**
struct	**WOLFTPM2_KEY**
struct	**WOLFTPM2_KEYBLOB**
struct	**WOLFTPM2_HASH**
struct	**WOLFTPM2_NV**
struct	**WOLFTPM2_HMAC**
struct	**WOLFTPM2_CSR**
struct	**WOLFTPM2_BUFFER**
struct	**WOLFTPM2_CAPS**
struct	**TpmCryptoDevCtx**

4.6.2 型

	Name
enum	**WOLFTPM2_MFG** { TPM_MFG_UNKNOWN = 0, TPM_MFG_INFINEON, TPM_MFG_STM, TPM_MFG_MCHP, TPM_MFG_NUVOTON, TPM_MFG_NATIONTECH}
typedef struct WOLFTPM2_SESSION**	
typedef struct WOLFTPM2_DEV**	
typedef struct WOLFTPM2_KEY**	
typedef struct WOLFTPM2_KEYBLOB**	
typedef struct WOLFTPM2_HASH**	
typedef struct WOLFTPM2_NV**	
typedef struct WOLFTPM2_HMAC**	
typedef struct WOLFTPM2_CSR**	
typedef struct WOLFTPM2_BUFFER**	
typedef enum WOLFTPM2_MFG**	
typedef struct WOLFTPM2_CAPS**	
typedef int*()(wc_CryptoInfo* info, struct TpmCryptoDevCtx *ctx)	**CheckWolfKeyCallbackFunc**
typedef struct TpmCryptoDevCtx**	

4.6.3 関数

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_Test** TPM の初期化をテストし、オプションで TPM 機能を受け取ることができます。
WOLFTPM_API int	**wolfTPM2_Init** TPM の初期化を完了します。
WOLFTPM_API int	**wolfTPM2_OpenExisting** 現在の TPM ローカリティで、既に初期化されている TPM を使用します。
WOLFTPM_API int	**wolfTPM2_Cleanup** TPM と wolfcrypt の初期化解除を行います。
WOLFTPM_API int	**wolfTPM2_Cleanup_ex** TPM (および使用されている場合は wolfcrypt) の初期化解除。
WOLFTPM_API int	**wolfTPM2_GetTpmDevId** TPM のデバイス ID を提供します。
WOLFTPM_API int	**wolfTPM2_SelfTest** TPM にセルフ テストを実行するように要求します。
WOLFTPM_API int	**wolfTPM2_GetCapabilities** 利用可能な TPM 機能を報告します。
WOLFTPM_API int	**wolfTPM2_UnsetAuth** インデックス番号が指す TPM 認証スロットの 1 つをクリアします。
WOLFTPM_API int	**wolfTPM2_SetAuth** 指定されたインデックス、セッション ハンドル、属性、および認証を使用して、TPM 承認スロットを設定します。
WOLFTPM_API int	**wolfTPM2_SetAuthPassword** 提供されたユーザー認証 (通常はパスワード) を使用して TPM 認証スロットを設定します。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_SetAuthHandle** wolfTPM2 ハンドルに関連付けられたユーザー認証を使用して TPM 認証スロットを設定します。
WOLFTPM_API int	**wolfTPM2_SetAuthSession** 指定された TPM セッション ハンドル、インデックス、およびセッション属性を使用して、TPM 承認スロットを設定します。
WOLFTPM_API int	**wolfTPM2_SetAuthHandleName** TPM セッションで使用される名前を、wolfTPM2 ハンドルに関連付けられた名前で更新します。
WOLFTPM_API int	**wolfTPM2_StartSession** TPM セッション、ポリシー、HMAC、またはトライアルを作成します。
WOLFTPM_API int	**wolfTPM2_CreateAuthSession_EkPolicy** デフォルトの EK ポリシーを満たすために、ポリシー シークレットを使用して TPM セッションを作成します。
WOLFTPM_API int	**wolfTPM2_CreatePrimaryKey** TPM 2.0 プライマリ鍵を準備および作成する単一の関数。
WOLFTPM_API int	**wolfTPM2_ChangeAuthKey** TPM 2.0 鍵の認証シークレットを変更します。
WOLFTPM_API int	**wolfTPM2_CreateKey** TPM 2.0 鍵を準備および作成する単一の関数。
WOLFTPM_API int	**wolfTPM2_LoadKey** TPM 2.0 鍵をロードする単一関数。
WOLFTPM_API int	**wolfTPM2_CreateAndLoadKey** 1 つのステップで TPM 2.0 鍵を作成してロードする単一の関数。
WOLFTPM_API int	**wolfTPM2_CreateLoadedKey** 単一の TPM 2.0 操作を使用して鍵を作成および読み込み、暗号化された秘密鍵マテリアルを保存します。
WOLFTPM_API int	**wolfTPM2_LoadPublicKey** 外部の鍵の公開部分をロードするラッパー。
WOLFTPM_API int	**wolfTPM2_LoadPrivateKey** 外部の秘密鍵をインポートし、TPM に 1 ステップでロードする単一機能。
WOLFTPM_API int	**wolfTPM2_ImportPrivateKey** 外部の秘密鍵をインポートし、TPM に 1 ステップでロードする単一機能。
WOLFTPM_API int	**wolfTPM2_LoadRsaPublicKey** 外部 RSA 鍵の公開部分をインポートするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_LoadRsaPublicKey_ex** 外部の RSA 鍵の公開部分をインポートするヘルパー関数の拡張関数。
WOLFTPM_API int	**wolfTPM2_ImportRsaPrivateKey** 外部の RSA 秘密鍵をインポートします。
WOLFTPM_API int	**wolfTPM2_LoadRsaPrivateKey** 外部の RSA 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_LoadRsaPrivateKey_ex** 外部の RSA 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数の拡張関数。
WOLFTPM_API int	**wolfTPM2_LoadEccPublicKey** 外部の ECC 鍵の公開部分をインポートするヘルパー関数。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_ImportEccPrivateKey** 外部の ECC 鍵のプライベート マテリアルをインポートするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_LoadEccPrivateKey** 外部の ECC 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_ReadPublicKey** ハンドルを使用して、読み込まれた TPM オブジェクトのパブリック部分を受け取るヘルパー関数。
WOLFTPM_API int	**wolfTPM2_CreateKeySeal** このラッパーを使用すると、シークレットを TPM 2.0 鍵内に封印できます。
WOLFTPM_API int	**wolfTPM2_ComputeName** TPM が期待する形式でオブジェクトのパブリック領域のハッシュを生成するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_SensitiveToPrivate** TPM2B_SENSITIVE を変換するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_RsaKey_TpmToWolf** RSA TPM 鍵を抽出し、それを wolfcrypt 鍵に変換します。
WOLFTPM_API int	**wolfTPM2_RsaKey_TpmToPemPub** 公開 RSA TPM 鍵を PEM 形式の公開鍵に変換する 注: pem と tempBuf は、同じサイズの異なるバッファーである必要があります。
WOLFTPM_API int	**wolfTPM2_RsaKey_WolfToTpm** RSA wolfcrypt 鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_RsaKey_WolfToTpm_ex** 特定のプライマリ キーまたは階層の下で RSA wolfcrypt 鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_RsaKey_PubPemToTpm** PEM 形式の公開鍵をファイルから TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_EccKey_TpmToWolf** ECC TPM 鍵を抽出し、wolfcrypt 鍵に変換します。
WOLFTPM_API int	**wolfTPM2_EccKey_WolfToTpm** ECC wolfcrypt 鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_EccKey_WolfToTpm_ex** ECC wolfcrypt 鍵を特定のプライマリ キーまたは階層の下の TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_EccKey_WolfToPubPoint** wolfcrypt 鍵から生成された ECC 公開鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_SignHash** TPM 鍵を使用して任意のデータに署名するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_SignHashScheme** TPM 鍵を使用して任意のデータに署名し、署名スキームとハッシュ アルゴリズムを指定する高度なヘルパー関数。
WOLFTPM_API int	**wolfTPM2_VerifyHash** TPM が生成した署名を検証するためのヘルパー関数。
WOLFTPM_API int	**wolfTPM2_VerifyHashScheme** TPM が生成した署名を検証するための高度なヘルパー関数。
WOLFTPM_API int	**wolfTPM2_ECDHGenKey** Diffie-Hellman 交換用の NULL 階層を持つ ECC 鍵ペアを生成してからロードします。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_ECDHGen** 一時鍵を生成し、Z (共有シークレット) を計算します
WOLFTPM_API int	**wolfTPM2_ECDHGenZ** pubPoint と読み込まれたプライベート ECC 鍵を使用して Z (共有シークレット) を計算します。
WOLFTPM_API int	**wolfTPM2_ECDHEGenKey** 一時的な ECC 鍵を生成し、配列インデックスを返します (2 フェーズメソッド)
WOLFTPM_API int	**wolfTPM2_ECDHEGenZ** pubPoint とカウンターを使用して Z (共有シークレット) を計算します (2 フェーズ法)
WOLFTPM_API int	**wolfTPM2_RsaEncrypt** TPM 2.0 鍵を使用して RSA 暗号化を実行します。
WOLFTPM_API int	**wolfTPM2_RsaDecrypt** TPM 2.0 鍵を使用して RSA 復号を実行します。
WOLFTPM_API int	**wolfTPM2_ReadPCR** 指定された TPM 2.0 プラットフォーム構成レジスタ (PCR) の値を読み取る。
WOLFTPM_API int	**wolfTPM2_ExtendPCR** ユーザー提供のダイジェストで PCR レジスタを拡張します。
WOLFTPM_API int	**wolfTPM2_NVCreateAuth** TPM の NVRAM にデータを格納するために後で使用する新しい NV インデックスを作成します。
WOLFTPM_API int	**wolfTPM2_NVWriteAuth** 指定されたオフセットで、ユーザー データを NV インデックスに格納します。
WOLFTPM_API int	**wolfTPM2_NVReadAuth** 指定されたオフセットから開始して、NV インデックスからユーザー データを読み取ります。
WOLFTPM_API int	**wolfTPM2_NVIncrement** NV 一方向カウンターをインクリメントします。
WOLFTPM_API int	**wolfTPM2_NVOpen** NV を開き、必要な認証と名前ハッシュを入力します。
WOLFTPM_API int	**wolfTPM2_NVDeleteAuth** 既存の NV インデックスを破棄します。
WOLFTPM_API int	**wolfTPM2_NVCreate 非推奨です。新しい API を使用してください。**
WOLFTPM_API int	**wolfTPM2_NVWrite 非推奨です。新しい API を使用してください。**
WOLFTPM_API int	**wolfTPM2_NVRead 非推奨です。新しい API を使用してください。**
WOLFTPM_API int	**wolfTPM2_NVDelete 非推奨です。新しい API を使用してください。**
WOLFTPM_API int	**wolfTPM2_NVReadPublic** 最大サイズなど、nvIndex に関する公開情報を抽出します。
WOLFTPM_API int	**wolfTPM2_NVStoreKey** TPM 2.0 キーを TPM の NVRAM に格納するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_NVDeleteKey** TPM の NVRAM から TPM 2.0 鍵を削除するヘルパー関数。
WOLFTPM_API struct WC_RNG *	**wolfTPM2_GetRng** wolfTPM に使用される wolfcrypt RNG インスタンスを取得します。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_GetRandom** TPM RNG または wolfcrypt RNG で生成された一連の乱数を取得します。
WOLFTPM_API int	**wolfTPM2_UnloadHandle** TPM がロードされたオブジェクトを破棄するために使用します。
WOLFTPM_API int	**wolfTPM2_Clear** wolfTPM と wolfcrypt を初期化解除します (有効な場合)
WOLFTPM_API int	**wolfTPM2_HashStart** TPM で生成されたハッシュを開始するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_HashUpdate** TPM で生成されたハッシュを新しいユーザー データで更新します。
WOLFTPM_API int	**wolfTPM2_HashFinish** TPM で生成されたハッシュをファイナライズし、ユーザー バッファーでダイジェスト出力を取得します。
WOLFTPM_API int	**wolfTPM2_LoadKeyedHashKey** 通常は HMAC 操作に使用される、KeyedHash 型の新しい TPM 鍵を作成して読み込みます。
WOLFTPM_API int	**wolfTPM2_HmacStart** TPM で生成された hmac を開始するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_HmacUpdate** TPM で生成された hmac を新しいユーザー データで更新します。
WOLFTPM_API int	**wolfTPM2_HmacFinish** TPM で生成された hmac をファイナライズし、ユーザー バッファーでダイジェスト出力を取得します。
WOLFTPM_API int	**wolfTPM2_LoadSymmetricKey** 外部対称鍵を TPM にロードします。
WOLFTPM_API int	**wolfTPM2_SetCommand** 他の制限付き TPM コマンドを有効にするために使用される、ベンダー固有の TPM コマンド。
WOLFTPM_API int	**wolfTPM2_Shutdown** TPM をシャットダウンまたはリセットするためのヘルパー関数。
WOLFTPM_API int	**wolfTPM2_UnloadHandles** 後続の TPM ハンドルをアンロードするためのワンショット API。
WOLFTPM_API int	**wolfTPM2_UnloadHandles_AllTransient** すべての一時的な TPM ハンドルをアンロードするためのワンショット API。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA** ユーザーが選択したオブジェクト属性に基づいて、新しい RSA 鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC** ユーザーが選択したオブジェクト属性に基づいて、新しい ECC 鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_Symmetric** 新しい対称鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_KeyedHash** 新しい KeyedHash 鍵の TPM パブリック テンプレートを準備します。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_KeySeal** シークレットを封印するための新しい鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA_EK** RSA タイプの TPM 承認鍵を生成するための TPM 公開テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC_EK** ECC タイプの TPM 承認鍵を生成するための TPM 公開テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA_SRK** RSA タイプの新しい TPM ストレージ 鍵を生成するための TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC_SRK** ECC タイプの新しい TPM ストレージ 鍵を生成するための TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA_AIK** RSA タイプの新しい TPM Attestation Key を生成するための TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC_AIK** ECC タイプの新しい TPM Attestation Key を生成するための TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_SetKeyTemplate_Unique** Create または CreatePrimary で使用されるパブリック テンプレートの一意の領域を設定します。
WOLFTPM_API int	**wolfTPM2_GetNvAttributesTemplate** TPM NV インデックス テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_CreateEK** ユーザーが選択したアルゴリズム、RSA または ECC に基づいて、新しい TPM 承認キーを生成します。
WOLFTPM_API int	**wolfTPM2_CreateSRK** 他の TPM キーのストレージ キーとして使用される新しい TPM プライマリ キーを生成します。
WOLFTPM_API int	**wolfTPM2_CreateAndLoadAIK** 指定されたストレージ鍵の下に新しい TPM 構成証明鍵を生成します。
WOLFTPM_API int	**wolfTPM2_GetTime** TPM 署名付きタイムスタンプを生成するワンショット API。
WOLFTPM_API int	**wolfTPM2_CSR_SetCustomExt** WOLFTPM2_CSR 構造体のカスタム要求拡張 oid と値の使用を設定する証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_SetKeyUsage** WOLFTPM2_CSR 構造のキー使用法を設定する証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_SetSubject** WOLFTPM2_CSR 構造体のサブジェクトを設定する証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_MakeAndSign_ex** TPM ベースの鍵 (件名と鍵の使用法が既に設定されている WOLFTPM2_KEY 構造) を使用した証明書署名要求 (CSR) 生成のヘルパー。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_CSR_MakeAndSign** sTPM ベースの鍵 (件名と鍵の使用法が既に設定されている WOLFTPM2_KEY 構造) を使用した証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_Generate_ex** TPM ベースの鍵 (WOLFTPM2_KEY) を使用した証明書署名要求 (CSR) 生成のヘルパー。TPM 鍵に基づいて CSR または自己署名証明書を出力するためのシングル ショット API。
WOLFTPM_API int	**wolfTPM2_CSR_Generate** TPM ベースの鍵 (WOLFTPM2_KEY) を使用した証明書署名要求 (CSR) 生成のヘルパー。TPM 鍵に基づいて CSR または自己署名証明書を出力するためのシングル ショット API。
WOLFTPM_API int	**wolfTPM2_CryptoDevCb** クリプトオフロードに TPM を使用するためのリファレンス クリプト コールバック API。このコールバック関数は、wolfTPM2_SetCryptoDevCb または wc_CryptoDev_RegisterDevice を使用して登録されます。
WOLFTPM_API int	**wolfTPM2_SetCryptoDevCb** 暗号コールバック関数を登録し、割り当てられた devId を返します。
WOLFTPM_API int	**wolfTPM2_ClearCryptoDevCb** 登録された暗号コールバックをクリアします。
WOLFTPM_API WOLFTPM2_DEV**	**wolfTPM2_New** WOLFTPM2_DEV を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_Free** wolfTPM2_New によって割り当てられた WOLFTPM2_DEV をクリーンアップして解放します。
WOLFTPM_API WOLFTPM2_KEYBLOB	**wolfTPM2_NewKeyBlob** WOLFTPM2_KEYBLOB を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreeKeyBlob** wolfTPM2_NewKeyBlob で割り当てられた WOLFTPM2_KEYBLOB を解放します。
WOLFTPM_API TPMT_PUBLIC	**wolfTPM2_NewPublicTemplate** TPMT_PUBLIC 構造体を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreePublicTemplate** wolfTPM2_NewPublicTemplate で割り当てられた TPMT_PUBLIC を解放します。
WOLFTPM_API WOLFTPM2_KEY	**wolfTPM2_NewKey** WOLFTPM2_KEY を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreeKey** wolfTPM2_NewKey で割り当てられた WOLFTPM2_KEY を解放します。
WOLFTPM_API WOLFTPM2_SESSION *	**wolfTPM2_NewSession** WOLFTPM2_SESSION を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreeSession** wolfTPM2_NewSession で割り当てられた WOLFTPM2_SESSION を解放します。
WOLFTPM_API WOLFTPM2_CSR *	**wolfTPM2_NewCSR** WOLFTPM2_CSR を割り当てて初期化します。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_FreeCSR** wolfTPM2_NewCSR で割り当てられた WOLFTPM2_CSR を解放します。
WOLFTPM_API WOLFTPM2_HANDLE *	**wolfTPM2_GetHandleRefFromKey** WOLFTPM2_KEY から WOLFTPM2_HANDLE を取得します。
WOLFTPM_API WOLFTPM2_HANDLE *	**wolfTPM2_GetHandleRefFromKeyBlob** WOLFTPM2_KEYBLOB から WOLFTPM2_HANDLE を取得します。
WOLFTPM_API WOLFTPM2_HANDLE *	**wolfTPM2_GetHandleRefFromSession** WOLFTPM2_SESSION から WOLFTPM2_HANDLE を取得します。
WOLFTPM_API TPM_HANDLE	**wolfTPM2_GetHandleValue** WOLFTPM2_HANDLE から 32 ビットのハンドル値を取得します。
WOLFTPM_API int	**wolfTPM2_SetKeyAuthPassword** 鍵の認証データを設定します。
WOLFTPM_API int	**wolfTPM2_GetKeyBlobAsBuffer** キーブロブからバイナリ バッファーにデータをマーシャリングします。これは、別のプロセスでロードするため、または電源の再投入後にディスクに保存できます。
WOLFTPM_API int	**wolfTPM2_SetKeyBlobFromBuffer** データを WOLFTPM2_KEYBLOB 構造体にアンマーシャリングします。これは、wolfTPM2_GetKeyBlobAsBuffer によって以前にマーシャリングされたキーブロブをロードするために使用できます。

4.6.4 属性

Name
C

4.6.5 型の詳細

4.6.5.1 enum WOLFTPM2_MFG

Enumerator	Value	Description
TPM_MFG_UNKNOWN	0	
TPM_MFG_INFINEON		
TPM_MFG_STM		
TPM_MFG_MCHP		
TPM_MFG_NUVOTON		
TPM_MFG_NATIONTECH		

4.6.5.2 typedef WOLFTPM2_SESSION

```
typedef struct WOLFTPM2_SESSION WOLFTPM2_SESSION;
```

4.6.5.3 typedef WOLFTPM2_DEV

```
typedef struct WOLFTPM2_DEV WOLFTPM2_DEV;
```

4.6.5.4 typedef WOLFTPM2_KEY

```
typedef struct WOLFTPM2_KEY WOLFTPM2_KEY;
```

4.6.5.5 typedef WOLFTPM2_KEYBLOB

```
typedef struct WOLFTPM2_KEYBLOB WOLFTPM2_KEYBLOB;
```

4.6.5.6 typedef WOLFTPM2_HASH

```
typedef struct WOLFTPM2_HASH WOLFTPM2_HASH;
```

4.6.5.7 typedef WOLFTPM2_NV

```
typedef struct WOLFTPM2_NV WOLFTPM2_NV;
```

4.6.5.8 typedef WOLFTPM2_HMAC

```
typedef struct WOLFTPM2_HMAC WOLFTPM2_HMAC;
```

4.6.5.9 typedef WOLFTPM2_CSR

```
typedef struct WOLFTPM2_CSR WOLFTPM2_CSR;
```

4.6.5.10 typedef WOLFTPM2_BUFFER

```
typedef struct WOLFTPM2_BUFFER WOLFTPM2_BUFFER;
```

4.6.5.11 typedef WOLFTPM2_MFG

```
typedef enum WOLFTPM2_MFG WOLFTPM2_MFG;
```

4.6.5.12 typedef WOLFTPM2_CAPS

```
typedef struct WOLFTPM2_CAPS WOLFTPM2_CAPS;
```

4.6.5.13 typedef CheckWolfKeyCallbackFunc

```
typedef int(* CheckWolfKeyCallbackFunc) (wc_CryptoInfo *info, struct
 ↪  TpmCryptoDevCtx *ctx);
```

4.6.5.14 typedef TpmCryptoDevCtx

```
typedef struct TpmCryptoDevCtx TpmCryptoDevCtx;
```

4.7 wolfTPM2 ラッパー

このモジュールでは、ラッパーと呼ばれる wolfTPM の豊富な API について説明します。

wolfTPM ラッパーは、主に 2 つのケースで使用されます。

- キーの生成や保存など、一般的な TPM 2.0 タスクを実行する
- 構成証明やパラメーター暗号化などの複雑な TPM 2.0 タスクを実行する wolfTPM は、その多くのラッパー関数のおかげで、TPM 2.0 を迅速かつ迅速に使用できるようにします。

4.7.1 関数

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_Test** TPM の初期化をテストし、オプションで TPM 機能を受け取ることができます。
WOLFTPM_API int	**wolfTPM2_Init** TPM の初期化を完了します。
WOLFTPM_API int	**wolfTPM2_OpenExisting** 現在の TPM ローカリティで、既に初期化されている TPM を使用します。
WOLFTPM_API int	**wolfTPM2_Cleanup** TPM と wolfcrypt の初期化解除を行います。
WOLFTPM_API int	**wolfTPM2_Cleanup_ex** TPM (および使用されている場合は wolfcrypt) の初期化解除。
WOLFTPM_API int	**wolfTPM2_GetTpmDevId** TPM のデバイス ID を提供します。
WOLFTPM_API int	**wolfTPM2_SelfTest** TPM にセルフ テストを実行するように要求します。
WOLFTPM_API int	**wolfTPM2_GetCapabilities** 利用可能な TPM 機能を報告します。
WOLFTPM_API int	**wolfTPM2_UnsetAuth** インデックス番号が指す TPM 認証スロットの 1 つをクリアします。
WOLFTPM_API int	**wolfTPM2_SetAuth** 指定されたインデックス、セッション ハンドル、属性、および認証を使用して、TPM 承認スロットを設定します。
WOLFTPM_API int	**wolfTPM2_SetAuthPassword** 提供されたユーザー認証 (通常はパスワード) を使用して TPM 認証スロットを設定します。
WOLFTPM_API int	**wolfTPM2_SetAuthHandle** wolfTPM2 ハンドルに関連付けられたユーザー認証を使用して TPM 認証スロットを設定します。
WOLFTPM_API int	**wolfTPM2_SetAuthSession** 指定された TPM セッション ハンドル、インデックス、およびセッション属性を使用して、TPM 承認スロットを設定します。
WOLFTPM_API int	**wolfTPM2_SetAuthHandleName** TPM セッションで使用される名前を、wolfTPM2 ハンドルに関連付けられた名前で更新します。
WOLFTPM_API int	**wolfTPM2_StartSession** TPM セッション、ポリシー、HMAC、またはトライアルを作成します。
WOLFTPM_API int	**wolfTPM2_CreateAuthSession_EkPolicy** デフォルトの EK ポリシーを満たすために、ポリシー シークレットを使用して TPM セッションを作成します。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_CreatePrimaryKey** TPM 2.0 プライマリ鍵を準備および作成する単一の関数。
WOLFTPM_API int	**wolfTPM2_ChangeAuthKey** TPM 2.0 鍵の認証シークレットを変更します。
WOLFTPM_API int	**wolfTPM2_CreateKey** TPM 2.0 鍵を準備および作成する単一の関数。
WOLFTPM_API int	**wolfTPM2_LoadKey** TPM 2.0 鍵をロードする単一関数。
WOLFTPM_API int	**wolfTPM2_CreateAndLoadKey** 1 つのステップで TPM 2.0 鍵を作成してロードする単一の関数。
WOLFTPM_API int	**wolfTPM2_CreateLoadedKey** 単一の TPM 2.0 操作を使用して鍵を作成および読み込み、暗号化された秘密鍵マテリアルを保存します。
WOLFTPM_API int	**wolfTPM2_LoadPublicKey** 外部の鍵の公開部分をロードするラッパー。
WOLFTPM_API int	**wolfTPM2_LoadPrivateKey** 外部の秘密鍵をインポートし、TPM に 1 ステップでロードする単一機能。
WOLFTPM_API int	**wolfTPM2_ImportPrivateKey** 外部の秘密鍵をインポートし、TPM に 1 ステップでロードする単一機能。
WOLFTPM_API int	**wolfTPM2_LoadRsaPublicKey** 外部 RSA 鍵の公開部分をインポートするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_LoadRsaPublicKey_ex** 外部の RSA 鍵の公開部分をインポートするヘルパー関数の拡張関数。
WOLFTPM_API int	**wolfTPM2_ImportRsaPrivateKey** 外部の RSA 秘密鍵をインポートします。
WOLFTPM_API int	**wolfTPM2_LoadRsaPrivateKey** 外部の RSA 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_LoadRsaPrivateKey_ex** 外部の RSA 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数の拡張関数。
WOLFTPM_API int	**wolfTPM2_LoadEccPublicKey** 外部の ECC 鍵の公開部分をインポートするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_ImportEccPrivateKey** 外部の ECC 鍵のプライベート マテリアルをインポートするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_LoadEccPrivateKey** 外部の ECC 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_ReadPublicKey** ハンドルを使用して、読み込まれた TPM オブジェクトのパブリック部分を受け取るヘルパー関数。
WOLFTPM_API int	**wolfTPM2_CreateKeySeal** このラッパーを使用すると、シークレットを TPM 2.0 鍵内に封印できます。
WOLFTPM_API int	**wolfTPM2_ComputeName** TPM が期待する形式でオブジェクトのパブリック領域のハッシュを生成するヘルパー関数。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_SensitiveToPrivate** TPM2B_SENSITIVE を変換するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_RsaKey_TpmToWolf** RSA TPM 鍵を抽出し、それを wolfcrypt 鍵に変換します。
WOLFTPM_API int	**wolfTPM2_RsaKey_TpmToPemPub** 公開 RSA TPM 鍵を PEM 形式の公開鍵に変換する 注: pem と tempBuf は、同じサイズの異なるバッファーである必要があります。
WOLFTPM_API int	**wolfTPM2_RsaKey_WolfToTpm** RSA wolfcrypt 鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_RsaKey_WolfToTpm_ex** 特定のプライマリ キーまたは階層の下で RSA wolfcrypt 鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_RsaKey_PubPemToTpm** PEM 形式の公開鍵をファイルから TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_EccKey_TpmToWolf** ECC TPM 鍵を抽出し、wolfcrypt 鍵に変換します。
WOLFTPM_API int	**wolfTPM2_EccKey_WolfToTpm** ECC wolfcrypt 鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_EccKey_WolfToTpm_ex** ECC wolfcrypt 鍵を特定のプライマリ キーまたは階層の下の TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_EccKey_WolfToPubPoint** wolfcrypt 鍵から生成された ECC 公開鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_SignHash** TPM 鍵を使用して任意のデータに署名するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_SignHashScheme** TPM 鍵を使用して任意のデータに署名し、署名スキームとハッシュ アルゴリズムを指定する高度なヘルパー関数。
WOLFTPM_API int	**wolfTPM2_VerifyHash** TPM が生成した署名を検証するためのヘルパー関数。
WOLFTPM_API int	**wolfTPM2_VerifyHashScheme** TPM が生成した署名を検証するための高度なヘルパー関数。
WOLFTPM_API int	**wolfTPM2_ECDHGenKey** Diffie-Hellman 交換用の NULL 階層を持つ ECC 鍵ペアを生成してからロードします。
WOLFTPM_API int	**wolfTPM2_ECDHGen** 一時鍵を生成し、Z (共有シークレット) を計算します
WOLFTPM_API int	**wolfTPM2_ECDHGenZ** pubPoint と読み込まれたプライベート ECC 鍵を使用して Z (共有シークレット) を計算します。
WOLFTPM_API int	**wolfTPM2_ECDHEGenKey** 一時的な ECC 鍵を生成し、配列インデックスを返します (2 フェーズ メソッド)
WOLFTPM_API int	**wolfTPM2_ECDHEGenZ** pubPoint とカウンターを使用して Z (共有シークレット) を計算します (2 フェーズ法)
WOLFTPM_API int	**wolfTPM2_RsaEncrypt** TPM 2.0 鍵を使用して RSA 暗号化を実行します。
WOLFTPM_API int	**wolfTPM2_RsaDecrypt** TPM 2.0 鍵を使用して RSA 復号を実行します。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_ReadPCR** 指定された TPM 2.0 プラットフォーム構成レジスタ (PCR) の値を読み取る。
WOLFTPM_API int	**wolfTPM2_ExtendPCR** ユーザー提供のダイジェストで PCR レジスタを拡張します。
WOLFTPM_API int	**wolfTPM2_NVCreateAuth** TPM の NVRAM にデータを格納するために後で使用する新しい NV インデックスを作成します。
WOLFTPM_API int	**wolfTPM2_NVWriteAuth** 指定されたオフセットで、ユーザー データを NV インデックスに格納します。
WOLFTPM_API int	**wolfTPM2_NVReadAuth** 指定されたオフセットから開始して、NV インデックスからユーザー データを読み取ります。
WOLFTPM_API int	**wolfTPM2_NVIncrement** NV 一方向カウンターをインクリメントします。
WOLFTPM_API int	**wolfTPM2_NVOpen** NV を開き、必要な認証と名前ハッシュを入力します。
WOLFTPM_API int	**wolfTPM2_NVDeleteAuth** 既存の NV インデックスを破棄します。
WOLFTPM_API int	**wolfTPM2_NVCreate 非推奨です。新しい API を使用してください。**
WOLFTPM_API int	**wolfTPM2_NVWrite 非推奨です。新しい API を使用してください。**
WOLFTPM_API int	**wolfTPM2_NVRead 非推奨です。新しい API を使用してください。**
WOLFTPM_API int	**wolfTPM2_NVDelete 非推奨です。新しい API を使用してください。**
WOLFTPM_API int	**wolfTPM2_NVReadPublic** 最大サイズなど、nvIndex に関する公開情報を抽出します。
WOLFTPM_API int	**wolfTPM2_NVStoreKey** TPM 2.0 キーを TPM の NVRAM に格納するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_NVDeleteKey** TPM の NVRAM から TPM 2.0 鍵を削除するヘルパー関数。
WOLFTPM_API struct WC_RNG *	**wolfTPM2_GetRng** wolfTPM に使用される wolfcrypt RNG インスタンスを取得します。
WOLFTPM_API int	**wolfTPM2_GetRandom** TPM RNG または wolfcrypt RNG で生成された一連の乱数を取得します。
WOLFTPM_API int	**wolfTPM2_UnloadHandle** TPM がロードされたオブジェクトを破棄するために使用します。
WOLFTPM_API int	**wolfTPM2_Clear** wolfTPM と wolfcrypt を初期化解除します (有効な場合)
WOLFTPM_API int	**wolfTPM2_HashStart** TPM で生成されたハッシュを開始するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_HashUpdate** TPM で生成されたハッシュを新しいユーザー データで更新します。
WOLFTPM_API int	**wolfTPM2_HashFinish** TPM で生成されたハッシュをファイナライズし、ユーザー バッファーでダイジェスト出力を取得します。
WOLFTPM_API int	**wolfTPM2_LoadKeyedHashKey** 通常は HMAC 操作に使用される、KeyedHash 型の新しい TPM 鍵を作成して読み込みます。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_HmacStart** TPM で生成された hmac を開始するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_HmacUpdate** TPM で生成された hmac を新しいユーザー データで更新します。
WOLFTPM_API int	**wolfTPM2_HmacFinish** TPM で生成された hmac をファイナライズし、ユーザー バッファーでダイジェスト出力を取得します。
WOLFTPM_API int	**wolfTPM2_LoadSymmetricKey** 外部対称鍵を TPM にロードします。
WOLFTPM_API int	**wolfTPM2_SetCommand** 他の制限付き TPM コマンドを有効にするために使用される、ベンダー固有の TPM コマンド。
WOLFTPM_API int	**wolfTPM2_Shutdown** TPM をシャットダウンまたはリセットするためのヘルパー関数。
WOLFTPM_API int	**wolfTPM2_UnloadHandles** 後続の TPM ハンドルをアンロードするためのワンショット API。
WOLFTPM_API int	**wolfTPM2_UnloadHandles_AllTransient** すべての一時的な TPM ハンドルをアンロードするためのワンショット API。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA** ユーザーが選択したオブジェクト属性に基づいて、新しい RSA 鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC** ユーザーが選択したオブジェクト属性に基づいて、新しい ECC 鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_Symmetric** 新しい対称鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_KeyedHash** 新しい KeyedHash 鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_KeySeal** シークレットを封印するための新しい鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA_EK** RSA タイプの TPM 承認鍵を生成するための TPM 公開テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC_EK** ECC タイプの TPM 承認鍵を生成するための TPM 公開テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA_SRK** RSA タイプの新しい TPM ストレージ 鍵を生成するための TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC_SRK** ECC タイプの新しい TPM ストレージ 鍵を生成するための TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA_AIK** RSA タイプの新しい TPM Attestation Key を生成するための TPM パブリック テンプレートを準備します。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC_AIK** ECC タイプの新しい TPM Attestation Key を生成するための TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_SetKeyTemplate_Unique** Create または CreatePrimary で使用されるパブリック テンプレートの一意の領域を設定します。
WOLFTPM_API int	**wolfTPM2_GetNvAttributesTemplate** TPM NV インデックス テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_CreateEK** ユーザーが選択したアルゴリズム、RSA または ECC に基づいて、新しい TPM 承認キーを生成します。
WOLFTPM_API int	**wolfTPM2_CreateSRK** 他の TPM キーのストレージ キーとして使用される新しい TPM プライマリ キーを生成します。
WOLFTPM_API int	**wolfTPM2_CreateAndLoadAIK** 指定されたストレージ鍵の下に新しい TPM 構成証明鍵を生成します。
WOLFTPM_API int	**wolfTPM2_GetTime** TPM 署名付きタイムスタンプを生成するワンショット API。
WOLFTPM_API int	**wolfTPM2_CSR_SetCustomExt** WOLFTPM2_CSR 構造体のカスタム要求拡張 oid と値の使用を設定する証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_SetKeyUsage** WOLFTPM2_CSR 構造のキー使用法を設定する証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_SetSubject** WOLFTPM2_CSR 構造体のサブジェクトを設定する証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_MakeAndSign_ex** TPM ベースの鍵 (件名と鍵の使用法が既に設定されている WOLFTPM2_KEY 構造) を使用した証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_MakeAndSign** sTPM ベースの鍵 (件名と鍵の使用法が既に設定されている WOLFTPM2_KEY 構造) を使用した証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_Generate_ex** TPM ベースの鍵 (WOLFTPM2_KEY) を使用した証明書署名要求 (CSR) 生成のヘルパー。TPM 鍵に基づいて CSR または自己署名証明書を出力するためのシングル ショット API。
WOLFTPM_API int	**wolfTPM2_CSR_Generate** TPM ベースの鍵 (WOLFTPM2_KEY) を使用した証明書署名要求 (CSR) 生成のヘルパー。TPM 鍵に基づいて CSR または自己署名証明書を出力するためのシングル ショット API。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_CryptoDevCb** クリプトオフロードに TPM を使用するためのリファレンス クリプト コールバック API。このコールバック関数は、wolfTPM2_SetCryptoDevCb または wc_CryptoDev_RegisterDevice を使用して登録されます。
WOLFTPM_API int	**wolfTPM2_SetCryptoDevCb** 暗号コールバック関数を登録し、割り当てられた devId を返します。
WOLFTPM_API int	**wolfTPM2_ClearCryptoDevCb** 登録された暗号コールバックをクリアします。
WOLFTPM_API WOLFTPM2_DEV**	**wolfTPM2_New** WOLFTPM2_DEV を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_Free** wolfTPM2_New によって割り当てられた WOLFTPM2_DEV をクリーンアップして解放します。
WOLFTPM_API WOLFTPM2_KEYBLOB	**wolfTPM2_NewKeyBlob** WOLFTPM2_KEYBLOB を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreeKeyBlob** wolfTPM2_NewKeyBlob で割り当てられた WOLFTPM2_KEYBLOB を解放します。
WOLFTPM_API TPMT_PUBLIC	**wolfTPM2_NewPublicTemplate** TPMT_PUBLIC 構造体を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreePublicTemplate** wolfTPM2_NewPublicTemplate で割り当てられた TPMT_PUBLIC を解放します。
WOLFTPM_API WOLFTPM2_KEY	**wolfTPM2_NewKey** WOLFTPM2_KEY を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreeKey** wolfTPM2_NewKey で割り当てられた WOLFTPM2_KEY を解放します。
WOLFTPM_API WOLFTPM2_SESSION *	**wolfTPM2_NewSession** WOLFTPM2_SESSION を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreeSession** wolfTPM2_NewSession で割り当てられた WOLFTPM2_SESSION を解放します。
WOLFTPM_API WOLFTPM2_CSR *	**wolfTPM2_NewCSR** WOLFTPM2_CSR を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreeCSR** wolfTPM2_NewCSR で割り当てられた WOLFTPM2_CSR を解放します。
WOLFTPM_API WOLFTPM2_HANDLE *	**wolfTPM2_GetHandleRefFromKey** WOLFTPM2_KEY から WOLFTPM2_HANDLE を取得します。
WOLFTPM_API WOLFTPM2_HANDLE *	**wolfTPM2_GetHandleRefFromKeyBlob** WOLFTPM2_KEYBLOB から WOLFTPM2_HANDLE を取得します。
WOLFTPM_API WOLFTPM2_HANDLE *	**wolfTPM2_GetHandleRefFromSession** WOLFTPM2_SESSION から WOLFTPM2_HANDLE を取得します。
WOLFTPM_API TPM_HANDLE	**wolfTPM2_GetHandleValue** WOLFTPM2_HANDLE から 32 ビットのハンドル値を取得します。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_SetKeyAuthPassword** 鍵の認証データを設定します。
WOLFTPM_API int	**wolfTPM2_GetKeyBlobAsBuffer** キーブロブからバイナリ バッファーにデータをマーシャリングします。これは、別のプロセスでロードするため、または電源の再投入後にディスクに保存できます。
WOLFTPM_API int	**wolfTPM2_SetKeyBlobFromBuffer** データを WOLFTPM2_KEYBLOB 構造体にアンマーシャリングします。これは、wolfTPM2_GetKeyBlobAsBuffer によって以前にマーシャリングされたキーブロブをロードするために使用できます。

