|  |  |  |
| --- | --- | --- |
| AN230694 | | |
| Semper SDKでSemper Secure NORフラッシュを使用するためのホスト要件   |  | | --- | | 著者：ヴィンセントハン | | 関連するパーツファミリ：S35HL-T / S35HS-T  S36HL-T/S36HS-T  S38HL-T/S38HS-T | | | |
| このアプリケーションノートでは、安全なデータの保存と転送のためにSemper Secure NORフラッシュでSemper™ソリューション開発キット（SDK）を使用するためのホスト側の要件について説明します。また、安全な目的でホストアプリケーションソフトウェアを実装するためのガイドラインと提案も提供します。 |

目次

[1 はじめに 1](#_Toc44022205)

[2 Semper SecureNORフラッシュセキュリティの機能と機能 2](#_Toc44022206)

[3 非対称システムアーキテクチャと対称システムアーキテクチャ 2](#_Toc44022207)

[3.1 非対称システムアーキテクチャ 2](#_Toc44022208)

[3.2 対称システムアーキテクチャ 3](#_Toc44022209)

[4 Semper SDK 4](#_Toc44022210)

[5 Semper SDKでSemper SecureNORフラッシュを使用するためのホスト要件 5](#_Toc44022211)

[5.1 非対称システム 5](#_Toc44022212)

[5.2 対称システム 6](#_Toc44022213)

[5.3 ハードウェアアブストラクションレイヤ（HAL） 8](#_Toc44022214)

[5.4 暗号化アルゴリズム検証（CAV）テスト機能 11](#_Toc44022215)

[5.5 主要な世代 12](#_Toc44022216)

[6 S-SDKでのSemperの安全な例 13](#_Toc44022217)

[6.1 プロビジョニング 13](#_Toc44022218)

[6.2 セキュア ブート 14](#_Toc44022219)

[6.3 安全な保管 15](#_Toc44022220)

[6.4 RMA 16](#_Toc44022221)

[7 結論 16](#_Toc44022222)

[8 リファレンス 16](#_Toc44022223)

[改訂履歴 17](#_Toc44022224)

# はじめに

Semper Secure NOR フラッシュファミリは、安全でセキュアな電子システムを構築するためのセキュリティ機能を提供します。Semper SDKには、フラッシュデバイスドライバの完全なセット、ハードウェアアブストラクションレイヤ（HAL）の例、およびアプリケーションコードの例が含まれており、ソフトウェア開発を迅速に開始させるのに役立ちます。

このアプリケーションノートでは、Semper Secure NORフラッシュとSemper SDKの主な機能を紹介し、Semper SecureとSemper SDKを使用するためのホスト側の要件を示し、ホスト側でのSemper SDKコード統合とアプリケーションソフトウェアに関するガイダンスを提供します。

Semper SDKでSemper Secureの使用を開始する前に、基本的な用語、安全な概念および使用法についてSemper Secureのデータシートを読むことを強くお勧めします。Semper Secure NORフラッシュの詳細については、リファレンスに記載されているアプリケーションノートを参照してください。

このドキュメントでは、他の指定子が使用されていない場合、Semper Secureフラッシュデバイスを「デバイス」、接続されたMCU / SoCを「ホスト」、Semperソリューション開発キットを「S-SDK」と呼びます。

# Semper Secure NORフラッシュセキュリティの特徴と機能

Semper Secure NORフラッシュファミリデバイスは、セキュリティ機能とともに機能安全を提供し、安全で安全なシステムの構築に役立ちます。主なセキュリティ機能は次のとおりです。

* **最大8つのセキュアリージョンを使用したアレイパーティショニング**

Semper Secure NORフラッシュは、顧客の機密データを保存するために最大8つのリージョンを提供します。リージョンごとに異なるセキュアレベルと特権レベルを選択できます。

* **非対称および対称キーアルゴリズムのサポート**

Semper Secure NORフラッシュは、非対称または対称キーアルゴリズム（異なる部品番号による）をサポートしており、さまざまなユーザーの製造およびアプリケーションのシナリオを満たすことができます。

