TRAVEO™T2G ファミリでの CRYPTO モジュールの使用方法

本書について

適用範囲と目的

AN220253 では、TRAVEO™ T2G ファミリの暗号ブロック (CRYPTO) モジュールの設定および使用方法について説明します。このアプリケーションノートでは、非対称/対称暗号化/復号化, ハッシュ値計算, 真性乱数と擬似乱数生成など、さまざまな暗号関連機能の使用方法について説明します。

対象者

このドキュメントは TRAVEO™ T2G ファミリ CRYPTO ファミリを使用するすべての人を対象にします。

関連製品ファミリ

TRAVEO™ T2G

目次

目次

1 はじめに

今日の組込みアプリケーションでは、セキュリティ機能に関する要件がますます高まっています。車載アプリケーションでは、一般的に次の機能を実行するためにセキュリティ機能が使用されます。

- 知的財産の保護
- 車内の他の電子制御ユニットまたは外部の電子制御ユニットとの間のメッセージの認証
- 改ざん防止 (例えば、エンジンパラメータの調整やトリップレコーダの変更防止など)
- アフターマーケットでの機能有効化の許可 (例えば、スピードリミッタの有効化など)
- OEM (オリジナルの機器メーカ) のスペアパーツのみ使用を許可する

TRAVEO™ T2G ファミリのマイクロコントローラは、このような使用事例を可能にするため特別に開発されたセキュリティ機能を提供します。デバッグインタフェースまたはマイクロコントローラで実行されるソフトウェアのアクセス許可を制御するライフサイクルおよび保護スキームに加え、このアプリケーションノートで説明する暗号ブロック (CRYPTO) も重要なコンポーネントの 1 つです。本アプリケーションノートでは、インフィニオンのサンプルドライバライブラリ (SDL) の一部である"Crypto"ドライバを用いて暗号機能について説明します。

2 暗号化機能概要と特長

2.1 乱数生成器 (Random Number Generator:RNG)

擬似乱数生成器 (Pseudo Random Number Generator: PRNG) と真性乱数生成器 (True Random Number Generator:TRNG) の 2 種類があります。

PRNG は、多項式と与えられた開始値 (以下「シード」と記載します) に基づいて、ランダムに見える数列を出力します。PRNG に関する知識をもつ攻撃者は、将来の乱数を予測できる場合があります。

TRNG は、真性のランダム性の根源として物理的性質 (例えば熱ノイズ) を使用します。通常、後処理ステップが実行され、TRNG の出力を無効にしようとする攻撃者によって引き起こされた TRNG の誤動作を検出するために、監視機能が実装されます。

PRNG の利点は、通常 TRNG よりも乱数生成が高速な点です。さらに、物理的な実装に応じて、アクティブな TRNG は特定のアプリケーションにおいては、電力損失を引き起こすことがあります (例えば、リングオシレータがダイナミックスイッチング電流を発生させるなど)。これらは、多くのアプリケーションが TRNG を使用して PRNG の真性のランダムシードを生成する理由です。

2.2 対称暗号化

対称暗号化は、メッセージの送信者と受信者の間の共有秘密情報に依存します。同じ秘密鍵が暗号化と復号化に使用されます。

2.3 非対称暗号化

非対称暗号化の場合、各当事者はプライベート鍵と公開鍵を持っています。非対称暗号化の原則は、送信者が受信者の公開鍵を使用して受信者が秘密鍵で解読できるメッセージを暗号化できるようにします。または、送信者はその秘密鍵を使用してメッセージのデジタル署名を生成でき、受信者は送信者の公開鍵を使用してこのメッセージを検証できます。

2.4 ハッシュ関数

ハッシュ関数は、可変長入力に対して固定長出力 ("ハッシュ値"または"ダイジェスト") を生成します。暗号ハッシュ関数は、その特性に関連してより強い要件を持つ特別な種類のハッシュ関数です。

ハッシュ関数への入力データとともにハッシュ値を使用して、入力データまたはハッシュ値の完全性をチェックできます。

2.5 メッセージ認証コード (MAC)

MAC は、暗号ハッシュ関数または対称暗号に基づいています。メッセージの完全性に加えて、送信者と受信者の間で共有される秘密のために、メッセージの信頼性を保証できます。

2.6 デジタル署名

デジタル署名は、暗号ハッシュ関数と組み合わせて非対称暗号を使用することによって生成されます。これらは、完全性, 真正性, 否認防止の 3 つのセキュリティ機能を提供します。つまり送信者は、送信者のプライベート鍵を使用して署名を生成し、受信者が送信者の公開鍵を使用して、これを証明できます 。

3 Crypto ドライバ

3.1 ドライバアーキテクチャ

Crypto ドライバの設計は、クライアントサーバアーキテクチャに基づいています。Crypto サーバは CM0+コアでのみ動作し、適切な Crypto Core 関数を実行することで Crypto ハードウェアと連携します。Crypto クライアントはどちらのコアでも実行できますが、このアプリケーションノートでは、クライアントが CM4 コアで動作するケースについてのみ説明します。CM4 を参照するすべての説明と図は、CM7 を搭載した TRAVEO™ T2G デバイスにも適用されます。クライアントとサーバは、プロセッサ間通信 (IPC) を介して通信します。通信に IPC を使用すると、異なるコアからの同時要求を処理するシンプルな同期メカニズムが提供されます。

Crypto ドライバは、CRYPTO ブロック以外の次のハードウェアリソースを使用します。

- クライアントとサーバ間のデータ交換用の 1 つの IPC チャネル構造
- 通知のための 2 つの IPC 割込み構造
- ハードウェアエラーのハンドリング用に 1 つの割込み

Crypto サーバとクライアント間の一般的な通信の概念を図 1 に示します。

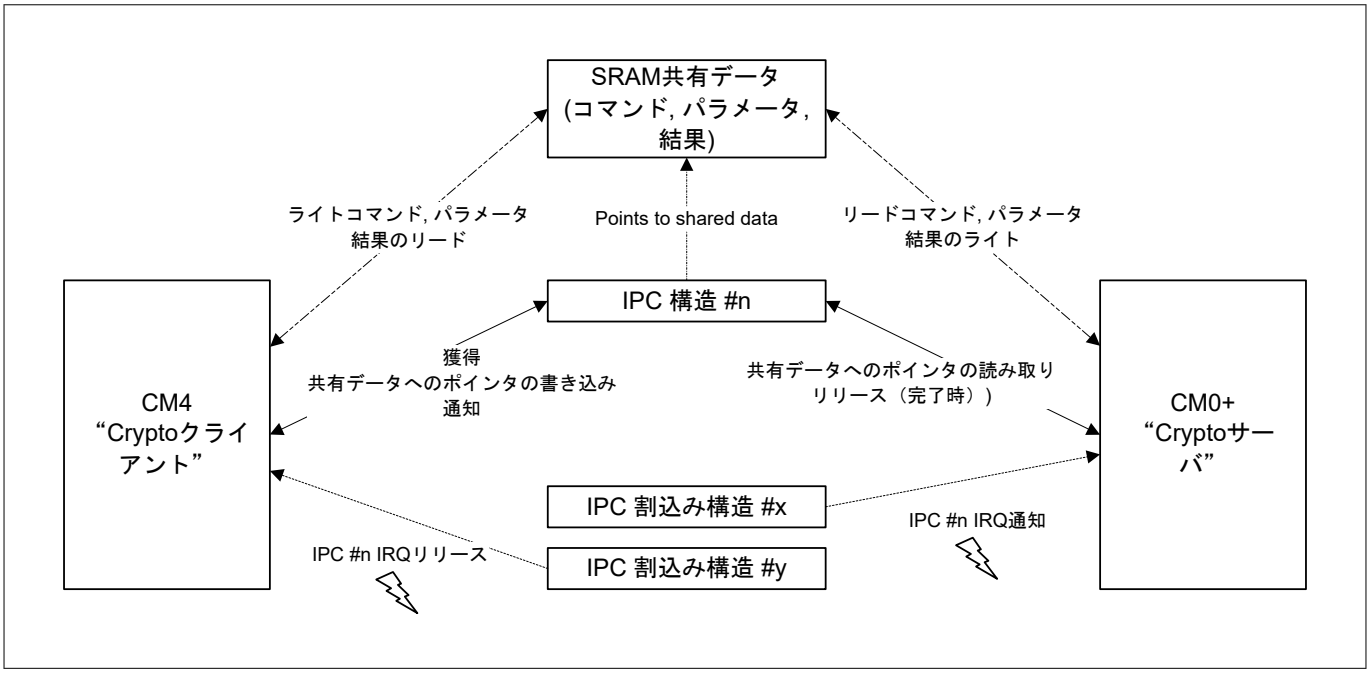

図 1 IPC 経由のサーバ/クライアント通信例

3.2 ドライバの初期化

ドライバを暗号操作に使用する前に、正しい順序でドライバを初期化する必要があります。

1. Crypto サーバは CM0+ で初期化する必要があります。
2. Crypto サーバの初期化が完了すると、クライアントは CM4 によって正常に初期化されます。

注: Crypto サーバは CM0+ 上で動作し、クライアントは CM4 上で実行できます。そして、両方のコアは互いに異なる周波数で独立して実行されるため、クライアントの初期化時にサーバの初期化が完了しないことがあります。理想的には、この状況を回避するために、CM0+ は Crypto サーバ初期化後、CM4 のリセットを解除します。あるいは、クライアント初期化の前に、サーバの状態を確認できます。