4.7.2 詳細な説明

このモジュールでは、ラッパーと呼ばれる wolfTPM の豊富な API について説明します。

wolfTPM ラッパーは、主に 2 つのケースで使用されます。

- キーの生成や保存など、一般的な TPM 2.0 タスクを実行する
- 構成証明やパラメーター暗号化などの複雑な TPM 2.0 タスクを実行する wolfTPM は、その多くのラッパー関数のおかげで、TPM 2.0 を迅速かつ迅速に使用できるようにします。

4.7.3 関数のドキュメント

wolfTPM2_Test

```
WOLFTPM_API int wolfTPM2_Test(
    TPM2HalIoCb ioCb,
    void * userCtx,
    WOLFTPM2_CAPS * caps
)
```

TPM の初期化をテストし、オプションで TPM 機能を受け取ることができます。

パラメータ:

- **ioCb** IO コールバック関数 (examples/tpm_io.h を参照)
- **userCtx** ユーザーコンテキストへのポインター (NULL 指定も可)
- **caps** TPM 機能を返却する為の WOLFTPM2_CAPS 構造体へのポインター (NULL 指定も可)

参考:

- wolfTPM2_Init
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_Init

```
WOLFTPM_API int wolfTPM2_Init(
    WOLFTPM2_DEV * dev,
```

```
    TPM2HalIoCb ioCb,
    void * userCtx
)
```

TPM の初期化を完了します。

パラメータ:

- **dev** WOLFTPM2_DEV 型の空の構造体へのポインター
- **ioCb** IO コールバック関数 (examples/tpm_io.h 参照)
- **userCtx** ユーザーコンテキストへのポインター (NULL 指定も可)

参考:

- wolfTPM2_OpenExisting
- wolfTPM2_Test
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信を確認すること)
- BAD_FUNC_ARG: 不正な引数

使用例

```
int rc;
WOLFTPM2_DEV dev;

rc = wolfTPM2_Init(&dev, TPM2_IoCb, userCtx);
if (rc != TPM_RC_SUCCESS) {
    //wolfTPM2_Init failed
    goto exit;
}
```

wolfTPM2_OpenExisting

```
WOLFTPM_API int wolfTPM2_OpenExisting(
    WOLFTPM2_DEV * dev,
    TPM2HalIoCb ioCb,
    void * userCtx
)
```

現在の TPM ローカリティで、既に初期化されている TPM を使用します。

パラメータ:

- **dev** WOLFTPM2_DEV 型の空の構造体へのポインター
- **ioCb** IO コールバック関数 (examples/tpm_io.h 参照)
- **userCtx** ユーザーコンテキストへのポインター (NULL 指定も可)

参考:

- wolfTPM2_Init
- wolfTPM2_Cleanup
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信を確認すること)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_Cleanup

```
WOLFTPM_API int wolfTPM2_Cleanup(
    WOLFTPM2_DEV * dev
)
```

TPM と wolfcrypt の初期化解除を行います。

パラメータ:

- **dev** データ設定済みの WOLFTPM2_DEV 型の構造体へのポインター

参考:

- wolfTPM2_OpenExisting
- wolfTPM2_Test
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信を確認すること)
- BAD_FUNC_ARG: 不正な引数

ノート:

適切な doShutdown パラメータを指定して wolfTPM2_Cleanup_ex を呼び出します

使用例

```
int rc;

rc = wolfTPM2_Cleanup(&dev);
if (rc != TPM_RC_SUCCESS) {
    //wolfTPM2_Cleanup failed
    goto exit;
}
```

wolfTPM2_Cleanup_ex

```
WOLFTPM_API int wolfTPM2_Cleanup_ex(
    WOLFTPM2_DEV * dev,
    int doShutdown
)
```

TPM (および使用されている場合は wolfcrypt) の初期化解除。

パラメータ:

- **dev** データ設定済みの WOLFTPM2_DEV 型の構造体へのポインター
- **doShutdown** フラグ値。true を指定の場合は TPM2_Shutdown が実行される

参考:

- wolfTPM2_OpenExisting
- wolfTPM2_Test
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信を確認すること)
- BAD_FUNC_ARG: 不正な引数

使用例

```
int rc;

//perform TPM2_Shutdown after deinitialization
rc = wolfTPM2_Cleanup_ex(&dev, 1);
if (rc != TPM_RC_SUCCESS) {
    //wolfTPM2_Cleanup_ex failed
    goto exit;
}
```

wolfTPM2_GetTpmDevId

```
WOLFTPM_API int wolfTPM2_GetTpmDevId(
    WOLFTPM2_DEV * dev
)
```

TPM のデバイス ID を提供します。

パラメータ:

- **dev** データ設定済みの WOLFTPM2_DEV 型の構造体へのポインター

参考:

- wolfTPM2_GetCapabilities
- wolfTPM2_Init

戻り値:

- 有効な TPM デバイス ID を示す整数値
- あるいは、TPM 初期化で DevID を取得できない場合は INVALID_DEVID を返します

使用例

```
int tpmDevId;

tpmDevId = wolfTPM2_GetTpmDevId(&dev);
if (tpmDevId != INVALID_DEVID) {
    //wolfTPM2_Cleanup_ex failed
    goto exit;
}
```

wolfTPM2_SelfTest

```
WOLFTPM_API int wolfTPM2_SelfTest(
    WOLFTPM2_DEV * dev
)
```

TPM にセルフ テストを実行するように要求します。

パラメータ:

- **dev** データ設定済みの WOLFTPM2_DEV 型の構造体へのポインター

参考:

- wolfTPM2_OpenExisting
- wolfTPM2_Test
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功

- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信と TPM リターンコードを確認すること)
- BAD_FUNC_ARG: 不正な引数

使用例

```
int rc;

//perform TPM2_Shutdown after deinitialization
rc = wolfTPM2_SelfTest(&dev);
if (rc != TPM_RC_SUCCESS) {
    //wolfTPM2_SelfTest failed
    goto exit;
}
```

wolfTPM2_GetCapabilities

```
WOLFTPM_API int wolfTPM2_GetCapabilities(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CAPS * caps
)
```

利用可能な TPM 機能を報告します。

パラメータ:

- **dev** データ設定済みの WOLFTPM2_DEV 型の構造体へのポインター
- **caps** 機能を返却する為の WOLFTPM2_CAPS 構造体へのポインター

参考:

- wolfTPM2_GetTpmDevId
- wolfTPM2_SelfTest
- wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信と TPM リターンコードを確認すること)
- BAD_FUNC_ARG: 不正な引数

使用例

```
int rc;
WOLFTPM2_CAPS caps;

//perform TPM2_Shutdown after deinitialization
rc = wolfTPM2_GetCapabilities(&dev, &caps);
if (rc != TPM_RC_SUCCESS) {
    //wolfTPM2_GetCapabilities failed
    goto exit;
}
```

wolfTPM2_UnsetAuth

```
WOLFTPM_API int wolfTPM2_UnsetAuth(
    WOLFTPM2_DEV * dev,
    int index
)
```

インデックス番号が指す TPM 認証スロットの 1 つをクリアします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定するための整数値（0～3）

参考:

- wolfTPM2_SetAuth
- wolfTPM2_SetAuthPassword
- wolfTPM2_SetAuthHandle
- wolfTPM2_SetAuthSession

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: TPM2 コンテキストのロックを取得できなかった
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_SetAuth

```
WOLFTPM_API int wolfTPM2_SetAuth(
    WOLFTPM2_DEV * dev,
    int index,
    TPM_HANDLE sessionHandle,
    const TPM2B_AUTH * auth,
    TPMA_SESSION sessionAttributes,
    const TPM2B_NAME * name
)
```

指定されたインデックス、セッション ハンドル、属性、および認証を使用して、TPM 承認スロットを設定します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定する整数値 (0 ～3)
- **sessionHandle** TPM_HANDLE 型の整数値
- **auth** TPM 承認を含む TPM2B_AUTH 型の構造体へのポインター
- **sessionAttributes** TPMA_SESSION 型の整数値。セッションの 1 つ以上の属性を選択します
- **name** TPM2B_NAME 構造体へのポインター

参考:

- wolfTPM2_SetAuthPassword
- wolfTPM2_SetAuthHandle
- wolfTPM2_SetAuthSession

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

wolfTPM2_SetAuthPassword など、他の wolfTPM2 ラッパーのいずれかを使用することをお勧めします。wolfTPM2_SetAuth ラッパーは、高度なユース ケースの TPM 認証スロットを完全に制御できるためです。ほとんどのシナリオでは、wolfTPM2_SetAuthHandle と SetAuthPassword が使用されます。

wolfTPM2_SetAuthPassword

```
WOLFTPM_API int wolfTPM2_SetAuthPassword(
    WOLFTPM2_DEV * dev,
    int index,
    const TPM2B_AUTH * auth
```

)

提供されたユーザー認証 (通常はパスワード) を使用して TPM 認証スロットを設定します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定する整数値 (0 〜3)
- **auth** TPM 承認を含む TPM2B_AUTH 型の構造体へのポインター

参考:

- wolfTPM2_SetAuthHandle
- wolfTPM2_SetAuthSession
- wolfTPM2_SetAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

多くの場合、プライマリ キーを含む TPM 鍵の読み込みと使用を承認するために使用されます。

wolfTPM2_SetAuthHandle

```
WOLFTPM_API int wolfTPM2_SetAuthHandle(
    WOLFTPM2_DEV * dev,
    int index,
    const WOLFTPM2_HANDLE * handle
)
```

wolfTPM2 ハンドルに関連付けられたユーザー認証を使用して TPM 認証スロットを設定します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定する整数値 (0 〜3)
- **handle** WOLFTPM2_HANDLE 構造体へのポインター

参考:

- wolfTPM2_SetAuth
- wolfTPM2_SetAuthPassword
- wolfTPM2_SetAuthHandle
- wolfTPM2_SetAuthSession

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

このラッパーは、TPM キーを複数の操作に使用し、TPM 承認が再び必要な場合に特に役立ちます。

wolfTPM2_SetAuthSession

```
WOLFTPM_API int wolfTPM2_SetAuthSession(
    WOLFTPM2_DEV * dev,
    int index,
    const WOLFTPM2_SESSION * tpmSession,
    TPMA_SESSION sessionAttributes
)
```

指定された TPM セッション ハンドル、インデックス、およびセッション属性を使用して、TPM 承認スロットを設定します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定する整数値 (0 ～3)
- **tpmSession** TPM_HANDLE 型のセッションハンドル整数値
- **sessionAttributes** TPMA_SESSION 型の整数値, 一つ以上のアトリビュートを選択する

参考:

- wolfTPM2_SetAuth
- wolfTPM2_SetAuthPassword
- wolfTPM2_SetAuthHandle

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

このラッパーは、TPM セッション、つまりパラメータ暗号化のセッションのコンフィギュレーションに有用です。

wolfTPM2_SetAuthHandleName

```
WOLFTPM_API int wolfTPM2_SetAuthHandleName(
    WOLFTPM2_DEV * dev,
    int index,
    const WOLFTPM2_HANDLE * handle
)
```

TPM セッションで使用される名前を、wolfTPM2 ハンドルに関連付けられた名前で更新します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定する整数値 (0 ～3)
- **handle** WOLFTPM2_HANDLE 構造体へのポインター

参考:

- wolfTPM2_SetAuth
- wolfTPM2_SetAuthPassword
- wolfTPM2_SetAuthHandle
- wolfTPM2_SetAuthSession

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

通常、このラッパーは別のラッパー（例えば wolfTPM2_NVWriteAuth）から使用され、非常に特殊なユースケースで使用されます

wolfTPM2_StartSession

```
WOLFTPM_API int wolfTPM2_StartSession(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_SESSION * session,
```

```
    WOLFTPM2_KEY * tpmKey,
    WOLFTPM2_HANDLE * bind,
    TPM_SE sesType,
    int encDecAlg
)
```

TPM セッション、ポリシー、HMAC、またはトライアルを作成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **session** WOLFTPM2_SESSION 構造体へのポインター
- **tpmKey** セッションのソルトとして使用する WOLFTPM2_KEY へのポインター
- **bind** セッションをバインドするために使用される WOLFTPM2_HANDLE へのポインタ
- **sesType** バイト値、セッション タイプ (HMAC、ポリシー、またはトライアル)
- **encDecAlg** パラメータ暗号化の場合のアルゴリズムを指定する整数値 (TPM_ALG_CFB または TPM_ALG_XOR)。CFB または XOR 以外の値はすべて NULL と見なされ、パラメーターの暗号化は無効になります。

参考:

- wolfTPM2_SetAuthSession

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

このラッパーは、パラメータ暗号化のために TPM セッションを開始するためにも使用できます。wolfTPM nvram または keygen の例を参照してください。

wolfTPM2_CreateAuthSession_EkPolicy

```
WOLFTPM_API int wolfTPM2_CreateAuthSession_EkPolicy(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_SESSION * tpmSession
)
```

デフォルトの EK ポリシーを満たすために、ポリシー シークレットを使用して TPM セッションを作成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **session** WOLFTPM2_SESSION 構造体へのポインター

参考:

- wolfTPM2_SetAuthSession
- wolfTPM2_StartSession

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数
- TPM_RC_FAILURE: TPM リターン コード、使用可能なハンドル、TPM IO などを要確認

ノート:

このラッパーは、EK 承認がデフォルトから変更されていない場合にのみ使用できます。

wolfTPM2_CreatePrimaryKey

```
WOLFTPM_API int wolfTPM2_CreatePrimaryKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    TPM_HANDLE primaryHandle,
    TPMT_PUBLIC * publicTemplate,
    const byte * auth,
    int authSz
)
```

TPM 2.0 プライマリ鍵を準備および作成する単一の関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **primaryHandle** 4 つの TPM 2.0 プライマリ シードのいずれかを指定する整数値: TPM_RH_OWNER、TPM_RH_ENDORSEMENT、TPM_RH_PLATFORM、または TPM_RH_NULL
- **publicTemplate** 手動で、または wolfTPM2_GetKeyTemplate_... ラッパーの 1 つを使用して設定された TPMT_PUBLIC 構造へのポインタ
- **auth** プライマリキーのパスワード認証を指定する文字列定数へのポインタ
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreateKey
- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 2.0 では、非対称 RSA または ECC プライマリ キーのみが許可されます。その後、対称鍵と非対称鍵の両方を TPM 2.0 プライマリ キーの下に作成できます。通常、プライマリ キーは TPM 2.0 鍵の階層を作成するために使用されます。TPM は主鍵を使用して、他の鍵をラップし、署名または復号します。

wolfTPM2_ChangeAuthKey

```
WOLFTPM_API int wolfTPM2_ChangeAuthKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    WOLFTPM2_HANDLE * parent,
    const byte * auth,
    int authSz
)
```

TPM 2.0 鍵の認証シークレットを変更します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される TPM 2.0 プライマリ キーを指定します
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreatePrimaryKey
- wolfTPM2_SetAuthHandle
- wolfTPM2_UnloadHandle

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

PM では、プライマリ キーの認証シークレットを変更することはできません。代わりに、wolfTPM2_CreatePrimary を使用して、新しい認証で同じ PrimaryKey を作成します。

wolfTPM2_CreateKey

```
WOLFTPM_API int wolfTPM2_CreateKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEYBLOB * keyBlob,
    WOLFTPM2_HANDLE * parent,
    TPMT_PUBLIC * publicTemplate,
    const byte * auth,
    int authSz
)
```

TPM 2.0 鍵を準備および作成する単一の関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **keyBlob** WOLFTPM2_KEYBLOB 型の空の構造体へのポインター
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される 2.0 プライマリ キーを指定します
- **publicTemplate** 手動で、または wolfTPM2_GetKeyTemplate_... ラッパーの 1 つを使用して設定された TPMT_PUBLIC 構造へのポインター
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_LoadKey
- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC
- wolfTPM2_CreatePrimaryKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

この関数は、キー マテリアルのみを作成し、それをキーブロブ引数に格納します。キーをロードするには、wolfTPM2_LoadKey を使用します

wolfTPM2_LoadKey

```
WOLFTPM_API int wolfTPM2_LoadKey(
    WOLFTPM2_DEV * dev,
```

```
    WOLFTPM2_KEYBLOB * keyBlob,
    WOLFTPM2_HANDLE * parent
)
```

TPM 2.0 鍵をロードする単一関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **keyBlob** WOLFTPM2_KEYBLOB 型の構造体へのポインター
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される TPM 2.0 プライマリ キーを指定します

参考:

- wolfTPM2_CreateKey
- wolfTPM2_CreatePrimaryKey
- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

- TPM 2.0 鍵をロードするには、この操作の前にその親 (プライマリ キー) もロードする必要があります。プライマリキー作成時にロードされます。

wolfTPM2_CreateAndLoadKey

```
WOLFTPM_API int wolfTPM2_CreateAndLoadKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    WOLFTPM2_HANDLE * parent,
    TPMT_PUBLIC * publicTemplate,
    const byte * auth,
    int authSz
)
```

1 つのステップで TPM 2.0 鍵を作成してロードする単一の関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される TPM 2.0 プライマリ キーを指定します
- **publicTemplate** 手動で、または wolfTPM2_GetKeyTemplate_... ラッパーの 1 つを使用して設定された TPMT_PUBLIC 構造へのポインタ
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreateKey
- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CreateLoadedKey

```
WOLFTPM_API int wolfTPM2_CreateLoadedKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEYBLOB * keyBlob,
    WOLFTPM2_HANDLE * parent,
    TPMT_PUBLIC * publicTemplate,
    const byte * auth,
    int authSz
)
```

単一の TPM 2.0 操作を使用して鍵を作成および読み込み、暗号化された秘密鍵マテリアルを保存します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **keyBlob** WOLFTPM2_KEYBLOB 型の空の構造体へのポインター。暗号化されたデータとして秘密鍵の素材が含まれます
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される TPM 2.0 プライマリ キーを指定します
- **publicTemplate** 手動で、または wolfTPM2_GetKeyTemplate_... ラッパーの 1 つを使用して設定された TPMT_PUBLIC 構造へのポインター
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreateAndLoadKey
- wolfTPM2_CreateKey
- wolfTPM2_LoadKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_LoadPublicKey

```
WOLFTPM_API int wolfTPM2_LoadPublicKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const TPM2B_PUBLIC * pub
)
```

外部の鍵の公開部分をロードするラッパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **pub** TPM2B_PUBLIC 構造体へのポインター

参考:

- wolfTPM2_LoadRsaPublicKey
- wolfTPM2_LoadEccPublicKey

- wolfTPM2_wolfTPM2_LoadPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

- 鍵は、TPM が期待する形式にフォーマットする必要があります。'pub' 引数と代替ラッパーを参照してください。

wolfTPM2_LoadPrivateKey

```
WOLFTPM_API int wolfTPM2_LoadPrivateKey(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEY * key,
    const TPM2B_PUBLIC * pub,
    TPM2B_SENSITIVE * sens
)
```

外部の秘密鍵をインポートし、TPM に 1 ステップでロードする単一機能。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL 指定が可)
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **pub** TPM2B_PUBLIC 構造体へのポインター
- **sens** TPM2B_SENSITIVE 型のデータが設定された構造体へのポインター

参考:

- wolfTPM2_CreateKey
- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

秘密鍵の素材は、TPM が期待する形式で準備する必要があります。"sens" 引数を参照してください

wolfTPM2_ImportPrivateKey

```
WOLFTPM_API int wolfTPM2_ImportPrivateKey(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEYBLOB * keyBlob,
    const TPM2B_PUBLIC * pub,
    TPM2B_SENSITIVE * sens
)
```

外部の秘密鍵をインポートし、TPM に 1 ステップでロードする単一機能。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL 指定が可)
- **keyBlob** WOLFTPM2_KEYBLOB]() 型の空の構造体へのポインター
- **pub** TPM2B_PUBLIC 構造体へのポインター
- **sens** TPM2B_SENSITIVE 型のデータが設定された構造体へのポインター

参考:

- wolfTPM2_ImportRsaPrivateKey
- wolfTPM2_ImportEccPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

プライマリ キー マテリアルは、TPM が期待する形式で準備する必要があります。“sens” 引数を参照してください。

wolfTPM2_LoadRsaPublicKey

```
WOLFTPM_API int wolfTPM2_LoadRsaPublicKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * rsaPub,
    word32 rsaPubSz,
    word32 exponent
)
```

外部 RSA 鍵の公開部分をインポートするヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **rsaPub** 公開鍵素材を含むバイトバッファへのポインター
- **rsaPubSz** バッファサイズを指定する word32 型の整数値
- **exponent** RSA 指数を指定する word32 型の整数値

参考:

- wolfTPM2_LoadRsaPublicKey_ex
- wolfTPM2_LoadPublicKey
- wolfTPM2_LoadEccPublicKey
- wolfTPM2_ReadPublicKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

パブリック部分の TPM 形式を必要としないため、使用を推奨

wolfTPM2_LoadRsaPublicKey_ex

```
WOLFTPM_API int wolfTPM2_LoadRsaPublicKey_ex(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * rsaPub,
    word32 rsaPubSz,
    word32 exponent,
    TPMI_ALG_RSA_SCHEME scheme,
    TPMI_ALG_HASH hashAlg
)
```

外部の RSA 鍵の公開部分をインポートするヘルパー関数の拡張関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **rsaPub** 公開鍵素材を含むバイトバッファへのポインター
- **rsaPubSz** バッファサイズを指定する word32 型の整数値
- **exponent** RSA 指数を指定する word32 型の整数値
- **scheme** RSA スキームを指定する TPMI_ALG_RSA_SCHEME タイプの値
- **hashAlg** TPM ハッシュアルゴリズムを指定する TPMI_ALG_HASH タイプの値

参考:

- wolfTPM2_LoadRsaPublicKey
- wolfTPM2_LoadPublicKey
- wolfTPM2_LoadEccPublicKey
- wolfTPM2_ReadPublicKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

開発者は TPM ハッシュ アルゴリズムと RSA スキームを指定できます

wolfTPM2_ImportRsaPrivateKey

```
WOLFTPM_API int wolfTPM2_ImportRsaPrivateKey(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEYBLOB * keyBlob,
    const byte * rsaPub,
    word32 rsaPubSz,
    word32 exponent,
    const byte * rsaPriv,
    word32 rsaPrivSz,
    TPMI_ALG_RSA_SCHEME scheme,
    TPMI_ALG_HASH hashAlg
)
```

外部の RSA 秘密鍵をインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL にすることができ、キーは OWNER 階層の下にインポートされます)
- **keyBlob** WOLFTPM2_KEYBLOB]() 型の空の構造体へのポインター
- **rsaPub** RSA 鍵の公開部分を含むバイトバッファへのポインタ
- **rsaPubSz** パブリック部分のバッファ サイズを指定する word32 型の整数値
- **exponent** RSA 指数を指定する word32 型の整数値
- **rsaPriv** RSA 鍵のプライベートマテリアルを含むバイトバッファへのポインター
- **rsaPrivSz** プライベート マテリアル バッファ サイズを指定する word32 型の整数値
- **scheme** RSA スキームを指定する TPMI_ALG_RSA_SCHEME タイプの値
- **hashAlg** サポートされている TPM 2.0 ハッシュ アルゴリズムを指定する TPMI_ALG_HASH 型の整数値

参考:

- wolfTPM2_LoadRsaPrivateKey
- wolfTPM2_LoadRsaPrivateKey_ex
- wolfTPM2_LoadPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数
- BUFFER_E: 引数のサイズが TPM バッファで許可されているサイズよりも大きい

wolfTPM2_LoadRsaPrivateKey

```
WOLFTPM_API int wolfTPM2_LoadRsaPrivateKey(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEY * key,
    const byte * rsaPub,
    word32 rsaPubSz,
    word32 exponent,
    const byte * rsaPriv,
    word32 rsaPrivSz
)
```

外部の RSA 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL にすることができ、キーは OWNER 階層の下にインポートされます)
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **rsaPub** RSA 鍵の公開部分を含むバイトバッファへのポインタ
- **rsaPubSz** パブリック部分のバッファ サイズを指定する word32 型の整数値
- **exponent** RSA 指数を指定する word32 型の整数値
- **rsaPriv** RSA 鍵のプライベートマテリアルを含むバイトバッファへのポインター
- **rsaPrivSz** プライベート マテリアル バッファ サイズを指定する word32 型の整数値

参考:

- wolfTPM2_ImportRsaPrivateKey
- wolfTPM2_LoadRsaPrivateKey_ex
- wolfTPM2_LoadPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_LoadRsaPrivateKey_ex

```
WOLFTPM_API int wolfTPM2_LoadRsaPrivateKey_ex(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEY * key,
    const byte * rsaPub,
    word32 rsaPubSz,
    word32 exponent,
    const byte * rsaPriv,
    word32 rsaPrivSz,
    TPMI_ALG_RSA_SCHEME scheme,
    TPMI_ALG_HASH hashAlg
)
```

外部の RSA 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数の拡張関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL にすることができ、キーは OWNER 階層の下にインポートされます)
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **rsaPub** RSA 鍵の公開部分を含むバイトバッファへのポインター
- **rsaPubSz** パブリック部分のバッファ サイズを指定する word32 型の整数値
- **exponent** RSA 指数を指定する word32 型の整数値
- **rsaPriv** RSA 鍵のプライベートマテリアルを含むバイトバッファへのポインター
- **rsaPrivSz** プライベート マテリアル バッファ サイズを指定する word32 型の整数値
- **scheme** RSA スキームを指定する TPMI_ALG_RSA_SCHEME タイプの値
- **hashAlg** TPM ハッシュアルゴリズムを指定する TPMI_ALG_HASH タイプの値

参考:

- wolfTPM2_LoadRsaPrivateKey
- wolfTPM2_LoadPrivateKey
- wolfTPM2_ImportRsaPrivateKey
- wolfTPM2_LoadEccPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_LoadEccPublicKey

```
WOLFTPM_API int wolfTPM2_LoadEccPublicKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    int curveId,
    const byte * eccPubX,
    word32 eccPubXSz,
    const byte * eccPubY,
    word32 eccPubYSz
)
```

外部の ECC 鍵の公開部分をインポートするヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **curveId** 整数値、受け入れられた TPM_ECC_CURVE 値の 1 つ
- **eccPubX** ポイント X のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubXSz** ポイント X バッファ サイズを指定する word32 型の整数値
- **eccPubY** ポイント Y のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubYSz** ポイント Y バッファ サイズを指定する word32 型の整数値

参考:

- wolfTPM2_LoadPublicKey
- wolfTPM2_LoadRsaPublicKey
- wolfTPM2_ReadPublicKey
- wolfTPM2_LoadEccPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

パブリック部分の TPM 形式を必要としないため、使用を推奨

wolfTPM2_ImportEccPrivateKey

```
WOLFTPM_API int wolfTPM2_ImportEccPrivateKey(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEYBLOB * keyBlob,
    int curveId,
    const byte * eccPubX,
    word32 eccPubXSz,
    const byte * eccPubY,
    word32 eccPubYSz,
    const byte * eccPriv,
    word32 eccPrivSz
)
```

外部の ECC 鍵のプライベート マテリアルをインポートするヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL にすることができ、キーは OWNER 階層の下にインポートされます)
- **keyBlob** WOLFTPM2_KEYBLOB]() 型の空の構造体へのポインター
- **curveId** 整数値、受け入れられた TPM_ECC_CURVE 値の 1 つ
- **eccPubX** ポイント X のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubXSz** ポイント X バッファ サイズを指定する word32 型の整数値
- **eccPubY** ポイント Y のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubYSz** ポイント Y バッファ サイズを指定する word32 型の整数値
- **eccPriv** プライベート マテリアルを含むバイト バッファへのポインタ
- **eccPrivSz** プライベート マテリアル サイズを指定する word32 型の整数値

参考:

- wolfTPM2_LoadEccPrivateKey
- wolfTPM2_LoadEccPrivateKey_ex
- wolfTPM2_LoadPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_LoadEccPrivateKey

```
WOLFTPM_API int wolfTPM2_LoadEccPrivateKey(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEY * key,
    int curveId,
    const byte * eccPubX,
    word32 eccPubXSz,
    const byte * eccPubY,
    word32 eccPubYSz,
    const byte * eccPriv,
    word32 eccPrivSz
)
```

外部の ECC 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL にすることができ、キーは OWNER 階層の下にインポートされます)
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **curveId** 整数値、受け入れられた TPM_ECC_CURVE 値の 1 つ
- **eccPubX** ポイント X のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubXSz** ポイント X バッファ サイズを指定する word32 型の整数値
- **eccPubY** ポイント Y のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubYSz** ポイント Y バッファ サイズを指定する word32 型の整数値
- **eccPriv** プライベート マテリアルを含むバイト バッファへのポインタ
- **eccPrivSz** プライベート マテリアル サイズを指定する word32 型の整数値

参考:

- wolfTPM2_ImportEccPrivateKey
- wolfTPM2_LoadEccPublicKey
- wolfTPM2_LoadPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ReadPublicKey

```
WOLFTPM_API int wolfTPM2_ReadPublicKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const TPM_HANDLE handle
)
```

ハンドルを使用して、読み込まれた TPM オブジェクトのパブリック部分を受け取るヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **handle** ロードされた TPM オブジェクトのハンドルを指定する、TPM_HANDLE 型の整数値

参考:

- wolfTPM2_LoadRsaPublicKey
- wolfTPM2_LoadEccPublicKey
- wolfTPM2_LoadPublicKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 対称鍵の公開部分には、TPM メタデータのみが含まれます

wolfTPM2_CreateKeySeal

```
WOLFTPM_API int wolfTPM2_CreateKeySeal(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEYBLOB * keyBlob,
    WOLFTPM2_HANDLE * parent,
    TPMT_PUBLIC * publicTemplate,
    const byte * auth,
    int authSz,
    const byte * sealData,
    int sealSize
)
```

このラッパーを使用すると、シークレットを TPM 2.0 鍵内に封印できます。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **keyBlob** WOLFTPM2_KEYBLOB]() 型の空の構造体へのポインター
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される 2.0 プライマリ キーを指定します。
- **publicTemplate** wolfTPM2_GetKeyTemplate_KeySeal のいずれかを使用して設定された TPMT_PUBLIC 構造へのポインタ
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値
- **sealData** シールするシークレット (ユーザー データ) を含むバイト バッファーへのポインター
- **sealSize** シールバッファのサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_GetKeyTemplate_KeySeal
- TPM2_Unseal
- wolfTPM2_CreatePrimary

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)

- BAD_FUNC_ARG: 不正な引数

ノート:

シークレットのサイズは 128 バイトを超えることはできません

wolfTPM2_ComputeName

```
WOLFTPM_API int wolfTPM2_ComputeName(
    const TPM2B_PUBLIC * pub,
    TPM2B_NAME * out
)
```

TPM が期待する形式でオブジェクトのパブリック領域のハッシュを生成するヘルパー関数。

パラメータ:

- **pub** TPM オブジェクトの公開部分を含んだ TPM2B_PUBLIC 構造体へのポインター
- **out** 計算された名前を格納するための TPM2B_NAME 型の空の構造体へのポインター

参考:

- wolfTPM2_ImportPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

計算された TPM 名には、TPM_ALG_ID のハッシュが含まれており、パブリックはオブジェクトのものです

wolfTPM2_SensitiveToPrivate

```
WOLFTPM_API int wolfTPM2_SensitiveToPrivate(
    TPM2B_SENSITIVE * sens,
    TPM2B_PRIVATE * priv,
    TPMI_ALG_HASH nameAlg,
    TPM2B_NAME * name,
    const WOLFTPM2_KEY * parentKey,
    TPMT_SYM_DEF_OBJECT * sym,
    TPM2B_ENCRYPTED_SECRET * symSeed
)
```

TPM2B_SENSITIVE を変換するヘルパー関数。

パラメータ:

- **sens** TPM2B_SENSITIVE 型の正しく設定された構造体へのポインター
- **priv** TPM2B_PRIVATE 型の空の構造体へのポインター
- **nameAlg** TPMI_ALG_HASH 型の整数値。有効な TPM2 ハッシュ アルゴリズムを指定します
- **name** TPM2B_NAME 構造体へのポインター
- **parentKey** WOLFTPM2_KEY 構造体へのポインター。parentKey が存在する場合はそれを指定します
- **sym** TPMT_SYM_DEF_OBJECT 型の構造体へのポインター
- **symSeed** TPM2B_ENCRYPTED_SECRET 型の構造体へのポインター

参考:

- wolfTPM2_ImportPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_RsaKey_TpmToWolf

```
WOLFTPM_API int wolfTPM2_RsaKey_TpmToWolf(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * tpmKey,
    RsaKey * wolfKey
)
```

RSA TPM 鍵を抽出し、それを wolfcrypt 鍵に変換します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **tpmKey** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **wolfKey** 変換されたキーを格納する RsaKey 型の空の構造体へのポインター

参考:

- wolfTPM2_RsaKey_WolfToTpm
- wolfTPM2_RsaKey_WolfToTpm_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_RsaKey_TpmToPemPub

```
WOLFTPM_API int wolfTPM2_RsaKey_TpmToPemPub(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * keyBlob,
    byte * pem,
    word32 * pemSz
)
```

公開 RSA TPM 鍵を PEM 形式の公開鍵に変換する 注: pem と tempBuf は、同じサイズの異なるバッファーである必要があります。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **keyBlob** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **pem** PEM 会話の一時ストレージとして使用される、バイト型の配列へのポインター
- **pemSz** 使用済みバッファサイズを格納する整数変数へのポインター

参考:

- wolfTPM2_RsaKey_TpmToWolf
- wolfTPM2_RsaKey_WolfToTpm

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_RsaKey_WolfToTpm

```
WOLFTPM_API int wolfTPM2_RsaKey_WolfToTpm(
    WOLFTPM2_DEV * dev,
    RsaKey * wolfKey,
    WOLFTPM2_KEY * tpmKey
)
```

RSA wolfcrypt 鍵を TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **wolfKey** wolfcrypt キーを保持する RsaKey 型の構造体へのポインター
- **tpmKey** インポートされた TPM キーを保持するための WOLFTPM2_KEY 型の空の構造体へのポインター

参考:

- wolfTPM2_RsaKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 2.0 で使用する wolfcrypt による外部生成鍵の使用を許可します

wolfTPM2_RsaKey_WolfToTpm_ex

```
WOLFTPM_API int wolfTPM2_RsaKey_WolfToTpm_ex(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    RsaKey * wolfKey,
    WOLFTPM2_KEY * tpmKey
)
```

特定のプライマリ キーまたは階層の下で RSA wolfcrypt 鍵を TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_KEY 構造体へのポインター、主キーまたは TPM 階層を指す
- **wolfKey** wolfcrypt キーを保持する RsaKey 型の構造体へのポインター
- **tpmKey** インポートされた TPM キーを保持するための WOLFTPM2_KEY 型の空の構造体へのポインター

参考:

- wolfTPM2_RsaKey_WolfToTpm
- wolfTPM2_RsaKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

wolfTPM で wolfcrypt が生成した鍵を使用できるようにします

wolfTPM2_RsaKey_PubPemToTpm

```
WOLFTPM_API int wolfTPM2_RsaKey_PubPemToTpm(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * tpmKey,
    const byte * pem,
    word32 pemSz
)
```

PEM 形式の公開鍵をファイルから TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **tpmKey** インポートされた TPM キーを保持するための WOLFTPM2_KEY 型の空の構造体へのポインター
- **pem** PEM 形式の公開鍵素材を含む、バイト型の配列へのポインター
- **pemSz** PEM キーデータのサイズを指定する整数変数へのポインター

参考:

- wolfTPM2_RsaKey_WolfToTpm
- wolfTPM2_RsaKey_TpmToPem
- wolfTPM2_RsaKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)

wolfTPM2_EccKey_TpmToWolf

```
WOLFTPM_API int wolfTPM2_EccKey_TpmToWolf(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * tpmKey,
    ecc_key * wolfKey
)
```

ECC TPM 鍵を抽出し、wolfcrypt 鍵に変換します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **tpmKey** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **wolfKey** 変換されたキーを格納するための、ecc_key 型の空の構造体へのポインター

参考:

- wolfTPM2_EccKey_WolfToTpm
- wolfTPM2_EccKey_WolfToTpm_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_EccKey_WolfToTpm

```
WOLFTPM_API int wolfTPM2_EccKey_WolfToTpm(
    WOLFTPM2_DEV * dev,
    ecc_key * wolfKey,
    WOLFTPM2_KEY * tpmKey
)
```

ECC wolfcrypt 鍵を TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **wolfKey** wolfcrypt キーを保持する、ecc_key タイプの構造体へのポインター
- **tpmKey** インポートされた TPM キーを保持するための WOLFTPM2_KEY 型の空の構造体へのポインター

参考:

- wolfTPM2_EccKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 2.0 で使用する wolfcrypt による外部生成鍵の使用を許可します

wolfTPM2_EccKey_WolfToTpm_ex

```
WOLFTPM_API int wolfTPM2_EccKey_WolfToTpm_ex(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * parentKey,
    ecc_key * wolfKey,
    WOLFTPM2_KEY * tpmKey
)
```

ECC wolfcrypt 鍵を特定のプライマリ キーまたは階層の下の TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_KEY 構造体へのポインター、主キーまたは TPM 階層を指す
- **wolfKey** wolfcrypt キーを保持する、ecc_key タイプの構造体へのポインター
- **tpmKey** インポートされた TPM キーを保持するための WOLFTPM2_KEY 型の空の構造体へのポインター

参考:

- wolfTPM2_EccKey_WolfToTPM
- wolfTPM2_EccKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

wolfTPM で wolfcrypt が生成した鍵を使用できるようにします

wolfTPM2_EccKey_WolfToPubPoint

```
WOLFTPM_API int wolfTPM2_EccKey_WolfToPubPoint(
    WOLFTPM2_DEV * dev,
    ecc_key * wolfKey,
    TPM2B_ECC_POINT * pubPoint
)
```

wolfcrypt 鍵から生成された ECC 公開鍵を TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **wolfKey** wolfcrypt 公開 ECC キーを保持する、ecc_key タイプの構造体へのポインター
- **pubPoint** TPM2B_ECC_POINT 型の空の構造体へのポインター

参考: wolfTPM2_EccKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 2.0 で使用する wolfcrypt によって外部で生成された公開 ECC 鍵の使用を許可します

wolfTPM2_SignHash

```
WOLFTPM_API int wolfTPM2_SignHash(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * digest,
    int digestSz,
    byte * sig,
    int * sigSz
)
```

TPM 鍵を使用して任意のデータに署名するヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター material
- **digest** 任意のデータを含むバイトバッファへのポインタ
- **digestSz** ダイジェスト バッファのサイズをバイト単位で指定する整数値
- **sig** 生成された署名を含むバイトバッファへのポインター
- **sigSz** 署名バッファのサイズをバイト単位で指定する整数値

参考:

- verifyHash
- signHashScheme
- verifyHashScheme

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_SignHashScheme

```
WOLFTPM_API int wolfTPM2_SignHashScheme(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * digest,
    int digestSz,
    byte * sig,
    int * sigSz,
```

```
    TPMI_ALG_SIG_SCHEME sigAlg,
    TPMI_ALG_HASH hashAlg
)
```

TPM 鍵を使用して任意のデータに署名し、署名スキームとハッシュ アルゴリズムを指定する高度なヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター material
- **digest** 任意のデータを含むバイトバッファへのポインタ
- **digestSz** ダイジェスト バッファのサイズをバイト単位で指定する整数値
- **sig** 生成された署名を含むバイトバッファへのポインター
- **sigSz** 署名バッファのサイズをバイト単位で指定する整数値
- **sigAlg** サポートされている TPM 2.0 署名スキームを指定する TPMI_ALG_SIG_SCHEME 型の整数値
- **hashAlg** サポートされている TPM 2.0 ハッシュ アルゴリズムを指定する TPMI_ALG_HASH 型の整数値

参考:

- signHash
- verifyHash
- verifyHashScheme

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_VerifyHash

```
WOLFTPM_API int wolfTPM2_VerifyHash(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * sig,
    int sigSz,
    const byte * digest,
    int digestSz
)
```

TPM が生成した署名を検証するためのヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM 2.0 キー マテリアルを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **sig** 生成された署名を含むバイトバッファへのポインター
- **sigSz** 署名バッファのサイズをバイト単位で指定する整数値
- **digest** 符号付きデータを含むバイトバッファへのポインター
- **digestSz** ダイジェスト バッファのサイズをバイト単位で指定する整数値

参考:

- signHash
- signHashScheme
- verifyHashScheme

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_VerifyHashScheme

```
WOLFTPM_API int wolfTPM2_VerifyHashScheme(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * sig,
    int sigSz,
    const byte * digest,
    int digestSz,
    TPMI_ALG_SIG_SCHEME sigAlg,
    TPMI_ALG_HASH hashAlg
)
```

TPM が生成した署名を検証するための高度なヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM 2.0 キー マテリアルを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **sig** 生成された署名を含むバイトバッファへのポインター
- **sigSz** 署名バッファのサイズをバイト単位で指定する整数値
- **digest** 符号付きデータを含むバイトバッファへのポインター
- **digestSz** ダイジェスト バッファのサイズをバイト単位で指定する整数値
- **sigAlg** サポートされている TPM 2.0 署名スキームを指定する TPMI_ALG_SIG_SCHEME 型の整数値
- **hashAlg** サポートされている TPM 2.0 ハッシュ アルゴリズムを指定する TPMI_ALG_HASH 型の整数値

参考:

- signHash
- signHashScheme
- verifyHash

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ECDHGenKey

```
WOLFTPM_API int wolfTPM2_ECDHGenKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * ecdhKey,
    int curve_id,
    const byte * auth,
    int authSz
)
```

Diffie-Hellman 交換用の NULL 階層を持つ ECC 鍵ペアを生成してからロードします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **ecdhKey** WOLFTPM2_KEY 型の空の構造体へのポインター
- **curve_id** 有効な TPM_ECC_CURVE 値を指定する整数値

- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_ECDHGen
- wolfTPM2_ECDHGenZ
- wolfTPM2_ECDHEGenKey
- wolfTPM2_ECDHEGenZ

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ECDHGen

```
WOLFTPM_API int wolfTPM2_ECDHGen(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * privKey,
    TPM2B_ECC_POINT * pubPoint,
    byte * out,
    int * outSz
)
```

一時鍵を生成し、Z (共有シークレット) を計算します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **privKey** WOLFTPM2_KEY 型の構造体へのポインター
- **pubPoint** TPM2B_ECC_POINT 型の空の構造体へのポインター
- **out** 生成された共有シークレットを格納するバイト バッファへのポインター
- **outSz** 共有シークレットのサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_ECDHGenZ
- wolfTPM2_ECDHGenKey
- wolfTPM2_ECDHEGenKey
- wolfTPM2_ECDHEGenZ

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

秘密鍵ハンドルを使用して鍵ペアを生成し、公開ポイントと共有秘密を返すワンショット API

wolfTPM2_ECDHGenZ

```
WOLFTPM_API int wolfTPM2_ECDHGenZ(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * privKey,
    const TPM2B_ECC_POINT * pubPoint,
    byte * out,
    int * outSz
)
```

pubPoint と読み込まれたプライベート ECC 鍵を使用して Z (共有シークレット) を計算します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **privKey** TPM handle を格納する WOLFTPM2_KEY 型の構造体へのポインター
- **pubPoint** TPM2B_ECC_POINT]() 型のデータが取り込まれた構造体へのポインター
- **out** 計算された共有シークレットを格納するバイト バッファへのポインター
- **outSz** 共有シークレットのサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_ECDHGen
- wolfTPM2_ECDHGenKey
- wolfTPM2_ECDHEGenKey
- wolfTPM2_ECDHEGenZ

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ECDHEGenKey

```
WOLFTPM_API int wolfTPM2_ECDHEGenKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * ecdhKey,
    int curve_id
)
```

一時的な ECC 鍵を生成し、配列インデックスを返します (2 フェーズ メソッド)

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **ecdhKey** WOLFTPM2_KEY 型の空の構造体へのポインター
- **curve_id** 有効な TPM_ECC_CURVE 値を指定する整数値

参考:

- wolfTPM2_ECDHEGenZ
- wolfTPM2_ECDHGen
- wolfTPM2_ECDHGenKey
- wolfTPM2_ECDHGenZ

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

One time use key

wolfTPM2_ECDHEGenZ

```
WOLFTPM_API int wolfTPM2_ECDHEGenZ(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEY * ecdhKey,
    const TPM2B_ECC_POINT * pubPoint,
```

```
    byte * out,
    int * outSz
)
```

pubPoint とカウンターを使用して Z (共有シークレット) を計算します (2 フェーズメソッド)

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** プライマリキーの有効な TPM ハンドルを含む WOLFTPM2_KEY 型の構造体へのポインター
- **ecdhKey** TPM handle を格納する WOLFTPM2_KEY 型の構造体へのポインター
- **pubPoint** TPM2B_ECC_POINT 型の空の構造体へのポインター
- **out** 計算された共有シークレットを格納するバイト バッファへのポインター
- **outSz** 共有シークレットのサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_ECDHEGenKey
- wolfTPM2_ECDHGen
- wolfTPM2_ECDHGenKey
- wolfTPM2_ECDHGenZ

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

カウンター、アレイ ID は 1 回だけ使用できます

wolfTPM2_RsaEncrypt

```
WOLFTPM_API int wolfTPM2_RsaEncrypt(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    TPM_ALG_ID padScheme,
    const byte * msg,
    int msgSz,
    byte * out,
    int * outSz
)
```

TPM 2.0 鍵を使用して RSA 暗号化を実行します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター material
- **padScheme** パディング スキームを指定する、TPM_ALG_ID 型の整数値
- **msg** 任意のデータを含むバイトバッファへのポインタ
- **msgSz** 任意のデータ バッファのサイズを指定する整数値
- **out** 暗号化されたデータが格納されるバイトバッファへのポインター
- **outSz** 暗号化されたデータ バッファのサイズを指定する整数値

参考:

- wolfTPM2_RsaDecrypt

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_RsaDecrypt

```
WOLFTPM_API int wolfTPM2_RsaDecrypt(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    TPM_ALG_ID padScheme,
    const byte * in,
    int inSz,
    byte * msg,
    int * msgSz
)
```

TPM 2.0 鍵を使用して RSA 復号を実行します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **padScheme** パディング スキームを指定する、TPM_ALG_ID 型の整数値
- **in** 暗号化されたデータを含むバイトバッファへのポインター
- **inSz** 暗号化されたデータ バッファのサイズを指定する整数値
- **msg** 復号されたデータを含むバイトバッファへのポインター
- **msgSz** 暗号化されたデータ バッファのサイズへのポインター。戻り時に実際のサイズが設定される

参考:

- wolfTPM2_RsaEncrypt

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ReadPCR

```
WOLFTPM_API int wolfTPM2_ReadPCR(
    WOLFTPM2_DEV * dev,
    int pcrIndex,
    int hashAlg,
    byte * digest,
    int * pDigestLen
)
```

指定された TPM 2.0 プラットフォーム構成レジスタ (PCR) の値を読み取る。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **pcrIndex** 0 から 23 までの有効な PCR インデックスを指定する整数値 (TPM の局所性は、アクセスの成功に影響を与える可能性があります)
- **hashAlg** アクセスする TPM_ALG_SHA256 または TPM_ALG_SHA1 レジスタを指定する整数値
- **digest** PCR 値が格納されるバイトバッファへのポインター
- **pDigestLen** ダイジェスト バッファのサイズが格納される整数変数へのポインター

参考: wolfTPM2_ExtendPCR

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

PCR レジスタには SHA256 用と SHA1 用の 2 つのセットがあるため、正しいハッシュ アルゴリズムを指定してください (非推奨ですが、引き続き読み取ることができます)。

wolfTPM2_ExtendPCR

```
WOLFTPM_API int wolfTPM2_ExtendPCR(
    WOLFTPM2_DEV * dev,
    int pcrIndex,
    int hashAlg,
    const byte * digest,
    int digestLen
)
```

ユーザー提供のダイジェストで PCR レジスタを拡張します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **pcrIndex** 0 から 23 までの有効な PCR インデックスを指定する整数値 (TPM の局所性は、アクセスの成功に影響を与える可能性があります)
- **hashAlg** アクセスする TPM_ALG_SHA256 または TPM_ALG_SHA1 レジスタを指定する整数値
- **digest** PCR に拡張されるダイジェスト値を含む、バイトバッファへのポインター
- **digestLen** ダイジェスト バッファのサイズ

参考:

- wolfTPM2_ReadPCR

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

正しいハッシュアルゴリズムを指定してください

wolfTPM2_NVCreateAuth

```
WOLFTPM_API int wolfTPM2_NVCreateAuth(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HANDLE * parent,
    WOLFTPM2_NV * nv,
    word32 nvIndex,
    word32 nvAttributes,
    word32 maxSize,
    const byte * auth,
    int authSz
)
```

TPM の NVRAM にデータを格納するために後で使用する新しい NV インデックスを作成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

- **parent** 新しい NV インデックスの TPM 階層を指定する、WOLFTPM2_HANDLE へのポインター
- **nv** WOLFTPM2_NV]() 型の空の構造体へのポインター。新しい NV インデックスを保持します。
- **nvIndex** 成功時に TPM によって与えられた NV インデックス ハンドルを保持する整数値
- **nvAttributes** 整数値、wolfTPM2_GetNvAttributesTemplate を使用して正しい値を作成します
- **maxSize** この NV インデックスで書き込まれる最大バイト数を指定する整数値
- **auth** この NV インデックスのパスワード認証を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_NVWriteAuth
- wolfTPM2_NVReadAuth
- wolfTPM2_NVDeleteAuth
- wolfTPM2_NVOpen

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

これは、TPM2_NV_DefineSpace の wolfTPM2 ラッパーです。

wolfTPM2_NVWriteAuth

```
WOLFTPM_API int wolfTPM2_NVWriteAuth(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_NV * nv,
    word32 nvIndex,
    byte * dataBuf,
    word32 dataSz,
    word32 offset
)
```

指定されたオフセットで、ユーザー データを NV インデックスに格納します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **nv** WOLFTPM2_NV 型のデータが取り込まれた構造体へのポインター
- **nvIndex** 既存の NV インデックス ハンドル値を保持する整数値
- **dataBuf** TPM の NVRAM に書き込まれるユーザー データを含むバイト バッファーへのポインター
- **dataSz** ユーザーデータバッファのサイズをバイト単位で指定する整数値
- **offset** NV インデックス メモリの開始点からのオフセットを指定する word32 型の整数値。ゼロの場合もあります。

参考:

- wolfTPM2_NVReadAuth
- wolfTPM2_NVCreateAuth
- wolfTPM2_NVDeleteAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

ユーザーデータのサイズは、wolfTPM2_CreateAuth を使用して指定された NV インデックスの maxSize 以下である必要があります

wolfTPM2_NVReadAuth

```
WOLFTPM_API int wolfTPM2_NVReadAuth(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_NV * nv,
    word32 nvIndex,
    byte * dataBuf,
    word32 * pDataSz,
    word32 offset
)
```

指定されたオフセットから開始して、NV インデックスからユーザー データを読み取ります。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **nv** WOLFTPM2_NV 型のデータが取り込まれた構造体へのポインター
- **nvIndex** 既存の NV インデックス ハンドル値を保持する整数値
- **dataBuf** TPM の NVRAM からの読み取りデータを格納するために使用される、空のバイト バッファーへのポインター
- **pDataSz** 整数変数へのポインター。NVRAM から読み取ったデータのサイズ (バイト単位) を格納するために使用されます。
- **offset** NV インデックス メモリの開始点からのオフセットを指定する word32 型の整数値。ゼロの場合もあります。

参考:

- wolfTPM2_NVWriteAuth
- wolfTPM2_NVCreateAuth
- wolfTPM2_NVDeleteAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

ユーザーデータのサイズは、wolfTPM2_CreateAuth を使用して指定された NV インデックスの maxSize 以下である必要があります

wolfTPM2_NVIncrement

```
WOLFTPM_API int wolfTPM2_NVIncrement(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_NV * nv
)
```

NV 一方向カウンターをインクリメントします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **nv** WOLFTPM2_NV 型のデータが取り込まれた構造体へのポインター

参考:

- wolfTPM2_NVOpen

- wolfTPM2_NVCreateAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_NVOpen

```
WOLFTPM_API int wolfTPM2_NVOpen(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_NV * nv,
    word32 nvIndex,
    const byte * auth,
    word32 authSz
)
```

NV を開き、必要な認証と名前ハッシュを入力します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **nv** WOLFTPM2_NV]() 型の空の構造体へのポインター。新しい NV インデックスを保持します。
- **nvIndex** 成功時に TPM によって与えられた NV インデックス ハンドルを保持する整数値
- **auth** この NV インデックスのパスワード認証を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_NVCreateAuth
- wolfTPM2_UnloadHandle

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_NVDeleteAuth

```
WOLFTPM_API int wolfTPM2_NVDeleteAuth(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HANDLE * parent,
    word32 nvIndex
)
```

既存の NV インデックスを破棄します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parent** 新しい NV インデックスの TPM 階層を指定する、WOLFTPM2_HANDLE へのポインター
- **nvIndex** 成功時に TPM によって与えられた NV インデックス ハンドルを保持する整数値

参考:

- wolfTPM2_NVCreateAuth
- wolfTPM2_NVWriteAuth
- wolfTPM2_NVReadAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_NVCreate

```
WOLFTPM_API int wolfTPM2_NVCreate(
    WOLFTPM2_DEV * dev,
    TPM_HANDLE authHandle,
    word32 nvIndex,
    word32 nvAttributes,
    word32 maxSize,
    const byte * auth,
    int authSz
)
```

非推奨です。新しい API を使用してください。

参考:

- wolfTPM2_NVCreateAuth

wolfTPM2_NVWrite

```
WOLFTPM_API int wolfTPM2_NVWrite(
    WOLFTPM2_DEV * dev,
    TPM_HANDLE authHandle,
    word32 nvIndex,
    byte * dataBuf,
    word32 dataSz,
    word32 offset
)
```

非推奨です。新しい API を使用してください。

参考:

- wolfTPM2_NVWriteAuth

wolfTPM2_NVRead

```
WOLFTPM_API int wolfTPM2_NVRead(
    WOLFTPM2_DEV * dev,
    TPM_HANDLE authHandle,
    word32 nvIndex,
    byte * dataBuf,
    word32 * dataSz,
    word32 offset
)
```

非推奨です。新しい API を使用してください。

参考:

- wolfTPM2_NVReadAuth

wolfTPM2_NVDelete

```
WOLFTPM_API int wolfTPM2_NVDelete(
    WOLFTPM2_DEV * dev,
    TPM_HANDLE authHandle,
```

```
    word32 nvIndex
)
```

非推奨です。新しい API を使用してください。

参考:

- wolfTPM2_NVDeleteAuth

wolfTPM2_NVReadPublic

```
WOLFTPM_API int wolfTPM2_NVReadPublic(
    WOLFTPM2_DEV * dev,
    word32 nvIndex,
    TPMS_NV_PUBLIC * nvPublic
)
```

最大サイズなど、nvIndex に関する公開情報を抽出します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **nvIndex** 成功時に TPM によって与えられた NV インデックス ハンドルを保持する整数値
- **nvPublic** TPMS_NV_PUBLIC へのポインター。抽出された nvIndex 公開情報を格納するために使用されます。

参考:

- wolfTPM2_NVCreateAuth
- wolfTPM2_NVDeleteAuth
- wolfTPM2_NVWriteAuth
- wolfTPM2_NVReadAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_NVStoreKey

```
WOLFTPM_API int wolfTPM2_NVStoreKey(
    WOLFTPM2_DEV * dev,
    TPM_HANDLE primaryHandle,
    WOLFTPM2_KEY * key,
    TPM_HANDLE persistentHandle
)
```

TPM 2.0 キーを TPM の NVRAM に格納するヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **primaryHandle** TPM 2.0 階層を指定する整数値。通常は TPM_RH_OWNER
- **key** WOLFTPM2_KEY 型の構造体へのポインター。格納用の TPM 2.0 キーを含みます
- **persistentHandle** 既存の nvIndex を指定する整数値

参考:

- wolfTPM2_NVDeleteKey
- wolfTPM2_NVCreateAuth
- wolfTPM2_NVDeleteAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_NVDeleteKey

```
WOLFTPM_API int wolfTPM2_NVDeleteKey(
    WOLFTPM2_DEV * dev,
    TPM_HANDLE primaryHandle,
    WOLFTPM2_KEY * key
)
```

TPM の NVRAM から TPM 2.0 鍵を削除するヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **primaryHandle** TPM 2.0 階層を指定する整数値。通常は TPM_RH_OWNER
- **key** nvIndex ハンドル値を含む WOLFTPM2_KEY 型の構造体へのポインター

参考:

- wolfTPM2_NVDeleteKey
- wolfTPM2_NVCreateAuth
- wolfTPM2_NVDeleteAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetRng

```
WOLFTPM_API struct WC_RNG * wolfTPM2_GetRng(
    WOLFTPM2_DEV * dev
)
```

wolfTPM に使用される wolfcrypt RNG インスタンスを取得します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

参考:

- wolfTPM2_GetRandom

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

wolfcrypt が有効で、TPM RNG の代わりに使用するように構成されている場合のみ

wolfTPM2_GetRandom

```
WOLFTPM_API int wolfTPM2_GetRandom(
    WOLFTPM2_DEV * dev,
    byte * buf,
```

```
    word32 len
)
```

TPM RNG または wolfcrypt RNG で生成された一連の乱数を取得します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **buf** 生成された乱数を格納するために使用されるバイトバッファへのポインター
- **len** バッファーのサイズ (バイト単位) を格納するために使用される word32 型の整数値

参考:

- wolfTPM2_GetRandom

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM RNG ソースを使用するように WOLFTPM2_USE_HW_RNG を定義します。

wolfTPM2_UnloadHandle

```
WOLFTPM_API int wolfTPM2_UnloadHandle(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HANDLE * handle
)
```

TPM がロードされたオブジェクトを破棄するために使用します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **handle** 有効な TPM 2.0 ハンドル値を持つ WOLFTPM2_HANDLE タイプの構造体へのポインター

参考:

- wolfTPM2_Clear

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_Clear

```
WOLFTPM_API int wolfTPM2_Clear(
    WOLFTPM2_DEV * dev
)
```

wolfTPM と wolfcrypt を初期化解除します (有効な場合)

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

参考:

- wolfTPM2_Clear

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_HashStart

```
WOLFTPM_API int wolfTPM2_HashStart(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HASH * hash,
    TPMI_ALG_HASH hashAlg,
    const byte * usageAuth,
    word32 usageAuthSz
)
```

TPM で生成されたハッシュを開始するヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hash** WOLFTPM2_HASH 構造体へのポインター
- **hashAlg** 有効な TPM 2.0 ハッシュ アルゴリズムを指定する整数値
- **usageAuth** ハッシュを後で使用するための承認を指定する文字列定数へのポインター
- **usageAuthSz** i 承認のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_HashUpdate
- wolfTPM2_HashFinish

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_HashUpdate

```
WOLFTPM_API int wolfTPM2_HashUpdate(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HASH * hash,
    const byte * data,
    word32 dataSz
)
```

TPM で生成されたハッシュを新しいユーザー データで更新します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hash** WOLFTPM2_HASH 構造体へのポインター
- **data** ハッシュに追加されるユーザーデータを含むバイトバッファーへのポインター
- **dataSz** ユーザーデータのサイズをバイト単位で指定する word32 型の整数値

参考:

- wolfTPM2_HashStart
- wolfTPM2_HashFinish

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)

- BAD_FUNC_ARG: 不正な引数

ノート:

認証が正しく設定されていることを確認してください

wolfTPM2_HashFinish

```
WOLFTPM_API int wolfTPM2_HashFinish(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HASH * hash,
    byte * digest,
    word32 * digestSz
)
```

TPM で生成されたハッシュをファイナライズし、ユーザー バッファーでダイジェスト出力を取得します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hash** WOLFTPM2_HASH 構造体へのポインター
- **digest** 結果のダイジェストを格納するために使用されるバイトバッファへのポインター
- **digestSz** ダイジェスト バッファーのサイズへのポインター。返されると、ダイジェスト バッファーに格納されているバイト数に設定されます。

参考:

- wolfTPM2_HashStart
- wolfTPM2_HashUpdate

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

認証が正しく設定されていることを確認してください

wolfTPM2_LoadKeyedHashKey

```
WOLFTPM_API int wolfTPM2_LoadKeyedHashKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    WOLFTPM2_HANDLE * parent,
    int hashAlg,
    const byte * keyBuf,
    word32 keySz,
    const byte * usageAuth,
    word32 usageAuthSz
)
```

通常は HMAC 操作に使用される、KeyedHash 型の新しい TPM 鍵を作成して読み込みます。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** 生成される鍵を格納するための WOLFTPM2_KEY 型の空の構造体へのポインター
- **parent** プライマリキーの有効な TPM ハンドルを含む WOLFTPM2_KEY 型の構造体へのポインター
- **hashAlg** 有効な TPM 2.0 ハッシュ アルゴリズムを指定する整数値
- **keyBuf** 新しい KeyedHash キーの派生値を含むバイト配列へのポインター

- **keySz** keyBuf に格納される派生値のサイズをバイト単位で指定する整数値
- **usageAuth** 新しいキーの承認を指定する文字列定数へのポインター
- **usageAuthSz** i 承認のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_HmacStart
- wolfTPM2_HmacUpdate
- wolfTPM2_HmacFinish

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM を使用して HMAC を生成するには、wolfTPM2_Hmac ラッパーを使用することをお勧めします。

wolfTPM2_HmacStart

```
WOLFTPM_API int wolfTPM2_HmacStart(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HMAC * hmac,
    WOLFTPM2_HANDLE * parent,
    TPMI_ALG_HASH hashAlg,
    const byte * keyBuf,
    word32 keySz,
    const byte * usageAuth,
    word32 usageAuthSz
)
```

TPM で生成された hmac を開始するヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hmac** WOLFTPM2_HMAC 構造体へのポインター
- **parent** プライマリキーの有効な TPM ハンドルを含む WOLFTPM2_KEY 型の構造体へのポインター
- **hashAlg** 有効な TPM 2.0 ハッシュ アルゴリズムを指定する整数値
- **keyBuf** 新しい KeyedHash キーの派生値を含むバイト配列へのポインター
- **keySz** keyBuf に格納される派生値のサイズをバイト単位で指定する整数値
- **usageAuth** 文字列定数へのポインター。後で hmac を使用するための承認を指定します。
- **usageAuthSz** i 承認のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_HmacUpdate
- wolfTPM2_HmacFinish
- wolfTPM2_LoadKeyedHashKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_HmacUpdate

```
WOLFTPM_API int wolfTPM2_HmacUpdate(
    WOLFTPM2_DEV * dev,
```

```
    WOLFTPM2_HMAC * hmac,
    const byte * data,
    word32 dataSz
)
```

TPM で生成された hmac を新しいユーザー データで更新します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hmac** WOLFTPM2_HMAC 構造体へのポインター
- **data** hmac に追加されるユーザー データを含むバイト バッファーへのポインター
- **dataSz** ユーザーデータのサイズをバイト単位で指定する word32 型の整数値

参考:

- wolfTPM2_HmacStart
- wolfTPM2_HMACFinish

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 認証が正しく設定されていることを確認してください

wolfTPM2_HmacFinish

```
WOLFTPM_API int wolfTPM2_HmacFinish(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HMAC * hmac,
    byte * digest,
    word32 * digestSz
)
```

TPM で生成された hmac をファイナライズし、ユーザー バッファーでダイジェスト出力を取得します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hmac** WOLFTPM2_HMAC 構造体へのポインター
- **digest** 結果の hmac ダイジェストを格納するために使用されるバイト バッファへのポインター
- **digestSz** ダイジェストのサイズをバイト単位で指定する word32 型の整数値

参考:

- wolfTPM2_HmacStart
- wolfTPM2_HmacUpdate

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 認証が正しく設定されていることを確認してください

wolfTPM2_LoadSymmetricKey

```
WOLFTPM_API int wolfTPM2_LoadSymmetricKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    int alg,
    const byte * keyBuf,
    word32 keySz
)
```

外部の対称鍵を TPM にロードします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **alg** 有効な TPM 2.0 対称キー アルゴリズム（AES CFB の TPM_ALG_CFB）を指定する整数値。
- **keyBuf** 対称鍵の秘密情報を含むバイト配列へのポインター
- **keySz** eyBuf に格納されているキー マテリアルのサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_EncryptDecryptBlock
- wolfTPM2_EncryptDecrypt
- TPM2_EncryptDecrypt2

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_SetCommand

```
WOLFTPM_API int wolfTPM2_SetCommand(
    WOLFTPM2_DEV * dev,
    TPM_CC commandCode,
    int enableFlag
)
```

他の制限付き TPM コマンドを有効にするために使用される、ベンダー固有の TPM コマンド。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **commandCode** 有効なベンダー コマンドを表す整数値
- **enableFlag** 整数値、ゼロ以外の値は「有効にする」ことを表します

参考: TPM2_GPIO_Config

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_Shutdown

```
WOLFTPM_API int wolfTPM2_Shutdown(
    WOLFTPM2_DEV * dev,
    int doStartup
)
```

TPM をシャットダウンまたはリセットするためのヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **doStartup** 整数値、ゼロ以外の値は「シャットダウン後にスタートアップを実行する」ことを表します

参考: wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

doStartup が設定されている場合、TPM2_Shutdown の直後に TPM2_Startup が実行されます。

wolfTPM2_UnloadHandles

```
WOLFTPM_API int wolfTPM2_UnloadHandles(
    WOLFTPM2_DEV * dev,
    word32 handleStart,
    word32 handleCount
)
```

後続の TPM ハンドルをアンロードするためのワンショット API。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **handleStart** 最初の TPM ハンドルの値を指定する word32 型の整数値
- **handleCount** ハンドル数を指定する word32 型の整数値

参考: wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_UnloadHandles_AllTransient

```
WOLFTPM_API int wolfTPM2_UnloadHandles_AllTransient(
    WOLFTPM2_DEV * dev
)
```

すべての一時的な TPM ハンドルをアンロードするためのワンショット API。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

参考:

- wolfTPM2_UnloadHandles
- wolfTPM2_CreatePrimary

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)

- BAD_FUNC_ARG: 不正な引数

ノート:

プライマリキーが一時オブジェクトとして存在する場合、TPM 鍵を使用する前にそれらを再作成する必要があります

wolfTPM2_GetKeyTemplate_RSA

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA(
    TPMT_PUBLIC * publicTemplate,
    TPMA_OBJECT objectAttributes
)
```

ユーザーが選択したオブジェクト属性に基づいて、新しい RSA 鍵の TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** TPMT_PUBLIC 型の空の構造体へのポインター。新しい RSA テンプレートを格納します。
- **objectAttributes** TPMA_OBJECT タイプの整数値。1 つ以上の属性（TPMA_OBJECT_fixedTPM 等）を含めることができます。

参考:

- wolfTPM2_GetKeyTemplate_ECC
- wolfTPM2_GetKeyTemplate_Symmetric
- wolfTPM2_GetKeyTemplate_KeyedHash
- wolfTPM2_GetKeyTemplate_KeySeal

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_ECC

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC(
    TPMT_PUBLIC * publicTemplate,
    TPMA_OBJECT objectAttributes,
    TPM_ECC_CURVE curve,
    TPM_ALG_ID sigScheme
)
```