* **認証/暗号化された読み取り、プログラム、および消去操作**

Semper Secure NORフラッシュは、マルチレベルのセキュリティ特権をサポートしています。さまざまなレベルのアプリケーションは、さまざまなセキュリティレベルのフラッシュ操作を持つことができるため、ホスト側でアプリケーションアクセスを管理できます。

* **エフェメラルセッションキーの生成**

Semper Secure NORフラッシュは、セッションベースのエフェメラルキーをサポートします。この動的なセッションキーメカニズムにより、システムはピン/ラインスニファ攻撃からより安全になります。

* **キーとユーザー単調カウンターの安全なストレージ**

Semper Secure NORフラッシュは最大64個のキーストレージをサポートし、関連するセキュアリージョンで動作します。また、最大64ビットの単調コマンドカウンターを提供し、ホストMCUとフラッシュ間の安全なトランザクション中にカウンター値を使用します。これにより、再生されたパッケージ攻撃からシステムを保護できます。

* **デバイス識別子構成エンジン（DICE）**

Semper Secure非対称デバイスは、Trusted Computing Group（TCG）DICE仕様に従って、すべてのパワーオンリセット（POR）でエイリアス証明書とDeviceID証明書を作成します。

* **高速で安全なブートサポート**

Semper Secure NORフラッシュは、高速セキュアブートをサポートし、ホストMCU、Semper Secureデバイス、およびファームウェアのブート信頼性を維持するセキュアブートコマンドを提供します。

主要なセキュリティ機能のパフォーマンスについては、Semper Secureデータシートの「Performance Summary」セクションを参照してください。

# 非対称システムアーキテクチャと対称システムアーキテクチャ

Semper Secure NORフラッシュファミリは、さまざまな使用モデルおよびアプリケーション向けに非対称および対称の両方の品番のデバイスを提供します。

## 非対称システムアーキテクチャ

このアーキテクチャは、製品が複数の場所で製造されているか、OEMによって外部委託されているシナリオで使用されます。プロセスの安全性と信頼性を維持するために、各部分には独自の秘密キーまたは公開キーのペアがあり、証明書と公開キーを相互に共有するだけです。この場合、ボンディングのためにこれらの証明書と公開キーを相互に転送するには、最初のペアリングが必要となります。

図1．非対称システムペアリング



## 対称システムアーキテクチャ

対称システムは、ホスト側とフラッシュ側の間で同じ秘密を共有します。ホストは、最初のペアリング中に共有シークレットをフラッシュにプログラムする必要があります。

図2．対称システムペアリング



# Semper SDK

S-SDKは、Semper NORフラッシュメモリを使用したアプリケーション開発を簡素化および高速化するために、複数の開発プラットフォームで機能することが証明されている使いやすいソフトウェアパッケージです。

図3．Semperソリューション開発キットのアーキテクチャ



S-SDKは、次のパーツを提供します。

* フラッシュ、セキュリティ、および機能安全のための製品グレードおよびMISRA-C準拠のリファレンスドライバでSemper NOR フラッシュファミリデバイスをサポート。
* システム統合を簡素化するためのすべてのハードウェアインターフェイス用HAL。
* 複数のアプリケーション例とリファレンスHAL実装。
* 複数の開発環境、ツールチェーンおよびIDEをサポートするクロスプラットフォームSDK。
* 評価キットとフラッシュメモリモジュール。
* プレシリコンソフトウェア開発用のフラッシュメモリCモデル。
* Semper Secureソリューションとのホストの互換性を確保するためのCAVPテストベクトルに基づく暗号化アルゴリズム検証モジュール。
* 評価用のwolfSSLのwolfCryptライブラリ。
* 追加のハードウェアなしでLinux上でSemper Secureサンプルアプリケーションをすぐに実行できるように事前に構成されたセットアップ。

Semper SecureおよびSemper SDKの詳細は、[Semper Secure Early Access Program（EAP）](https://go.cypress.com/semper-secure-early-access-program)にアクセスしてください。

# Semper SDKでSemper Secure NORフラッシュを使用するためのホスト要件

S-SDKは、フラッシュドライバ、安全なドライバ、機能安全ドライバ、および対応するHALの例を提供します。これらは、安全なソフトウェアを独自のシステムに実装するのに役立ちます。