図 2 に、Crypto サーバとクライアントの初期化を示します。

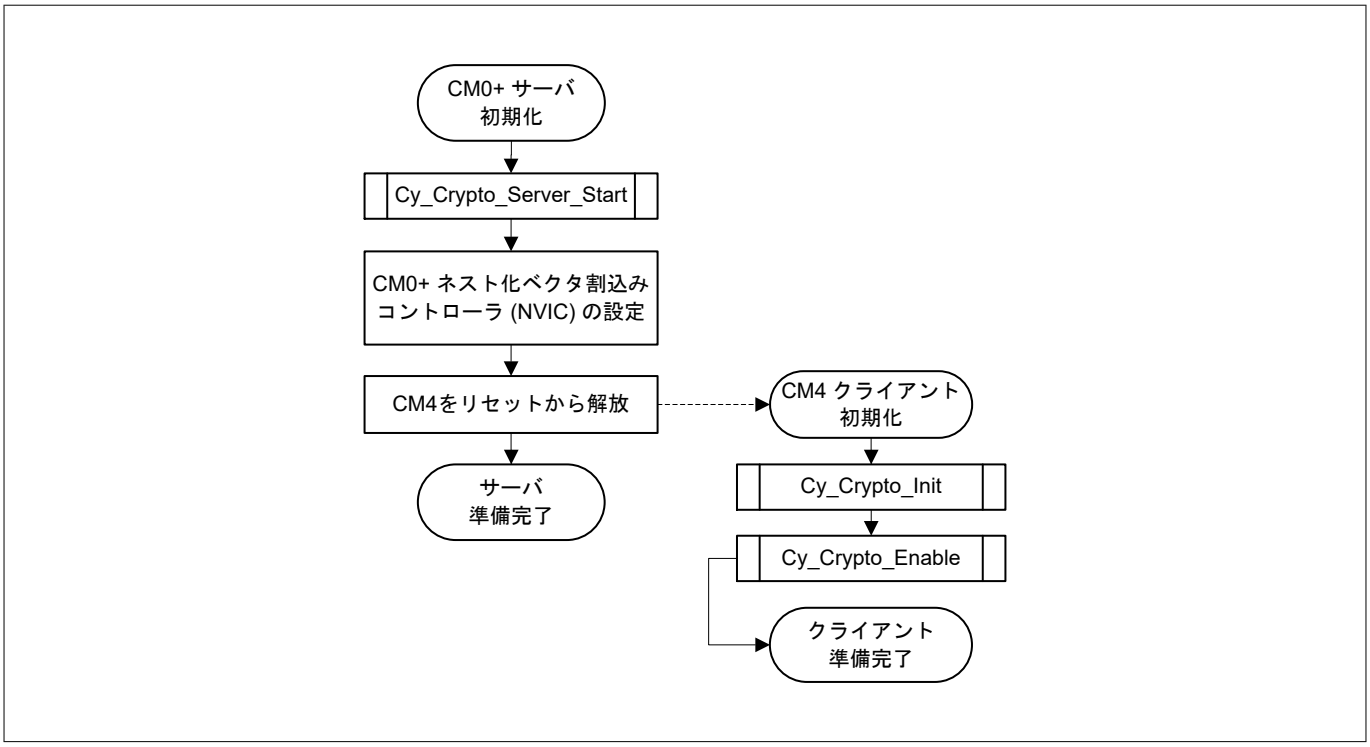

図 2　　サーバ/クライアントの初期化フローチャート例

3.3　　ドライバの使い方

サーバとクライアントの初期化が完了すると、クライアントはサーバから暗号操作を要求できます。CM0+ で実行されているサーバが要求された操作を処理している間、CM4 は他のタスクを処理できます。操作の終了は、3 つの異なる方法で決定できます。

1. Cryptoドライバ関数 `Cy_Crypto_Sync` をブロッキング モードで呼び出します。関数が戻ると、クライアントは前の操作が完了したことを確認できます。
2. Cryptoドライバ関数を `Cy_Crypto_Sync` を定期的にノンブロッキング モードで呼び出し、戻り値をチェックします。戻り値がサーバの準備ができていることを示している場合、前操作は完了です。
3. クライアントは、IPC リリース割込みを設定して、サーバ操作が完了した直後に通知を受け取れます。

このアプリケーションノートでは、第 1 の方法 `Cy_Crypto_Sync`(Cy_Crypto_Sync をブロッキングモードで呼び出すこと) が使用され、関数パラメータと "block = true"値によってもフローチャートに示されます。図 3 に、初期化されたCryptoドライバの一般的な使い方を示します。

3 Crypto ドライバ

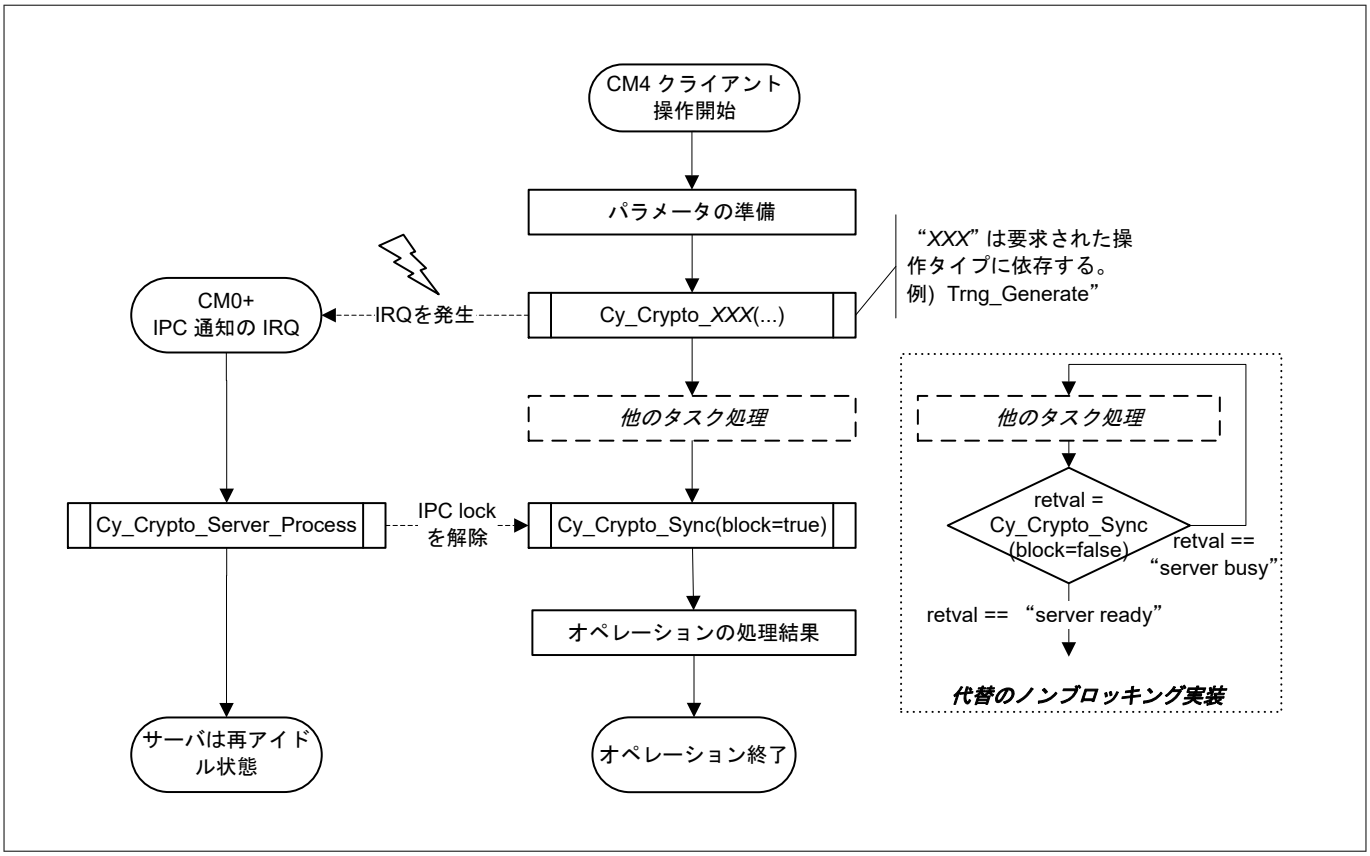

図 3　　一般的なドライバの使用方法のフローチャート

4 巡回冗長検査 (CRC)