ユーザーが選択したオブジェクト属性に基づいて、新しい ECC 鍵の TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** TPMT_PUBLIC 型の空の構造体へのポインター。新しい ECC キー テンプレートを格納します。
- **objectAttributes** TPMA_OBJECT タイプの整数値。1 つ以上の属性（TPMA_OBJECT_fixedTPM 等）を含めることができます。

- **curve** TPM_ECC_CURVE タイプの整数値。TPM がサポートする ECC 曲線 ID を指定します。
- **sigScheme** TPM_ALG_ID 型の整数値。TPM がサポートする署名方式を指定します。

参考:

- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_Symmetric

- wolfTPM2_GetKeyTemplate_KeyedHash
- wolfTPM2_GetKeyTemplate_KeySeal

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_Symmetric

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_Symmetric(
    TPMT_PUBLIC * publicTemplate,
    int keyBits,
    TPM_ALG_ID algMode,
    int isSign,
    int isDecrypt
)
```

新しい対称鍵の TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** TPMT_PUBLIC 型の空の構造体へのポインター。新しい対称キー テンプレートを格納します。
- **keyBits** 整数値。対称キーのサイズを指定します。通常は 128 ビットまたは 256 ビットです。
- **algMode** TPM_ALG_ID 型の整数値。TPM がサポートする対称アルゴリズム（AES CFB の TPM_ALG_CFB）を指定します。

- **isSign** 整数値、ゼロ以外の値は「署名鍵」を表します
- **isDecrypt** 整数値、ゼロ以外の値は「復号鍵」を表します

参考:

- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC
- wolfTPM2_GetKeyTemplate_KeyedHash
- wolfTPM2_GetKeyTemplate_KeySeal

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_KeyedHash

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_KeyedHash(
    TPMT_PUBLIC * publicTemplate,
    TPM_ALG_ID hashAlg,
    int isSign,
    int isDecrypt
)
```

新しい KeyedHash 鍵の TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター
- **hashAlg** TPM_ALG_ID 型の整数値。TPM がサポートするハッシュ アルゴリズム（SHA 256 の場合は TPM_ALG_SHA256）を指定します。

- **isSign** 整数値、ゼロ以外の値は「署名鍵」を表します

- **isDecrypt** 整数値、ゼロ以外の値は「復号鍵」を表します

参考:

- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC
- wolfTPM2_GetKeyTemplate_Symmetric
- wolfTPM2_GetKeyTemplate_KeySeal

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_KeySeal

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_KeySeal(
    TPMT_PUBLIC * publicTemplate,
    TPM_ALG_ID nameAlg
)
```

シークレットを封印するための新しい鍵の TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター
- **nameAlg** TPM_ALG_ID 型の整数値。TPM がサポートするハッシュ アルゴリズム（SHA 256 の場合は TPM_ALG_SHA256）を指定します。

参考:

- wolfTPM2_GetKeyTemplate_ECC
- wolfTPM2_GetKeyTemplate_Symmetric
- wolfTPM2_GetKeyTemplate_KeyedHash
- wolfTPM2_GetKeyTemplate_KeySeal

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

鍵封印には厳しい要件があるため、ほとんどの鍵パラメータはラッパーによって事前に決定されます。

wolfTPM2_GetKeyTemplate_RSA_EK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA_EK(
    TPMT_PUBLIC * publicTemplate
)
```

RSA タイプの TPM 承認鍵を生成するための TPM 公開テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_ECC_EK
- wolfTPM2_GetKeyTemplate_RSA_SRK
- wolfTPM2_GetKeyTemplate_RSA_AIK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_ECC_EK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC_EK(
    TPMT_PUBLIC * publicTemplate
)
```

ECC タイプの TPM 承認鍵を生成するための TPM 公開テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_RSA_EK
- wolfTPM2_GetKeyTemplate_ECC_SRK
- wolfTPM2_GetKeyTemplate_ECC_AIK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_RSA_SRK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA_SRK(
    TPMT_PUBLIC * publicTemplate
)
```

RSA タイプの新しい TPM ストレージ 鍵を生成するための TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_ECC_SRK
- wolfTPM2_GetKeyTemplate_RSA_AIK
- wolfTPM2_GetKeyTemplate_RSA_EK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_ECC_SRK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC_SRK(
    TPMT_PUBLIC * publicTemplate
)
```

ECC タイプの新しい TPM ストレージ 鍵を生成するための TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_RSA_SRK
- wolfTPM2_GetKeyTemplate_ECC_AIK

- wolfTPM2_GetKeyTemplate_ECC_EK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_RSA_AIK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA_AIK(
    TPMT_PUBLIC * publicTemplate
)
```

RSA タイプの新しい TPM Attestation Key を生成するための TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_ECC_AIK
- wolfTPM2_GetKeyTemplate_RSA_SRK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_ECC_AIK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC_AIK(
    TPMT_PUBLIC * publicTemplate
)
```

ECC タイプの新しい TPM Attestation Key を生成するための TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_RSA_AIK
- wolfTPM2_GetKeyTemplate_ECC_SRK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_SetKeyTemplate_Unique

```
WOLFTPM_API int wolfTPM2_SetKeyTemplate_Unique(
    TPMT_PUBLIC * publicTemplate,
    const byte * unique,
    int uniqueSz
)
```

Create または CreatePrimary で使用されるパブリック テンプレートの一意の領域を設定します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

- **unique** パブリック テンプレートの一意の領域を設定するためのバッファーへのオプションのポインター。NULL の場合、バッファはゼロ化されます。
- **uniqueSz** 一意のフィールドを埋めるサイズ。ゼロの場合、キー サイズが使用されます。

参考:

- wolfTPM2_CreateKey
- wolfTPM2_CreatePrimaryKey

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetNvAttributesTemplate

```
WOLFTPM_API int wolfTPM2_GetNvAttributesTemplate(
    TPM_HANDLE auth,
    word32 * nvAttributes
)
```

TPM NV インデックス テンプレートを準備します。

パラメータ:

- **auth** 新しい TPM NV インデックスが作成される TPM 階層を表す整数値
- **nvAttributes** NV 属性を格納するための空の整数変数へのポインター

参考:

- wolfTPM2_CreateAuth
- wolfTPM2_WriteAuth
- wolfTPM2_ReadAuth
- wolfTPM2_DeleteAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CreateEK

```
WOLFTPM_API int wolfTPM2_CreateEK(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * ekKey,
    TPM_ALG_ID alg
)
```

ユーザーが選択したアルゴリズム、RSA または ECC に基づいて、新しい TPM 承認キーを生成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **ekKey** 新しい EK に関する情報を格納するための空の WOLFTPM2_KEY 構造体へのポインター
- **alg** TPM_ALG_RSA または TPM_ALG_ECC のみを指定できます。上記の注を参照してください

参考:

- wolfTPM2_CreateSRK
- wolfTPM2_GetKeyTemplate_RSA_EK
- wolfTPM2_GetKeyTemplate_ECC_EK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

EK に使用できるのは RSA と ECC のみですが、TPM で対称鍵を作成して使用できます

wolfTPM2_CreateSRK

```
WOLFTPM_API int wolfTPM2_CreateSRK(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * srkKey,
    TPM_ALG_ID alg,
    const byte * auth,
    int authSz
)
```

他の TPM キーのストレージ キーとして使用される新しい TPM プライマリ キーを生成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **srkKey** 新しい EK に関する情報を格納するための空の WOLFTPM2_KEY 構造体へのポインター
- **alg** TPM_ALG_RSA または TPM_ALG_ECC のみを指定できます。上記の注を参照してください
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreateEK
- wolfTPM2_CreateAndLoadAIK
- wolfTPM2_GetKeyTemplate_RSA_SRK
- wolfTPM2_GetKeyTemplate_ECC_SRK

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

EK に使用できるのは RSA と ECC のみですが、TPM で対称鍵を作成して使用できます

wolfTPM2_CreateAndLoadAIK

```
WOLFTPM_API int wolfTPM2_CreateAndLoadAIK(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * aikKey,
    TPM_ALG_ID alg,
    WOLFTPM2_KEY * srkKey,
    const byte * auth,
    int authSz
)
```

指定されたストレージ鍵の下に新しい TPM 構成証明鍵を生成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **aikKey** 新しく生成された TPM キーを格納するための空の WOLFTPM2_KEY 構造体へのポインター
- **alg** TPM_ALG_RSA または TPM_ALG_ECC のみを指定できます

- **srkKey** WOLFTPM2_KEY 構造体へのポインター。ロードされたストレージ キーの有効な TPM ハンドルを指します。
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreateSRK
- wolfTPM2_GetKeyTemplate_RSA_AIK
- wolfTPM2_GetKeyTemplate_ECC_AIK

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetTime

```
WOLFTPM_API int wolfTPM2_GetTime(
    WOLFTPM2_KEY * aikKey,
    GetTime_Out * getTimeOut
)
```

TPM 署名付きタイムスタンプを生成するワンショット API。

パラメータ:

- **aikKey** WOLFTPM2_KEY 構造体へのポインター。ロードされた認証キーの有効な TPM ハンドルを含みます
- **getTimeOut** コマンドの出力を格納するための GetTime_Out 型の空の構造体へのポインター

参考:

- wolfTPM2_CreateSRK
- wolfTPM2_GetKeyTemplate_RSA_EK
- wolfTPM2_GetKeyTemplate_ECC_EK

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

この呼び出しの前に、構成証明鍵を生成してロードする必要があります

wolfTPM2_CSR_SetCustomExt

```
WOLFTPM_API int wolfTPM2_CSR_SetCustomExt(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CSR * csr,
    int critical,
    const char * oid,
    const byte * der,
    word32 derSz
)
```

WOLFTPM2_CSR 構造体のカスタム要求拡張 oid と値の使用を設定する証明書署名要求 (CSR) 生成のヘルパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター (not used)
- **csr** WOLFTPM2_CSR 構造体へのポインター
- **critical** 0 の場合、拡張機能はクリティカルとマークされません。それ以外の場合、クリティカルとマークされます。
- **oid** 文字列としてのドット区切りの oid。たとえば、「1.2.840.10045.3.1.7」
- **der** 拡張子のコンテンツの der エンコーディング
- **derSz** der エンコーディングのバイト単位のサイズ

参考:

- wolfTPM2_CSR_SetSubject
- wolfTPM2_CSR_SetKeyUsage
- wolfTPM2_CSR_MakeAndSign
- wolfTPM2_CSR_MakeAndSign_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_SetKeyUsage

```
WOLFTPM_API int wolfTPM2_CSR_SetKeyUsage(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CSR * csr,
    const char * keyUsage
)
```

WOLFTPM2_CSR 構造のキー使用法を設定する証明書署名要求 (CSR) 生成のヘルパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター (not used)
- **csr** WOLFTPM2_CSR 構造体へのポインター
- **keyUsage** コンマ区切りのキー使用属性の文字列リスト。可能な値: any、serverAuth、clientAuth、codeSigning、emailProtection、timeStamping、および OCSPSigning デフォルト: “serverAuth,clientAuth,codeSigning”

参考:

- wolfTPM2_CSR_SetSubject
- wolfTPM2_CSR_SetCustomExt
- wolfTPM2_CSR_MakeAndSign
- wolfTPM2_CSR_MakeAndSign_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_SetSubject

```
WOLFTPM_API int wolfTPM2_CSR_SetSubject(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CSR * csr,
    const char * subject
)
```

WOLFTPM2_CSR 構造体のサブジェクトを設定する証明書署名要求 (CSR) 生成のヘルパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター (not used)
- **csr** WOLFTPM2_CSR 構造体へのポインター
- **subject** /CN= 構文を使用した識別名文字列。例: "/C=US/ST=Washington/L=Seattle/O=wolfSSL/OU=Development/C

参考:

- wolfTPM2_CSR_SetKeyUsage
- wolfTPM2_CSR_SetCustomExt
- wolfTPM2_CSR_MakeAndSign
- wolfTPM2_CSR_MakeAndSign_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_MakeAndSign_ex

```
WOLFTPM_API int wolfTPM2_CSR_MakeAndSign_ex(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CSR * csr,
    WOLFTPM2_KEY * key,
    int outFormat,
    byte * out,
    int outSz,
    int sigType,
    int selfSignCert,
    int devId
)
```

TPM ベースの鍵 (件名と鍵の使用法が既に設定されている WOLFTPM2_KEY 構造) を使用した証明書署名要求 (CSR) 生成のヘルパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **csr** WOLFTPM2_CSR 構造体へのポインター
- **key** WOLFTPM2_KEY 構造体へのポインター
- **outFormat** WOLFSSL_FILETYPE_ASN1 または WOLFSSL_FILETYPE_PEM
- **out** ASN.1/DER または PEM としての CSR の宛先バッファ
- **outSz** 宛先バッファの最大サイズ
- **sigType** keyType (CTC_SHA256wRSA または CTC_SHA256wECDSA) に基づいて SHA2-256 を自動的に選択するには、0 を使用します。可能な値のリストについては、wolfCrypt の「enum Ctc_SigType」を参照してください。
- **selfSignCert** 1 (ゼロ以外) に設定すると、結果は自己署名証明書になります。ゼロ (0) は、CA によって使用される CSR (証明書署名要求) を生成します。
- **devId** 暗号コールバックの登録時に使用されるデバイス識別子。INVALID_DEVID (-2) を使用して、必要な暗号化コールバックを自動的に登録します。

参考:

- wolfTPM2_CSR_SetSubject
- wolfTPM2_CSR_SetKeyUsage
- wolfTPM2_CSR_SetCustomExt
- wolfTPM2_CSR_MakeAndSign

戻り値:

- Success: 正の整数 (出力のサイズ)
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)

- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_MakeAndSign

```
WOLFTPM_API int wolfTPM2_CSR_MakeAndSign(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CSR * csr,
    WOLFTPM2_KEY * key,
    int outFormat,
    byte * out,
    int outSz
)
```

TPM ベースの鍵 (件名と鍵の使用法が既に設定されている WOLFTPM2_KEY 構造) を使用した証明書署名要求 (CSR) 生成のヘルパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **csr** WOLFTPM2_CSR 構造体へのポインター
- **key** WOLFTPM2_KEY 構造体へのポインター
- **outFormat** WOLFSSL_FILETYPE_ASN1 または WOLFSSL_FILETYPE_PEM
- **out** ASN.1/DER または PEM としての CSR の宛先バッファ
- **outSz** 宛先バッファの最大サイズ

参考:

- wolfTPM2_CSR_SetSubject
- wolfTPM2_CSR_SetKeyUsage
- wolfTPM2_CSR_SetCustomExt
- wolfTPM2_CSR_MakeAndSign_ex

戻り値:

- Success: 正の整数 (出力のサイズ)
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_Generate_ex

```
WOLFTPM_API int wolfTPM2_CSR_Generate_ex(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const char * subject,
    const char * keyUsage,
    int outFormat,
    byte * out,
    int outSz,
    int sigType,
    int selfSignCert,
    int devId
)
```

TPM ベースの鍵 (WOLFTPM2_KEY) を使用した証明書署名要求 (CSR) 生成のヘルパー。TPM 鍵に基づいて CSR または自己署名証明書を出力するためのシングル ショット API。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 構造体へのポインター

- **subject** /CN= 構文を使用した識別名文字列。例: "/C=US/ST=Washington/L=Seattle/O=wolfSSL/OU=Development/C
- **keyUsage** コンマ区切りのキー使用属性の文字列リスト。可能な値: any、serverAuth、clientAuth、codeSigning、emailProtection、timeStamping、および OCSPSigning デフォルト: "serverAuth,clientAuth,codeSigning"
- **outFormat** WOLFSSL_FILETYPE_ASN1 または WOLFSSL_FILETYPE_PEM
- **out** ASN.1/DER または PEM としての CSR の宛先バッファ
- **outSz** 宛先バッファの最大サイズ
- **sigType** keyType (CTC_SHA256wRSA または CTC_SHA256wECDSA) に基づいて SHA2-256 を自動的に選択するには、0 を使用します。可能な値のリストについては、wolfCrypt の「enum Ctc_SigType」を参照してください。
- **selfSignCert** 1 (ゼロ以外) に設定すると、結果は自己署名証明書になります。ゼロ (0) は、CA によって使用される CSR (証明書署名要求) を生成します。
- **devId** 暗号コールバックの登録時に使用されるデバイス識別子。INVALID_DEVID (-2) を使用して、必要な暗号化コールバックを自動的に登録します。

参考:

- wolfTPM2_SetCryptoDevCb
- wolfTPM2_CSR_Generate

戻り値:

- Success: 正の整数 (出力のサイズ)
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_Generate

```
WOLFTPM_API int wolfTPM2_CSR_Generate(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const char * subject,
    const char * keyUsage,
    int outFormat,
    byte * out,
    int outSz
)
```

TPM ベースの鍵 (WOLFTPM2_KEY) を使用した証明書署名要求 (CSR) 生成のヘルパー。TPM 鍵に基づいて CSR または自己署名証明書を出力するためのシングル ショット API。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** pointer to a loaded WOLFTPM2_KEY 構造体へのポインター
- **subject** /CN= 構文を使用した識別名文字列。例: "/C=US/ST=Washington/L=Seattle/O=wolfSSL/OU=Development/C
- **keyUsage** コンマ区切りのキー使用属性の文字列リスト。可能な値: any、serverAuth、clientAuth、codeSigning、emailProtection、timeStamping、および OCSPSigning デフォルト: "serverAuth,clientAuth,codeSigning"
- **outFormat** WOLFSSL_FILETYPE_ASN1 または WOLFSSL_FILETYPE_PEM
- **out** ASN.1/DER または PEM としての CSR の宛先バッファ
- **outSz** 宛先バッファの最大サイズ

参考:

- wolfTPM2_SetCryptoDevCb
- wolfTPM2_CSR_Generate_ex

戻り値:

- Success: 正の整数 (出力のサイズ)
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CryptoDevCb

```
WOLFTPM_API int wolfTPM2_CryptoDevCb(
    int devId,
    wc_CryptoInfo * info,
    void * ctx
)
```

クリプトオフロードに TPM を使用するためのリファレンス クリプト コールバック API。このコールバック関数は、wolfTPM2_SetCryptoDevCb または wc_CryptoDev_RegisterDevice を使用して登録されます。

パラメータ:

- **devId** コールバックの登録時に使用される devId。INVALID_DEVID 以外の符号付き整数値
- **info** 暗号タイプとパラメーターに関する詳細情報を含む wc_CryptoInfo 構造を指す
- **ctx** コールバックが wolfTPM2_SetCryptoDevCb に登録されたときに提供されたユーザー コンテキスト

参考:

- wolfTPM2_SetCryptoDevCb
- wolfTPM2_ClearCryptoDevCb

戻り値:

- TPM_RC_SUCCESS: 成功
- CRYPTOCB_UNAVAILABLE: TPM ハードウェアを使用せず、デフォルトのソフトウェア暗号にフォールバックします。
- WC_HW_E: 一般的なハードウェア障害

wolfTPM2_SetCryptoDevCb

```
WOLFTPM_API int wolfTPM2_SetCryptoDevCb(
    WOLFTPM2_DEV * dev,
    CryptoDevCallbackFunc cb,
    TpmCryptoDevCtx * tpmCtx,
    int * pDevId
)
```

暗号コールバック関数を登録し、割り当てられた devId を返します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **cb** wolfTPM2_CryptoDevCb API はテンプレートですが、独自のものを提供することもできます
- **tpmCtx** ユーザー指定のコンテキスト。wolfTPM2_CryptoDevCb には TpmCryptoDevCtx を使用しますが、独自のものにすることもできます。
- **pDevId** 自動的に割り当てられたデバイス ID へのポインター

参考:

- wolfTPM2_CryptoDevCb
- wolfTPM2_ClearCryptoDevCb

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)

- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ClearCryptoDevCb

```
WOLFTPM_API int wolfTPM2_ClearCryptoDevCb(
    WOLFTPM2_DEV * dev,
    int devId
)
```

登録された暗号コールバックをクリアします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **devId** コールバックの登録時に使用される devId

参考:

- wolfTPM2_CryptoDevCb
- wolfTPM2_SetCryptoDevCb

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_New

```
WOLFTPM_API WOLFTPM2_DEV * wolfTPM2_New(
    void
)
```

WOLFTPM2_DEV を割り当てて初期化します。

参考:

- wolfTPM2_Free

戻り値:

- 新しいデバイス構造体へのポインター
- NULL: エラー時

wolfTPM2_Free

```
WOLFTPM_API int wolfTPM2_Free(
    WOLFTPM2_DEV * dev
)
```

wolfTPM2_New によって割り当てられた WOLFTPM2_DEV をクリーンアップして解放します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

参考:

- wolfTPM2_New

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_NewKeyBlob

```
WOLFTPM_API WOLFTPM2_KEYBLOB * wolfTPM2_NewKeyBlob(
    void
)
```

WOLFTPM2_KEYBLOB を割り当てて初期化します。

参考:

- wolfTPM2_FreeKeyBlob

戻り値:

- 新しく初期化されたポインタ WOLFTPM2_KEYBLOB
- エラーの場合は NULL

wolfTPM2_FreeKeyBlob

```
WOLFTPM_API int wolfTPM2_FreeKeyBlob(
    WOLFTPM2_KEYBLOB * blob
)
```

wolfTPM2_NewKeyBlob で割り当てられた WOLFTPM2_KEYBLOB を解放します。

パラメータ:

- **blob** wolfTPM2_NewKeyBlob によって割り当てられた WOLFTPM2_KEYBLOB へのポインター

参考:

- wolfTPM2_NewKeyBlob

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_NewPublicTemplate

```
WOLFTPM_API TPMT_PUBLIC * wolfTPM2_NewPublicTemplate(
    void
)
```

TPMT_PUBLIC 構造体を割り当てて初期化します。

参考:

- wolfTPM2_FreePublicTemplate

戻り値:

- 新しく初期化されたポインタ
- エラーの場合は NULL

wolfTPM2_FreePublicTemplate

```
WOLFTPM_API int wolfTPM2_FreePublicTemplate(
    TPMT_PUBLIC * PublicTemplate
)
```

wolfTPM2_NewPublicTemplate で割り当てられた TPMT_PUBLIC を解放します。

パラメータ:

- **PublicTemplate** wolfTPM2_NewPublicTemplate で割り当てられた TPMT_PUBLIC へのポインター

参考:

- wolfTPM2_NewPublicTemplate

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_NewKey

```
WOLFTPM_API WOLFTPM2_KEY * wolfTPM2_NewKey(
    void
)
```

WOLFTPM2_KEY を割り当てて初期化します。

参考:

- wolfTPM2_FreeKey

戻り値:

- 新しく初期化されたポインタ WOLFTPM2_KEY
- エラーの場合は NULL

wolfTPM2_FreeKey

```
WOLFTPM_API int wolfTPM2_FreeKey(
    WOLFTPM2_KEY * key
)
```

wolfTPM2_NewKey で割り当てられた WOLFTPM2_KEY を解放します。

パラメータ:

- **key** wolfTPM2_NewKey によって割り当てられた WOLFTPM2_KEY へのポインター

参考:

- wolfTPM2_NewKey

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_NewSession

```
WOLFTPM_API WOLFTPM2_SESSION * wolfTPM2_NewSession(
    void
)
```

WOLFTPM2_SESSION を割り当てて初期化します。

参考:

- wolfTPM2_FreeSession

戻り値:

- 新しく初期化されたポインタ WOLFTPM2_SESSION
- エラーの場合は NULL

wolfTPM2_FreeSession

```
WOLFTPM_API int wolfTPM2_FreeSession(
    WOLFTPM2_SESSION * session
)
```

wolfTPM2_NewSession で割り当てられた WOLFTPM2_SESSION を解放します。

パラメータ:

- **blob** wolfTPM2_NewSession によって割り当てられた WOLFTPM2_KEYBLOB へのポインター

参考:

- wolfTPM2_NewSession

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_NewCSR

```
WOLFTPM_API WOLFTPM2_CSR * wolfTPM2_NewCSR(
    void
)
```

WOLFTPM2_CSR を割り当てて初期化します。

参考:

- wolfTPM2_FreeCSR

戻り値:

- 新しく初期化されたポインタ WOLFTPM2_CSR
- エラーの場合は NULL

wolfTPM2_FreeCSR

```
WOLFTPM_API int wolfTPM2_FreeCSR(
    WOLFTPM2_CSR * csr
)
```

wolfTPM2_NewCSR で割り当てられた WOLFTPM2_CSR を解放します。

パラメータ:

- **blob** wolfTPM2_NewCSR によって割り当てられた WOLFTPM2_CSR へのポインター

参考:

- wolfTPM2_NewCSR

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_GetHandleRefFromKey

```
WOLFTPM_API WOLFTPM2_HANDLE * wolfTPM2_GetHandleRefFromKey(
    WOLFTPM2_KEY * key
)
```

WOLFTPM2_KEY から WOLFTPM2_HANDLE を取得します。

パラメータ:

- **key** WOLFTPM2_KEY 構造体へのポインター

戻り値:

- 鍵構造体内のハンドルへのポインター
- キーポインタが NULL の場合は NULL

wolfTPM2_GetHandleRefFromKeyBlob

```
WOLFTPM_API WOLFTPM2_HANDLE * wolfTPM2_GetHandleRefFromKeyBlob(
    WOLFTPM2_KEYBLOB * keyBlob
)
```

WOLFTPM2_KEYBLOB から WOLFTPM2_HANDLE を取得します。

パラメータ:

- **key** WOLFTPM2_KEYBLOB 構造体へのポインター

戻り値:

- キー blob 構造体で内のハンドルへのポインター
- キーポインタが NULL の場合は NULL

wolfTPM2_GetHandleRefFromSession

```
WOLFTPM_API WOLFTPM2_HANDLE * wolfTPM2_GetHandleRefFromSession(
    WOLFTPM2_SESSION * session
)
```

WOLFTPM2_SESSION から WOLFTPM2_HANDLE を取得します。

パラメータ:

- **key** pointer to a WOLFTPM2_SESSION struct

戻り値:

- セッション構造体内のハンドルへのポインター
- キーポインタが NULL の場合は NULL

wolfTPM2_GetHandleValue

```
WOLFTPM_API TPM_HANDLE wolfTPM2_GetHandleValue(
    WOLFTPM2_HANDLE * handle
)
```

WOLFTPM2_HANDLE から 32 ビットのハンドル値を取得します。

パラメータ:

- **handle** WOLFTPM2_HANDLE 構造体へのポインター

戻り値:

- TPM_HANDLE 値

wolfTPM2_SetKeyAuthPassword

```
WOLFTPM_API int wolfTPM2_SetKeyAuthPassword(
    WOLFTPM2_KEY * key,
    const byte * auth,
    int authSz
)
```

鍵の認証データを設定します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **auth** 認証データへのポインター
- **authSz** 認証データの長さ (バイト単位)

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyBlobAsBuffer

```
WOLFTPM_API int wolfTPM2_GetKeyBlobAsBuffer(
    byte * buffer,
    word32 bufferSz,
    WOLFTPM2_KEYBLOB * key
)
```

キーブロブからバイナリ バッファーにデータをマーシャリングします。これは、別のプロセスでロードするため、または電源の再投入後にディスクに保存できます。

パラメータ:

- **buffer** マーシャリングされたキーブロブを格納するバッファへのポインター
- **bufferSz** 上記のバッファのサイズ
- **key** マーシャリングするキーブロブへのポインター

参考:

- wolfTPM2_SetKeyBlobFromBuffer

戻り値:

- 正の整数 (出力のサイズ)
- BUFFER_E: 提供されたバッファに十分なスペースがありません
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_SetKeyBlobFromBuffer

```
WOLFTPM_API int wolfTPM2_SetKeyBlobFromBuffer(
    WOLFTPM2_KEYBLOB * key,
    byte * buffer,
    word32 bufferSz
)
```

データを WOLFTPM2_KEYBLOB 構造体にアンマーシャリングします。これは、wolfTPM2_GetKeyBlobAsBuffer によって以前にマーシャリングされたキーブロブをロードするために使用できます。

パラメータ:

- **key** データをロードしてアンマーシャリングするキーブロブへのポインター
- **buffer** ロード元のマーシャリングされたキーブロブを含むバッファへのポインター
- **bufferSz** 上記のバッファのサイズ

参考:

- wolfTPM2_GetKeyBlobAsBuffer

戻り値:

- TPM_RC_SUCCESS: 成功
- BUFFER_E: バッファーが小さすぎるか、非整列化されていない余分なデータが残っています
- BAD_FUNC_ARG: 不正な引数

4.7.4 属性の詳細

4.7.4.1 variable C

```
C {
#endif


typedef struct WOLFTPM2_HANDLE {
    TPM_HANDLE      hndl;
    TPM2B_AUTH      auth;
    TPMT_SYM_DEF    symmetric;
    TPM2B_NAME      name;
    int             policyAuth;
    unsigned int    nameLoaded : 1;
} WOLFTPM2_HANDLE;
```

4.7.5 ソースコード

```
/* tpm2_wrap.h
 *
 * Copyright (C) 2006-2022 wolfSSL Inc.
 *
 * This file is part of wolfTPM.
 *
 * wolfTPM is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * wolfTPM is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1335, USA
 */

#ifndef __TPM2_WRAP_H__
#define __TPM2_WRAP_H__

#include <wolftpm/tpm2.h>

#ifdef __cplusplus
    extern "C" {
#endif


typedef struct WOLFTPM2_HANDLE {
    TPM_HANDLE      hndl;
    TPM2B_AUTH      auth;       /* Used if policyAuth is not set */
    TPMT_SYM_DEF    symmetric;
    TPM2B_NAME      name;
```

```
    int             policyAuth; /* Handle requires Policy, not password Auth */
    unsigned int    nameLoaded : 1; /* flag to indicate if "name" was loaded
     ↪  and computed */
} WOLFTPM2_HANDLE;

#define TPM_SES_PWD 0xFF /* Session type for Password that fits in one byte */

typedef struct WOLFTPM2_SESSION {
    TPM_ST          type;         /* Trial, Policy or HMAC; or TPM_SES_PWD */
    WOLFTPM2_HANDLE handle;       /* Session handle from StartAuthSession */
    TPM2B_NONCE     nonceTPM;     /* Value from StartAuthSession */
    TPM2B_NONCE     nonceCaller;  /* Fresh nonce at each command */
    TPM2B_DIGEST    salt;         /* User defined */
    TPMI_ALG_HASH   authHash;
} WOLFTPM2_SESSION;

typedef struct WOLFTPM2_DEV {
    TPM2_CTX ctx;
    TPM2_AUTH_SESSION session[MAX_SESSION_NUM];
} WOLFTPM2_DEV;

typedef struct WOLFTPM2_KEY {
    WOLFTPM2_HANDLE   handle;
    TPM2B_PUBLIC      pub;
} WOLFTPM2_KEY;

typedef struct WOLFTPM2_KEYBLOB {
    WOLFTPM2_HANDLE   handle;
    TPM2B_PUBLIC      pub;
    TPM2B_NAME        name;
    TPM2B_PRIVATE     priv;
} WOLFTPM2_KEYBLOB;

typedef struct WOLFTPM2_HASH {
    WOLFTPM2_HANDLE handle;
} WOLFTPM2_HASH;

typedef struct WOLFTPM2_NV {
    WOLFTPM2_HANDLE handle;
} WOLFTPM2_NV;

typedef struct WOLFTPM2_HMAC {
    WOLFTPM2_HASH   hash;
    WOLFTPM2_KEY    key;

    /* option bits */
    word16 hmacKeyLoaded:1;
    word16 hmacKeyKeep:1;
} WOLFTPM2_HMAC;

#ifdef WOLFTPM2_CERT_GEN
typedef struct WOLFTPM2_CSR {
    Cert req;
} WOLFTPM2_CSR;
```

```
#endif

#ifndef WOLFTPM2_MAX_BUFFER
    #define WOLFTPM2_MAX_BUFFER 2048
#endif

typedef struct WOLFTPM2_BUFFER {
    int size;
    byte buffer[WOLFTPM2_MAX_BUFFER];
} WOLFTPM2_BUFFER;

typedef enum WOLFTPM2_MFG {
    TPM_MFG_UNKNOWN = 0,
    TPM_MFG_INFINEON,
    TPM_MFG_STM,
    TPM_MFG_MCHP,
    TPM_MFG_NUVOTON,
    TPM_MFG_NATIONTECH,
} WOLFTPM2_MFG;

typedef struct WOLFTPM2_CAPS {
    WOLFTPM2_MFG mfg;
    char mfgStr[4 + 1];
    char vendorStr[(4 * 4) + 1];
    word32 tpmType;
    word16 fwVerMajor;
    word16 fwVerMinor;
    word32 fwVerVendor;

    /* bits */
    word16 fips140_2 : 1; /* using FIPS mode */
    word16 cc_eal4   : 1; /* Common Criteria EAL4+ */
    word16 req_wait_state : 1; /* requires SPI wait state */
} WOLFTPM2_CAPS;

/* NV Handles */
#define TPM2_NV_RSA_EK_CERT 0x01C00002
#define TPM2_NV_ECC_EK_CERT 0x01C0000A


/* Wrapper API's to simplify TPM use */

/* For devtpm and swtpm builds, the ioCb and userCtx are not used and should be
 ↪  set to NULL */

WOLFTPM_API int wolfTPM2_Test(TPM2HalIoCb ioCb, void* userCtx, WOLFTPM2_CAPS*
 ↪  caps);

WOLFTPM_API int wolfTPM2_Init(WOLFTPM2_DEV* dev, TPM2HalIoCb ioCb, void*
 ↪  userCtx);

WOLFTPM_API int wolfTPM2_OpenExisting(WOLFTPM2_DEV* dev, TPM2HalIoCb ioCb,
 ↪  void* userCtx);
```

```
WOLFTPM_API int wolfTPM2_Cleanup(WOLFTPM2_DEV* dev);

WOLFTPM_API int wolfTPM2_Cleanup_ex(WOLFTPM2_DEV* dev, int doShutdown);

WOLFTPM_API int wolfTPM2_GetTpmDevId(WOLFTPM2_DEV* dev);

WOLFTPM_API int wolfTPM2_SelfTest(WOLFTPM2_DEV* dev);

WOLFTPM_API int wolfTPM2_GetCapabilities(WOLFTPM2_DEV* dev, WOLFTPM2_CAPS*
 ↪  caps);

WOLFTPM_API int wolfTPM2_UnsetAuth(WOLFTPM2_DEV* dev, int index);

WOLFTPM_API int wolfTPM2_SetAuth(WOLFTPM2_DEV* dev, int index,
    TPM_HANDLE sessionHandle, const TPM2B_AUTH* auth, TPMA_SESSION
 ↪  sessionAttributes,
    const TPM2B_NAME* name);

WOLFTPM_API int wolfTPM2_SetAuthPassword(WOLFTPM2_DEV* dev, int index, const
 ↪  TPM2B_AUTH* auth);

WOLFTPM_API int wolfTPM2_SetAuthHandle(WOLFTPM2_DEV* dev, int index, const
 ↪  WOLFTPM2_HANDLE* handle);

WOLFTPM_API int wolfTPM2_SetAuthSession(WOLFTPM2_DEV* dev, int index,
    const WOLFTPM2_SESSION* tpmSession, TPMA_SESSION sessionAttributes);

WOLFTPM_API int wolfTPM2_SetAuthHandleName(WOLFTPM2_DEV* dev, int index, const
 ↪  WOLFTPM2_HANDLE* handle);

WOLFTPM_API int wolfTPM2_StartSession(WOLFTPM2_DEV* dev,
    WOLFTPM2_SESSION* session, WOLFTPM2_KEY* tpmKey,
    WOLFTPM2_HANDLE* bind, TPM_SE sesType, int encDecAlg);

WOLFTPM_API int wolfTPM2_CreateAuthSession_EkPolicy(WOLFTPM2_DEV* dev,
                                                    WOLFTPM2_SESSION* tpmSession);