S-SDKでSemper Secureを有効にするには、ホストMCU / SoCに少なくとも次の機能が必要です。

* Semper Secureデータシートに記載されている暗号化アルゴリズムを使用したハードウェアまたはソフトウェアによる暗号化機能（HASH / ECC）。
* 非対称システムの場合はデバイスID（UDS\_HOST）を格納するための不揮発性セキュアストレージ、または対称システムの場合はMASTER\_KEYを格納するための不揮発性セキュアストレージ。
* UDS\_HOSTまたはMASTER\_KEYから派生したキーを格納するための不揮発性または揮発性の安全なストレージ。
* 速い茶色のキツネは怠惰な犬を飛び越えます。

## 非対称システム

非対称システムの場合、フラッシュに保存されるホスト証明書を生成するには、マスター秘密/公開キーのペアとマスターセッションキーが必要です。

非対称システムの特定のホスト要件は次のとおりです。

* **暗号化関数**
* ハッシュ関数（SHA2-256）
  + Semper Secureでデータの有効性を検証できる、メモリ操作を認証するためにホスト側でHMAC値を生成するための測定。
  + ホストとSemper Secureの間で交換されるパケットのHMAC値を生成および検証するためのパケット認証用のMAC生成。
  + マスター/リージョンセッションキーの作成中にホスト側でHMAC値を生成するためのキー導出。
* 楕円曲線暗号（ECDH P-256）
  + キー導出、NIST SP.800-56A、ホスト側でのマスター/リージョンセッションキーの作成中に中間キーを生成するためのc（0e、2s）スキーム。
  + ホストまたはOEM証明書の署名生成用のECDSAP-256。
* AES（AES-GCM 256）（オプション）
  + 暗号化された読み取り、プログラム、消去操作のために、ホスト側で暗号化されたパケットを暗号化または復号化するための暗号化トランザクション（AES-GCM 256）。
  + パケット認証用のMAC生成-AES-GCMタグは、認証された暗号化操作のためにホスト上で転送されるパケットのMACとして使用されます。
* Semper Secureファームウェアの検証、キープログラミング、およびマスター/リージョンセッションキーの作成中にホスト側でナンスを生成する真のランダムジェネレータ。ReadNonceコマンドを使用して、Semper Secureから32バイトのナンス値を取得することもできます。
* **不揮発性セキュアストレージ（OTP / eFUSE / eフラッシュ）**
* 不揮発性の安全なストレージに保存されているホスト固有のデバイスシークレット（UDS\_HOST）。これは、デバイスのペアリングとキーまたは証明書の生成に必要です。
* （HL0 | DEVICEID\_PUB | OEM\_PUB）（256ビット）（推奨）のハッシュ結果は、すべてのPORでセキュアブート用の不揮発性セキュアストレージに保存されます。これは推奨される例であり、変更できます。
* **揮発性のセキュアストレージ（RAM）**
* 最大64のHOST\_REGION\_PRIV＆amp; PUBキーペア（オプション）
* REGION\_SESSION最大8つのリージョンのキー（オプション）
* **不揮発性セキュアストレージまたは揮発性セキュアストレージ**

不揮発性の安全なストレージに保存されるか、UDS\_HOSTから派生し、実行時に揮発性の安全なストレージに保存されます。

* HOST\_MASTER\_PRIV＆amp; PUBキーペア
* （RMA\_PRIV＆amp; PUBキーペア、RMA\_SESSIONキー）（オプション）
* MASTER\_SESSIONキー

したがって、非対称フラッシュデバイスは、必須機能およびオプション機能をサポートするための関連するキーと証明書を提供します。

* **キー（Semper Secure）**
* ProgramKeyコマンドによるHOST\_REGION\_PUB（オプション）
* ProgramKeyコマンドによるRMA\_KEY（オプション）
* GetRegionPublicKeyコマンドで生成されたフラッシュ\_REGIONキーペア（オプション）
* Create / StoreSessionKeyコマンドで生成されたMASTER\_SESSION\_KEY
* **証明書（Semper Secure）**
* InstallCertificateコマンドを介してインストールされたOEM\_CERT
* InstallCertificateコマンドを介してインストールされたHOST\_CERT
* ReadCertificateコマンドで生成されたALIAS\_CERT
* ReadCertificateコマンドで生成されたALIAS\_PRIV1
* ReadCertificateコマンドで生成されたDEVICEID\_CERT1
* ReadCertificateコマンドで生成されたDEVICEID\_PRIV1
* サイプレス工場に設置されたCY\_CERT
* **ROTP / ROM（Semper Secure）**
* SetInitialConfigコマンドで生成されたUDS
* サイプレス工場に設置されたCY\_PUB