4.1 ユースケース

メモリ領域の CRC 結果 (例えば、フラッシュのパラメータ) を格納できます。ソフトウェアは、環境の影響やハードウェアの欠陥などの原因によって領域の内容が壊れていないことを保証します。

注: CRC は暗号操作ではないため、このような領域を悪意のある変更から保護するために使用しないでください。攻撃者は変更された領域の CRC 結果を簡単に計算できます。

4.2 ドライバ関数

Crypto ドライバは、次の CRC 関連の関数を提供します。

4.2.1 Cy_Crypto_Crc_Init

この関数は、基本的な CRC 関連の設定を初期化し、後で複数の CRC 計算で再利用できます。広く使用されている CRC32 や CRC16-CCITT などの多くの CRC 標準は、使用されている多項式（および長さ）と入力/出力設定が異なります。これらの設定は、`Cy_Crypto_Crc_Init` を呼び出すことによって定義されます。

4.2.2 Cy_Crypto_Crc_Run

実際の CRC 計算は、この関数を呼び出すことによって要求されます。計算, 開始アドレス, およびサイズの初期値が関数に渡され、CRC 結果が返されます。

CRC 計算の初期値は、使用する CRC 標準によって定義され、新しい CRC 計算の開始に適用されます。CRC が複数の非連続メモリ領域にわたって計算される場合、`Cy_Crypto_Crc_Run` は複数回呼び出されなければならず (現在の操作が終了した後)、前のブロックの結果を次のブロックの初期値として使用する必要があります。

4.3 フローチャート

図 4 に、Crypto ドライバを使用して CRC 値を計算する一般的なフローを示します。

4 巡回冗長検査 (CRC)

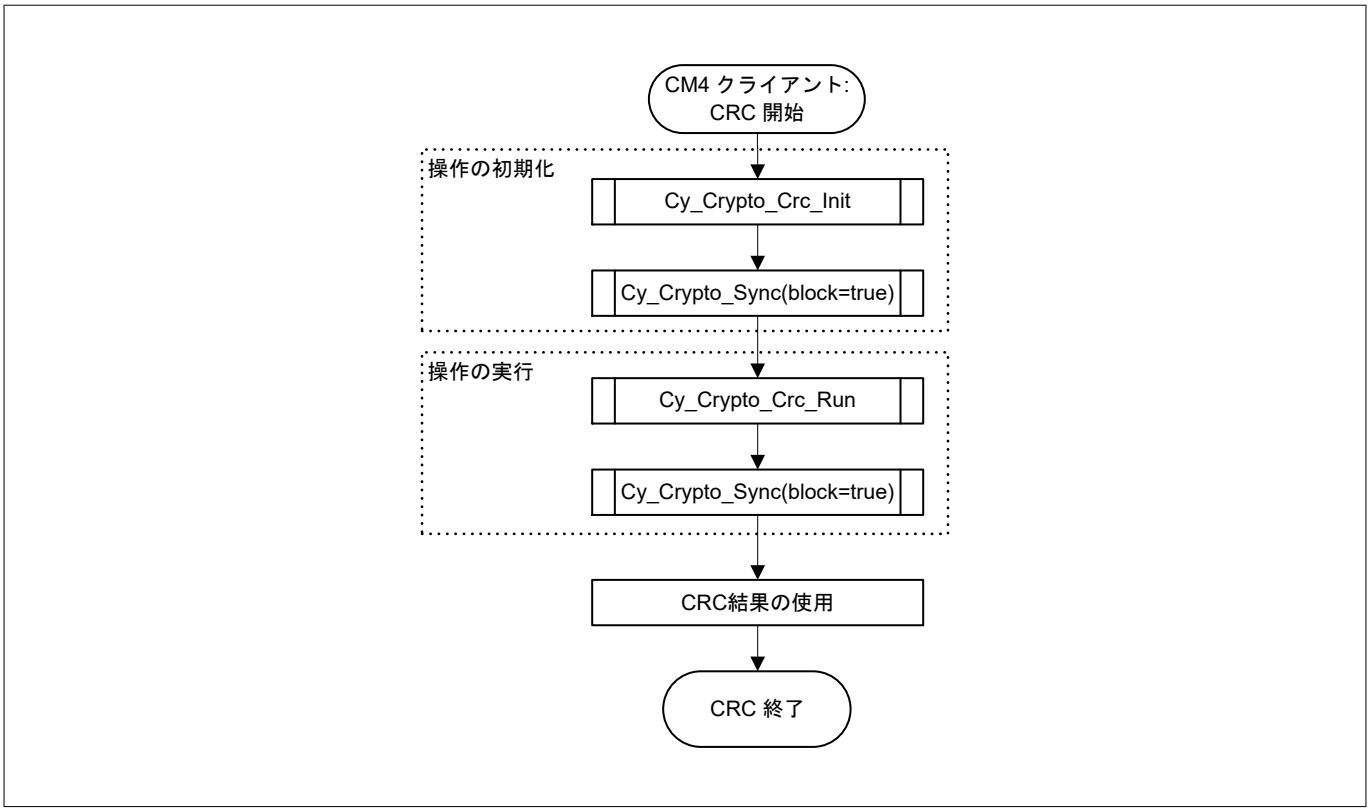

図 4　　CRC 計算のフローチャート

***注**:　　CRC 演算の繰り返し使用のために、関連パラメータ(例えば多項式) の 1 つが変化しない限り、`Cy_Crypto_Crc_Init` を再度呼び出す必要はありません。*

5 擬似乱数生成器 (PRNG)

5.1 ユースケース

擬似乱数生成器を使用して、以下の生成が可能です。

- 通信セッションの対称キー
- "暗号ソルト (cryptographic salt)"、辞書内のすべての単語に対してあらかじめ計算されたハッシュ値の巨大なデータベースを使用する辞書攻撃を防ぐためのハッシュ関数用の追加のランダム入力用データ
- チャレンジレスポンス認証プロトコルにおける"チャレンジ"値

***注**: PRNG シード値は、強力なセキュリティ特性を提供するために、エントロピーが高い真性のランダム値で初期化する必要があります。その目的で TRNG を使用できます。非ランダムまたは一定のシード値は、ソフトウェアの開発やテストなどの一時的な状況で再現可能な PRNG 出力などの決定的な動作が必要な場合にのみ使用してください。*

5.2 ドライバ関数

Crypto ドライバは、PRNG に関連する次の関数を提供します。

5.2.1 Cy_Crypto_Prng_Init

この関数は、PRNG を構成する 3 つの線形フィードバックシフトレジスタを初期化するためにシード値を決定します。

5.2.2 Cy_Crypto_Prng_Generate

この関数を呼び出すことによって疑似乱数が生成されます。擬似乱数の上限は、関数が 0 から指定された制限の間の数値を返すように指定できます。上限は 2^{32}-1 です。

***注**: `Cy_Crypto_Prng_Init` は、後でシード値を変更する場合にのみ呼び出す必要があります。複数の乱数が必要な場合は、`Cy_Crypto_Prng_Generate` を呼び出すだけで十分です。*

5.3 フローチャート

図 5 に、Crypto ドライバを使用して疑似乱数を生成する一般的なフローを示します。

5 擬似乱数生成器 (PRNG)