WOLFTPM_API int wolfTPM2_CreatePrimaryKey(WOLFTPM2_DEV* dev,
    WOLFTPM2_KEY* key, TPM_HANDLE primaryHandle, TPMT_PUBLIC* publicTemplate,
    const byte* auth, int authSz);

WOLFTPM_API int wolfTPM2_ChangeAuthKey(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    WOLFTPM2_HANDLE* parent, const byte* auth, int authSz);

WOLFTPM_API int wolfTPM2_CreateKey(WOLFTPM2_DEV* dev,
    WOLFTPM2_KEYBLOB* keyBlob, WOLFTPM2_HANDLE* parent,
    TPMT_PUBLIC* publicTemplate, const byte* auth, int authSz);

WOLFTPM_API int wolfTPM2_LoadKey(WOLFTPM2_DEV* dev,
    WOLFTPM2_KEYBLOB* keyBlob, WOLFTPM2_HANDLE* parent);

WOLFTPM_API int wolfTPM2_CreateAndLoadKey(WOLFTPM2_DEV* dev,
    WOLFTPM2_KEY* key, WOLFTPM2_HANDLE* parent, TPMT_PUBLIC* publicTemplate,
    const byte* auth, int authSz);
```

```
WOLFTPM_API int wolfTPM2_CreateLoadedKey(WOLFTPM2_DEV* dev, WOLFTPM2_KEYBLOB*
 ↪  keyBlob,
    WOLFTPM2_HANDLE* parent, TPMT_PUBLIC* publicTemplate,
    const byte* auth, int authSz);

WOLFTPM_API int wolfTPM2_LoadPublicKey(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    const TPM2B_PUBLIC* pub);

WOLFTPM_API int wolfTPM2_LoadPrivateKey(WOLFTPM2_DEV* dev,
    const WOLFTPM2_KEY* parentKey, WOLFTPM2_KEY* key, const TPM2B_PUBLIC* pub,
    TPM2B_SENSITIVE* sens);

WOLFTPM_API int wolfTPM2_ImportPrivateKey(WOLFTPM2_DEV* dev,
    const WOLFTPM2_KEY* parentKey, WOLFTPM2_KEYBLOB* keyBlob, const
     ↪  TPM2B_PUBLIC* pub,
    TPM2B_SENSITIVE* sens);

WOLFTPM_API int wolfTPM2_LoadRsaPublicKey(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    const byte* rsaPub, word32 rsaPubSz, word32 exponent);

WOLFTPM_API int wolfTPM2_LoadRsaPublicKey_ex(WOLFTPM2_DEV* dev, WOLFTPM2_KEY*
 ↪  key,
    const byte* rsaPub, word32 rsaPubSz, word32 exponent,
    TPMI_ALG_RSA_SCHEME scheme, TPMI_ALG_HASH hashAlg);

WOLFTPM_API int wolfTPM2_ImportRsaPrivateKey(WOLFTPM2_DEV* dev,
    const WOLFTPM2_KEY* parentKey, WOLFTPM2_KEYBLOB* keyBlob,
    const byte* rsaPub, word32 rsaPubSz, word32 exponent,
    const byte* rsaPriv, word32 rsaPrivSz,
    TPMI_ALG_RSA_SCHEME scheme, TPMI_ALG_HASH hashAlg);

WOLFTPM_API int wolfTPM2_LoadRsaPrivateKey(WOLFTPM2_DEV* dev,
    const WOLFTPM2_KEY* parentKey, WOLFTPM2_KEY* key,
    const byte* rsaPub, word32 rsaPubSz, word32 exponent,
    const byte* rsaPriv, word32 rsaPrivSz);

WOLFTPM_API int wolfTPM2_LoadRsaPrivateKey_ex(WOLFTPM2_DEV* dev,
    const WOLFTPM2_KEY* parentKey, WOLFTPM2_KEY* key,
    const byte* rsaPub, word32 rsaPubSz, word32 exponent,
    const byte* rsaPriv, word32 rsaPrivSz,
    TPMI_ALG_RSA_SCHEME scheme, TPMI_ALG_HASH hashAlg);

WOLFTPM_API int wolfTPM2_LoadEccPublicKey(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    int curveId, const byte* eccPubX, word32 eccPubXSz,
    const byte* eccPubY, word32 eccPubYSz);

WOLFTPM_API int wolfTPM2_ImportEccPrivateKey(WOLFTPM2_DEV* dev,
    const WOLFTPM2_KEY* parentKey, WOLFTPM2_KEYBLOB* keyBlob,
    int curveId, const byte* eccPubX, word32 eccPubXSz,
    const byte* eccPubY, word32 eccPubYSz,
    const byte* eccPriv, word32 eccPrivSz);

WOLFTPM_API int wolfTPM2_LoadEccPrivateKey(WOLFTPM2_DEV* dev,
```

```
    const WOLFTPM2_KEY* parentKey, WOLFTPM2_KEY* key,
    int curveId, const byte* eccPubX, word32 eccPubXSz,
    const byte* eccPubY, word32 eccPubYSz,
    const byte* eccPriv, word32 eccPrivSz);

WOLFTPM_API int wolfTPM2_ReadPublicKey(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    const TPM_HANDLE handle);

WOLFTPM_API int wolfTPM2_CreateKeySeal(WOLFTPM2_DEV* dev,
    WOLFTPM2_KEYBLOB* keyBlob, WOLFTPM2_HANDLE* parent,
    TPMT_PUBLIC* publicTemplate, const byte* auth, int authSz,
    const byte* sealData, int sealSize);

WOLFTPM_API int wolfTPM2_ComputeName(const TPM2B_PUBLIC* pub, TPM2B_NAME* out);

WOLFTPM_API int wolfTPM2_SensitiveToPrivate(TPM2B_SENSITIVE* sens,
 ↪  TPM2B_PRIVATE* priv,
    TPMI_ALG_HASH nameAlg, TPM2B_NAME* name, const WOLFTPM2_KEY* parentKey,
    TPMT_SYM_DEF_OBJECT* sym, TPM2B_ENCRYPTED_SECRET* symSeed);

#ifndef WOLFTPM2_NO_WOLFCRYPT
#ifndef NO_RSA

WOLFTPM_API int wolfTPM2_RsaKey_TpmToWolf(WOLFTPM2_DEV* dev, WOLFTPM2_KEY*
 ↪  tpmKey,
    RsaKey* wolfKey);

WOLFTPM_API int wolfTPM2_RsaKey_TpmToPemPub(WOLFTPM2_DEV* dev,
                                            WOLFTPM2_KEY* keyBlob,
                                            byte* pem, word32* pemSz);

WOLFTPM_API int wolfTPM2_RsaKey_WolfToTpm(WOLFTPM2_DEV* dev, RsaKey* wolfKey,
    WOLFTPM2_KEY* tpmKey);

WOLFTPM_API int wolfTPM2_RsaKey_WolfToTpm_ex(WOLFTPM2_DEV* dev,
    const WOLFTPM2_KEY* parentKey, RsaKey* wolfKey, WOLFTPM2_KEY* tpmKey);

WOLFTPM_API int wolfTPM2_RsaKey_PubPemToTpm(WOLFTPM2_DEV* dev,
    WOLFTPM2_KEY* tpmKey, const byte* pem, word32 pemSz);
#endif
#ifdef HAVE_ECC

WOLFTPM_API int wolfTPM2_EccKey_TpmToWolf(WOLFTPM2_DEV* dev, WOLFTPM2_KEY*
 ↪  tpmKey,
    ecc_key* wolfKey);

WOLFTPM_API int wolfTPM2_EccKey_WolfToTpm(WOLFTPM2_DEV* dev, ecc_key* wolfKey,
    WOLFTPM2_KEY* tpmKey);

WOLFTPM_API int wolfTPM2_EccKey_WolfToTpm_ex(WOLFTPM2_DEV* dev, WOLFTPM2_KEY*
 ↪  parentKey,
    ecc_key* wolfKey, WOLFTPM2_KEY* tpmKey);
```

```
WOLFTPM_API int wolfTPM2_EccKey_WolfToPubPoint(WOLFTPM2_DEV* dev, ecc_key*
 ↪ wolfKey,
    TPM2B_ECC_POINT* pubPoint);
#endif
#endif

WOLFTPM_API int wolfTPM2_SignHash(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    const byte* digest, int digestSz, byte* sig, int* sigSz);

WOLFTPM_API int wolfTPM2_SignHashScheme(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    const byte* digest, int digestSz, byte* sig, int* sigSz,
    TPMI_ALG_SIG_SCHEME sigAlg, TPMI_ALG_HASH hashAlg);

WOLFTPM_API int wolfTPM2_VerifyHash(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    const byte* sig, int sigSz, const byte* digest, int digestSz);

WOLFTPM_API int wolfTPM2_VerifyHash_ex(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    const byte* sig, int sigSz, const byte* digest, int digestSz,
    int hashAlg);

WOLFTPM_API int wolfTPM2_VerifyHashScheme(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    const byte* sig, int sigSz, const byte* digest, int digestSz,
    TPMI_ALG_SIG_SCHEME sigAlg, TPMI_ALG_HASH hashAlg);

WOLFTPM_API int wolfTPM2_ECDHGenKey(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* ecdhKey,
    int curve_id, const byte* auth, int authSz);

WOLFTPM_API int wolfTPM2_ECDHGen(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* privKey,
    TPM2B_ECC_POINT* pubPoint, byte* out, int* outSz);

WOLFTPM_API int wolfTPM2_ECDHGenZ(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* privKey,
    const TPM2B_ECC_POINT* pubPoint, byte* out, int* outSz);

WOLFTPM_API int wolfTPM2_ECDHEGenKey(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* ecdhKey,
    int curve_id);

WOLFTPM_API int wolfTPM2_ECDHEGenZ(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* parentKey,
    WOLFTPM2_KEY* ecdhKey, const TPM2B_ECC_POINT* pubPoint,
    byte* out, int* outSz);

WOLFTPM_API int wolfTPM2_RsaEncrypt(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    TPM_ALG_ID padScheme, const byte* msg, int msgSz, byte* out, int* outSz);

WOLFTPM_API int wolfTPM2_RsaDecrypt(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    TPM_ALG_ID padScheme, const byte* in, int inSz, byte* msg, int* msgSz);


WOLFTPM_API int wolfTPM2_ReadPCR(WOLFTPM2_DEV* dev,
    int pcrIndex, int hashAlg, byte* digest, int* pDigestLen);

WOLFTPM_API int wolfTPM2_ExtendPCR(WOLFTPM2_DEV* dev, int pcrIndex, int
 ↪ hashAlg,
    const byte* digest, int digestLen);
```

```
/* Newer API's that use WOLFTPM2_NV context and support auth */

WOLFTPM_API int wolfTPM2_NVCreateAuth(WOLFTPM2_DEV* dev, WOLFTPM2_HANDLE*
 ↪  parent,
    WOLFTPM2_NV* nv, word32 nvIndex, word32 nvAttributes, word32 maxSize,
    const byte* auth, int authSz);

WOLFTPM_API int wolfTPM2_NVWriteAuth(WOLFTPM2_DEV* dev, WOLFTPM2_NV* nv,
    word32 nvIndex, byte* dataBuf, word32 dataSz, word32 offset);

WOLFTPM_API int wolfTPM2_NVReadAuth(WOLFTPM2_DEV* dev, WOLFTPM2_NV* nv,
    word32 nvIndex, byte* dataBuf, word32* pDataSz, word32 offset);

WOLFTPM_API int wolfTPM2_NVIncrement(WOLFTPM2_DEV* dev, WOLFTPM2_NV* nv);

WOLFTPM_API int wolfTPM2_NVOpen(WOLFTPM2_DEV* dev, WOLFTPM2_NV* nv,
    word32 nvIndex, const byte* auth, word32 authSz);

WOLFTPM_API int wolfTPM2_NVDeleteAuth(WOLFTPM2_DEV* dev, WOLFTPM2_HANDLE*
 ↪  parent,
    word32 nvIndex);

/* older API's with improper auth support, kept only for backwards
 ↪  compatibility */
WOLFTPM_API int wolfTPM2_NVCreate(WOLFTPM2_DEV* dev, TPM_HANDLE authHandle,
    word32 nvIndex, word32 nvAttributes, word32 maxSize, const byte* auth, int
 ↪  authSz);
WOLFTPM_API int wolfTPM2_NVWrite(WOLFTPM2_DEV* dev, TPM_HANDLE authHandle,
    word32 nvIndex, byte* dataBuf, word32 dataSz, word32 offset);
WOLFTPM_API int wolfTPM2_NVRead(WOLFTPM2_DEV* dev, TPM_HANDLE authHandle,
    word32 nvIndex, byte* dataBuf, word32* dataSz, word32 offset);
WOLFTPM_API int wolfTPM2_NVDelete(WOLFTPM2_DEV* dev, TPM_HANDLE authHandle,
    word32 nvIndex);

WOLFTPM_API int wolfTPM2_NVReadPublic(WOLFTPM2_DEV* dev, word32 nvIndex,
    TPMS_NV_PUBLIC* nvPublic);

WOLFTPM_API int wolfTPM2_NVStoreKey(WOLFTPM2_DEV* dev, TPM_HANDLE
 ↪  primaryHandle,
    WOLFTPM2_KEY* key, TPM_HANDLE persistentHandle);

WOLFTPM_API int wolfTPM2_NVDeleteKey(WOLFTPM2_DEV* dev, TPM_HANDLE
 ↪  primaryHandle,
    WOLFTPM2_KEY* key);

WOLFTPM_API struct WC_RNG* wolfTPM2_GetRng(WOLFTPM2_DEV* dev);

WOLFTPM_API int wolfTPM2_GetRandom(WOLFTPM2_DEV* dev, byte* buf, word32 len);

WOLFTPM_API int wolfTPM2_UnloadHandle(WOLFTPM2_DEV* dev, WOLFTPM2_HANDLE*
 ↪  handle);

WOLFTPM_API int wolfTPM2_Clear(WOLFTPM2_DEV* dev);
```

```
WOLFTPM_API int wolfTPM2_HashStart(WOLFTPM2_DEV* dev, WOLFTPM2_HASH* hash,
    TPMI_ALG_HASH hashAlg, const byte* usageAuth, word32 usageAuthSz);

WOLFTPM_API int wolfTPM2_HashUpdate(WOLFTPM2_DEV* dev, WOLFTPM2_HASH* hash,
    const byte* data, word32 dataSz);

WOLFTPM_API int wolfTPM2_HashFinish(WOLFTPM2_DEV* dev, WOLFTPM2_HASH* hash,
    byte* digest, word32* digestSz);

WOLFTPM_API int wolfTPM2_LoadKeyedHashKey(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    WOLFTPM2_HANDLE* parent, int hashAlg, const byte* keyBuf, word32 keySz,
    const byte* usageAuth, word32 usageAuthSz);

WOLFTPM_API int wolfTPM2_HmacStart(WOLFTPM2_DEV* dev, WOLFTPM2_HMAC* hmac,
    WOLFTPM2_HANDLE* parent, TPMI_ALG_HASH hashAlg, const byte* keyBuf, word32
 ↪  keySz,
    const byte* usageAuth, word32 usageAuthSz);

WOLFTPM_API int wolfTPM2_HmacUpdate(WOLFTPM2_DEV* dev, WOLFTPM2_HMAC* hmac,
    const byte* data, word32 dataSz);

WOLFTPM_API int wolfTPM2_HmacFinish(WOLFTPM2_DEV* dev, WOLFTPM2_HMAC* hmac,
    byte* digest, word32* digestSz);

WOLFTPM_API int wolfTPM2_LoadSymmetricKey(WOLFTPM2_DEV* dev,
    WOLFTPM2_KEY* key, int alg, const byte* keyBuf, word32 keySz);

#define WOLFTPM2_ENCRYPT NO
#define WOLFTPM2_DECRYPT YES
WOLFTPM_API int wolfTPM2_EncryptDecryptBlock(WOLFTPM2_DEV* dev, WOLFTPM2_KEY*
 ↪  key,
    const byte* in, byte* out, word32 inOutSz, byte* iv, word32 ivSz,
    int isDecrypt);
WOLFTPM_API int wolfTPM2_EncryptDecrypt(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    const byte* in, byte* out, word32 inOutSz,
    byte* iv, word32 ivSz, int isDecrypt);

WOLFTPM_API int wolfTPM2_SetCommand(WOLFTPM2_DEV* dev, TPM_CC commandCode,
    int enableFlag);

WOLFTPM_API int wolfTPM2_Shutdown(WOLFTPM2_DEV* dev, int doStartup);

WOLFTPM_API int wolfTPM2_UnloadHandles(WOLFTPM2_DEV* dev, word32 handleStart,
    word32 handleCount);

WOLFTPM_API int wolfTPM2_UnloadHandles_AllTransient(WOLFTPM2_DEV* dev);

/* Utility functions */

WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA(TPMT_PUBLIC* publicTemplate,
    TPMA_OBJECT objectAttributes);

WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC(TPMT_PUBLIC* publicTemplate,
    TPMA_OBJECT objectAttributes, TPM_ECC_CURVE curve, TPM_ALG_ID sigScheme);
```

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_Symmetric(TPMT_PUBLIC* publicTemplate,
    int keyBits, TPM_ALG_ID algMode, int isSign, int isDecrypt);

WOLFTPM_API int wolfTPM2_GetKeyTemplate_KeyedHash(TPMT_PUBLIC* publicTemplate,
    TPM_ALG_ID hashAlg, int isSign, int isDecrypt);

WOLFTPM_API int wolfTPM2_GetKeyTemplate_KeySeal(TPMT_PUBLIC* publicTemplate,
 ↪  TPM_ALG_ID nameAlg);

WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA_EK(TPMT_PUBLIC* publicTemplate);

WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC_EK(TPMT_PUBLIC* publicTemplate);

WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA_SRK(TPMT_PUBLIC* publicTemplate);

WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC_SRK(TPMT_PUBLIC* publicTemplate);

WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA_AIK(TPMT_PUBLIC* publicTemplate);

WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC_AIK(TPMT_PUBLIC* publicTemplate);

WOLFTPM_API int wolfTPM2_SetKeyTemplate_Unique(TPMT_PUBLIC* publicTemplate,
 ↪  const byte* unique, int uniqueSz);

WOLFTPM_API int wolfTPM2_GetNvAttributesTemplate(TPM_HANDLE auth, word32*
 ↪  nvAttributes);

WOLFTPM_API int wolfTPM2_CreateEK(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* ekKey,
 ↪  TPM_ALG_ID alg);

WOLFTPM_API int wolfTPM2_CreateSRK(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* srkKey,
 ↪  TPM_ALG_ID alg,
    const byte* auth, int authSz);
WOLFTPM_API int wolfTPM2_CreateAndLoadAIK(WOLFTPM2_DEV* dev, WOLFTPM2_KEY*
 ↪  aikKey,
    TPM_ALG_ID alg, WOLFTPM2_KEY* srkKey, const byte* auth, int authSz);

WOLFTPM_API int wolfTPM2_GetTime(WOLFTPM2_KEY* aikKey, GetTime_Out*
 ↪  getTimeOut);


#ifdef WOLFTPM2_CERT_GEN

WOLFTPM_API int wolfTPM2_CSR_SetCustomExt(WOLFTPM2_DEV* dev, WOLFTPM2_CSR* csr,
    int critical, const char *oid, const byte *der, word32 derSz);

WOLFTPM_API int wolfTPM2_CSR_SetKeyUsage(WOLFTPM2_DEV* dev, WOLFTPM2_CSR* csr,
    const char* keyUsage);

WOLFTPM_API int wolfTPM2_CSR_SetSubject(WOLFTPM2_DEV* dev, WOLFTPM2_CSR* csr,
    const char* subject);
```

```
WOLFTPM_API int wolfTPM2_CSR_MakeAndSign_ex(WOLFTPM2_DEV* dev, WOLFTPM2_CSR*
 ↪ csr,
    WOLFTPM2_KEY* key, int outFormat, byte* out, int outSz,
    int sigType, int selfSignCert, int devId);

WOLFTPM_API int wolfTPM2_CSR_MakeAndSign(WOLFTPM2_DEV* dev, WOLFTPM2_CSR* csr,
    WOLFTPM2_KEY* key, int outFormat, byte* out, int outSz);

WOLFTPM_API int wolfTPM2_CSR_Generate_ex(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    const char* subject, const char* keyUsage, int outFormat,
    byte* out, int outSz, int sigType, int selfSignCert, int devId);

WOLFTPM_API int wolfTPM2_CSR_Generate(WOLFTPM2_DEV* dev, WOLFTPM2_KEY* key,
    const char* subject, const char* keyUsage, int outFormat,
    byte* out, int outSz);

#endif /* WOLFTPM2_CERT_GEN */


/* moved to tpm.h native code. macros here for backwards compatibility */
#define wolfTPM2_SetupPCRSel  TPM2_SetupPCRSel
#define wolfTPM2_GetAlgName   TPM2_GetAlgName
#define wolfTPM2_GetRCString  TPM2_GetRCString
#define wolfTPM2_GetCurveSize TPM2_GetCurveSize

/* for salted auth sessions */
WOLFTPM_LOCAL int wolfTPM2_RSA_Salt(struct WOLFTPM2_DEV* dev, WOLFTPM2_KEY*
 ↪ tpmKey,
    TPM2B_DIGEST *salt, TPM2B_ENCRYPTED_SECRET *encSalt, TPMT_PUBLIC
 ↪ *publicArea);
WOLFTPM_LOCAL int wolfTPM2_EncryptSalt(struct WOLFTPM2_DEV* dev, WOLFTPM2_KEY*
 ↪ tpmKey,
    StartAuthSession_In* in, TPM2B_AUTH* bindAuth, TPM2B_DIGEST* salt);


#ifdef WOLFTPM_CRYPTOCB
struct TpmCryptoDevCtx;
typedef int (*CheckWolfKeyCallbackFunc)(wc_CryptoInfo* info, struct
 ↪ TpmCryptoDevCtx* ctx);

typedef struct TpmCryptoDevCtx {
    WOLFTPM2_DEV* dev;
#ifndef NO_RSA
    WOLFTPM2_KEY* rsaKey;  /* RSA */
#endif
#ifdef HAVE_ECC
    WOLFTPM2_KEY* eccKey;  /* ECDSA */
    #ifndef WOLFTPM2_USE_SW_ECDHE
    WOLFTPM2_KEY* ecdhKey; /* ECDH */
    #endif
#endif
    WOLFTPM2_KEY* storageKey;
#ifdef WOLFTPM_USE_SYMMETRIC
```

```
    unsigned short useSymmetricOnTPM:1; /* if set indicates desire to use
     ↪ symmetric algorithms on TPM */
#endif
    unsigned short useFIPSMode:1; /* if set requires FIPS mode on TPM and no
     ↪ fallback to software algos */
} TpmCryptoDevCtx;

WOLFTPM_API int wolfTPM2_CryptoDevCb(int devId, wc_CryptoInfo* info, void*
 ↪ ctx);

WOLFTPM_API int wolfTPM2_SetCryptoDevCb(WOLFTPM2_DEV* dev,
 ↪ CryptoDevCallbackFunc cb,
    TpmCryptoDevCtx* tpmCtx, int* pDevId);

WOLFTPM_API int wolfTPM2_ClearCryptoDevCb(WOLFTPM2_DEV* dev, int devId);

#endif /* WOLFTPM_CRYPTOCB */

#ifndef WOLFTPM2_NO_HEAP

WOLFTPM_API WOLFTPM2_DEV* wolfTPM2_New(void);

WOLFTPM_API int wolfTPM2_Free(WOLFTPM2_DEV *dev);

WOLFTPM_API WOLFTPM2_KEYBLOB* wolfTPM2_NewKeyBlob(void);

WOLFTPM_API int wolfTPM2_FreeKeyBlob(WOLFTPM2_KEYBLOB* blob);

WOLFTPM_API TPMT_PUBLIC* wolfTPM2_NewPublicTemplate(void);

WOLFTPM_API int wolfTPM2_FreePublicTemplate(TPMT_PUBLIC* PublicTemplate);


WOLFTPM_API WOLFTPM2_KEY* wolfTPM2_NewKey(void);

WOLFTPM_API int wolfTPM2_FreeKey(WOLFTPM2_KEY* key);


WOLFTPM_API WOLFTPM2_SESSION* wolfTPM2_NewSession(void);

WOLFTPM_API int wolfTPM2_FreeSession(WOLFTPM2_SESSION* session);

#ifdef WOLFTPM2_CERT_GEN

WOLFTPM_API WOLFTPM2_CSR* wolfTPM2_NewCSR(void);
WOLFTPM_API int wolfTPM2_FreeCSR(WOLFTPM2_CSR* csr);
#endif
#endif /* !WOLFTPM2_NO_HEAP */

WOLFTPM_API WOLFTPM2_HANDLE* wolfTPM2_GetHandleRefFromKey(WOLFTPM2_KEY* key);

WOLFTPM_API WOLFTPM2_HANDLE*
 ↪ wolfTPM2_GetHandleRefFromKeyBlob(WOLFTPM2_KEYBLOB* keyBlob);
```

```
WOLFTPM_API WOLFTPM2_HANDLE*
 ↪  wolfTPM2_GetHandleRefFromSession(WOLFTPM2_SESSION* session);

WOLFTPM_API TPM_HANDLE wolfTPM2_GetHandleValue(WOLFTPM2_HANDLE* handle);

WOLFTPM_API int wolfTPM2_SetKeyAuthPassword(WOLFTPM2_KEY *key, const byte*
 ↪  auth,
    int authSz);

WOLFTPM_API int wolfTPM2_GetKeyBlobAsBuffer(byte *buffer, word32 bufferSz,
    WOLFTPM2_KEYBLOB* key);

WOLFTPM_API int wolfTPM2_SetKeyBlobFromBuffer(WOLFTPM2_KEYBLOB* key,
    byte *buffer, word32 bufferSz);

#ifdef __cplusplus
    }  /* extern "C" */
#endif

#endif /* __TPM2_WRAP_H__ */
```

4.8 wolfTPM2 ラッパー

このモジュールでは、ラッパーと呼ばれる wolfTPM の豊富な API について説明します。

wolfTPM ラッパーは、主に 2 つのケースで使用されます。

- キーの生成や保存など、一般的な TPM 2.0 タスクを実行する
- 構成証明やパラメーター暗号化などの複雑な TPM 2.0 タスクを実行する wolfTPM は、その多くのラッパー関数のおかげで、TPM 2.0 を迅速かつ迅速に使用できるようにします。

4.8.1 関数

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_Test** TPM の初期化をテストし、オプションで TPM 機能を受け取ることができます。
WOLFTPM_API int	**wolfTPM2_Init** TPM の初期化を完了します。
WOLFTPM_API int	**wolfTPM2_OpenExisting** 現在の TPM ローカリティで、既に初期化されている TPM を使用します。
WOLFTPM_API int	**wolfTPM2_Cleanup** TPM と wolfcrypt の初期化解除を行います。
WOLFTPM_API int	**wolfTPM2_Cleanup_ex** TPM (および使用されている場合は wolfcrypt) の初期化解除。
WOLFTPM_API int	**wolfTPM2_GetTpmDevId** TPM のデバイス ID を提供します。
WOLFTPM_API int	**wolfTPM2_SelfTest** TPM にセルフ テストを実行するように要求します。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_GetCapabilities** 利用可能な TPM 機能を報告します。
WOLFTPM_API int	**wolfTPM2_UnsetAuth** インデックス番号が指す TPM 認証スロットの 1 つをクリアします。
WOLFTPM_API int	**wolfTPM2_SetAuth** 指定されたインデックス、セッション ハンドル、属性、および認証を使用して、TPM 承認スロットを設定します。
WOLFTPM_API int	**wolfTPM2_SetAuthPassword** 提供されたユーザー認証 (通常はパスワード) を使用して TPM 認証スロットを設定します。
WOLFTPM_API int	**wolfTPM2_SetAuthHandle** wolfTPM2 ハンドルに関連付けられたユーザー認証を使用して TPM 認証スロットを設定します。
WOLFTPM_API int	**wolfTPM2_SetAuthSession** 指定された TPM セッション ハンドル、インデックス、およびセッション属性を使用して、TPM 承認スロットを設定します。
WOLFTPM_API int	**wolfTPM2_SetAuthHandleName** TPM セッションで使用される名前を、wolfTPM2 ハンドルに関連付けられた名前で更新します。
WOLFTPM_API int	**wolfTPM2_StartSession** TPM セッション、ポリシー、HMAC、またはトライアルを作成します。
WOLFTPM_API int	**wolfTPM2_CreateAuthSession_EkPolicy** デフォルトの EK ポリシーを満たすために、ポリシー シークレットを使用して TPM セッションを作成します。
WOLFTPM_API int	**wolfTPM2_CreatePrimaryKey** TPM 2.0 プライマリ鍵を準備および作成する単一の関数。
WOLFTPM_API int	**wolfTPM2_ChangeAuthKey** TPM 2.0 鍵の認証シークレットを変更します。
WOLFTPM_API int	**wolfTPM2_CreateKey** TPM 2.0 鍵を準備および作成する単一の関数。
WOLFTPM_API int	**wolfTPM2_LoadKey** TPM 2.0 鍵をロードする単一関数。
WOLFTPM_API int	**wolfTPM2_CreateAndLoadKey** 1 つのステップで TPM 2.0 鍵を作成してロードする単一の関数。
WOLFTPM_API int	**wolfTPM2_CreateLoadedKey** 単一の TPM 2.0 操作を使用して鍵を作成および読み込み、暗号化された秘密鍵マテリアルを保存します。
WOLFTPM_API int	**wolfTPM2_LoadPublicKey** 外部の鍵の公開部分をロードするラッパー。
WOLFTPM_API int	**wolfTPM2_LoadPrivateKey** 外部の秘密鍵をインポートし、TPM に 1 ステップでロードする単一機能。
WOLFTPM_API int	**wolfTPM2_ImportPrivateKey** 外部の秘密鍵をインポートし、TPM に 1 ステップでロードする単一機能。
WOLFTPM_API int	**wolfTPM2_LoadRsaPublicKey** 外部 RSA 鍵の公開部分をインポートするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_LoadRsaPublicKey_ex** 外部の RSA 鍵の公開部分をインポートするヘルパー関数の拡張関数。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_ImportRsaPrivateKey** 外部の RSA 秘密鍵をインポートします。
WOLFTPM_API int	**wolfTPM2_LoadRsaPrivateKey** 外部の RSA 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_LoadRsaPrivateKey_ex** 外部の RSA 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数の拡張関数。
WOLFTPM_API int	**wolfTPM2_LoadEccPublicKey** 外部の ECC 鍵の公開部分をインポートするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_ImportEccPrivateKey** 外部の ECC 鍵のプライベート マテリアルをインポートするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_LoadEccPrivateKey** 外部の ECC 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数。
WOLFTPM_API int	**wolfTPM2_ReadPublicKey** ハンドルを使用して、読み込まれた TPM オブジェクトのパブリック部分を受け取るヘルパー関数。
WOLFTPM_API int	**wolfTPM2_CreateKeySeal** このラッパーを使用すると、シークレットを TPM 2.0 鍵内に封印できます。
WOLFTPM_API int	**wolfTPM2_ComputeName** TPM が期待する形式でオブジェクトのパブリック領域のハッシュを生成するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_SensitiveToPrivate** TPM2B_SENSITIVE を変換するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_RsaKey_TpmToWolf** RSA TPM 鍵を抽出し、それを wolfcrypt 鍵に変換します。
WOLFTPM_API int	**wolfTPM2_RsaKey_TpmToPemPub** 公開 RSA TPM 鍵を PEM 形式の公開鍵に変換する 注: pem と tempBuf は、同じサイズの異なるバッファーである必要があります。
WOLFTPM_API int	**wolfTPM2_RsaKey_WolfToTpm** RSA wolfcrypt 鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_RsaKey_WolfToTpm_ex** 特定のプライマリ キーまたは階層の下で RSA wolfcrypt 鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_RsaKey_PubPemToTpm** PEM 形式の公開鍵をファイルから TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_EccKey_TpmToWolf** ECC TPM 鍵を抽出し、wolfcrypt 鍵に変換します。
WOLFTPM_API int	**wolfTPM2_EccKey_WolfToTpm** ECC wolfcrypt 鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_EccKey_WolfToTpm_ex** ECC wolfcrypt 鍵を特定のプライマリ キーまたは階層の下の TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_EccKey_WolfToPubPoint** wolfcrypt 鍵から生成された ECC 公開鍵を TPM にインポートします。
WOLFTPM_API int	**wolfTPM2_SignHash** TPM 鍵を使用して任意のデータに署名するヘルパー関数。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_SignHashScheme** TPM 鍵を使用して任意のデータに署名し、署名スキームとハッシュ アルゴリズムを指定する高度なヘルパー関数。
WOLFTPM_API int	**wolfTPM2_VerifyHash** TPM が生成した署名を検証するためのヘルパー関数。
WOLFTPM_API int	**wolfTPM2_VerifyHashScheme** TPM が生成した署名を検証するための高度なヘルパー関数。
WOLFTPM_API int	**wolfTPM2_ECDHGenKey** Diffie-Hellman 交換用の NULL 階層を持つ ECC 鍵ペアを生成してからロードします。
WOLFTPM_API int	**wolfTPM2_ECDHGen** 一時鍵を生成し、Z (共有シークレット) を計算します
WOLFTPM_API int	**wolfTPM2_ECDHGenZ** pubPoint と読み込まれたプライベート ECC 鍵を使用して Z (共有シークレット) を計算します。
WOLFTPM_API int	**wolfTPM2_ECDHEGenKey** 一時的な ECC 鍵を生成し、配列インデックスを返します (2 フェーズ メソッド)
WOLFTPM_API int	**wolfTPM2_ECDHEGenZ** pubPoint とカウンターを使用して Z (共有シークレット) を計算します (2 フェーズ法)
WOLFTPM_API int	**wolfTPM2_RsaEncrypt** TPM 2.0 鍵を使用して RSA 暗号化を実行します。
WOLFTPM_API int	**wolfTPM2_RsaDecrypt** TPM 2.0 鍵を使用して RSA 復号を実行します。
WOLFTPM_API int	**wolfTPM2_ReadPCR** 指定された TPM 2.0 プラットフォーム構成レジスタ (PCR) の値を読み取る。
WOLFTPM_API int	**wolfTPM2_ExtendPCR** ユーザー提供のダイジェストで PCR レジスタを拡張します。
WOLFTPM_API int	**wolfTPM2_NVCreateAuth** TPM の NVRAM にデータを格納するために後で使用する新しい NV インデックスを作成します。
WOLFTPM_API int	**wolfTPM2_NVWriteAuth** 指定されたオフセットで、ユーザー データを NV インデックスに格納します。
WOLFTPM_API int	**wolfTPM2_NVReadAuth** 指定されたオフセットから開始して、NV インデックスからユーザー データを読み取ります。
WOLFTPM_API int	**wolfTPM2_NVIncrement** NV 一方向カウンターをインクリメントします。
WOLFTPM_API int	**wolfTPM2_NVOpen** NV を開き、必要な認証と名前ハッシュを入力します。
WOLFTPM_API int	**wolfTPM2_NVDeleteAuth** 既存の NV インデックスを破棄します。
WOLFTPM_API int	**wolfTPM2_NVCreate 非推奨です。新しい API を使用してください。**
WOLFTPM_API int	**wolfTPM2_NVWrite 非推奨です。新しい API を使用してください。**
WOLFTPM_API int	**wolfTPM2_NVRead 非推奨です。新しい API を使用してください。**
WOLFTPM_API int	**wolfTPM2_NVDelete 非推奨です。新しい API を使用してください。**