1. MASTER\_SESSIONキーはリージョンキーに使用でき、ALIAS\_PUBキーはフラッシュ\_REGIONキーに使用できます。

\*\*

## 対称システム

対称システムの場合、ホストはマスターキーを生成し、デバイスボンディングのためにこのシークレットをフラッシュと共有します。

対称システムの特定のホスト要件は次のとおりです。

* **暗号化関数**
* ハッシュ関数（SHA2-256）
  + Semper Secureでのデータの有効性を検証できる、メモリ操作を認証するためにホスト上でHMAC値を生成するための測定。
  + ホストとSemper Secureの間で交換されるパケットのHMAC値を生成および検証するためのパケット認証用のMAC生成。
  + マスター/リージョンセッションキーの作成中にホスト上でHMAC値を生成するためのキー導出。
* 楕円曲線暗号（ECDH P-256）
  + キー導出、NIST SP.800-56A、ホストでのマスター/リージョンセッションキーの作成中に中間キーを生成するためのc（0e、2s）スキーム。
* AES (AES-GCM 256)
  + 暗号化された読み取り、プログラム、消去操作のために、ホスト上の暗号化されたパケットを暗号化または復号化するための暗号化トランザクション（AES-GCM 256）。
  + パケット認証用のMAC生成-AES-GCMタグは、認証された暗号化操作のためにホスト上で転送されるパケットのMACとして使用されます。
* Semper Secureファームウェアの検証、キープログラミング、およびマスター/リージョンセッションキーの作成中にホスト上でナンスを生成する真のランダムジェネレータ。ReadNonceコマンドを使用して、Semper Secureから32バイトのナンス値を取得することもできます。
* **不揮発性セキュアストレージ（OTP / eFUSE / eフラッシュ）**
* Semper Secure側からの共有シークレットとして不揮発性セキュアストレージに格納されているMASTERキー。
* デバイスのペアリングとキーまたは証明書の生成に必要な、不揮発性の安全なストレージに保存されている一意のデバイスシークレット（オプション）。
* **揮発性のセキュアストレージ（RAM）**
* REGIONキー（オプション）
* REGION\_SESSION最大8つのリージョンのキー（オプション）
* **不揮発性セキュアストレージまたは揮発性セキュアストレージ**

不揮発性の安全なストレージに保存されるか、UDS\_HOSTから派生し、実行時に揮発性の安全なストレージに保存されます。

* MASTER\_SESSIONキー
* （RMAキー、RMA\_Sessionキー）（オプション）

したがって、対称フラッシュデバイスは、必須およびオプションの機能をサポートするための関連するキーと証明書を提供します。

* **キー（Semper Secure）**
* ProgramKeyコマンドによるREGIONキー（オプション）
* ProgramKeyコマンドによるRMA\_KEY（オプション）
* CreateKey / ProgramKey / Create / StoreSessionKeyコマンドによるMASTER\_KEY
* **証明書（Semper Secure）**
* サイプレス工場に設置されたCY\_CERT
* **ROTP / ROM（Semper Secure）**
* SetInitialConfigコマンドで生成されたUDS
* サイプレス工場に設置されたCY\_PUB

1. MASTERキーはリージョンキーに使用できます。高速で安全な起動を行うには、MASTERキーをOTP / eFUSEに保存することをお勧めします。

## ハードウェアアブストラクションレイヤ（HAL）

SDKSDKドライバの間を橋渡しするために、コードの移行と適応が必要です。HALは、S-SDKユーザーが準備または移行する必要がある主要なコンポーネントです。

図4．Semper SDKとユーザーアプリケーション関数が階層を呼び出す



S-SDKに必要なHALコンポーネントは次のとおりです。

* 暗号
* SMC（SPIメモリコントローラー）
* ユーティリティ（オプション）

### 暗号

Crypto HALは、安全な操作のためにSemper Secureが必要とする暗号化アルゴリズムを実装する必要があります。Crypto HALのテンプレートCファイルは、SEMPER\_SDK / Templateフォルダーにあります。このテンプレートで関数の本体を実装できます。