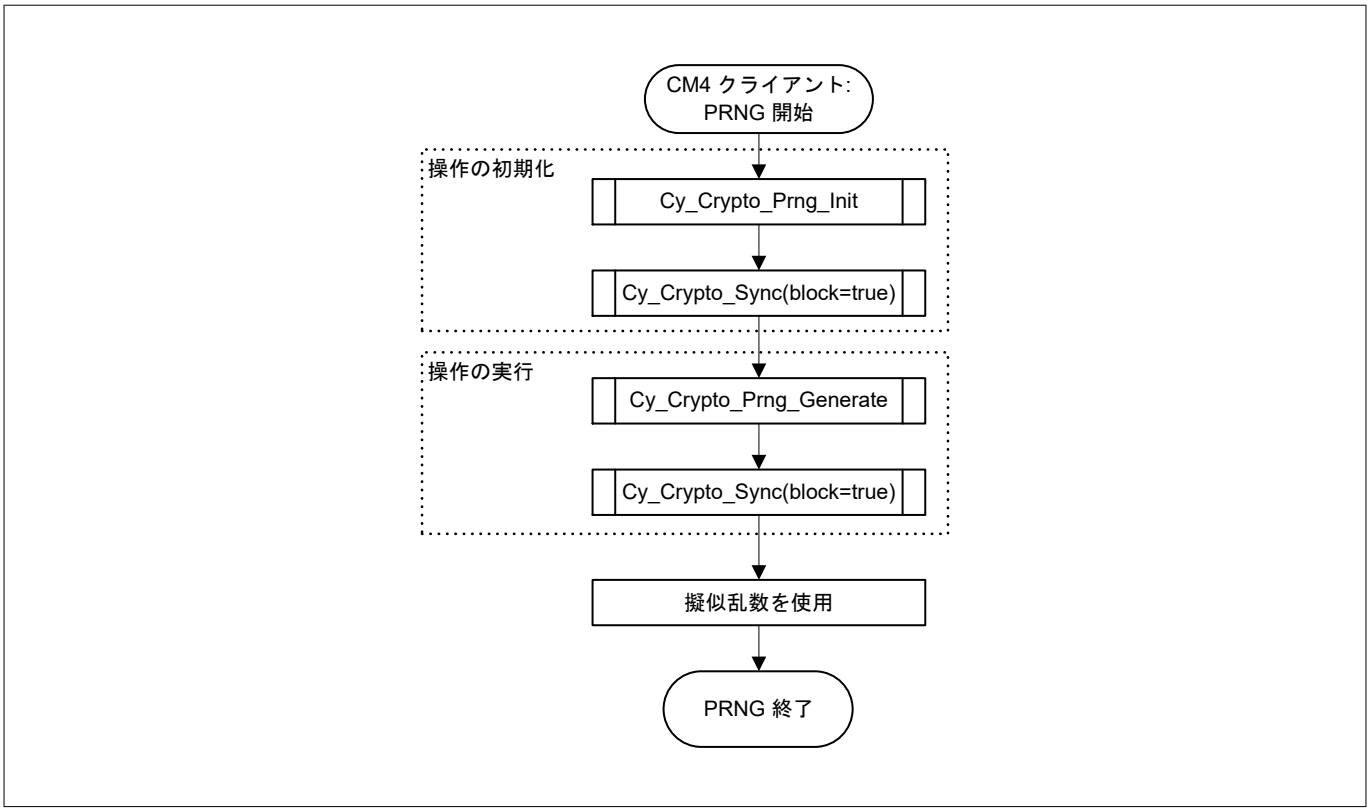

図 5　　疑似乱数生成のフローチャート

***注**:　　`Cy_Crypto_Prng_Init` は、シード値が変更された場合にのみ呼び出される必要があります。PRNG 値の繰り返し生成のためにこの関数を呼び出す必要はありません。*

6 真性乱数生成器 (TRNG)

6.1 ユースケース

TRNG は通常、PRNG のエントロピーが高い真性のランダムシード値を生成するためにのみ使用されます。これは消費電流が比較的高く、乱数ビットがかなりゆっくりと生成されるためです。

6.2 ドライバ関数

Crypto ドライバは、次の TRNG 関連の関数を提供します。

6.2.1 Cy_Crypto_Trng_Generate

この関数は、指定されたビット長の真性乱数を生成します。この関数を呼び出すときには、物理的な構成 (リングオシレータ) に関連するいくつかのパラメータを記述する必要があります。

6.3 フローチャート

図 6 に、真性乱数を生成するために Crypto ドライバを使用する一般的なフローを示します。

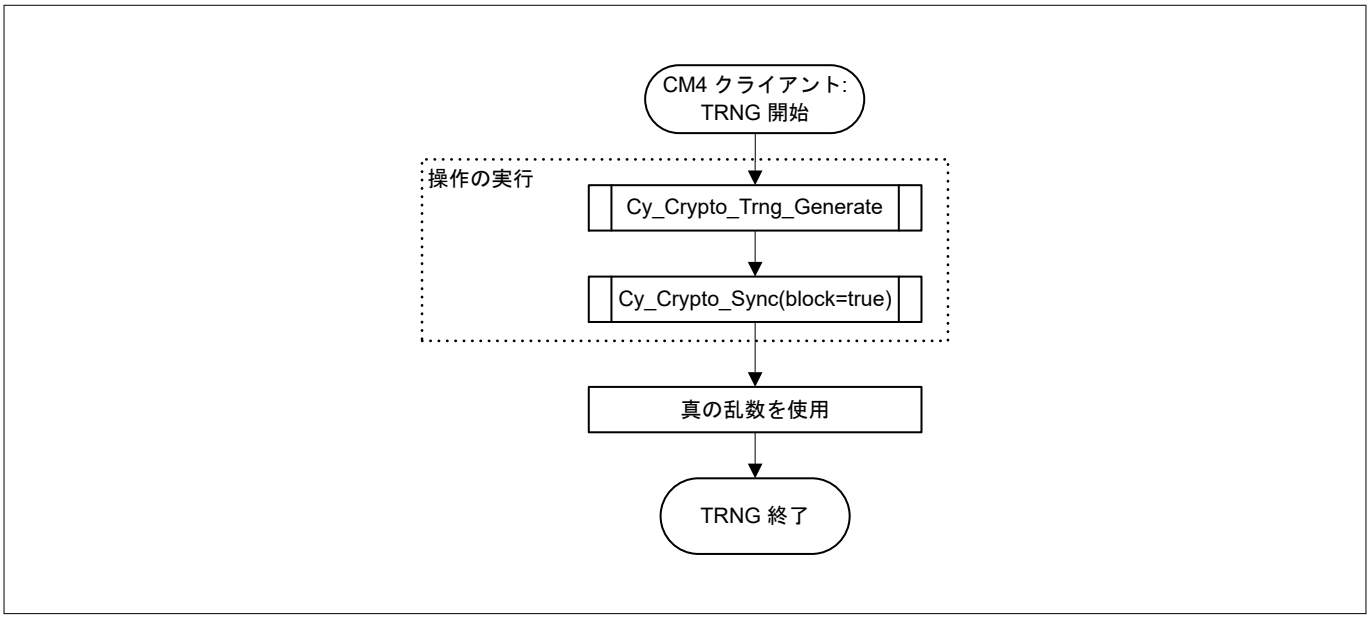

図 6 真性乱数生成のフローチャート

7 Advanced Encryption Standard (AES) を使用した対称鍵暗号化

Crypto ブロックはいくつかの対称鍵暗号化規格をサポートし、Advanced Encryption Standard（AES）は最も広く普及しています。

AES 動作自体は、128 ビットの単一のデータブロックに基づいています。複数の関連データブロックが暗号化されている場合、NIST（National Institute of Standard and Technology）Special Publication 800-38A などの科学文献に記載されているブロックチェーンモードの 1 つを使用することを推奨します。それぞれのブロックを独立して暗号化することは、通常、不十分なセキュリティレベルを提供します。たとえば、同じプレーンテキストのブロックも同じ暗号テキストを持つ場合があります。

Crypto ドライバは、次の AES 動作モードをサポートしています。

- ECB (Electronic Code Book Mode)
- このモードでは、連鎖することなく独立してブロックを暗号化/復号化するため、関連するデータの複数のブロックに低レベルのセキュリティが提供されます。
- CBC (Cipher Block Chaining Mode)
- 1 つのブロックの暗号文は、次のブロックのプレーンテキストと結合されます。最初のブロックに先行するブロックがないので、いわゆる初期化ベクトル(IV)が必要であり、それが最初のブロックのプレーンテキストと組み合わされるためです。
- CFB (Cipher Feedback Mode)
- このモードは、任意の長さのプレーンテキストのストリーム暗号として使用できます。プレーンテキストは AES 暗号化操作の出力と XOR されます。暗号文は、後続のブロックの暗号化入力として使用されます。最初のブロックには、CBC モードと同様の初期化ベクトルが必要です。
- CTR (Counter Mode)
- カウンタ動作モードは、ストリーム暗号としても使用されます。ここでも、プレーンテキストは AES 暗号化操作の出力と XOR されます。AES 暗号化への入力は、いわゆるノンスとカウンタの組み合わせ、または連結です。カウンタはブロックごとに変化し、最も単純なケースでは、それは単に増分されているだけです。
- 暗号文脈におけるノンスは、通常、暗号操作のために一度だけ使用される任意の数です。