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_NVReadPublic** 最大サイズなど、nvIndex に関する公開情報を抽出します。
WOLFTPM_API int	**wolfTPM2_NVStoreKey** TPM 2.0 キーを TPM の NVRAM に格納するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_NVDeleteKey** TPM の NVRAM から TPM 2.0 鍵を削除するヘルパー関数。
WOLFTPM_API struct WC_RNG *	**wolfTPM2_GetRng** wolfTPM に使用される wolfcrypt RNG インスタンスを取得します。
WOLFTPM_API int	**wolfTPM2_GetRandom** TPM RNG または wolfcrypt RNG で生成された一連の乱数を取得します。
WOLFTPM_API int	**wolfTPM2_UnloadHandle** TPM がロードされたオブジェクトを破棄するために使用します。
WOLFTPM_API int	**wolfTPM2_Clear** wolfTPM と wolfcrypt を初期化解除します (有効な場合)
WOLFTPM_API int	**wolfTPM2_HashStart** TPM で生成されたハッシュを開始するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_HashUpdate** TPM で生成されたハッシュを新しいユーザー データで更新します。
WOLFTPM_API int	**wolfTPM2_HashFinish** TPM で生成されたハッシュをファイナライズし、ユーザー バッファーでダイジェスト出力を取得します。
WOLFTPM_API int	**wolfTPM2_LoadKeyedHashKey** 通常は HMAC 操作に使用される、KeyedHash 型の新しい TPM 鍵を作成して読み込みます。
WOLFTPM_API int	**wolfTPM2_HmacStart** TPM で生成された hmac を開始するヘルパー関数。
WOLFTPM_API int	**wolfTPM2_HmacUpdate** TPM で生成された hmac を新しいユーザー データで更新します。
WOLFTPM_API int	**wolfTPM2_HmacFinish** TPM で生成された hmac をファイナライズし、ユーザー バッファーでダイジェスト出力を取得します。
WOLFTPM_API int	**wolfTPM2_LoadSymmetricKey** 外部対称鍵を TPM にロードします。
WOLFTPM_API int	**wolfTPM2_SetCommand** 他の制限付き TPM コマンドを有効にするために使用される、ベンダー固有の TPM コマンド。
WOLFTPM_API int	**wolfTPM2_Shutdown** TPM をシャットダウンまたはリセットするためのヘルパー関数。
WOLFTPM_API int	**wolfTPM2_UnloadHandles** 後続の TPM ハンドルをアンロードするためのワンショット API。
WOLFTPM_API int	**wolfTPM2_UnloadHandles_AllTransient** すべての一時的な TPM ハンドルをアンロードするためのワンショット API。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA** ユーザーが選択したオブジェクト属性に基づいて、新しい RSA 鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC** ユーザーが選択したオブジェクト属性に基づいて、新しい ECC 鍵の TPM パブリック テンプレートを準備します。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_Symmetric** 新しい対称鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_KeyedHash** 新しい KeyedHash 鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_KeySeal** シークレットを封印するための新しい鍵の TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA_EK** RSA タイプの TPM 承認鍵を生成するための TPM 公開テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC_EK** ECC タイプの TPM 承認鍵を生成するための TPM 公開テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA_SRK** RSA タイプの新しい TPM ストレージ 鍵を生成するための TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC_SRK** ECC タイプの新しい TPM ストレージ 鍵を生成するための TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_RSA_AIK** RSA タイプの新しい TPM Attestation Key を生成するための TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_GetKeyTemplate_ECC_AIK** ECC タイプの新しい TPM Attestation Key を生成するための TPM パブリック テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_SetKeyTemplate_Unique** Create または CreatePrimary で使用されるパブリック テンプレートの一意の領域を設定します。
WOLFTPM_API int	**wolfTPM2_GetNvAttributesTemplate** TPM NV インデックス テンプレートを準備します。
WOLFTPM_API int	**wolfTPM2_CreateEK** ユーザーが選択したアルゴリズム、RSA または ECC に基づいて、新しい TPM 承認キーを生成します。
WOLFTPM_API int	**wolfTPM2_CreateSRK** 他の TPM キーのストレージ キーとして使用される新しい TPM プライマリ キーを生成します。
WOLFTPM_API int	**wolfTPM2_CreateAndLoadAIK** 指定されたストレージ鍵の下に新しい TPM 構成証明鍵を生成します。
WOLFTPM_API int	**wolfTPM2_GetTime** TPM 署名付きタイムスタンプを生成するワンショット API。
WOLFTPM_API int	**wolfTPM2_CSR_SetCustomExt** WOLFTPM2_CSR 構造体のカスタム要求拡張 oid と値の使用を設定する証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_SetKeyUsage** WOLFTPM2_CSR 構造のキー使用法を設定する証明書署名要求 (CSR) 生成のヘルパー。

戻り値	関数名および機能概要
WOLFTPM_API int	**wolfTPM2_CSR_SetSubject** WOLFTPM2_CSR 構造体のサブジェクトを設定する証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_MakeAndSign_ex** TPM ベースの鍵 (件名と鍵の使用法が既に設定されている WOLFTPM2_KEY 構造) を使用した証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_MakeAndSign** sTPM ベースの鍵 (件名と鍵の使用法が既に設定されている WOLFTPM2_KEY 構造) を使用した証明書署名要求 (CSR) 生成のヘルパー。
WOLFTPM_API int	**wolfTPM2_CSR_Generate_ex** TPM ベースの鍵 (WOLFTPM2_KEY) を使用した証明書署名要求 (CSR) 生成のヘルパー。TPM 鍵に基づいて CSR または自己署名証明書を出力するためのシングル ショット API。
WOLFTPM_API int	**wolfTPM2_CSR_Generate** TPM ベースの鍵 (WOLFTPM2_KEY) を使用した証明書署名要求 (CSR) 生成のヘルパー。TPM 鍵に基づいて CSR または自己署名証明書を出力するためのシングル ショット API。
WOLFTPM_API int	**wolfTPM2_CryptoDevCb** クリプトオフロードに TPM を使用するためのリファレンス クリプト コールバック API。このコールバック関数は、wolfTPM2_SetCryptoDevCb または wc_CryptoDev_RegisterDevice を使用して登録されます。
WOLFTPM_API int	**wolfTPM2_SetCryptoDevCb** 暗号コールバック関数を登録し、割り当てられた devId を返します。
WOLFTPM_API int	**wolfTPM2_ClearCryptoDevCb** 登録された暗号コールバックをクリアします。
WOLFTPM_API WOLFTPM2_DEV**	**wolfTPM2_New** WOLFTPM2_DEV を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_Free** wolfTPM2_New によって割り当てられた WOLFTPM2_DEV をクリーンアップして解放します。
WOLFTPM_API WOLFTPM2_KEYBLOB	**wolfTPM2_NewKeyBlob** WOLFTPM2_KEYBLOB を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreeKeyBlob** wolfTPM2_NewKeyBlob で割り当てられた WOLFTPM2_KEYBLOB を解放します。
WOLFTPM_API TPMT_PUBLIC	**wolfTPM2_NewPublicTemplate** TPMT_PUBLIC 構造体を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreePublicTemplate** wolfTPM2_NewPublicTemplate で割り当てられた TPMT_PUBLIC を解放します。
WOLFTPM_API WOLFTPM2_KEY	**wolfTPM2_NewKey** WOLFTPM2_KEY を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreeKey** wolfTPM2_NewKey で割り当てられた WOLFTPM2_KEY を解放します。

戻り値	関数名および機能概要
WOLFTPM_API WOLFTPM2_SESSION *	**wolfTPM2_NewSession** WOLFTPM2_SESSION を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreeSession** wolfTPM2_NewSession で割り当てられた WOLFTPM2_SESSION を解放します。
WOLFTPM_API WOLFTPM2_CSR *	**wolfTPM2_NewCSR** WOLFTPM2_CSR を割り当てて初期化します。
WOLFTPM_API int	**wolfTPM2_FreeCSR** wolfTPM2_NewCSR で割り当てられた WOLFTPM2_CSR を解放します。
WOLFTPM_API WOLFTPM2_HANDLE *	**wolfTPM2_GetHandleRefFromKey** WOLFTPM2_KEY から WOLFTPM2_HANDLE を取得します。
WOLFTPM_API WOLFTPM2_HANDLE *	**wolfTPM2_GetHandleRefFromKeyBlob** WOLFTPM2_KEYBLOB から WOLFTPM2_HANDLE を取得します。
WOLFTPM_API WOLFTPM2_HANDLE *	**wolfTPM2_GetHandleRefFromSession** WOLFTPM2_SESSION から WOLFTPM2_HANDLE を取得します。
WOLFTPM_API TPM_HANDLE	**wolfTPM2_GetHandleValue** WOLFTPM2_HANDLE から 32 ビットのハンドル値を取得します。
WOLFTPM_API int	**wolfTPM2_SetKeyAuthPassword** 鍵の認証データを設定します。
WOLFTPM_API int	**wolfTPM2_GetKeyBlobAsBuffer** キーブロブからバイナリ バッファーにデータをマーシャリングします。これは、別のプロセスでロードするため、または電源の再投入後にディスクに保存できます。
WOLFTPM_API int	**wolfTPM2_SetKeyBlobFromBuffer** データを WOLFTPM2_KEYBLOB 構造体にアンマーシャリングします。これは、wolfTPM2_GetKeyBlobAsBuffer によって以前にマーシャリングされたキーブロブをロードするために使用できます。

4.8.2 詳細な説明

このモジュールでは、ラッパーと呼ばれる wolfTPM の豊富な API について説明します。

wolfTPM ラッパーは、主に 2 つのケースで使用されます。

- キーの生成や保存など、一般的な TPM 2.0 タスクを実行する
- 構成証明やパラメーター暗号化などの複雑な TPM 2.0 タスクを実行する wolfTPM は、その多くのラッパー関数のおかげで、TPM 2.0 を迅速かつ迅速に使用できるようにします。

4.8.3 関数のドキュメント

wolfTPM2_Test

```
WOLFTPM_API int wolfTPM2_Test(
    TPM2HalIoCb ioCb,
    void * userCtx,
    WOLFTPM2_CAPS * caps
```

)

TPM の初期化をテストし、オプションで TPM 機能を受け取ることができます。

パラメータ:

- **ioCb** IO コールバック関数 (examples/tpm_io.h を参照)
- **userCtx** ユーザーコンテキストへのポインター (NULL 指定も可)
- **caps** TPM 機能を返却する為の WOLFTPM2_CAPS 構造体へのポインター (NULL 指定も可)

参考:

- wolfTPM2_Init
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_Init

```
WOLFTPM_API int wolfTPM2_Init(
    WOLFTPM2_DEV * dev,
    TPM2HalIoCb ioCb,
    void * userCtx
)
```

TPM の初期化を完了します。

パラメータ:

- **dev** WOLFTPM2_DEV 型の空の構造体へのポインター
- **ioCb** IO コールバック関数 (examples/tpm_io.h 参照)
- **userCtx** ユーザーコンテキストへのポインター (NULL 指定も可)

参考:

- wolfTPM2_OpenExisting
- wolfTPM2_Test
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信を確認すること)
- BAD_FUNC_ARG: 不正な引数

使用例

```
int rc;
WOLFTPM2_DEV dev;

rc = wolfTPM2_Init(&dev, TPM2_IoCb, userCtx);
if (rc != TPM_RC_SUCCESS) {
    //wolfTPM2_Init failed
    goto exit;
}
```

wolfTPM2_OpenExisting

```
WOLFTPM_API int wolfTPM2_OpenExisting(
    WOLFTPM2_DEV * dev,
    TPM2HalIoCb ioCb,
    void * userCtx
)
```

現在の TPM ローカリティで、既に初期化されている TPM を使用します。

パラメータ:

- **dev** WOLFTPM2_DEV 型の空の構造体へのポインター
- **ioCb** IO コールバック関数 (examples/tpm_io.h 参照)
- **userCtx** ユーザーコンテキストへのポインター (NULL 指定も可)

参考:

- wolfTPM2_Init
- wolfTPM2_Cleanup
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信を確認すること)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_Cleanup

```
WOLFTPM_API int wolfTPM2_Cleanup(
    WOLFTPM2_DEV * dev
)
```

TPM と wolfcrypt の初期化解除を行います。

パラメータ:

- **dev** データ設定済みの WOLFTPM2_DEV 型の構造体へのポインター

参考:

- wolfTPM2_OpenExisting
- wolfTPM2_Test
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信を確認すること)
- BAD_FUNC_ARG: 不正な引数

ノート:

適切な doShutdown パラメータを指定して wolfTPM2_Cleanup_ex を呼び出します

使用例

```
int rc;

rc = wolfTPM2_Cleanup(&dev);
if (rc != TPM_RC_SUCCESS) {
    //wolfTPM2_Cleanup failed
    goto exit;
}
```

wolfTPM2_Cleanup_ex

```
WOLFTPM_API int wolfTPM2_Cleanup_ex(
    WOLFTPM2_DEV * dev,
    int doShutdown
)
```

TPM (および使用されている場合は wolfcrypt) の初期化解除。

パラメータ:

- **dev** データ設定済みの WOLFTPM2_DEV 型の構造体へのポインター
- **doShutdown** フラグ値。true を指定の場合は TPM2_Shutdown が実行される

参考:

- wolfTPM2_OpenExisting
- wolfTPM2_Test
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信を確認すること)
- BAD_FUNC_ARG: 不正な引数

使用例

```
int rc;

//perform TPM2_Shutdown after deinitialization
rc = wolfTPM2_Cleanup_ex(&dev, 1);
if (rc != TPM_RC_SUCCESS) {
    //wolfTPM2_Cleanup_ex failed
    goto exit;
}
```

wolfTPM2_GetTpmDevId

```
WOLFTPM_API int wolfTPM2_GetTpmDevId(
    WOLFTPM2_DEV * dev
)
```

TPM のデバイス ID を提供します。

パラメータ:

- **dev** データ設定済みの WOLFTPM2_DEV 型の構造体へのポインター

参考:

- wolfTPM2_GetCapabilities
- wolfTPM2_Init

戻り値:

- 有効な TPM デバイス ID を示す整数値
- あるいは、TPM 初期化で DevID を取得できない場合は INVALID_DEVID を返します

使用例

```
int tpmDevId;

tpmDevId = wolfTPM2_GetTpmDevId(&dev);
```

```
if (tpmDevId != INVALID_DEVID) {
    //wolfTPM2_Cleanup_ex failed
    goto exit;
}
```

wolfTPM2_SelfTest

```
WOLFTPM_API int wolfTPM2_SelfTest(
    WOLFTPM2_DEV * dev
)
```

TPM にセルフ テストを実行するように要求します。

パラメータ:

- **dev** データ設定済みの WOLFTPM2_DEV 型の構造体へのポインター

参考:

- wolfTPM2_OpenExisting
- wolfTPM2_Test
- TPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信と TPM リターンコードを確認すること)
- BAD_FUNC_ARG: 不正な引数

使用例

```
int rc;

//perform TPM2_Shutdown after deinitialization
rc = wolfTPM2_SelfTest(&dev);
if (rc != TPM_RC_SUCCESS) {
    //wolfTPM2_SelfTest failed
    goto exit;
}
```

wolfTPM2_GetCapabilities

```
WOLFTPM_API int wolfTPM2_GetCapabilities(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CAPS * caps
)
```

利用可能な TPM 機能を報告します。

パラメータ:

- **dev** データ設定済みの WOLFTPM2_DEV 型の構造体へのポインター
- **caps** 機能を返却する為の WOLFTPM2_CAPS 構造体へのポインター

参考:

- wolfTPM2_GetTpmDevId
- wolfTPM2_SelfTest
- wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功

- TPM_RC_FAILURE: 一般的なエラー (TPM IO 通信と TPM リターンコードを確認すること)
- BAD_FUNC_ARG: 不正な引数

使用例

```
int rc;
WOLFTPM2_CAPS caps;

//perform TPM2_Shutdown after deinitialization
rc = wolfTPM2_GetCapabilities(&dev, &caps);
if (rc != TPM_RC_SUCCESS) {
    //wolfTPM2_GetCapabilities failed
    goto exit;
}
```

wolfTPM2_UnsetAuth

```
WOLFTPM_API int wolfTPM2_UnsetAuth(
    WOLFTPM2_DEV * dev,
    int index
)
```

インデックス番号が指す TPM 認証スロットの 1 つをクリアします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定するための整数値（0～3）

参考:

- wolfTPM2_SetAuth
- wolfTPM2_SetAuthPassword
- wolfTPM2_SetAuthHandle
- wolfTPM2_SetAuthSession

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: TPM2 コンテキストのロックを取得できなかった
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_SetAuth

```
WOLFTPM_API int wolfTPM2_SetAuth(
    WOLFTPM2_DEV * dev,
    int index,
    TPM_HANDLE sessionHandle,
    const TPM2B_AUTH * auth,
    TPMA_SESSION sessionAttributes,
    const TPM2B_NAME * name
)
```

指定されたインデックス、セッション ハンドル、属性、および認証を使用して、TPM 承認スロットを設定します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定する整数値 (0 ～3)
- **sessionHandle** TPM_HANDLE 型の整数値

- **auth** TPM 承認を含む TPM2B_AUTH 型の構造体へのポインター
- **sessionAttributes** TPMA_SESSION 型の整数値。セッションの 1 つ以上の属性を選択します
- **name** TPM2B_NAME 構造体へのポインター

参考:

- wolfTPM2_SetAuthPassword
- wolfTPM2_SetAuthHandle
- wolfTPM2_SetAuthSession

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

wolfTPM2_SetAuthPassword など、他の wolfTPM2 ラッパーのいずれかを使用することをお勧めします。wolfTPM2_SetAuth ラッパーは、高度なユース ケースの TPM 認証スロットを完全に制御できるためです。ほとんどのシナリオでは、wolfTPM2_SetAuthHandle と SetAuthPassword が使用されます。

wolfTPM2_SetAuthPassword

```
WOLFTPM_API int wolfTPM2_SetAuthPassword(
    WOLFTPM2_DEV * dev,
    int index,
    const TPM2B_AUTH * auth
)
```

提供されたユーザー認証 (通常はパスワード) を使用して TPM 認証スロットを設定します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定する整数値 (0 ～3)
- **auth** TPM 承認を含む TPM2B_AUTH 型の構造体へのポインター

参考:

- wolfTPM2_SetAuthHandle
- wolfTPM2_SetAuthSession
- wolfTPM2_SetAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

多くの場合、プライマリ キーを含む TPM 鍵の読み込みと使用を承認するために使用されます。

wolfTPM2_SetAuthHandle

```
WOLFTPM_API int wolfTPM2_SetAuthHandle(
    WOLFTPM2_DEV * dev,
    int index,
    const WOLFTPM2_HANDLE * handle
)
```

wolfTPM2 ハンドルに関連付けられたユーザー認証を使用して TPM 認証スロットを設定します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定する整数値 (0 〜3)
- **handle** WOLFTPM2_HANDLE 構造体へのポインター

参考:

- wolfTPM2_SetAuth
- wolfTPM2_SetAuthPassword
- wolfTPM2_SetAuthHandle
- wolfTPM2_SetAuthSession

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

このラッパーは、TPM キーを複数の操作に使用し、TPM 承認が再び必要な場合に特に役立ちます。

wolfTPM2_SetAuthSession

```
WOLFTPM_API int wolfTPM2_SetAuthSession(
    WOLFTPM2_DEV * dev,
    int index,
    const WOLFTPM2_SESSION * tpmSession,
    TPMA_SESSION sessionAttributes
)
```

指定された TPM セッション ハンドル、インデックス、およびセッション属性を使用して、TPM 承認スロットを設定します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定する整数値 (0 〜3)
- **tpmSession** TPM_HANDLE 型のセッションハンドル整数値
- **sessionAttributes** TPMA_SESSION 型の整数値, 一つ以上のアトリビュートを選択する

参考:

- wolfTPM2_SetAuth
- wolfTPM2_SetAuthPassword
- wolfTPM2_SetAuthHandle

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

このラッパーは、TPM セッション、つまりパラメータ暗号化のセッションのコンフィギュレーションに有用です。

wolfTPM2_SetAuthHandleName

```
WOLFTPM_API int wolfTPM2_SetAuthHandleName(
    WOLFTPM2_DEV * dev,
    int index,
    const WOLFTPM2_HANDLE * handle
)
```

TPM セッションで使用される名前を、wolfTPM2 ハンドルに関連付けられた名前で更新します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **index** TPM 認証スロットを指定する整数値 (0 〜3)
- **handle** WOLFTPM2_HANDLE 構造体へのポインター

参考:

- wolfTPM2_SetAuth
- wolfTPM2_SetAuthPassword
- wolfTPM2_SetAuthHandle
- wolfTPM2_SetAuthSession

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

通常、このラッパーは別のラッパー（例えば wolfTPM2_NVWriteAuth）から使用され、非常に特殊なユースケースで使用されます

wolfTPM2_StartSession

```
WOLFTPM_API int wolfTPM2_StartSession(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_SESSION * session,
    WOLFTPM2_KEY * tpmKey,
    WOLFTPM2_HANDLE * bind,
    TPM_SE sesType,
    int encDecAlg
)
```

TPM セッション、ポリシー、HMAC、またはトライアルを作成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **session** WOLFTPM2_SESSION 構造体へのポインター
- **tpmKey** セッションのソルトとして使用する WOLFTPM2_KEY へのポインター
- **bind** セッションをバインドするために使用される WOLFTPM2_HANDLE へのポインタ
- **sesType** バイト値、セッション タイプ (HMAC、ポリシー、またはトライアル)
- **encDecAlg** パラメータ暗号化の場合のアルゴリズムを指定する整数値 (TPM_ALG_CFB または TPM_ALG_XOR)。CFB または XOR 以外の値はすべて NULL と見なされ、パラメーターの暗号化は無効になります。

参考:

- wolfTPM2_SetAuthSession

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

このラッパーは、パラメータ暗号化のために TPM セッションを開始するためにも使用できます。wolfTPM nvram または keygen の例を参照してください。

wolfTPM2_CreateAuthSession_EkPolicy

```
WOLFTPM_API int wolfTPM2_CreateAuthSession_EkPolicy(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_SESSION * tpmSession
)
```

デフォルトの EK ポリシーを満たすために、ポリシー シークレットを使用して TPM セッションを作成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **session** WOLFTPM2_SESSION 構造体へのポインター

参考:

- wolfTPM2_SetAuthSession
- wolfTPM2_StartSession

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数
- TPM_RC_FAILURE: TPM リターン コード、使用可能なハンドル、TPM IO などを要確認

ノート:

このラッパーは、EK 承認がデフォルトから変更されていない場合にのみ使用できます。

wolfTPM2_CreatePrimaryKey

```
WOLFTPM_API int wolfTPM2_CreatePrimaryKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    TPM_HANDLE primaryHandle,
    TPMT_PUBLIC * publicTemplate,
    const byte * auth,
    int authSz
)
```

TPM 2.0 プライマリ鍵を準備および作成する単一の関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **primaryHandle** 4 つの TPM 2.0 プライマリ シードのいずれかを指定する整数値: TPM_RH_OWNER、TPM_RH_ENDORSEMENT、TPM_RH_PLATFORM、または TPM_RH_NULL
- **publicTemplate** 手動で、または wolfTPM2_GetKeyTemplate_... ラッパーの 1 つを使用して設定された TPMT_PUBLIC 構造へのポインタ
- **auth** プライマリキーのパスワード認証を指定する文字列定数へのポインタ
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreateKey
- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)

- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 2.0 では、非対称 RSA または ECC プライマリ キーのみが許可されます。その後、対称鍵と非対称鍵の両方を TPM 2.0 プライマリ キーの下に作成できます。通常、プライマリ キーは TPM 2.0 鍵の階層を作成するために使用されます。TPM は主鍵を使用して、他の鍵をラップし、署名または復号します。

wolfTPM2_ChangeAuthKey

```
WOLFTPM_API int wolfTPM2_ChangeAuthKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    WOLFTPM2_HANDLE * parent,
    const byte * auth,
    int authSz
)
```

TPM 2.0 鍵の認証シークレットを変更します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される TPM 2.0 プライマリ キーを指定します
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreatePrimaryKey
- wolfTPM2_SetAuthHandle
- wolfTPM2_UnloadHandle

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

PM では、プライマリ キーの認証シークレットを変更することはできません。代わりに、wolfTPM2_CreatePrimary を使用して、新しい認証で同じ PrimaryKey を作成します。

wolfTPM2_CreateKey

```
WOLFTPM_API int wolfTPM2_CreateKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEYBLOB * keyBlob,
    WOLFTPM2_HANDLE * parent,
    TPMT_PUBLIC * publicTemplate,
    const byte * auth,
    int authSz
)
```

TPM 2.0 鍵を準備および作成する単一の関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

- **keyBlob** WOLFTPM2_KEYBLOB 型の空の構造体へのポインター
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される 2.0 プライマリ キーを指定します
- **publicTemplate** 手動で、または wolfTPM2_GetKeyTemplate_... ラッパーの 1 つを使用して設定された TPMT_PUBLIC 構造へのポインター
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_LoadKey
- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC
- wolfTPM2_CreatePrimaryKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

この関数は、キー マテリアルのみを作成し、それをキーブロブ引数に格納します。キーをロードするには、wolfTPM2_LoadKey を使用します

wolfTPM2_LoadKey

```
WOLFTPM_API int wolfTPM2_LoadKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEYBLOB * keyBlob,
    WOLFTPM2_HANDLE * parent
)
```

TPM 2.0 鍵をロードする単一関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **keyBlob** WOLFTPM2_KEYBLOB 型の構造体へのポインター
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される TPM 2.0 プライマリ キーを指定します

参考:

- wolfTPM2_CreateKey
- wolfTPM2_CreatePrimaryKey
- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

- TPM 2.0 鍵をロードするには、この操作の前にその親 (プライマリ キー) もロードする必要があります。プライマリキー作成時にロードされます。

wolfTPM2_CreateAndLoadKey

```
WOLFTPM_API int wolfTPM2_CreateAndLoadKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    WOLFTPM2_HANDLE * parent,
    TPMT_PUBLIC * publicTemplate,
    const byte * auth,
    int authSz
)
```

1 つのステップで TPM 2.0 鍵を作成してロードする単一の関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される TPM 2.0 プライマリ キーを指定します
- **publicTemplate** 手動で、または wolfTPM2_GetKeyTemplate_... ラッパーの 1 つを使用して設定された TPMT_PUBLIC 構造へのポインタ
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreateKey
- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CreateLoadedKey

```
WOLFTPM_API int wolfTPM2_CreateLoadedKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEYBLOB * keyBlob,
    WOLFTPM2_HANDLE * parent,
    TPMT_PUBLIC * publicTemplate,
    const byte * auth,
    int authSz
)
```

単一の TPM 2.0 操作を使用して鍵を作成および読み込み、暗号化された秘密鍵マテリアルを保存します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **keyBlob** WOLFTPM2_KEYBLOB 型の空の構造体へのポインター。暗号化されたデータとして秘密鍵の素材が含まれます
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される TPM 2.0 プライマリ キーを指定します
- **publicTemplate** 手動で、または wolfTPM2_GetKeyTemplate_... ラッパーの 1 つを使用して設定された TPMT_PUBLIC 構造へのポインター
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreateAndLoadKey
- wolfTPM2_CreateKey
- wolfTPM2_LoadKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_LoadPublicKey

```
WOLFTPM_API int wolfTPM2_LoadPublicKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const TPM2B_PUBLIC * pub
)
```

外部の鍵の公開部分をロードするラッパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **pub** TPM2B_PUBLIC 構造体へのポインター

参考:

- wolfTPM2_LoadRsaPublicKey
- wolfTPM2_LoadEccPublicKey
- wolfTPM2_wolfTPM2_LoadPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

- 鍵は、TPM が期待する形式にフォーマットする必要があります。'pub' 引数と代替ラッパーを参照してください。

wolfTPM2_LoadPrivateKey

```
WOLFTPM_API int wolfTPM2_LoadPrivateKey(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEY * key,
    const TPM2B_PUBLIC * pub,
    TPM2B_SENSITIVE * sens
)
```

外部の秘密鍵をインポートし、TPM に 1 ステップでロードする単一機能。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL 指定が可)
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **pub** TPM2B_PUBLIC 構造体へのポインター

- **sens** TPM2B_SENSITIVE 型のデータが設定された構造体へのポインター

参考:

- wolfTPM2_CreateKey
- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

秘密鍵の素材は、TPM が期待する形式で準備する必要があります。“sens” 引数を参照してください

wolfTPM2_ImportPrivateKey

```
WOLFTPM_API int wolfTPM2_ImportPrivateKey(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEYBLOB * keyBlob,
    const TPM2B_PUBLIC * pub,
    TPM2B_SENSITIVE * sens
)
```

外部の秘密鍵をインポートし、TPM に 1 ステップでロードする単一機能。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL 指定が可)
- **keyBlob** WOLFTPM2_KEYBLOB]() 型の空の構造体へのポインター
- **pub** TPM2B_PUBLIC 構造体へのポインター
- **sens** TPM2B_SENSITIVE 型のデータが設定された構造体へのポインター

参考:

- wolfTPM2_ImportRsaPrivateKey
- wolfTPM2_ImportEccPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

プライマリ キー マテリアルは、TPM が期待する形式で準備する必要があります。“sens” 引数を参照してください。

wolfTPM2_LoadRsaPublicKey

```
WOLFTPM_API int wolfTPM2_LoadRsaPublicKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * rsaPub,
    word32 rsaPubSz,
    word32 exponent
```

)

外部 RSA 鍵の公開部分をインポートするヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **rsaPub** 公開鍵素材を含むバイトバッファへのポインター
- **rsaPubSz** バッファサイズを指定する word32 型の整数値
- **exponent** RSA 指数を指定する word32 型の整数値

参考:

- wolfTPM2_LoadRsaPublicKey_ex
- wolfTPM2_LoadPublicKey
- wolfTPM2_LoadEccPublicKey
- wolfTPM2_ReadPublicKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

パブリック部分の TPM 形式を必要としないため、使用を推奨

wolfTPM2_LoadRsaPublicKey_ex

```
WOLFTPM_API int wolfTPM2_LoadRsaPublicKey_ex(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * rsaPub,
    word32 rsaPubSz,
    word32 exponent,
    TPMI_ALG_RSA_SCHEME scheme,
    TPMI_ALG_HASH hashAlg
)
```

外部の RSA 鍵の公開部分をインポートするヘルパー関数の拡張関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **rsaPub** 公開鍵素材を含むバイトバッファへのポインター
- **rsaPubSz** バッファサイズを指定する word32 型の整数値
- **exponent** RSA 指数を指定する word32 型の整数値
- **scheme** RSA スキームを指定する TPMI_ALG_RSA_SCHEME タイプの値
- **hashAlg** TPM ハッシュアルゴリズムを指定する TPMI_ALG_HASH タイプの値

参考:

- wolfTPM2_LoadRsaPublicKey
- wolfTPM2_LoadPublicKey
- wolfTPM2_LoadEccPublicKey
- wolfTPM2_ReadPublicKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

開発者は TPM ハッシュ アルゴリズムと RSA スキームを指定できます

wolfTPM2_ImportRsaPrivateKey

```
WOLFTPM_API int wolfTPM2_ImportRsaPrivateKey(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEYBLOB * keyBlob,
    const byte * rsaPub,
    word32 rsaPubSz,
    word32 exponent,
    const byte * rsaPriv,
    word32 rsaPrivSz,
    TPMI_ALG_RSA_SCHEME scheme,
    TPMI_ALG_HASH hashAlg
)
```

外部の RSA 秘密鍵をインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL にすることができ、キーは OWNER 階層の下にインポートされます)
- **keyBlob** WOLFTPM2_KEYBLOB]() 型の空の構造体へのポインター
- **rsaPub** RSA 鍵の公開部分を含むバイトバッファへのポインタ
- **rsaPubSz** パブリック部分のバッファ サイズを指定する word32 型の整数値
- **exponent** RSA 指数を指定する word32 型の整数値
- **rsaPriv** RSA 鍵のプライベートマテリアルを含むバイトバッファへのポインター
- **rsaPrivSz** プライベート マテリアル バッファ サイズを指定する word32 型の整数値
- **scheme** RSA スキームを指定する TPMI_ALG_RSA_SCHEME タイプの値
- **hashAlg** サポートされている TPM 2.0 ハッシュ アルゴリズムを指定する TPMI_ALG_HASH 型の整数値

参考:

- wolfTPM2_LoadRsaPrivateKey
- wolfTPM2_LoadRsaPrivateKey_ex
- wolfTPM2_LoadPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数
- BUFFER_E: 引数のサイズが TPM バッファで許可されているサイズよりも大きい

wolfTPM2_LoadRsaPrivateKey

```
WOLFTPM_API int wolfTPM2_LoadRsaPrivateKey(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEY * key,
    const byte * rsaPub,
```

```
    word32 rsaPubSz,
    word32 exponent,
    const byte * rsaPriv,
    word32 rsaPrivSz
)
```

外部の RSA 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL にすることができ、キーは OWNER 階層の下にインポートされます)
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **rsaPub** RSA 鍵の公開部分を含むバイトバッファへのポインタ
- **rsaPubSz** パブリック部分のバッファ サイズを指定する word32 型の整数値
- **exponent** RSA 指数を指定する word32 型の整数値
- **rsaPriv** RSA 鍵のプライベートマテリアルを含むバイトバッファへのポインター
- **rsaPrivSz** プライベート マテリアル バッファ サイズを指定する word32 型の整数値

参考:

- wolfTPM2_ImportRsaPrivateKey
- wolfTPM2_LoadRsaPrivateKey_ex
- wolfTPM2_LoadPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_LoadRsaPrivateKey_ex

```
WOLFTPM_API int wolfTPM2_LoadRsaPrivateKey_ex(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEY * key,
    const byte * rsaPub,
    word32 rsaPubSz,
    word32 exponent,
    const byte * rsaPriv,
    word32 rsaPrivSz,
    TPMI_ALG_RSA_SCHEME scheme,
    TPMI_ALG_HASH hashAlg
)
```

外部の RSA 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数の拡張関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL にすることができ、キーは OWNER 階層の下にインポートされます)
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **rsaPub** RSA 鍵の公開部分を含むバイトバッファへのポインター
- **rsaPubSz** パブリック部分のバッファ サイズを指定する word32 型の整数値
- **exponent** RSA 指数を指定する word32 型の整数値
- **rsaPriv** RSA 鍵のプライベートマテリアルを含むバイトバッファへのポインター

- **rsaPrivSz** プライベート マテリアル バッファ サイズを指定する word32 型の整数値
- **scheme** RSA スキームを指定する TPMI_ALG_RSA_SCHEME タイプの値
- **hashAlg** TPM ハッシュアルゴリズムを指定する TPMI_ALG_HASH タイプの値

参考:

- wolfTPM2_LoadRsaPrivateKey
- wolfTPM2_LoadPrivateKey
- wolfTPM2_ImportRsaPrivateKey
- wolfTPM2_LoadEccPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_LoadEccPublicKey

```
WOLFTPM_API int wolfTPM2_LoadEccPublicKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    int curveId,
    const byte * eccPubX,
    word32 eccPubXSz,
    const byte * eccPubY,
    word32 eccPubYSz
)
```

外部の ECC 鍵の公開部分をインポートするヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **curveId** 整数値、受け入れられた TPM_ECC_CURVE 値の 1 つ
- **eccPubX** ポイント X のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubXSz** ポイント X バッファ サイズを指定する word32 型の整数値
- **eccPubY** ポイント Y のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubYSz** ポイント Y バッファ サイズを指定する word32 型の整数値

参考:

- wolfTPM2_LoadPublicKey
- wolfTPM2_LoadRsaPublicKey
- wolfTPM2_ReadPublicKey
- wolfTPM2_LoadEccPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

パブリック部分の TPM 形式を必要としないため、使用を推奨

wolfTPM2_ImportEccPrivateKey

```
WOLFTPM_API int wolfTPM2_ImportEccPrivateKey(
    WOLFTPM2_DEV * dev,
```

```
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEYBLOB * keyBlob,
    int curveId,
    const byte * eccPubX,
    word32 eccPubXSz,
    const byte * eccPubY,
    word32 eccPubYSz,
    const byte * eccPriv,
    word32 eccPrivSz
)
```