以下は、rypto HALAPIのリストです。

SSDK\_HAL\_CRYPTO\_T MY\_CRYPTO\_HAL =

{

  .Initialize         = HAL\_Crypto\_Initialize,

  .Uninitialize       = HAL\_Crypto\_Uninitialize,

  .GetHostCertificate  = HAL\_Crypto\_GetHostCertificate,

  .DecodeCertificate   = HAL\_Crypto\_DecodeCertificate,

  .ValidateCertificate = HAL\_Crypto\_ValidateCertificate,

  .EcdhP256           = HAL\_Crypto\_EcdhP256\_KH,

  .HkdfSha256         = HAL\_Crypto\_HkdfSha256\_KH,

  .GetPublicKey       = HAL\_Crypto\_GetPublicKey,

  .GenerateNonce      = HAL\_Crypto\_GenerateNonce,

  .HashSha256Init     = HAL\_Crypto\_HashSha256Init,

  .HashSha256Update   = HAL\_Crypto\_HashSha256Update,

  .HashSha256Final    = HAL\_Crypto\_HashSha256Final,

  .HmacSha256Init     = HAL\_Crypto\_HmacSha256Init\_KH,

  .HmacSha256Update   = HAL\_Crypto\_HmacSha256Update,

  .HmacSha256Final    = HAL\_Crypto\_HmacSha256Final,

  .AesGcmEncrypt      = HAL\_Crypto\_AesGcmEncrypt\_KH,

  .AesGcmDecrypt      = HAL\_Crypto\_AesGcmDecrypt\_KH,

  .StoreDeviceHash    = HAL\_Crypto\_StoreDeviceHash,

  .ValidateDeviceHash = HAL\_Crypto\_ValidateDeviceHash,

  .ValidateAlgorithm  = HAL\_Crypto\_RunCAV

};

Crypto HALのHAL\_Crypto\_GenrateNonce関数は、ユーザー側で何をする必要があるかを示す例として使用されます。

S-SDKサンプルコードSEMPER\_SDK / Platform / Linux\_x86-64\_GCC / host / ssdk\_hal\_ctypto\_wc.cでは、CryptoHALのアルゴリズムはwolfCryptで実装されています。このファイルでは、HAL\_Crypto\_GenerateNonceは、wolfCrypt関数wc\_RNG\_GenerateBlockを呼び出して、ホスト上にナンスを生成することによって実装されます。

static int32\_t HAL\_Crypto\_GenerateNonce(uint8\_t \*NoncePtr, uint32\_t NonceSize)

{

  WC\_RNG  WcRng;

  wc\_InitRng(&WcRng);

  int wc\_ret = wc\_RNG\_GenerateBlock(&WcRng, NoncePtr, NonceSize);

  wc\_FreeRng(&WcRng);

  return wc\_ret == 0 ? SSDK\_E\_NO\_ERROR : (wc\_ret + SSDK\_E\_HAL\_CRYPTO\_BASE);

}

SEMPER\_SDK/Template/ssdk\_hal\_crypto.cに同じ名前の関数があります。これは、空のプレースホルダー関数です。

/\*\*

 \* \brief   Generate nonce.

 \* \param   [out] NoncePtr  - Pointer to buffer for nonce

 \* \param   [in]  NonceSize - Size of nonce to generate

 \* \return   0    Success (SSDK\_E\_NO\_ERROR)

 \* \return  <0    Error   (SSDK\_E\_xxx)

 \*                        <SSDK\_E\_HAL\_CRYPTO\_BASE for HAL specific error code

 \*/

static int32\_t HAL\_Crypto\_GenerateNonce(uint8\_t \*NoncePtr, uint32\_t NonceSize)

{

  /\* TODO: Add Your Code \*/

  return SSDK\_E\_NO\_ERROR;