7.1 ユースケース

AES の性能要件は非対称鍵暗号の場合と比べはるかに低いため、通常、暗号鍵を持つデバイス間の安全な通信に使用されます。対照的に、非対称鍵暗号化の多くは通信プロトコルによって関係するすべての当事者間で鍵を交換することで通信セッションを確立するためだけに使用されます。この鍵は、対称暗号化と復号化に使用されます。

7.2 ドライバ関数

Crypto ドライバは、以下の AES 関連の関数を提供します。AES の初期化は不要です。

7.2.1 Cy_Crypto_Aes_Ecb_Run

この関数は、単一の 128 ビットデータブロックを暗号化または復号化します。ユーザは、方向 (暗号化/復号化), 鍵の場所とサイズ, および送信元と送信先のブロックの場所を指定するよう要求されます。

7.2.2 Cy_Crypto_Aes_Cbc_Run

`Cy_Crypto_Aes_Ecb_Run` に加えて、この関数は初期化ベクトルと暗号化/復号化されるデータの合計サイズを求めます。

7.2.3 Cy_Crypto_Aes_Cfb_Run

この関数は、`Cy_Crypto_Aes_Cbc_Run` と同じ引数をとります。

7.2.4 Cy_Crypto_Aes_Ctr_Run

この関数は、`Cy_Crypto_Aes_Cbc_Run` と同じ引数をとります。初期化ベクトルは、開始カウンタ値と組み合わせたノンスです。ドライバは、新しいブロック暗号化ごとにカウンタ部分を内部的にインクリメントします。

7.3 フローチャート

図 7 に、Cryptoドライバを使用して AES 操作を実行する一般的なフローを示します。

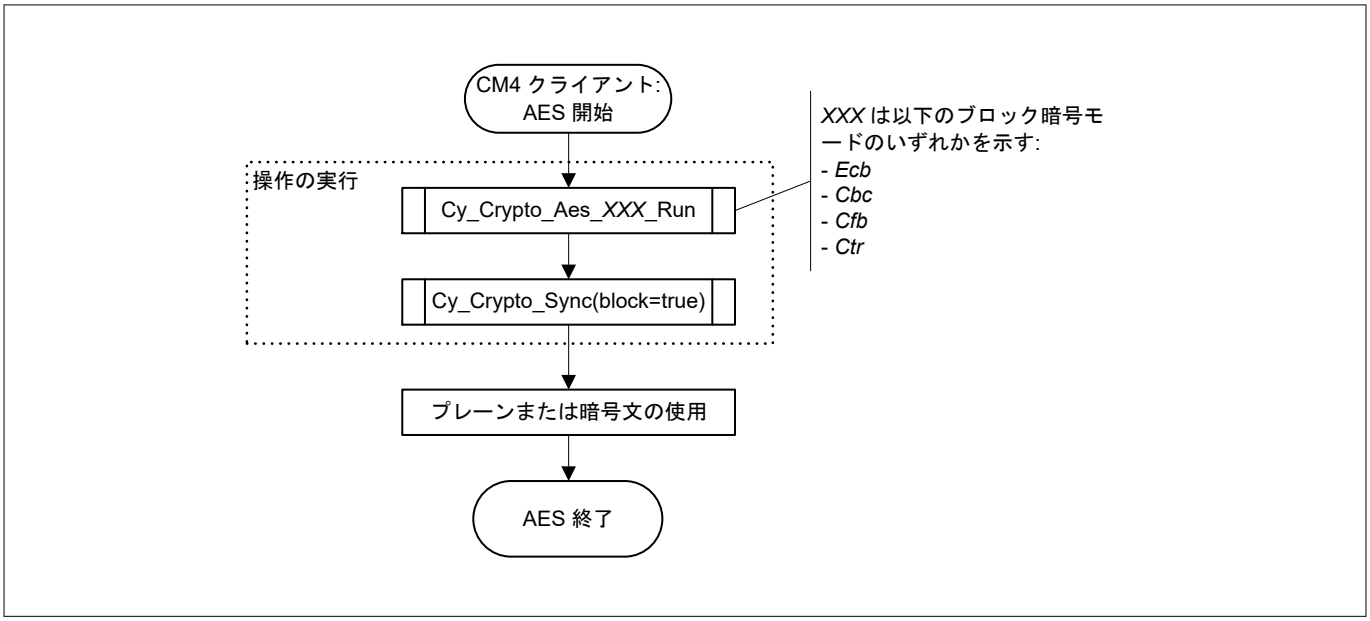

図 7 AES 操作のフローチャート

7.4 他の対称鍵アルゴリズム

一般的な AES アルゴリズムに加えて、Cryptoドライバとハードウェアは次のアルゴリズムもサポートしています。

- データ暗号化標準（DES）
- `Cy_Crypto_Des_Run` 関数は、Cryptoドライバによって提供されます。
- TDES (トリプル DES, 3DES とも略されます)
- 対応するドライバ関数は `Cy_Crypto_Tdes_Run` です。
- ChaCha：ドライバ関数 `Cy_Crypto_Chacha_Run` によってサポートされます。

8 SHA 暗号ハッシュ関数ファミリ

SHA は Secure Hash Algorithm の略で、標準化された暗号ハッシュ関数です。Crypto ドライバは、SHA-1, SHA-256, SHA-512, SHA-512/256, SHA3-256, SHA3-512, SHAKE256 などの標準化されたすべてのバリエーションをサポートしています。

8.1 ユースケース

SHA は、実際のプレーンテキストパスワードを直接格納する代わりに、ユーザパスワード(理想的には「暗号ソルト値」と組合せ)からハッシュ値を導き出し、このハッシュ値(ソルトと共に)をデータベースに格納するためによく使用されます。セキュリティ違反が発生した場合、データベースは他のシステムへのログインに使用されるユーザパスワードを明らかにしません。

SHA は、他の暗号化ブロックと組み合わせて使用して、より高いレベルのセキュリティ目標を達成します。例えば、ハッシュベースのメッセージ認証コード(ハッシュベースおよび暗号ベースのメッセージ認証コード (HMAC, CMAC)を参照)またはデジタル署名(RSA と SHA を使用したデジタル署名検証を参照)です。

8.2 ドライバ関数

Crypto ドライバは、次の SHA 関連の関数を提供します。SHA の初期化は必要ありません。

8.2.1 Cy_Crypto_Sha_Run

この関数は、選択された SHA モードごとにメッセージダイジェストを生成します。これは、関数を呼び出すときに指定する必要があります。さらに、メッセージの場所, サイズ, およびダイジェストを格納する場所は、関数によって求められます。