外部の ECC 鍵のプライベート マテリアルをインポートするヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL にすることができ、キーは OWNER 階層の下にインポートされます)
- **keyBlob** WOLFTPM2_KEYBLOB]() 型の空の構造体へのポインター
- **curveId** 整数値、受け入れられた TPM_ECC_CURVE 値の 1 つ
- **eccPubX** ポイント X のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubXSz** ポイント X バッファ サイズを指定する word32 型の整数値
- **eccPubY** ポイント Y のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubYSz** ポイント Y バッファ サイズを指定する word32 型の整数値
- **eccPriv** プライベート マテリアルを含むバイト バッファへのポインタ
- **eccPrivSz** プライベート マテリアル サイズを指定する word32 型の整数値

参考:

- wolfTPM2_LoadEccPrivateKey
- wolfTPM2_LoadEccPrivateKey_ex
- wolfTPM2_LoadPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_LoadEccPrivateKey

```
WOLFTPM_API int wolfTPM2_LoadEccPrivateKey(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEY * key,
    int curveId,
    const byte * eccPubX,
    word32 eccPubXSz,
    const byte * eccPubY,
    word32 eccPubYSz,
    const byte * eccPriv,
    word32 eccPrivSz
)
```

外部の ECC 秘密鍵を 1 ステップでインポートおよびロードするヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

- **parentKey** WOLFTPM2_HANDLE タイプの構造体へのポインター (外部キーの場合は NULL にすることができ、キーは OWNER 階層の下にインポートされます)
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **curveId** 整数値、受け入れられた TPM_ECC_CURVE 値の 1 つ
- **eccPubX** ポイント X のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubXSz** ポイント X バッファ サイズを指定する word32 型の整数値
- **eccPubY** ポイント Y のパブリック マテリアルを含むバイト バッファへのポインタ
- **eccPubYSz** ポイント Y バッファ サイズを指定する word32 型の整数値
- **eccPriv** プライベート マテリアルを含むバイト バッファへのポインタ
- **eccPrivSz** プライベート マテリアル サイズを指定する word32 型の整数値

参考:

- wolfTPM2_ImportEccPrivateKey
- wolfTPM2_LoadEccPublicKey
- wolfTPM2_LoadPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ReadPublicKey

```
WOLFTPM_API int wolfTPM2_ReadPublicKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const TPM_HANDLE handle
)
```

ハンドルを使用して、読み込まれた TPM オブジェクトのパブリック部分を受け取るヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **handle** ロードされた TPM オブジェクトのハンドルを指定する、TPM_HANDLE 型の整数値

参考:

- wolfTPM2_LoadRsaPublicKey
- wolfTPM2_LoadEccPublicKey
- wolfTPM2_LoadPublicKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 対称鍵の公開部分には、TPM メタデータのみが含まれます

wolfTPM2_CreateKeySeal

```
WOLFTPM_API int wolfTPM2_CreateKeySeal(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEYBLOB * keyBlob,
    WOLFTPM2_HANDLE * parent,
    TPMT_PUBLIC * publicTemplate,
```

```
    const byte * auth,
    int authSz,
    const byte * sealData,
    int sealSize
)
```

このラッパーを使用すると、シークレットを TPM 2.0 鍵内に封印できます。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **keyBlob** WOLFTPM2_KEYBLOB]() 型の空の構造体へのポインター
- **parent** WOLFTPM2_HANDLE タイプの構造体へのポインター。親 (ストレージ キー) として使用される 2.0 プライマリ キーを指定します。
- **publicTemplate** wolfTPM2_GetKeyTemplate_KeySeal のいずれかを使用して設定された TPMT_PUBLIC 構造へのポインタ
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値
- **sealData** シールするシークレット (ユーザー データ) を含むバイト バッファーへのポインター
- **sealSize** シールバッファのサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_GetKeyTemplate_KeySeal
- TPM2_Unseal
- wolfTPM2_CreatePrimary

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

シークレットのサイズは 128 バイトを超えることはできません

wolfTPM2_ComputeName

```
WOLFTPM_API int wolfTPM2_ComputeName(
    const TPM2B_PUBLIC * pub,
    TPM2B_NAME * out
)
```

TPM が期待する形式でオブジェクトのパブリック領域のハッシュを生成するヘルパー関数。

パラメータ:

- **pub** TPM オブジェクトの公開部分を含んだ TPM2B_PUBLIC 構造体へのポインター
- **out** 計算された名前を格納するための TPM2B_NAME 型の空の構造体へのポインター

参考:

- wolfTPM2_ImportPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

計算された TPM 名には、TPM_ALG_ID のハッシュが含まれており、パブリックはオブジェクトのものです

wolfTPM2_SensitiveToPrivate

```
WOLFTPM_API int wolfTPM2_SensitiveToPrivate(
    TPM2B_SENSITIVE * sens,
    TPM2B_PRIVATE * priv,
    TPMI_ALG_HASH nameAlg,
    TPM2B_NAME * name,
    const WOLFTPM2_KEY * parentKey,
    TPMT_SYM_DEF_OBJECT * sym,
    TPM2B_ENCRYPTED_SECRET * symSeed
)
```

TPM2B_SENSITIVE を変換するヘルパー関数。

パラメータ:

- **sens** TPM2B_SENSITIVE 型の正しく設定された構造体へのポインター
- **priv** TPM2B_PRIVATE 型の空の構造体へのポインター
- **nameAlg** TPMI_ALG_HASH 型の整数値。有効な TPM2 ハッシュ アルゴリズムを指定します
- **name** TPM2B_NAME 構造体へのポインター
- **parentKey** WOLFTPM2_KEY 構造体へのポインター。parentKey が存在する場合はそれを指定します
- **sym** TPMT_SYM_DEF_OBJECT 型の構造体へのポインター
- **symSeed** TPM2B_ENCRYPTED_SECRET 型の構造体へのポインター

参考:

- wolfTPM2_ImportPrivateKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_RsaKey_TpmToWolf

```
WOLFTPM_API int wolfTPM2_RsaKey_TpmToWolf(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * tpmKey,
    RsaKey * wolfKey
)
```

RSA TPM 鍵を抽出し、それを wolfcrypt 鍵に変換します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **tpmKey** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **wolfKey** 変換されたキーを格納する RsaKey 型の空の構造体へのポインター

参考:

- wolfTPM2_RsaKey_WolfToTpm
- wolfTPM2_RsaKey_WolfToTpm_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)

- BAD_FUNC_ARG: 不正な引数

wolfTPM2_RsaKey_TpmToPemPub

```
WOLFTPM_API int wolfTPM2_RsaKey_TpmToPemPub(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * keyBlob,
    byte * pem,
    word32 * pemSz
)
```

公開 RSA TPM 鍵を PEM 形式の公開鍵に変換する 注: pem と tempBuf は、同じサイズの異なるバッファーである必要があります。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **keyBlob** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **pem** PEM 会話の一時ストレージとして使用される、バイト型の配列へのポインター
- **pemSz** 使用済みバッファサイズを格納する整数変数へのポインター

参考:

- wolfTPM2_RsaKey_TpmToWolf
- wolfTPM2_RsaKey_WolfToTpm

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_RsaKey_WolfToTpm

```
WOLFTPM_API int wolfTPM2_RsaKey_WolfToTpm(
    WOLFTPM2_DEV * dev,
    RsaKey * wolfKey,
    WOLFTPM2_KEY * tpmKey
)
```

RSA wolfcrypt 鍵を TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **wolfKey** wolfcrypt キーを保持する RsaKey 型の構造体へのポインター
- **tpmKey** インポートされた TPM キーを保持するための WOLFTPM2_KEY 型の空の構造体へのポインター

参考:

- wolfTPM2_RsaKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 2.0 で使用する wolfcrypt による外部生成鍵の使用を許可します

wolfTPM2_RsaKey_WolfToTpm_ex

```
WOLFTPM_API int wolfTPM2_RsaKey_WolfToTpm_ex(
    WOLFTPM2_DEV * dev,
    const WOLFTPM2_KEY * parentKey,
    RsaKey * wolfKey,
    WOLFTPM2_KEY * tpmKey
)
```

特定のプライマリ キーまたは階層の下で RSA wolfcrypt 鍵を TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_KEY 構造体へのポインター、主キーまたは TPM 階層を指す
- **wolfKey** wolfcrypt キーを保持する RsaKey 型の構造体へのポインター
- **tpmKey** インポートされた TPM キーを保持するための WOLFTPM2_KEY 型の空の構造体へのポインター

参考:

- wolfTPM2_RsaKey_WolfToTpm
- wolfTPM2_RsaKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

wolfTPM で wolfcrypt が生成した鍵を使用できるようにします

wolfTPM2_RsaKey_PubPemToTpm

```
WOLFTPM_API int wolfTPM2_RsaKey_PubPemToTpm(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * tpmKey,
    const byte * pem,
    word32 pemSz
)
```

PEM 形式の公開鍵をファイルから TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **tpmKey** インポートされた TPM キーを保持するための WOLFTPM2_KEY 型の空の構造体へのポインター
- **pem** PEM 形式の公開鍵素材を含む、バイト型の配列へのポインター
- **pemSz** PEM キーデータのサイズを指定する整数変数へのポインター

参考:

- wolfTPM2_RsaKey_WolfToTpm
- wolfTPM2_RsaKey_TpmToPem
- wolfTPM2_RsaKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)

wolfTPM2_EccKey_TpmToWolf

```
WOLFTPM_API int wolfTPM2_EccKey_TpmToWolf(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * tpmKey,
    ecc_key * wolfKey
)
```

ECC TPM 鍵を抽出し、wolfcrypt 鍵に変換します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **tpmKey** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **wolfKey** 変換されたキーを格納するための、ecc_key 型の空の構造体へのポインター

参考:

- wolfTPM2_EccKey_WolfToTpm
- wolfTPM2_EccKey_WolfToTpm_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_EccKey_WolfToTpm

```
WOLFTPM_API int wolfTPM2_EccKey_WolfToTpm(
    WOLFTPM2_DEV * dev,
    ecc_key * wolfKey,
    WOLFTPM2_KEY * tpmKey
)
```

ECC wolfcrypt 鍵を TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **wolfKey** wolfcrypt キーを保持する、ecc_key タイプの構造体へのポインター
- **tpmKey** インポートされた TPM キーを保持するための WOLFTPM2_KEY 型の空の構造体へのポインター

参考:

- wolfTPM2_EccKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 2.0 で使用する wolfcrypt による外部生成鍵の使用を許可します

wolfTPM2_EccKey_WolfToTpm_ex

```
WOLFTPM_API int wolfTPM2_EccKey_WolfToTpm_ex(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * parentKey,
    ecc_key * wolfKey,
    WOLFTPM2_KEY * tpmKey
)
```

ECC wolfcrypt 鍵を特定のプライマリ キーまたは階層の下の TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** WOLFTPM2_KEY 構造体へのポインター、主キーまたは TPM 階層を指す
- **wolfKey** wolfcrypt キーを保持する、ecc_key タイプの構造体へのポインター
- **tpmKey** インポートされた TPM キーを保持するための WOLFTPM2_KEY 型の空の構造体へのポインター

参考:

- wolfTPM2_EccKey_WolfToTPM
- wolfTPM2_EccKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

wolfTPM で wolfcrypt が生成した鍵を使用できるようにします

wolfTPM2_EccKey_WolfToPubPoint

```
WOLFTPM_API int wolfTPM2_EccKey_WolfToPubPoint(
    WOLFTPM2_DEV * dev,
    ecc_key * wolfKey,
    TPM2B_ECC_POINT * pubPoint
)
```

wolfcrypt 鍵から生成された ECC 公開鍵を TPM にインポートします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **wolfKey** wolfcrypt 公開 ECC キーを保持する、ecc_key タイプの構造体へのポインター
- **pubPoint** TPM2B_ECC_POINT 型の空の構造体へのポインター

参考: wolfTPM2_EccKey_TpmToWolf

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 2.0 で使用する wolfcrypt によって外部で生成された公開 ECC 鍵の使用を許可します

wolfTPM2_SignHash

```
WOLFTPM_API int wolfTPM2_SignHash(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * digest,
    int digestSz,
    byte * sig,
    int * sigSz
)
```

TPM 鍵を使用して任意のデータに署名するヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター material
- **digest** 任意のデータを含むバイトバッファへのポインタ
- **digestSz** ダイジェスト バッファのサイズをバイト単位で指定する整数値
- **sig** 生成された署名を含むバイトバッファへのポインター
- **sigSz** 署名バッファのサイズをバイト単位で指定する整数値

参考:

- verifyHash
- signHashScheme
- verifyHashScheme

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_SignHashScheme

```
WOLFTPM_API int wolfTPM2_SignHashScheme(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * digest,
    int digestSz,
    byte * sig,
    int * sigSz,
    TPMI_ALG_SIG_SCHEME sigAlg,
    TPMI_ALG_HASH hashAlg
)
```

TPM 鍵を使用して任意のデータに署名し、署名スキームとハッシュ アルゴリズムを指定する高度なヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター material
- **digest** 任意のデータを含むバイトバッファへのポインタ
- **digestSz** ダイジェスト バッファのサイズをバイト単位で指定する整数値
- **sig** 生成された署名を含むバイトバッファへのポインター
- **sigSz** 署名バッファのサイズをバイト単位で指定する整数値
- **sigAlg** サポートされている TPM 2.0 署名スキームを指定する TPMI_ALG_SIG_SCHEME 型の整数値
- **hashAlg** サポートされている TPM 2.0 ハッシュ アルゴリズムを指定する TPMI_ALG_HASH 型の整数値

参考:

- signHash
- verifyHash
- verifyHashScheme

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)

- BAD_FUNC_ARG: 不正な引数

wolfTPM2_VerifyHash

```
WOLFTPM_API int wolfTPM2_VerifyHash(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * sig,
    int sigSz,
    const byte * digest,
    int digestSz
)
```

TPM が生成した署名を検証するためのヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM 2.0 キー マテリアルを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **sig** 生成された署名を含むバイトバッファへのポインター
- **sigSz** 署名バッファのサイズをバイト単位で指定する整数値
- **digest** 符号付きデータを含むバイトバッファへのポインター
- **digestSz** ダイジェスト バッファのサイズをバイト単位で指定する整数値

参考:

- signHash
- signHashScheme
- verifyHashScheme

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_VerifyHashScheme

```
WOLFTPM_API int wolfTPM2_VerifyHashScheme(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const byte * sig,
    int sigSz,
    const byte * digest,
    int digestSz,
    TPMI_ALG_SIG_SCHEME sigAlg,
    TPMI_ALG_HASH hashAlg
)
```

TPM が生成した署名を検証するための高度なヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM 2.0 キー マテリアルを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **sig** 生成された署名を含むバイトバッファへのポインター
- **sigSz** 署名バッファのサイズをバイト単位で指定する整数値
- **digest** 符号付きデータを含むバイトバッファへのポインター
- **digestSz** ダイジェスト バッファのサイズをバイト単位で指定する整数値
- **sigAlg** サポートされている TPM 2.0 署名スキームを指定する TPMI_ALG_SIG_SCHEME 型の整数値

- **hashAlg** サポートされている TPM 2.0 ハッシュ アルゴリズムを指定する TPMI_ALG_HASH 型の整数値

参考:

- signHash
- signHashScheme
- verifyHash

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ECDHGenKey

```
WOLFTPM_API int wolfTPM2_ECDHGenKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * ecdhKey,
    int curve_id,
    const byte * auth,
    int authSz
)
```

Diffie-Hellman 交換用の NULL 階層を持つ ECC 鍵ペアを生成してからロードします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **ecdhKey** WOLFTPM2_KEY 型の空の構造体へのポインター
- **curve_id** 有効な TPM_ECC_CURVE 値を指定する整数値
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_ECDHGen
- wolfTPM2_ECDHGenZ
- wolfTPM2_ECDHEGenKey
- wolfTPM2_ECDHEGenZ

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ECDHGen

```
WOLFTPM_API int wolfTPM2_ECDHGen(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * privKey,
    TPM2B_ECC_POINT * pubPoint,
    byte * out,
    int * outSz
)
```

一時鍵を生成し、Z (共有シークレット) を計算します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **privKey** WOLFTPM2_KEY 型の構造体へのポインター
- **pubPoint** TPM2B_ECC_POINT 型の空の構造体へのポインター
- **out** 生成された共有シークレットを格納するバイト バッファへのポインター
- **outSz** 共有シークレットのサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_ECDHGenZ
- wolfTPM2_ECDHGenKey
- wolfTPM2_ECDHEGenKey
- wolfTPM2_ECDHEGenZ

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

秘密鍵ハンドルを使用して鍵ペアを生成し、公開ポイントと共有秘密を返すワンショット API

wolfTPM2_ECDHGenZ

```
WOLFTPM_API int wolfTPM2_ECDHGenZ(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * privKey,
    const TPM2B_ECC_POINT * pubPoint,
    byte * out,
    int * outSz
)
```

pubPoint と読み込まれたプライベート ECC 鍵を使用して Z (共有シークレット) を計算します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **privKey** TPM handle を格納する WOLFTPM2_KEY 型の構造体へのポインター
- **pubPoint** TPM2B_ECC_POINT]() 型のデータが取り込まれた構造体へのポインター
- **out** 計算された共有シークレットを格納するバイト バッファへのポインター
- **outSz** 共有シークレットのサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_ECDHGen
- wolfTPM2_ECDHGenKey
- wolfTPM2_ECDHEGenKey
- wolfTPM2_ECDHEGenZ

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ECDHEGenKey

```
WOLFTPM_API int wolfTPM2_ECDHEGenKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * ecdhKey,
    int curve_id
```

)

一時的な ECC 鍵を生成し、配列インデックスを返します (2 フェーズ メソッド)

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **ecdhKey** WOLFTPM2_KEY 型の空の構造体へのポインター
- **curve_id** 有効な TPM_ECC_CURVE 値を指定する整数値

参考:

- wolfTPM2_ECDHEGenZ
- wolfTPM2_ECDHGen
- wolfTPM2_ECDHGenKey
- wolfTPM2_ECDHGenZ

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

One time use key

wolfTPM2_ECDHEGenZ

```
WOLFTPM_API int wolfTPM2_ECDHEGenZ(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * parentKey,
    WOLFTPM2_KEY * ecdhKey,
    const TPM2B_ECC_POINT * pubPoint,
    byte * out,
    int * outSz
)
```

pubPoint とカウンターを使用して Z (共有シークレット) を計算します (2 フェーズメソッド)

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parentKey** プライマリキーの有効な TPM ハンドルを含む WOLFTPM2_KEY 型の構造体へのポインター
- **ecdhKey** TPM handle を格納する WOLFTPM2_KEY 型の構造体へのポインター
- **pubPoint** TPM2B_ECC_POINT 型の空の構造体へのポインター
- **out** 計算された共有シークレットを格納するバイト バッファへのポインター
- **outSz** 共有シークレットのサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_ECDHEGenKey
- wolfTPM2_ECDHGen
- wolfTPM2_ECDHGenKey
- wolfTPM2_ECDHGenZ

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

カウンター、アレイ ID は 1 回だけ使用できます

wolfTPM2_RsaEncrypt

```
WOLFTPM_API int wolfTPM2_RsaEncrypt(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    TPM_ALG_ID padScheme,
    const byte * msg,
    int msgSz,
    byte * out,
    int * outSz
)
```

TPM 2.0 鍵を使用して RSA 暗号化を実行します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター material
- **padScheme** パディング スキームを指定する、TPM_ALG_ID 型の整数値
- **msg** 任意のデータを含むバイトバッファへのポインタ
- **msgSz** 任意のデータ バッファのサイズを指定する整数値
- **out** 暗号化されたデータが格納されるバイトバッファへのポインター
- **outSz** 暗号化されたデータ バッファのサイズを指定する整数値

参考:

- wolfTPM2_RsaDecrypt

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_RsaDecrypt

```
WOLFTPM_API int wolfTPM2_RsaDecrypt(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    TPM_ALG_ID padScheme,
    const byte * in,
    int inSz,
    byte * msg,
    int * msgSz
)
```

TPM 2.0 鍵を使用して RSA 復号を実行します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** TPM キーを保持する WOLFTPM2_KEY 型の構造体へのポインター
- **padScheme** パディング スキームを指定する、TPM_ALG_ID 型の整数値
- **in** 暗号化されたデータを含むバイトバッファへのポインター
- **inSz** 暗号化されたデータ バッファのサイズを指定する整数値
- **msg** 復号されたデータを含むバイトバッファへのポインター
- **msgSz** 暗号化されたデータ バッファのサイズへのポインター。戻り時に実際のサイズが設定される

参考:

- wolfTPM2_RsaEncrypt

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ReadPCR

```
WOLFTPM_API int wolfTPM2_ReadPCR(
    WOLFTPM2_DEV * dev,
    int pcrIndex,
    int hashAlg,
    byte * digest,
    int * pDigestLen
)
```

指定された TPM 2.0 プラットフォーム構成レジスタ (PCR) の値を読み取る。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **pcrIndex** 0 から 23 までの有効な PCR インデックスを指定する整数値 (TPM の局所性は、アクセスの成功に影響を与える可能性があります)
- **hashAlg** アクセスする TPM_ALG_SHA256 または TPM_ALG_SHA1 レジスタを指定する整数値
- **digest** PCR 値が格納されるバイトバッファへのポインター
- **pDigestLen** ダイジェスト バッファのサイズが格納される整数変数へのポインター

参考: wolfTPM2_ExtendPCR

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

PCR レジスタには SHA256 用と SHA1 用の 2 つのセットがあるため、正しいハッシュ アルゴリズムを指定してください (非推奨ですが、引き続き読み取ることができます)。

wolfTPM2_ExtendPCR

```
WOLFTPM_API int wolfTPM2_ExtendPCR(
    WOLFTPM2_DEV * dev,
    int pcrIndex,
    int hashAlg,
    const byte * digest,
    int digestLen
)
```

ユーザー提供のダイジェストで PCR レジスタを拡張します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **pcrIndex** 0 から 23 までの有効な PCR インデックスを指定する整数値 (TPM の局所性は、アクセスの成功に影響を与える可能性があります)
- **hashAlg** アクセスする TPM_ALG_SHA256 または TPM_ALG_SHA1 レジスタを指定する整数値

- **digest** PCR に拡張されるダイジェスト値を含む、バイトバッファへのポインター
- **digestLen** ダイジェスト バッファのサイズ

参考:

- wolfTPM2_ReadPCR

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

正しいハッシュアルゴリズムを指定してください

wolfTPM2_NVCreateAuth

```
WOLFTPM_API int wolfTPM2_NVCreateAuth(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HANDLE * parent,
    WOLFTPM2_NV * nv,
    word32 nvIndex,
    word32 nvAttributes,
    word32 maxSize,
    const byte * auth,
    int authSz
)
```

TPM の NVRAM にデータを格納するために後で使用する新しい NV インデックスを作成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parent** 新しい NV インデックスの TPM 階層を指定する、WOLFTPM2_HANDLE へのポインター
- **nv** WOLFTPM2_NV]() 型の空の構造体へのポインター。新しい NV インデックスを保持します。
- **nvIndex** 成功時に TPM によって与えられた NV インデックス ハンドルを保持する整数値
- **nvAttributes** 整数値、wolfTPM2_GetNvAttributesTemplate を使用して正しい値を作成します
- **maxSize** この NV インデックスで書き込まれる最大バイト数を指定する整数値
- **auth** この NV インデックスのパスワード認証を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_NVWriteAuth
- wolfTPM2_NVReadAuth
- wolfTPM2_NVDeleteAuth
- wolfTPM2_NVOpen

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

これは、TPM2_NV_DefineSpace の wolfTPM2 ラッパーです。

wolfTPM2_NVWriteAuth

```
WOLFTPM_API int wolfTPM2_NVWriteAuth(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_NV * nv,
    word32 nvIndex,
    byte * dataBuf,
    word32 dataSz,
    word32 offset
)
```

指定されたオフセットで、ユーザー データを NV インデックスに格納します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **nv** WOLFTPM2_NV 型のデータが取り込まれた構造体へのポインター
- **nvIndex** 既存の NV インデックス ハンドル値を保持する整数値
- **dataBuf** TPM の NVRAM に書き込まれるユーザー データを含むバイト バッファーへのポインター
- **dataSz** ユーザーデータバッファのサイズをバイト単位で指定する整数値
- **offset** NV インデックス メモリの開始点からのオフセットを指定する word32 型の整数値。ゼロの場合もあります。

参考:

- wolfTPM2_NVReadAuth
- wolfTPM2_NVCreateAuth
- wolfTPM2_NVDeleteAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

ユーザーデータのサイズは、wolfTPM2_CreateAuth を使用して指定された NV インデックスの maxSize 以下である必要があります

wolfTPM2_NVReadAuth

```
WOLFTPM_API int wolfTPM2_NVReadAuth(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_NV * nv,
    word32 nvIndex,
    byte * dataBuf,
    word32 * pDataSz,
    word32 offset
)
```

指定されたオフセットから開始して、NV インデックスからユーザー データを読み取ります。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **nv** WOLFTPM2_NV 型のデータが取り込まれた構造体へのポインター
- **nvIndex** 既存の NV インデックス ハンドル値を保持する整数値
- **dataBuf** TPM の NVRAM からの読み取りデータを格納するために使用される、空のバイト バッファーへのポインター
- **pDataSz** 整数変数へのポインター。NVRAM から読み取ったデータのサイズ (バイト単位) を格納するために使用されます。

- **offset** NV インデックス メモリの開始点からのオフセットを指定する word32 型の整数値。ゼロの場合もあります。

参考:

- wolfTPM2_NVWriteAuth
- wolfTPM2_NVCreateAuth
- wolfTPM2_NVDeleteAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

ユーザーデータのサイズは、wolfTPM2_CreateAuth を使用して指定された NV インデックスの maxSize 以下である必要があります

wolfTPM2_NVIncrement

```
WOLFTPM_API int wolfTPM2_NVIncrement(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_NV * nv
)
```

NV 一方向カウンターをインクリメントします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **nv** WOLFTPM2_NV 型のデータが取り込まれた構造体へのポインター

参考:

- wolfTPM2_NVOpen
- wolfTPM2_NVCreateAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_NVOpen

```
WOLFTPM_API int wolfTPM2_NVOpen(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_NV * nv,
    word32 nvIndex,
    const byte * auth,
    word32 authSz
)
```

NV を開き、必要な認証と名前ハッシュを入力します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **nv** WOLFTPM2_NV]() 型の空の構造体へのポインター。新しい NV インデックスを保持します。
- **nvIndex** 成功時に TPM によって与えられた NV インデックス ハンドルを保持する整数値
- **auth** この NV インデックスのパスワード認証を指定する文字列定数へのポインター

- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_NVCreateAuth
- wolfTPM2_UnloadHandle

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_NVDeleteAuth

```
WOLFTPM_API int wolfTPM2_NVDeleteAuth(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HANDLE * parent,
    word32 nvIndex
)
```

既存の NV インデックスを破棄します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **parent** 新しい NV インデックスの TPM 階層を指定する、WOLFTPM2_HANDLE へのポインター
- **nvIndex** 成功時に TPM によって与えられた NV インデックス ハンドルを保持する整数値

参考:

- wolfTPM2_NVCreateAuth
- wolfTPM2_NVWriteAuth
- wolfTPM2_NVReadAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_NVCreate

```
WOLFTPM_API int wolfTPM2_NVCreate(
    WOLFTPM2_DEV * dev,
    TPM_HANDLE authHandle,
    word32 nvIndex,
    word32 nvAttributes,
    word32 maxSize,
    const byte * auth,
    int authSz
)
```

非推奨です。新しい API を使用してください。

参考:

- wolfTPM2_NVCreateAuth

wolfTPM2_NVWrite

```
WOLFTPM_API int wolfTPM2_NVWrite(
    WOLFTPM2_DEV * dev,
```

```
    TPM_HANDLE authHandle,
    word32 nvIndex,
    byte * dataBuf,
    word32 dataSz,
    word32 offset
)
```

非推奨です。新しい API を使用してください。

参考:

- wolfTPM2_NVWriteAuth

wolfTPM2_NVRead

```
WOLFTPM_API int wolfTPM2_NVRead(
    WOLFTPM2_DEV * dev,
    TPM_HANDLE authHandle,
    word32 nvIndex,
    byte * dataBuf,
    word32 * dataSz,
    word32 offset
)
```

非推奨です。新しい API を使用してください。

参考:

- wolfTPM2_NVReadAuth

wolfTPM2_NVDelete

```
WOLFTPM_API int wolfTPM2_NVDelete(
    WOLFTPM2_DEV * dev,
    TPM_HANDLE authHandle,
    word32 nvIndex
)
```

非推奨です。新しい API を使用してください。

参考:

- wolfTPM2_NVDeleteAuth

wolfTPM2_NVReadPublic

```
WOLFTPM_API int wolfTPM2_NVReadPublic(
    WOLFTPM2_DEV * dev,
    word32 nvIndex,
    TPMS_NV_PUBLIC * nvPublic
)
```

最大サイズなど、nvIndex に関する公開情報を抽出します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **nvIndex** 成功時に TPM によって与えられた NV インデックス ハンドルを保持する整数値
- **nvPublic** TPMS_NV_PUBLIC へのポインター。抽出された nvIndex 公開情報を格納するために使用されます。

参考:

- wolfTPM2_NVCreateAuth
- wolfTPM2_NVDeleteAuth
- wolfTPM2_NVWriteAuth
- wolfTPM2_NVReadAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_NVStoreKey

```
WOLFTPM_API int wolfTPM2_NVStoreKey(
    WOLFTPM2_DEV * dev,
    TPM_HANDLE primaryHandle,
    WOLFTPM2_KEY * key,
    TPM_HANDLE persistentHandle
)
```

TPM 2.0 キーを TPM の NVRAM に格納するヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **primaryHandle** TPM 2.0 階層を指定する整数値。通常は TPM_RH_OWNER
- **key** WOLFTPM2_KEY 型の構造体へのポインター。格納用の TPM 2.0 キーを含みます
- **persistentHandle** 既存の nvIndex を指定する整数値

参考:

- wolfTPM2_NVDeleteKey
- wolfTPM2_NVCreateAuth
- wolfTPM2_NVDeleteAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_NVDeleteKey

```
WOLFTPM_API int wolfTPM2_NVDeleteKey(
    WOLFTPM2_DEV * dev,
    TPM_HANDLE primaryHandle,
    WOLFTPM2_KEY * key
)
```

TPM の NVRAM から TPM 2.0 鍵を削除するヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **primaryHandle** TPM 2.0 階層を指定する整数値。通常は TPM_RH_OWNER
- **key** nvIndex ハンドル値を含む WOLFTPM2_KEY 型の構造体へのポインター

参考:

- wolfTPM2_NVDeleteKey
- wolfTPM2_NVCreateAuth
- wolfTPM2_NVDeleteAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetRng

```
WOLFTPM_API struct WC_RNG * wolfTPM2_GetRng(
    WOLFTPM2_DEV * dev
)
```

wolfTPM に使用される wolfcrypt RNG インスタンスを取得します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

参考:

- wolfTPM2_GetRandom

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

wolfcrypt が有効で、TPM RNG の代わりに使用するように構成されている場合のみ

wolfTPM2_GetRandom

```
WOLFTPM_API int wolfTPM2_GetRandom(
    WOLFTPM2_DEV * dev,
    byte * buf,
    word32 len
)
```

TPM RNG または wolfcrypt RNG で生成された一連の乱数を取得します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **buf** 生成された乱数を格納するために使用されるバイトバッファへのポインター
- **len** バッファーのサイズ (バイト単位) を格納するために使用される word32 型の整数値

参考:

- wolfTPM2_GetRandom

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM RNG ソースを使用するように WOLFTPM2_USE_HW_RNG を定義します。

wolfTPM2_UnloadHandle

```
WOLFTPM_API int wolfTPM2_UnloadHandle(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HANDLE * handle
)
```

TPM がロードされたオブジェクトを破棄するために使用します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **handle** 有効な TPM 2.0 ハンドル値を持つ WOLFTPM2_HANDLE タイプの構造体へのポインター

参考:

- wolfTPM2_Clear

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_Clear

```
WOLFTPM_API int wolfTPM2_Clear(
    WOLFTPM2_DEV * dev
)
```

wolfTPM と wolfcrypt を初期化解除します (有効な場合)

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

参考:

- wolfTPM2_Clear

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_HashStart

```
WOLFTPM_API int wolfTPM2_HashStart(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HASH * hash,
    TPMI_ALG_HASH hashAlg,
    const byte * usageAuth,
    word32 usageAuthSz
)
```

TPM で生成されたハッシュを開始するヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hash** WOLFTPM2_HASH 構造体へのポインター
- **hashAlg** 有効な TPM 2.0 ハッシュ アルゴリズムを指定する整数値
- **usageAuth** ハッシュを後で使用するための承認を指定する文字列定数へのポインター
- **usageAuthSz** i 承認のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_HashUpdate
- wolfTPM2_HashFinish

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_HashUpdate

```
WOLFTPM_API int wolfTPM2_HashUpdate(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HASH * hash,
    const byte * data,
    word32 dataSz
)
```

TPM で生成されたハッシュを新しいユーザー データで更新します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hash** WOLFTPM2_HASH 構造体へのポインター
- **data** ハッシュに追加されるユーザーデータを含むバイトバッファーへのポインター
- **dataSz** ユーザーデータのサイズをバイト単位で指定する word32 型の整数値

参考:

- wolfTPM2_HashStart
- wolfTPM2_HashFinish

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

認証が正しく設定されていることを確認してください

wolfTPM2_HashFinish

```
WOLFTPM_API int wolfTPM2_HashFinish(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HASH * hash,
    byte * digest,
    word32 * digestSz
)
```

TPM で生成されたハッシュをファイナライズし、ユーザー バッファーでダイジェスト出力を取得します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hash** WOLFTPM2_HASH 構造体へのポインター
- **digest** 結果のダイジェストを格納するために使用されるバイトバッファへのポインター
- **digestSz** ダイジェスト バッファーのサイズへのポインター。返されると、ダイジェスト バッファーに格納されているバイト数に設定されます。

参考:

- wolfTPM2_HashStart
- wolfTPM2_HashUpdate

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

認証が正しく設定されていることを確認してください

wolfTPM2_LoadKeyedHashKey

```
WOLFTPM_API int wolfTPM2_LoadKeyedHashKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    WOLFTPM2_HANDLE * parent,
    int hashAlg,
    const byte * keyBuf,
    word32 keySz,
    const byte * usageAuth,
    word32 usageAuthSz
)
```

通常は HMAC 操作に使用される、KeyedHash 型の新しい TPM 鍵を作成して読み込みます。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** 生成される鍵を格納するための WOLFTPM2_KEY 型の空の構造体へのポインター
- **parent** プライマリキーの有効な TPM ハンドルを含む WOLFTPM2_KEY 型の構造体へのポインター
- **hashAlg** 有効な TPM 2.0 ハッシュ アルゴリズムを指定する整数値
- **keyBuf** 新しい KeyedHash キーの派生値を含むバイト配列へのポインター
- **keySz** keyBuf に格納される派生値のサイズをバイト単位で指定する整数値
- **usageAuth** 新しいキーの承認を指定する文字列定数へのポインター
- **usageAuthSz** i 承認のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_HmacStart
- wolfTPM2_HmacUpdate
- wolfTPM2_HmacFinish

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM を使用して HMAC を生成するには、wolfTPM2_Hmac ラッパーを使用することをお勧めします。

wolfTPM2_HmacStart

```
WOLFTPM_API int wolfTPM2_HmacStart(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HMAC * hmac,
    WOLFTPM2_HANDLE * parent,
```

```
    TPMI_ALG_HASH hashAlg,
    const byte * keyBuf,
    word32 keySz,
    const byte * usageAuth,
    word32 usageAuthSz
)
```