}

ホストでソフトウェアライブラリまたはハードウェア暗号化エンジンを使用して、同じ機能で本体を実装する必要があります。

### シリアルメモリコントローラー（SMC）

以下は、SMCHALのAPIのリストです。

SSDK\_HAL\_SMCTRL\_T SSDK\_HAL\_SMCTRL\_0 =

{

  .Initialize      = HAL\_SMC\_Initialize,

  .Uninitialize    = HAL\_SMC\_Uninitialize,

  .GetCapabilities = HAL\_SMC\_GetCapabilities,

  .Transfer        = HAL\_SMC\_Transfer,

  .Reset           = HAL\_SMC\_Reset

};

これらの機能は、ターゲットシステムの対応するソフトウェアまたはハードウェアSPIコントローラー（または他のシリアルメモリコントローラー、ドライバ付きメモリインターフェイス、BSP）ドライバを使用して実装する必要があります。

### ユーティリティ

Utils HALには、memcpy、memcmp、memset、delayなどのユニバーサル関数が含まれています。SEMPER\_SDK/Driver/Common/ssdk\_utils.cのデフォルトAPIは、MISRA-C準拠のCコードを提供します。また、ハードウェアタイマー、グローバルタイムスタンプ、または特定のクロスコンパイラーによって提供される最適化されたライブラリを使用してAPIを実装し、効率を向上させることもできます。

## 暗号化アルゴリズム検証（CAV）テスト機能

Semper Secureとコード間の暗号化アルゴリズムの一貫性を維持するには、ユーザーが実装した暗号化アルゴリズムまたはハードウェア暗号化エンジン出力の有効性と互換性をテストするための信頼できるテストベクトルが必要です。

Semper SDKのCryptoHALテンプレートはCAV関数を提供するため、Semper Secureデバイスと通信する前に実装を確認できます。

/\*\*

 \* \anchor page\_getting\_started\_ssdk\_crypto\_cavp

 \* \brief   Run Cryptographic Algorithm Validation.

 \* \return   0  Success (SSDK\_E\_NO\_ERROR)

 \* \return  <0  Error   (SSDK\_E\_xxx)

 \*/

static int32\_t HAL\_Crypto\_RunCAV(void)

{

  int32\_t err;

  err = HAL\_Crypto\_CAV\_EcdhP256();

  printf("HAL\_Crypto\_CAV\_EcdhP256:      %d\n", err);

  err = HAL\_Crypto\_CAV\_HkdfSha256();

  printf("HAL\_Crypto\_CAV\_HkdfSha256:    %d\n", err);

  err = HAL\_Crypto\_CAV\_HashSha256();

  printf("HAL\_Crypto\_CAV\_HashSha256:    %d\n", err);

  err = HAL\_Crypto\_CAV\_HmacSha256();

  printf("HAL\_Crypto\_CAV\_HmacSha256:    %d\n", err);

  err = HAL\_Crypto\_CAV\_AesGcmEncrypt();

  printf("HAL\_Crypto\_CAV\_AesGcmEncrypt: %d\n", err);

  err = HAL\_Crypto\_CAV\_AesGcmDecrypt();

  printf("HAL\_Crypto\_CAV\_AesGcmDecrypt: %d\n", err);

  return SSDK\_E\_NO\_ERROR;

}

テンプレート内のCAVテストベクトルは、[NIST CAVPWebサイト](https://csrc.nist.gov/Projects/Cryptographic-Algorithm-Validation-Program)からのものです。

## 主要な世代

Semper Secureは、相互キー生成にECDH P-256（NIST SP.800-56A、c（0e、2s）スキームを使用）を使用し、署名生成にECDSAP-256を使用します。

S-SDKの例の暗号HALファイルSEMPER\_SDK/Platform/Linux\_x86-64\_GCC/host/ssdk\_hal\_ctypto\_wc.cでは、次の静的キーペアがアレイにハードコーディングされています。

非対称デバイス：

* HOST\_MASTER\_PRIV
* HOST\_MASTER\_PUB
* HOST\_REGION\_PRIV
* HOST\_REGION \_PUB
* RMA\_PRIV
* RMA\_PUB

対称デバイス：

* MASTER\_KEY
* REGION\_KEY
* RMA\_ KEY

このファイルでは、wolfCryptを使用して暗号化アルゴリズムを実装しています。次の手順に従って、ホスト側でキーペアを生成できます。

  WC\_RNG  rng;e

  ecc\_key key;

  byte buf[128];

  word32 bufsz = sizeof(buf);