8.3 フローチャート

図 8 に、SHA 操作を実行するために Crypto ドライバを使用する一般的なフローを示します。

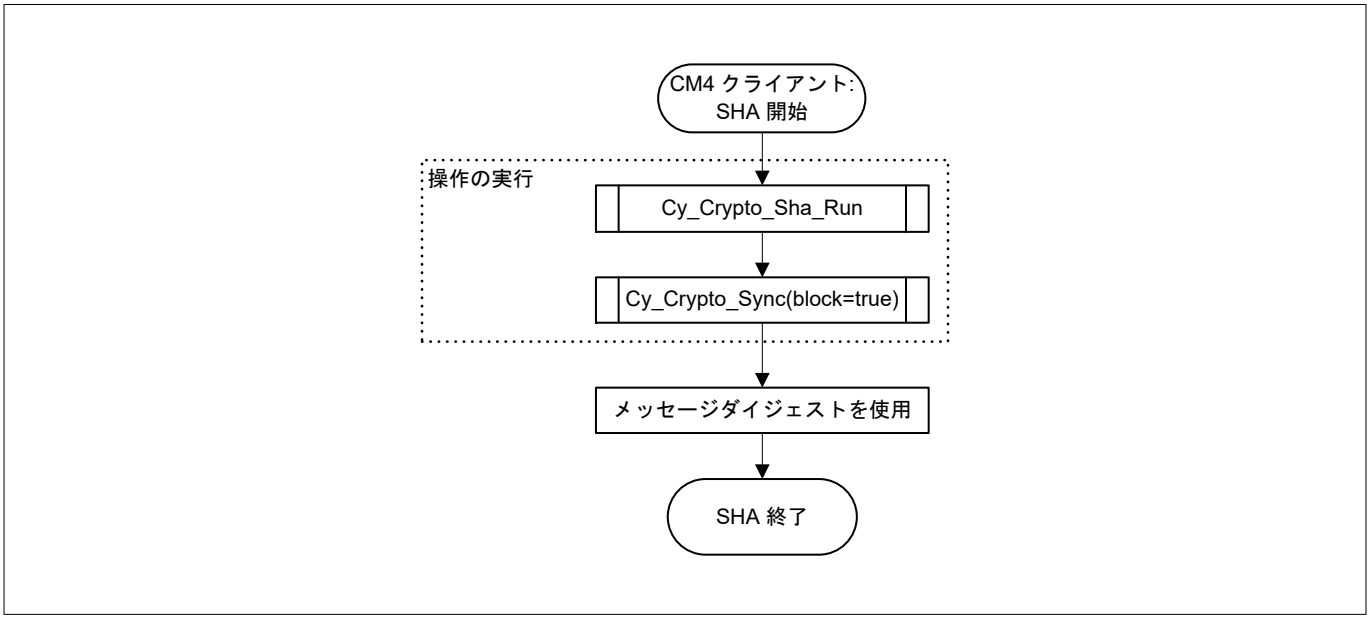

図 8 SHA 操作のフローチャート

9 ハッシュベースおよび暗号ベースのメッセージ認証コード (HMAC, CMAC)

メッセージ認証コード (MAC) は、暗号化ハッシュ関数に似ていますが、セキュリティ要件は異なります。これらのセキュリティ要件は、MAC を生成するための基礎として、暗号ハッシュ関数またはブロック暗号のいずれかとともに秘密暗号鍵を使用することによって達成できます。この操作は、ハッシュベースの MAC (HMAC) または暗号ベースの MAC (CMAC) と呼ばれます。Crypto ドライバは、HMAC 操作には暗号ハードウェアの SHA ブロックを使用し、CMAC 操作には AES ブロックを使用します。

9.1 ユースケース

MAC は、メッセージの受信者がそのメッセージが本物の送信者から発信されたことを知ることが重要であるときに使用されます。一例は、自動車車内のネットワーク中の分散された電子制御ユニットが、すべての重要なメッセージに対して MAC を使用して攻撃者がメッセージをネットワークに注入できないことを保証するケースです。これは、MAC 生成および検証プロセスで使用される送信側と受信側の間の共有秘密鍵によって実現されます。

9.2 ドライバ関数

Crypto ドライバは、次の MAC 関連の関数を提供します。MAC 操作の初期化は必要ありません。

9.2.1 Cy_Crypto_Hmac_Run

この関数を呼び出す場合、目的の SHA モード, キーの格納先とサイズ, メッセージの場所とサイズ, 計算された MAC が格納される場所を渡す必要があります。

9.2.2 Cy_Crypto_Cmac_Run

この関数は、`Cy_Crypto_Hmac_Run` と同様のパラメータを求めますが、SHA モードを使用します。

9.3 フローチャート

図 9 に、Crypto ドライバを使用して MAC 操作を実行する一般的なフローを示します。

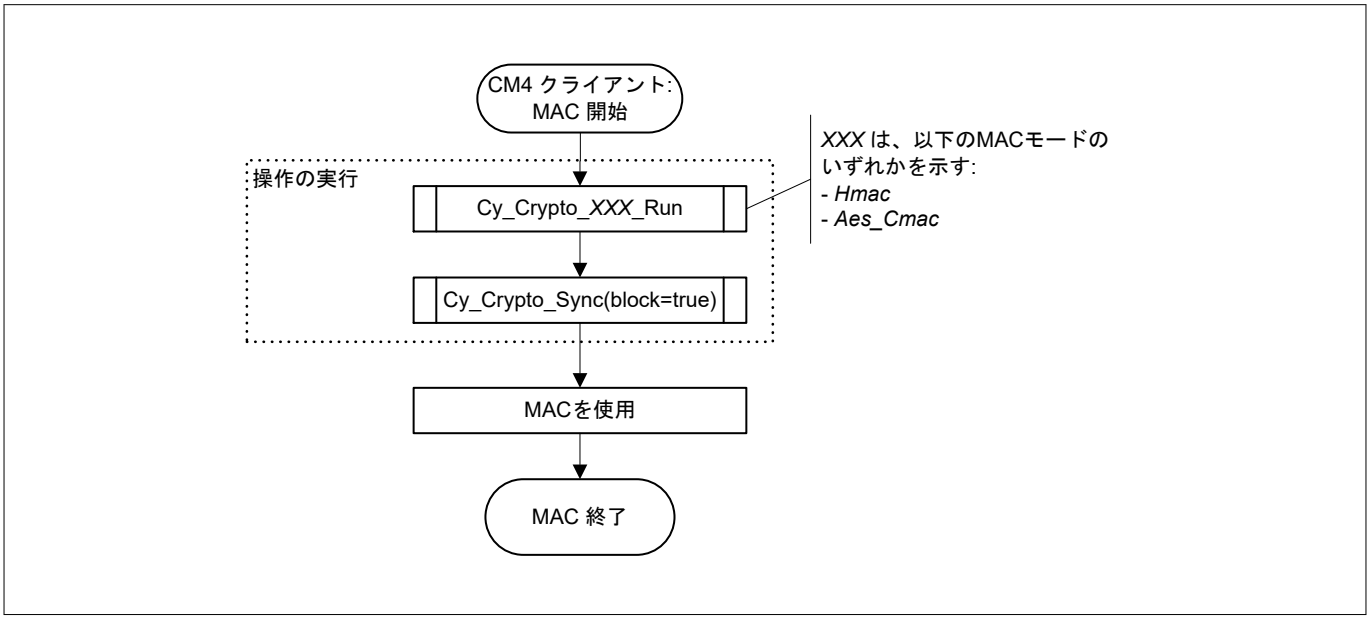

図 9 MAC 操作のフローチャート

10 RSA を使用した非対称鍵暗号化

非対称鍵暗号化 (公開鍵暗号としても知られている) のための様々なアルゴリズムが過去数十年に提案され、一般に使用されています。最も一般的なものの 1 つは、その発明者 Rivest, Shamir, Adleman によって命名された RSA です。Crypto ドライバは RSA アルゴリズムをサポートしています。

RSA と SHA に基づいてデジタル署名を検証するには、RSA と SHA に関連する Crypto ドライバ API を組み合わせて使用する必要があります。RSA と SHA を使用したデジタル署名検証を参照してください。

10.1 ユースケース

非対称暗号化を使用する暗号化と復号化は、通常、対称暗号化または復号化よりもはるかに低速です。これは、関係するすべての当事者間で共通鍵を交換するための通信チャネルの確立中にのみ使用されることが多い理由です。

10.2 ドライバ関数

Crypto ドライバは、RSA 暗号化および復号化に関連する次の関数を提供します。

10.2.1 Cy_Crypto_Rsa_InvertEndianness

プレーンテキスト, 暗号文, モジュラス, パブリックおよびプライベート指数などの RSA 操作に必要なマルチバイトパラメータは、通常ビッグエンディアン形式で定義されます。Crypto ドライバは内部においてリトルエンディアン形式で動作するため、すべての入力パラメータをリトルエンディアン形式で使用可能にする必要があります。この変換は、実行時に `Cy_Crypto_Rsa_InvertEndianness` を呼び出してソフトウェアで行えます。Crypto ドライバは、リトルエンディアン形式の `Cy_Crypto_Rsa_Proc` の出力 (プレーンテキストまたは暗号テキスト) を格納します。この関数を使用して、必要に応じて出力をビッグエンディアン形式に変換できます。

10.2.2 Cy_Crypto_Rsa_CalcCoefs

この関数は、RSA 処理を高速化する RSA 計算で使用される特定の係数とパラメータを事前に計算できます。計算係数は、RSA 関連の暗号化ドライバコンテキストに添付され、モジュラスなどの依存パラメータが変更されない限り、複数の RSA オペレーションにわたって再利用できます。