TPM で生成された hmac を開始するヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hmac** WOLFTPM2_HMAC 構造体へのポインター
- **parent** プライマリキーの有効な TPM ハンドルを含む WOLFTPM2_KEY 型の構造体へのポインター
- **hashAlg** 有効な TPM 2.0 ハッシュ アルゴリズムを指定する整数値
- **keyBuf** 新しい KeyedHash キーの派生値を含むバイト配列へのポインター
- **keySz** keyBuf に格納される派生値のサイズをバイト単位で指定する整数値
- **usageAuth** 文字列定数へのポインター。後で hmac を使用するための承認を指定します。
- **usageAuthSz** i 承認のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_HmacUpdate
- wolfTPM2_HmacFinish
- wolfTPM2_LoadKeyedHashKey

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_HmacUpdate

```
WOLFTPM_API int wolfTPM2_HmacUpdate(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HMAC * hmac,
    const byte * data,
    word32 dataSz
)
```

TPM で生成された hmac を新しいユーザー データで更新します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hmac** WOLFTPM2_HMAC 構造体へのポインター
- **data** hmac に追加されるユーザー データを含むバイト バッファーへのポインター
- **dataSz** ユーザーデータのサイズをバイト単位で指定する word32 型の整数値

参考:

- wolfTPM2_HmacStart
- wolfTPM2_HMACFinish

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 認証が正しく設定されていることを確認してください

wolfTPM2_HmacFinish

```
WOLFTPM_API int wolfTPM2_HmacFinish(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_HMAC * hmac,
    byte * digest,
    word32 * digestSz
)
```

TPM で生成された hmac をファイナライズし、ユーザー バッファーでダイジェスト出力を取得します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **hmac** WOLFTPM2_HMAC 構造体へのポインター
- **digest** 結果の hmac ダイジェストを格納するために使用されるバイト バッファへのポインター
- **digestSz** ダイジェストのサイズをバイト単位で指定する word32 型の整数値

参考:

- wolfTPM2_HmacStart
- wolfTPM2_HmacUpdate

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

TPM 認証が正しく設定されていることを確認してください

wolfTPM2_LoadSymmetricKey

```
WOLFTPM_API int wolfTPM2_LoadSymmetricKey(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    int alg,
    const byte * keyBuf,
    word32 keySz
)
```

外部の対称鍵を TPM にロードします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 型の空の構造体へのポインター
- **alg** 有効な TPM 2.0 対称キー アルゴリズム（AES CFB の TPM_ALG_CFB）を指定する整数値。
- **keyBuf** 対称鍵の秘密情報を含むバイト配列へのポインター
- **keySz** eyBuf に格納されているキー マテリアルのサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_EncryptDecryptBlock
- wolfTPM2_EncryptDecrypt
- TPM2_EncryptDecrypt2

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_SetCommand

```
WOLFTPM_API int wolfTPM2_SetCommand(
    WOLFTPM2_DEV * dev,
    TPM_CC commandCode,
    int enableFlag
)
```

他の制限付き TPM コマンドを有効にするために使用される、ベンダー固有の TPM コマンド。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **commandCode** 有効なベンダー コマンドを表す整数値
- **enableFlag** 整数値、ゼロ以外の値は「有効にする」ことを表します

参考: TPM2_GPIO_Config

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_Shutdown

```
WOLFTPM_API int wolfTPM2_Shutdown(
    WOLFTPM2_DEV * dev,
    int doStartup
)
```

TPM をシャットダウンまたはリセットするためのヘルパー関数。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **doStartup** 整数値、ゼロ以外の値は「シャットダウン後にスタートアップを実行する」ことを表します

参考: wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

doStartup が設定されている場合、TPM2_Shutdown の直後に TPM2_Startup が実行されます。

wolfTPM2_UnloadHandles

```
WOLFTPM_API int wolfTPM2_UnloadHandles(
    WOLFTPM2_DEV * dev,
    word32 handleStart,
    word32 handleCount
)
```

後続の TPM ハンドルをアンロードするためのワンショット API。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **handleStart** 最初の TPM ハンドルの値を指定する word32 型の整数値
- **handleCount** ハンドル数を指定する word32 型の整数値

参考: wolfTPM2_Init

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_UnloadHandles_AllTransient

```
WOLFTPM_API int wolfTPM2_UnloadHandles_AllTransient(
    WOLFTPM2_DEV * dev
)
```

すべての一時的な TPM ハンドルをアンロードするためのワンショット API。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

参考:

- wolfTPM2_UnloadHandles
- wolfTPM2_CreatePrimary

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

プライマリキーが一時オブジェクトとして存在する場合、TPM 鍵を使用する前にそれらを再作成する必要があります

wolfTPM2_GetKeyTemplate_RSA

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA(
    TPMT_PUBLIC * publicTemplate,
    TPMA_OBJECT objectAttributes
)
```

ユーザーが選択したオブジェクト属性に基づいて、新しい RSA 鍵の TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** TPMT_PUBLIC 型の空の構造体へのポインター。新しい RSA テンプレートを格納します。
- **objectAttributes** TPMA_OBJECT タイプの整数値。1 つ以上の属性（TPMA_OBJECT_fixedTPM 等）を含めることができます。

参考:

- wolfTPM2_GetKeyTemplate_ECC
- wolfTPM2_GetKeyTemplate_Symmetric

- wolfTPM2_GetKeyTemplate_KeyedHash
- wolfTPM2_GetKeyTemplate_KeySeal

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_ECC

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC(
    TPMT_PUBLIC * publicTemplate,
    TPMA_OBJECT objectAttributes,
    TPM_ECC_CURVE curve,
    TPM_ALG_ID sigScheme
)
```

ユーザーが選択したオブジェクト属性に基づいて、新しい ECC 鍵の TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** TPMT_PUBLIC 型の空の構造体へのポインター。新しい ECC キー テンプレートを格納します。
- **objectAttributes** TPMA_OBJECT タイプの整数値。1 つ以上の属性（TPMA_OBJECT_fixedTPM 等）を含めることができます。
- **curve** TPM_ECC_CURVE タイプの整数値。TPM がサポートする ECC 曲線 ID を指定します。
- **sigScheme** TPM_ALG_ID 型の整数値。TPM がサポートする署名方式を指定します。

参考:

- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_Symmetric
- wolfTPM2_GetKeyTemplate_KeyedHash
- wolfTPM2_GetKeyTemplate_KeySeal

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_Symmetric

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_Symmetric(
    TPMT_PUBLIC * publicTemplate,
    int keyBits,
    TPM_ALG_ID algMode,
    int isSign,
    int isDecrypt
)
```

新しい対称鍵の TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** TPMT_PUBLIC 型の空の構造体へのポインター。新しい対称キー テンプレートを格納します。
- **keyBits** 整数値。対称キーのサイズを指定します。通常は 128 ビットまたは 256 ビットです。

- **algMode** TPM_ALG_ID 型の整数値。TPM がサポートする対称アルゴリズム（AES CFB の TPM_ALG_CFB）を指定します。

- **isSign** 整数値、ゼロ以外の値は「署名鍵」を表します
- **isDecrypt** 整数値、ゼロ以外の値は「復号鍵」を表します

参考:

- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC
- wolfTPM2_GetKeyTemplate_KeyedHash
- wolfTPM2_GetKeyTemplate_KeySeal

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_KeyedHash

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_KeyedHash(
    TPMT_PUBLIC * publicTemplate,
    TPM_ALG_ID hashAlg,
    int isSign,
    int isDecrypt
)
```

新しい KeyedHash 鍵の TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター
- **hashAlg** TPM_ALG_ID 型の整数値。TPM がサポートするハッシュ アルゴリズム（SHA 256 の場合は TPM_ALG_SHA256）を指定します。

- **isSign** 整数値、ゼロ以外の値は「署名鍵」を表します
- **isDecrypt** 整数値、ゼロ以外の値は「復号鍵」を表します

参考:

- wolfTPM2_GetKeyTemplate_RSA
- wolfTPM2_GetKeyTemplate_ECC
- wolfTPM2_GetKeyTemplate_Symmetric
- wolfTPM2_GetKeyTemplate_KeySeal

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_KeySeal

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_KeySeal(
    TPMT_PUBLIC * publicTemplate,
    TPM_ALG_ID nameAlg
)
```

シークレットを封印するための新しい鍵の TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

- **nameAlg** TPM_ALG_ID 型の整数値。TPM がサポートするハッシュ アルゴリズム（SHA 256 の場合は TPM_ALG_SHA256）を指定します。

参考:

- wolfTPM2_GetKeyTemplate_ECC
- wolfTPM2_GetKeyTemplate_Symmetric
- wolfTPM2_GetKeyTemplate_KeyedHash
- wolfTPM2_GetKeyTemplate_KeySeal

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

鍵封印には厳しい要件があるため、ほとんどの鍵パラメータはラッパーによって事前に決定されます。

wolfTPM2_GetKeyTemplate_RSA_EK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA_EK(
    TPMT_PUBLIC * publicTemplate
)
```

RSA タイプの TPM 承認鍵を生成するための TPM 公開テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_ECC_EK
- wolfTPM2_GetKeyTemplate_RSA_SRK
- wolfTPM2_GetKeyTemplate_RSA_AIK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_ECC_EK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC_EK(
    TPMT_PUBLIC * publicTemplate
)
```

ECC タイプの TPM 承認鍵を生成するための TPM 公開テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_RSA_EK
- wolfTPM2_GetKeyTemplate_ECC_SRK
- wolfTPM2_GetKeyTemplate_ECC_AIK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_RSA_SRK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA_SRK(
    TPMT_PUBLIC * publicTemplate
)
```

RSA タイプの新しい TPM ストレージ 鍵を生成するための TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_ECC_SRK
- wolfTPM2_GetKeyTemplate_RSA_AIK
- wolfTPM2_GetKeyTemplate_RSA_EK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_ECC_SRK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC_SRK(
    TPMT_PUBLIC * publicTemplate
)
```

ECC タイプの新しい TPM ストレージ 鍵を生成するための TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_RSA_SRK
- wolfTPM2_GetKeyTemplate_ECC_AIK
- wolfTPM2_GetKeyTemplate_ECC_EK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_RSA_AIK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_RSA_AIK(
    TPMT_PUBLIC * publicTemplate
)
```

RSA タイプの新しい TPM Attestation Key を生成するための TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_ECC_AIK
- wolfTPM2_GetKeyTemplate_RSA_SRK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyTemplate_ECC_AIK

```
WOLFTPM_API int wolfTPM2_GetKeyTemplate_ECC_AIK(
    TPMT_PUBLIC * publicTemplate
)
```

ECC タイプの新しい TPM Attestation Key を生成するための TPM パブリック テンプレートを準備します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター

参考:

- wolfTPM2_GetKeyTemplate_RSA_AIK
- wolfTPM2_GetKeyTemplate_ECC_SRK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_SetKeyTemplate_Unique

```
WOLFTPM_API int wolfTPM2_SetKeyTemplate_Unique(
    TPMT_PUBLIC * publicTemplate,
    const byte * unique,
    int uniqueSz
)
```

Create または CreatePrimary で使用されるパブリック テンプレートの一意の領域を設定します。

パラメータ:

- **publicTemplate** 新しいテンプレートを格納する TPMT_PUBLIC 型の空の構造体へのポインター
- **unique** パブリック テンプレートの一意の領域を設定するためのバッファーへのオプションのポインター。NULL の場合、バッファはゼロ化されます。
- **uniqueSz** 一意のフィールドを埋めるサイズ。ゼロの場合、キー サイズが使用されます。

参考:

- wolfTPM2_CreateKey
- wolfTPM2_CreatePrimaryKey

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetNvAttributesTemplate

```
WOLFTPM_API int wolfTPM2_GetNvAttributesTemplate(
    TPM_HANDLE auth,
    word32 * nvAttributes
)
```

TPM NV インデックス テンプレートを準備します。

パラメータ:

- **auth** 新しい TPM NV インデックスが作成される TPM 階層を表す整数値
- **nvAttributes** NV 属性を格納するための空の整数変数へのポインター

参考:

- wolfTPM2_CreateAuth
- wolfTPM2_WriteAuth
- wolfTPM2_ReadAuth
- wolfTPM2_DeleteAuth

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CreateEK

```
WOLFTPM_API int wolfTPM2_CreateEK(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * ekKey,
    TPM_ALG_ID alg
)
```

ユーザーが選択したアルゴリズム、RSA または ECC に基づいて、新しい TPM 承認キーを生成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **ekKey** 新しい EK に関する情報を格納するための空の WOLFTPM2_KEY 構造体へのポインター
- **alg** TPM_ALG_RSA または TPM_ALG_ECC のみを指定できます。上記の注を参照してください

参考:

- wolfTPM2_CreateSRK
- wolfTPM2_GetKeyTemplate_RSA_EK
- wolfTPM2_GetKeyTemplate_ECC_EK

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

ノート:

EK に使用できるのは RSA と ECC のみですが、TPM で対称鍵を作成して使用できます

wolfTPM2_CreateSRK

```
WOLFTPM_API int wolfTPM2_CreateSRK(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * srkKey,
    TPM_ALG_ID alg,
    const byte * auth,
    int authSz
)
```

他の TPM キーのストレージ キーとして使用される新しい TPM プライマリ キーを生成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **srkKey** 新しい EK に関する情報を格納するための空の WOLFTPM2_KEY 構造体へのポインター
- **alg** TPM_ALG_RSA または TPM_ALG_ECC のみを指定できます。上記の注を参照してください
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreateEK
- wolfTPM2_CreateAndLoadAIK
- wolfTPM2_GetKeyTemplate_RSA_SRK
- wolfTPM2_GetKeyTemplate_ECC_SRK

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

EK に使用できるのは RSA と ECC のみですが、TPM で対称鍵を作成して使用できます

wolfTPM2_CreateAndLoadAIK

```
WOLFTPM_API int wolfTPM2_CreateAndLoadAIK(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * aikKey,
    TPM_ALG_ID alg,
    WOLFTPM2_KEY * srkKey,
    const byte * auth,
    int authSz
)
```

指定されたストレージ鍵の下に新しい TPM 構成証明鍵を生成します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **aikKey** 新しく生成された TPM キーを格納するための空の WOLFTPM2_KEY 構造体へのポインター
- **alg** TPM_ALG_RSA または TPM_ALG_ECC のみを指定できます
- **srkKey** WOLFTPM2_KEY 構造体へのポインター。ロードされたストレージ キーの有効な TPM ハンドルを指します。
- **auth** TPM 2.0 キーのパスワード承認を指定する文字列定数へのポインター
- **authSz** パスワード認証のサイズをバイト単位で指定する整数値

参考:

- wolfTPM2_CreateSRK
- wolfTPM2_GetKeyTemplate_RSA_AIK
- wolfTPM2_GetKeyTemplate_ECC_AIK

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetTime

```
WOLFTPM_API int wolfTPM2_GetTime(
    WOLFTPM2_KEY * aikKey,
    GetTime_Out * getTimeOut
)
```

TPM 署名付きタイムスタンプを生成するワンショット API。

パラメータ:

- **aikKey** WOLFTPM2_KEY 構造体へのポインター。ロードされた認証キーの有効な TPM ハンドルを含みます
- **getTimeOut** コマンドの出力を格納するための GetTime_Out 型の空の構造体へのポインター

参考:

- wolfTPM2_CreateSRK
- wolfTPM2_GetKeyTemplate_RSA_EK
- wolfTPM2_GetKeyTemplate_ECC_EK

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

ノート:

この呼び出しの前に、構成証明鍵を生成してロードする必要があります

wolfTPM2_CSR_SetCustomExt

```
WOLFTPM_API int wolfTPM2_CSR_SetCustomExt(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CSR * csr,
    int critical,
    const char * oid,
    const byte * der,
    word32 derSz
)
```

WOLFTPM2_CSR 構造体のカスタム要求拡張 oid と値の使用を設定する証明書署名要求 (CSR) 生成のヘルパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター (not used)
- **csr** WOLFTPM2_CSR 構造体へのポインター
- **critical** 0 の場合、拡張機能はクリティカルとマークされません。それ以外の場合、クリティカルとマークされます。
- **oid** 文字列としてのドット区切りの oid。たとえば、「1.2.840.10045.3.1.7」
- **der** 拡張子のコンテンツの der エンコーディング
- **derSz** der エンコーディングのバイト単位のサイズ

参考:

- wolfTPM2_CSR_SetSubject
- wolfTPM2_CSR_SetKeyUsage
- wolfTPM2_CSR_MakeAndSign
- wolfTPM2_CSR_MakeAndSign_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_SetKeyUsage

```
WOLFTPM_API int wolfTPM2_CSR_SetKeyUsage(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CSR * csr,
    const char * keyUsage
```

)

WOLFTPM2_CSR 構造のキー使用法を設定する証明書署名要求 (CSR) 生成のヘルパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター (not used)
- **csr** WOLFTPM2_CSR 構造体へのポインター
- **keyUsage** コンマ区切りのキー使用属性の文字列リスト。可能な値: any、serverAuth、clientAuth、codeSigning、emailProtection、timeStamping、および OCSPSigning デフォルト: "serverAuth,clientAuth,codeSigning"

参考:

- wolfTPM2_CSR_SetSubject
- wolfTPM2_CSR_SetCustomExt
- wolfTPM2_CSR_MakeAndSign
- wolfTPM2_CSR_MakeAndSign_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_SetSubject

```
WOLFTPM_API int wolfTPM2_CSR_SetSubject(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CSR * csr,
    const char * subject
)
```

WOLFTPM2_CSR 構造体のサブジェクトを設定する証明書署名要求 (CSR) 生成のヘルパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター (not used)
- **csr** WOLFTPM2_CSR 構造体へのポインター
- **subject** /CN= 構文を使用した識別名文字列。例: "/C=US/ST=Washington/L=Seattle/O=wolfSSL/OU=Development/C

参考:

- wolfTPM2_CSR_SetKeyUsage
- wolfTPM2_CSR_SetCustomExt
- wolfTPM2_CSR_MakeAndSign
- wolfTPM2_CSR_MakeAndSign_ex

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_MakeAndSign_ex

```
WOLFTPM_API int wolfTPM2_CSR_MakeAndSign_ex(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CSR * csr,
    WOLFTPM2_KEY * key,
    int outFormat,
    byte * out,
    int outSz,
    int sigType,
```

```
    int selfSignCert,
    int devId
)
```

TPM ベースの鍵 (件名と鍵の使用法が既に設定されている WOLFTPM2_KEY 構造) を使用した証明書署名要求 (CSR) 生成のヘルパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **csr** WOLFTPM2_CSR 構造体へのポインター
- **key** WOLFTPM2_KEY 構造体へのポインター
- **outFormat** WOLFSSL_FILETYPE_ASN1 または WOLFSSL_FILETYPE_PEM
- **out** ASN.1/DER または PEM としての CSR の宛先バッファ
- **outSz** 宛先バッファの最大サイズ
- **sigType** keyType (CTC_SHA256wRSA または CTC_SHA256wECDSA) に基づいて SHA2-256 を自動的に選択するには、0 を使用します。可能な値のリストについては、wolfCrypt の「enum Ctc_SigType」を参照してください。
- **selfSignCert** 1 (ゼロ以外) に設定すると、結果は自己署名証明書になります。ゼロ (0) は、CA によって使用される CSR (証明書署名要求) を生成します。
- **devId** 暗号コールバックの登録時に使用されるデバイス識別子。INVALID_DEVID (-2) を使用して、必要な暗号化コールバックを自動的に登録します。

参考:

- wolfTPM2_CSR_SetSubject
- wolfTPM2_CSR_SetKeyUsage
- wolfTPM2_CSR_SetCustomExt
- wolfTPM2_CSR_MakeAndSign

戻り値:

- Success: 正の整数 (出力のサイズ)
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_MakeAndSign

```
WOLFTPM_API int wolfTPM2_CSR_MakeAndSign(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_CSR * csr,
    WOLFTPM2_KEY * key,
    int outFormat,
    byte * out,
    int outSz
)
```

TPM ベースの鍵 (件名と鍵の使用法が既に設定されている WOLFTPM2_KEY 構造) を使用した証明書署名要求 (CSR) 生成のヘルパー。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **csr** WOLFTPM2_CSR 構造体へのポインター
- **key** WOLFTPM2_KEY 構造体へのポインター
- **outFormat** WOLFSSL_FILETYPE_ASN1 または WOLFSSL_FILETYPE_PEM
- **out** ASN.1/DER または PEM としての CSR の宛先バッファ
- **outSz** 宛先バッファの最大サイズ

参考:

- wolfTPM2_CSR_SetSubject
- wolfTPM2_CSR_SetKeyUsage
- wolfTPM2_CSR_SetCustomExt
- wolfTPM2_CSR_MakeAndSign_ex

戻り値:

- Success: 正の整数 (出力のサイズ)
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_Generate_ex

```
WOLFTPM_API int wolfTPM2_CSR_Generate_ex(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const char * subject,
    const char * keyUsage,
    int outFormat,
    byte * out,
    int outSz,
    int sigType,
    int selfSignCert,
    int devId
)
```

TPM ベースの鍵 (WOLFTPM2_KEY) を使用した証明書署名要求 (CSR) 生成のヘルパー。TPM 鍵に基づいて CSR または自己署名証明書を出力するためのシングル ショット API。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** WOLFTPM2_KEY 構造体へのポインター
- **subject** /CN= 構文を使用した識別名文字列。例: "/C=US/ST=Washington/L=Seattle/O=wolfSSL/OU=Development/C
- **keyUsage** コンマ区切りのキー使用属性の文字列リスト。可能な値: any、serverAuth、clientAuth、codeSigning、emailProtection、timeStamping、および OCSPSigning デフォルト: "serverAuth,clientAuth,codeSigning"
- **outFormat** WOLFSSL_FILETYPE_ASN1 または WOLFSSL_FILETYPE_PEM
- **out** ASN.1/DER または PEM としての CSR の宛先バッファ
- **outSz** 宛先バッファの最大サイズ
- **sigType** keyType (CTC_SHA256wRSA または CTC_SHA256wECDSA) に基づいて SHA2-256 を自動的に選択するには、0 を使用します。可能な値のリストについては、wolfCrypt の「enum Ctc_SigType」を参照してください。
- **selfSignCert** 1 (ゼロ以外) に設定すると、結果は自己署名証明書になります。ゼロ (0) は、CA によって使用される CSR (証明書署名要求) を生成します。
- **devId** 暗号コールバックの登録時に使用されるデバイス識別子。INVALID_DEVID (-2) を使用して、必要な暗号化コールバックを自動的に登録します。

参考:

- wolfTPM2_SetCryptoDevCb
- wolfTPM2_CSR_Generate

戻り値:

- Success: 正の整数 (出力のサイズ)
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CSR_Generate

```
WOLFTPM_API int wolfTPM2_CSR_Generate(
    WOLFTPM2_DEV * dev,
    WOLFTPM2_KEY * key,
    const char * subject,
    const char * keyUsage,
    int outFormat,
    byte * out,
    int outSz
)
```

TPM ベースの鍵 (WOLFTPM2_KEY) を使用した証明書署名要求 (CSR) 生成のヘルパー。TPM 鍵に基づいて CSR または自己署名証明書を出力するためのシングル ショット API。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **key** pointer to a loaded WOLFTPM2_KEY 構造体へのポインター
- **subject** /CN= 構文を使用した識別名文字列。例: "/C=US/ST=Washington/L=Seattle/O=wolfSSL/OU=Development/C
- **keyUsage** コンマ区切りのキー使用属性の文字列リスト。可能な値: any、serverAuth、clientAuth、codeSigning、emailProtection、timeStamping、および OCSPSigning デフォルト: "serverAuth,clientAuth,codeSigning"
- **outFormat** WOLFSSL_FILETYPE_ASN1 または WOLFSSL_FILETYPE_PEM
- **out** ASN.1/DER または PEM としての CSR の宛先バッファ
- **outSz** 宛先バッファの最大サイズ

参考:

- wolfTPM2_SetCryptoDevCb
- wolfTPM2_CSR_Generate_ex

戻り値:

- Success: 正の整数 (出力のサイズ)
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_CryptoDevCb

```
WOLFTPM_API int wolfTPM2_CryptoDevCb(
    int devId,
    wc_CryptoInfo * info,
    void * ctx
)
```

クリプトオフロードに TPM を使用するためのリファレンス クリプト コールバック API。このコールバック関数は、wolfTPM2_SetCryptoDevCb または wc_CryptoDev_RegisterDevice を使用して登録されます。

パラメータ:

- **devId** コールバックの登録時に使用される devId。INVALID_DEVID 以外の符号付き整数値
- **info** 暗号タイプとパラメーターに関する詳細情報を含む wc_CryptoInfo 構造を指す
- **ctx** コールバックが wolfTPM2_SetCryptoDevCb に登録されたときに提供されたユーザー コンテキスト

参考:

- wolfTPM2_SetCryptoDevCb
- wolfTPM2_ClearCryptoDevCb

戻り値:

- TPM_RC_SUCCESS: 成功
- CRYPTOCB_UNAVAILABLE: TPM ハードウェアを使用せず、デフォルトのソフトウェア暗号にフォールバックします。
- WC_HW_E: 一般的なハードウェア障害

wolfTPM2_SetCryptoDevCb

```
WOLFTPM_API int wolfTPM2_SetCryptoDevCb(
    WOLFTPM2_DEV * dev,
    CryptoDevCallbackFunc cb,
    TpmCryptoDevCtx * tpmCtx,
    int * pDevId
)
```

暗号コールバック関数を登録し、割り当てられた devId を返します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **cb** wolfTPM2_CryptoDevCb API はテンプレートですが、独自のものを提供することもできます
- **tpmCtx** ユーザー指定のコンテキスト。wolfTPM2_CryptoDevCb には TpmCryptoDevCtx を使用しますが、独自のものにすることもできます。
- **pDevId** 自動的に割り当てられたデバイス ID へのポインター

参考:

- wolfTPM2_CryptoDevCb
- wolfTPM2_ClearCryptoDevCb

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_ClearCryptoDevCb

```
WOLFTPM_API int wolfTPM2_ClearCryptoDevCb(
    WOLFTPM2_DEV * dev,
    int devId
)
```

登録された暗号コールバックをクリアします。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **devId** コールバックの登録時に使用される devId

参考:

- wolfTPM2_CryptoDevCb
- wolfTPM2_SetCryptoDevCb

戻り値:

- TPM_RC_SUCCESS: 成功
- TPM_RC_FAILURE: 一般的なエラー (TPM IO と TPM リターンコードを確認のこと)
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_New

```
WOLFTPM_API WOLFTPM2_DEV * wolfTPM2_New(
    void
)
```

WOLFTPM2_DEV を割り当てて初期化します。

参考:

- wolfTPM2_Free

戻り値:

- 新しいデバイス構造体へのポインター
- NULL: エラー時

wolfTPM2_Free

```
WOLFTPM_API int wolfTPM2_Free(
    WOLFTPM2_DEV * dev
)
```

wolfTPM2_New によって割り当てられた WOLFTPM2_DEV をクリーンアップして解放します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター

参考:

- wolfTPM2_New

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_NewKeyBlob

```
WOLFTPM_API WOLFTPM2_KEYBLOB * wolfTPM2_NewKeyBlob(
    void
)
```

WOLFTPM2_KEYBLOB を割り当てて初期化します。

参考:

- wolfTPM2_FreeKeyBlob

戻り値:

- 新しく初期化されたポインタ WOLFTPM2_KEYBLOB
- エラーの場合は NULL

wolfTPM2_FreeKeyBlob

```
WOLFTPM_API int wolfTPM2_FreeKeyBlob(
    WOLFTPM2_KEYBLOB * blob
)
```

wolfTPM2_NewKeyBlob で割り当てられた WOLFTPM2_KEYBLOB を解放します。

パラメータ:

- **blob** wolfTPM2_NewKeyBlob によって割り当てられた WOLFTPM2_KEYBLOB へのポインター

参考:

- wolfTPM2_NewKeyBlob

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_NewPublicTemplate

```
WOLFTPM_API TPMT_PUBLIC * wolfTPM2_NewPublicTemplate(
    void
)
```

TPMT_PUBLIC 構造体を割り当てて初期化します。

参考:

- wolfTPM2_FreePublicTemplate

戻り値:

- 新しく初期化されたポインタ
- エラーの場合は NULL

wolfTPM2_FreePublicTemplate

```
WOLFTPM_API int wolfTPM2_FreePublicTemplate(
    TPMT_PUBLIC * PublicTemplate
)
```

wolfTPM2_NewPublicTemplate で割り当てられた TPMT_PUBLIC を解放します。

パラメータ:

- **PublicTemplate** wolfTPM2_NewPublicTemplate で割り当てられた TPMT_PUBLIC へのポインター

参考:

- wolfTPM2_NewPublicTemplate

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_NewKey

```
WOLFTPM_API WOLFTPM2_KEY * wolfTPM2_NewKey(
    void
)
```

WOLFTPM2_KEY を割り当てて初期化します。

参考:

- wolfTPM2_FreeKey

戻り値:

- 新しく初期化されたポインタ WOLFTPM2_KEY
- エラーの場合は NULL

wolfTPM2_FreeKey

```
WOLFTPM_API int wolfTPM2_FreeKey(
    WOLFTPM2_KEY * key
)
```

wolfTPM2_NewKey で割り当てられた WOLFTPM2_KEY を解放します。

パラメータ:

- **key** wolfTPM2_NewKey によって割り当てられた WOLFTPM2_KEY へのポインター

参考:

- wolfTPM2_NewKey

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_NewSession

```
WOLFTPM_API WOLFTPM2_SESSION * wolfTPM2_NewSession(
    void
)
```

WOLFTPM2_SESSION を割り当てて初期化します。

参考:

- wolfTPM2_FreeSession

戻り値:

- 新しく初期化されたポインタ WOLFTPM2_SESSION
- エラーの場合は NULL

wolfTPM2_FreeSession

```
WOLFTPM_API int wolfTPM2_FreeSession(
    WOLFTPM2_SESSION * session
)
```

wolfTPM2_NewSession で割り当てられた WOLFTPM2_SESSION を解放します。

パラメータ:

- **blob** wolfTPM2_NewSession によって割り当てられた WOLFTPM2_KEYBLOB へのポインター

参考:

- wolfTPM2_NewSession

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_NewCSR

```
WOLFTPM_API WOLFTPM2_CSR * wolfTPM2_NewCSR(
    void
)
```

WOLFTPM2_CSR を割り当てて初期化します。

参考:

- wolfTPM2_FreeCSR

戻り値:

- 新しく初期化されたポインタ WOLFTPM2_CSR
- エラーの場合は NULL

wolfTPM2_FreeCSR

```
WOLFTPM_API int wolfTPM2_FreeCSR(
    WOLFTPM2_CSR * csr
)
```

wolfTPM2_NewCSR で割り当てられた WOLFTPM2_CSR を解放します。

パラメータ:

- **blob** wolfTPM2_NewCSR によって割り当てられた WOLFTPM2_CSR へのポインター

参考:

- wolfTPM2_NewCSR

戻り値:

- TPM_RC_SUCCESS: 成功

wolfTPM2_GetHandleRefFromKey

```
WOLFTPM_API WOLFTPM2_HANDLE * wolfTPM2_GetHandleRefFromKey(
    WOLFTPM2_KEY * key
)
```

WOLFTPM2_KEY から WOLFTPM2_HANDLE を取得します。

パラメータ:

- **key** WOLFTPM2_KEY 構造体へのポインター

戻り値:

- 鍵構造体内のハンドルへのポインター
- キーポインタが NULL の場合は NULL

wolfTPM2_GetHandleRefFromKeyBlob

```
WOLFTPM_API WOLFTPM2_HANDLE * wolfTPM2_GetHandleRefFromKeyBlob(
    WOLFTPM2_KEYBLOB * keyBlob
)
```

WOLFTPM2_KEYBLOB から WOLFTPM2_HANDLE を取得します。

パラメータ:

- **key** WOLFTPM2_KEYBLOB 構造体へのポインター

戻り値:

- キー blob 構造体で内のハンドルへのポインター
- キーポインタが NULL の場合は NULL

wolfTPM2_GetHandleRefFromSession

```
WOLFTPM_API WOLFTPM2_HANDLE * wolfTPM2_GetHandleRefFromSession(
    WOLFTPM2_SESSION * session
)
```

WOLFTPM2_SESSION から WOLFTPM2_HANDLE を取得します。

パラメータ:

- **key** pointer to a WOLFTPM2_SESSION struct

戻り値:

- セッション構造体内のハンドルへのポインター
- キーポインタが NULL の場合は NULL

wolfTPM2_GetHandleValue

```
WOLFTPM_API TPM_HANDLE wolfTPM2_GetHandleValue(
    WOLFTPM2_HANDLE * handle
)
```

WOLFTPM2_HANDLE から 32 ビットのハンドル値を取得します。

パラメータ:

- **handle** WOLFTPM2_HANDLE 構造体へのポインター

戻り値:

- TPM_HANDLE 値

wolfTPM2_SetKeyAuthPassword

```
WOLFTPM_API int wolfTPM2_SetKeyAuthPassword(
    WOLFTPM2_KEY * key,
    const byte * auth,
    int authSz
)
```

鍵の認証データを設定します。

パラメータ:

- **dev** TPM2_DEV 構造体へのポインター
- **auth** 認証データへのポインター
- **authSz** 認証データの長さ (バイト単位)

戻り値:

- TPM_RC_SUCCESS: 成功
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_GetKeyBlobAsBuffer

```
WOLFTPM_API int wolfTPM2_GetKeyBlobAsBuffer(
    byte * buffer,
    word32 bufferSz,
    WOLFTPM2_KEYBLOB * key
)
```

キーブロブからバイナリ バッファーにデータをマーシャリングします。これは、別のプロセスでロードするため、または電源の再投入後にディスクに保存できます。

パラメータ:

- **buffer** マーシャリングされたキーブロブを格納するバッファへのポインター
- **bufferSz** 上記のバッファのサイズ
- **key** マーシャリングするキーブロブへのポインター

参考:

- wolfTPM2_SetKeyBlobFromBuffer

戻り値:

- 正の整数 (出力のサイズ)
- BUFFER_E: 提供されたバッファに十分なスペースがありません
- BAD_FUNC_ARG: 不正な引数

wolfTPM2_SetKeyBlobFromBuffer

```
WOLFTPM_API int wolfTPM2_SetKeyBlobFromBuffer(
    WOLFTPM2_KEYBLOB * key,
    byte * buffer,
    word32 bufferSz
)
```

データを WOLFTPM2_KEYBLOB 構造体にアンマーシャリングします。これは、wolfTPM2_GetKeyBlobAsBuffer によって以前にマーシャリングされたキーブロブをロードするために使用できます。

パラメータ:

- **key** データをロードしてアンマーシャリングするキーブロブへのポインター
- **buffer** ロード元のマーシャリングされたキーブロブを含むバッファへのポインター
- **bufferSz** 上記のバッファのサイズ

参考:

- wolfTPM2_GetKeyBlobAsBuffer

戻り値:

- TPM_RC_SUCCESS: 成功
- BUFFER_E: バッファーが小さすぎるか、非整列化されていない余分なデータが残っています
- BAD_FUNC_ARG: 不正な引数

---

5 引用元

[1.] Wikipedia contributors. (2018, May 30). Trusted Platform Module. In *Wikipedia, The Free Encyclopedia*. Retrieved 22:46, June 20, 2018, from https://en.wikipedia.org/w/index.php?title=Trusted_Platform_Module&oldid=

[2.] Arthur W., Challener D., Goldman K. (2015) Platform Configuration Registers. In: A Practical Guide to TPM 2.0. Apress, Berkeley, CA