/\* Initialization RNG and ECC module in WolfCrypt \*/

  wc\_InitRng(&rng);

  wc\_ecc\_init(&key);

/\* Create New Key Pair with ECDSA Signature \*/

  wc\_ecc\_make\_key\_ex(&rng, 32, &key, ECC\_SECP256R1);

  /\* Export Public Key \*/

  wc\_ecc\_export\_x963(&key, buf, &bufsz);

  /\* Export Private Key \*/

  wc\_ecc\_export\_private\_only(&key, buf, &bufsz);

  wc\_FreeRng(&rng);

  wc\_ecc\_free(&key);

これらの手順を実行した後、ターミナルまたはバッファを介してキーを出力し、さらに使用するために内容をアレイにコピーできます。キーの作成にwolfCryptを使用することは、S-SDKのほんの一例です。必要に応じて、ハードウェア暗号化エンジンまたはopenSSLなどの他のソフトウェアライブラリを使用して、キーペアを生成できます。

# S-SDKでのSemperの安全な例

学習曲線を短縮し、アプリケーション開発に集中できるように、S-SDKには、Semper Secureの一般的な使用シナリオを含むサンプルコードがいくつか用意されています。

4つの主要な例は、SEMPER\_SDK/Example/ssdk\_secure\_example.cにあります。

## プロビジョニング

「Secure\_Eg\_Provisioning」関数は、Semper Secureを初期化してホストにペアリングし、セキュアリージョンを構成し、チップステータスをNORMALからPROVISION、次にSECUREに転送する方法の例を示しています。図5に、プロビジョニングのサンプルコードフローを示します。

図5．S-SDK Semper Secure-プロビジョニングのサンプルコードフロー



## セキュアブート

「Secure\_Eg\_SecureBoot」関数は、フラッシュ上のリソースを使用してセキュアブートを実装する方法の例を示しています。セキュアブートはシステムレベルの機能であり、ホストとフラッシュ間の相乗効果が必要です。この例では、フラッシュ操作を実装するだけです。システムのセキュアブート手順を完了するには、他の重要な部分（次のレベルのブートイメージの認証、セキュアリージョンの機密データにアクセスするためのセッションキーの作成など）を追加する必要があります。図6に、セキュアブートのサンプルコードフローを示します。

図6．S-SDK Semper Secure - セキュアブートのサンプルコードフロー



## 安全な保管

「Secure\_Eg\_SecureStorage」関数は、セキュアトランザクションを使用してSemper Secureにアクセスする方法と、単調カウンターを使用する方法の例を示しています。図7は、セキュアストレージのサンプルコードフローを示しています。

図7．S-SDK Semper Secure - セキュアストレージのサンプルコードフロー



## RMA

「Secure\_Eg\_RMA」関数は、フラッシュステータスをSECUREからRMAに移行する方法の例を示しています。RMAモードは、初期のセキュアパラメータで設定可能であり、RMA移行は必須ではありません。図8は、RMAのサンプルコードフローを示しています。

図8．S-SDK Semper Secure RMAサンプルコードフロー



# 結論

サイプレスS-SDKは、独自のアプリケーションを開発したり、提供されたコードを直接使用したりできるように設計されています。S-SDKは、MISRA-C準拠の低レベルドライバとHALコードを提供します。これにより、開発をスピードアップし、市場投入までの時間を短縮できます。S-SDKコードの例に従うか、プラットフォームに依存しないCコードを使用して、このドキュメントに記載されている手順を実行できます。

# リファレンス

002-26101 データシートS35HS-T、S35HL-TSemper™セキュアフラッシュ、クアッドSPI

002-28332 AN228332 Cypress Semper™セキュアNORフラッシュでの初期プロビジョニング

002-28825 AN228825セキュアストレージ用のSemper™セキュアフラッシュのセットアップ

002-29503 AN229503 RMA用のSemper™セキュアNORフラッシュのセットアップ

# 改訂履歴

文書タイトル: AN230694 – Semper SDKでSemper Secure NOR Flashを使用するためのホスト要件

文書番号: 002-30694

| **版** | **技術変更届** | **発行日** | **変更内容** |
| --- | --- | --- | --- |
|  |  |  |  |