RSA 操作の前処理中に関数が呼び出されない場合、`Cy_Crypto_Rsa_Proc` は呼び出されるごとにこれらパラメータを内部的に計算します。

10.2.3 Cy_Crypto_Rsa_Proc

これは、RSA を使用してプレーンテキストを暗号化することや、暗号テキストを復号する主な関数です。

関数インタフェースは汎用です。入力メッセージの配置とそのサイズ, 出力メッセージの位置, モジュラス, キー (すなわち指数), およびオプションで `Cy_Crypto_Rsa_CalcCoefs` によって計算された係数の位置を要求します。

`Cy_Crypto_Rsa_Proc` 関数は、提供された入力メッセージをモジュロと指数で処理し、その出力を出力メッセージの場所に格納します。提供される入力メッセージのタイプ (プレーンテキストまたは暗号テキスト) と提供される指数の種類 (公開または非公開) によって異なります。この関数は、暗号化の有無を問わず、内部的には操作を区別しません。

10.3 フローチャート

図 10 に、RSA の暗号化および復号化操作に Crypto ドライバを使用する一般的なフローを示します。

10 RSA を使用した非対称鍵暗号化

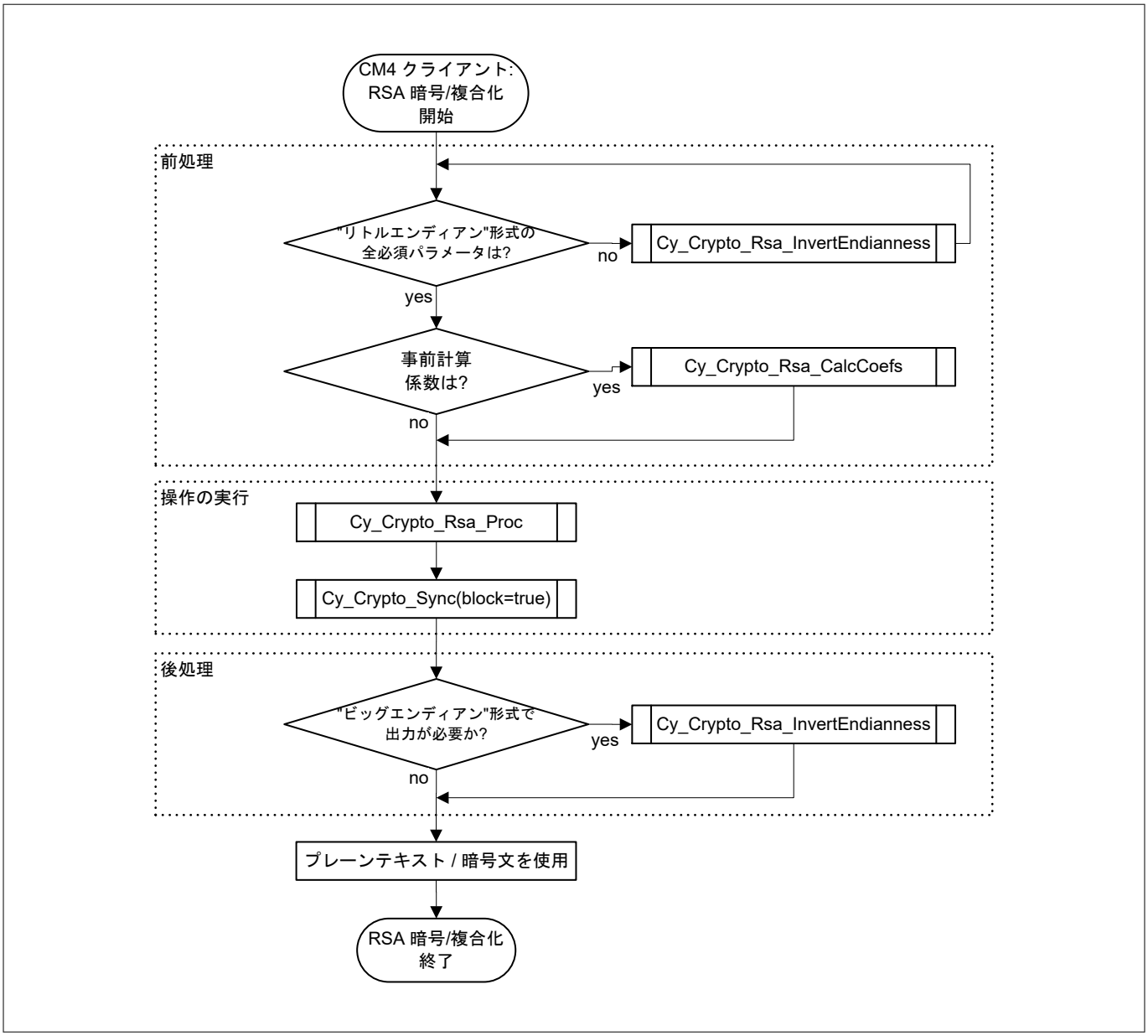

図 10　　RSA 暗号化と復号化操作のフローチャート

11 RSA と SHA を使用したデジタル署名検証

デジタル署名は広く使用されており、しばしば RSA と暗号強度を備えたハッシュ関数に基づいています。

公開鍵暗号標準 (PKCS) は、非対称鍵暗号のさまざまな側面を標準化しようとする仕様を集めたものです。これらの側面の 1 つは、デジタル署名の生成および検証方法を定義することです。Crypto ドライバは RSASSA-PKCS1-v1_5 に基づくデジタル署名の検証のみをサポートし、SHA 暗号化ハッシュ関数を使用します。より詳細な情報については、次のドキュメントを参照してください。http://www.emc.com/collateral/white-papers/h11300-pkcs-1v2-2-rsa-cryptography-standard-wp.pdf

11.1 ユースケース

デジタル署名のアプリケーションの 1 つは、インストールまたは実行前のファームウェアイメージの検証です。例えば、ファームウェア更新プロセスは、安全でない通信チャネルを介してファームウェアイメージおよび関連するデジタル署名を受信します。更新プロセスは、生産中に製造業者によってデバイスにプログラムされた公開鍵を使用してデジタル署名を検証することにより、ファームウェアイメージの信頼性および起点を証明できます。

11.2 ドライバ関数

Crypto ドライバは、RSA および SHA を使用してデジタル署名を検証することに関連する次の関数を提供します。

11.2.1 Cy_Crypto_Rsa_InvertEndianness

Cy_Crypto_Rsa_InvertEndianness を参照してください。

11.2.2 Cy_Crypto_Rsa_CalcCoefs

Cy_Crypto_Rsa_CalcCoefs を参照してください。

11.2.3 Cy_Crypto_Rsa_Proc

Cy_Crypto_Rsa_Proc を参照してください。

デジタル署名を検証するには、この関数と署名者の公開鍵を使用して署名を復号化する必要があります。

11.2.4 Cy_Crypto_Sha_Run

Cy_Crypto_Sha_Run を参照してください。

署名されたメッセージのダイジェストを計算するには、デジタル署名の生成に使用されたものと同じ SHA モードを使用する必要があります。

11.2.5 Cy_Crypto_Rsa_Verify

この関数は実際の検証を行います。解読されたデジタル署名を受け取り、サポートされている RSASSA-PKCS1-v1_5 標準に従ってその署名の形式を検証し、提供されたメッセージダイジェストがデジタル署名に含まれているものと一致するかどうかをチェックします。

11.3 フローチャート

図 11 に、RSA および SHA 操作を使用してデジタル署名を検証するための、Crypto ドライバの使用方法の一般的なフローを示します。

11 RSA と SHA を使用したデジタル署名検証

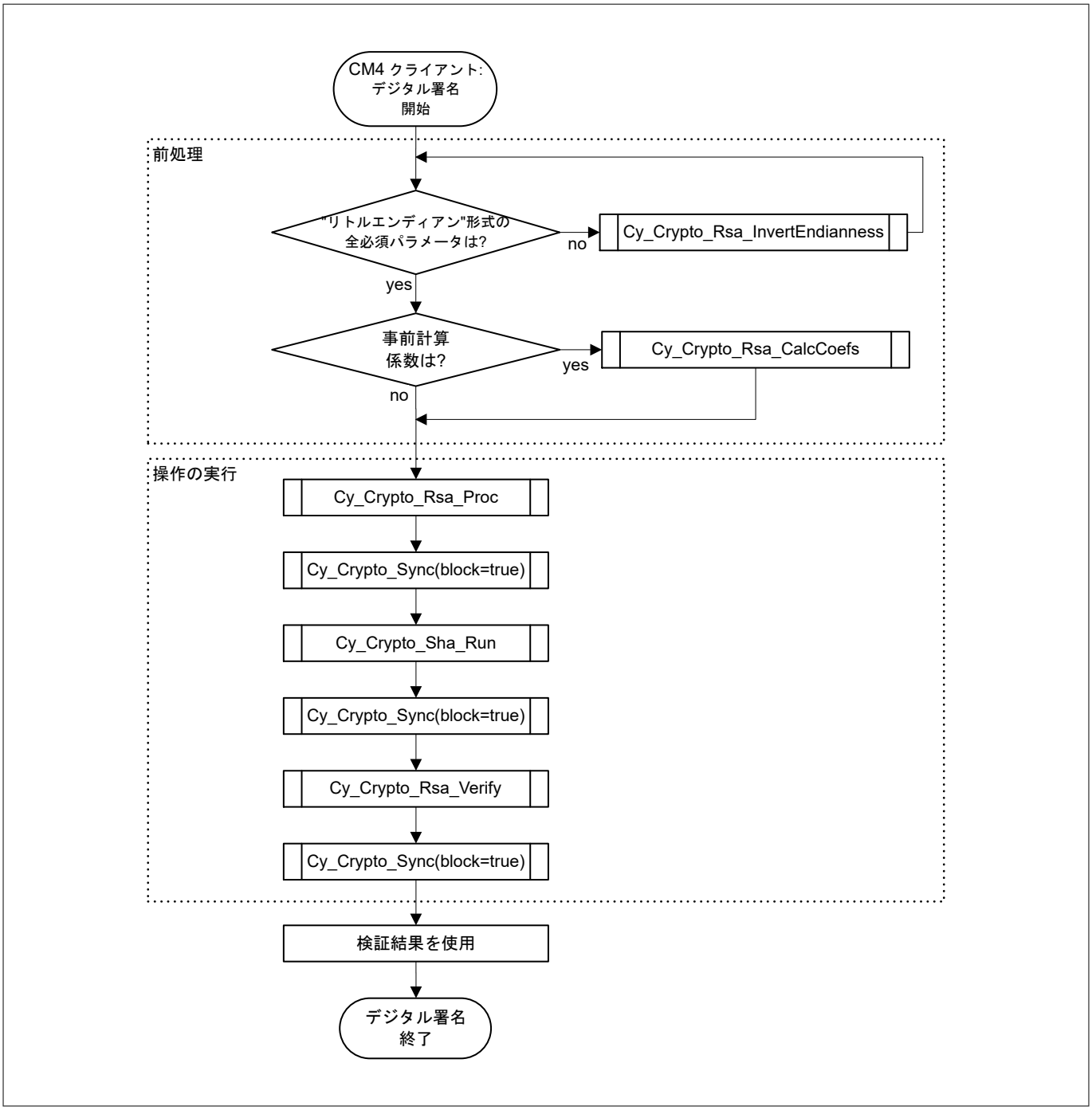

図 11　　RSA および SHA 操作によるデジタル署名の検証のフローチャート

用語集

用語	説明
AES	Advanced Encryption Standard (高度暗号化標準)
CRC	Cyclic Redundancy Check (巡回冗長検査)
PRNG	Pseudo Random Number Generator (擬似乱数生成器)
TRNG	True Random Number Generator (真性乱数生成器)
SHA	Secure Hash Algorithm (セキュア ハッシュ アルゴリズム)
DES	Data Encryption Standard (データ暗号化標準)
TDES	トリプル DES
CHACHA	ストリーム暗号アルゴリズム
ECDSA	Elliptic Curve Digital Signature Algorithm (楕円曲線デジタル署名アルゴリズム)
RSA	非対称鍵暗号 - Rivest, Shamir, および Adleman

関連資料

以下は、TRAVEO™ T2G ファミリシリーズのデータシートとテクニカルリファレンスマニュアルです。これらドキュメントの入手についてはテクニカルサポートに連絡してください。

[1] デバイス データシート

- CYT2B6 datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
- CYT2B7 datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
- CYT2B9 datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
- CYT2BL datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
- CYT3BB/4BB datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family
- CYT4BF datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family
- CYT6BJ datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family (Doc No. 002-33466)
- CYT3DL datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family
- CYT4DN datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family
- CYT4EN datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family (Doc No. 002-30842)
- CYT2CL datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family

[2] ボディコントローラ Entry ファミリ

- TRAVEO™ T2G automotive body controller entry family architecture technical reference manual (TRM)
- TRAVEO™ T2G automotive body controller entry registers technical reference manual (TRM) for CYT2B7
- TRAVEO™ T2G automotive body controller entry registers technical reference manual (TRM) for CYT2B9
- TRAVEO™ T2G automotive body controller high registers technical reference manual (TRM) for CYT2BL (Doc No. 002-29852)

[3] ボディコントローラ High ファミリ

- TRAVEO™ T2G automotive body controller high family architecture technical reference manual (TRM)
- TRAVEO™ T2G automotive body controller high registers technical reference manual (TRM) for CYT3BB/4BB
- TRAVEO™ T2G automotive body controller high registers technical reference manual (TRM) for CYT4BF
- TRAVEO™ T2G automotive body controller high registers technical reference manual (TRM) for CYT6BJ (Doc No. 002-36068)

[4] Cluster 2D ファミリ

- TRAVEO™ T2G automotive cluster 2D architecture technical reference manual (TRM)
- TRAVEO™ T2G automotive cluster 2D registers technical reference manual (TRM) for CYT3DL
- TRAVEO™ T2G automotive cluster 2D registers technical reference manual (TRM) for CYT4DN
- TRAVEO™ T2G automotive cluster 2D registers technical reference manual (TRM) for CYT4EN (Doc No. 002-35181)

[5] Cluster Entry ファミリ

- TRAVEO™ T2G automotive cluster entry family architecture technical reference manual (TRM)
- TRAVEO™ T2G automotive cluster entry registers technical reference manual (TRM) for CYT2CL

その他の関連資料

インフィニオンは、各種周辺機器にアクセスするためのサンプルソフトウェアとして、スタートアップを含むサンプルドライバライブラリ (SDL) を提供しています。SDL は、公式の AUTOSAR 製品でカバーされないドライバについて、お客様へのリファレンスとしても機能します。SDL は自動車用規格に適合していないため、量産用としては使用できません。このアプリケーションノートに記載されているのプログラムコードは、SDL の一部です。SDL を入手する場合は、テクニカルサポートに連絡してください。

改訂履歴

版数	発行日	変更内容
**	2018-07-13	これは英語版 002-20253 Rev. **を翻訳した日本語版 002-24018 Rev. **です。
*A	2021-05-17	テンプレートの変更を実施。これは英語版 002-20253 Rev. *A を翻訳した日本語版 Rev. *A です。
*B	2024-05-30	これは英語版 002-20253 Rev. *B を翻訳した日本語版 Rev. *B です。英語版の改訂内容: Template update; no content update
*C	2025-05-13	これは英語版 002-20253 Rev. *C を翻訳した日本語版 Rev. *C です。英語版の改訂内容: Added 用語集, 関連資料, and その他の関連資料 sections.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2025-05-13

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-hxg1682579047664

重要事項

本手引書に記載された情報は、本製品の使用に関する手引きとして提供されるものであり、いかなる場合も、本製品における特定の機能性能や品質について保証するものではありません。本製品の使用の前に、当該手引書の受領者は実際の使用環境の下であらゆる本製品の機能及びその他本手引書に記された一切の技術的情報について確認する義務が有ります。インフィニオンテクノロジーズはここに当該手引書内で記される情報につき、第三者の知的所有権の不侵害の保証を含むがこれに限らず、あらゆる種類の一切の保証および責任を否定いたします。

本文書に含まれるデータは、技術的訓練を受けた従業員のみを対象としています。本製品の対象用途への適合性、およびこれら用途に関連して本文書に記載された製品情報の完全性についての評価は、お客様の技術部門の責任にて実施してください。

警告事項

技術的要件に伴い、製品には危険物質が含まれる可能性があります。当該種別の詳細については、インフィニオンの最寄りの営業所までお問い合わせください。

インフィニオンの正式代表者が署名した書面を通じ、インフィニオンによる明示の承認が存在する場合を除き、インフィニオンの製品は、当該製品の障害またはその使用に関する一切の結果が、合理的に人的傷害を招く恐れのある一切の用途に使用することはできないこと予めご了承ください。