

RXファミリ

セキュアなユーザ鍵の注入と更新

要旨

暗号技術は、セキュリティソリューションの重要な側面である真正性、機密性、完全性のためのソリューションを実装するためのツールを提供するため、重要です。近年の暗号システムでは、システムのセキュリティは、使用されるアルゴリズムの機密性ではなく、鍵の機密性に依存します。

ルネサスの MCU セキュリティは、統合セキュリティエンジンを中心に展開しています。RX MCU にはさまざまなタイプのセキュリティエンジンがあります。ユーザは、それぞれの MCU で使用されているセキュリティエンジンを、ユーザーズマニュアル ハードウェア編から確認することができます。

セキュリティエンジンは、Protected Mode と Compatibility Mode と呼ばれる 2 つの異なるモードで動作できます。このアプリケーションノートのコンテキストでは、次の通りの運用上の違いを基に説明しています。

- Compatibility Mode では、セキュリティエンジンは平文鍵だけでなく、鍵の機密性を担保する安全な形式の鍵でも注入できます。鍵注入は、RX ファミリ Flexible Software Package(FSP) API を使用して実行する必要があります。
- Protected Mode では、セキュリティエンジンは鍵の機密性を担保する安全な形式の鍵のみを注入できます。鍵注入は、MCU のブートファームウェアのインタフェースを使用して実行する必要があります。

このアプリケーションノートでは、次のセキュアな鍵注入プロセスを示します。

- RSIP-E50D Compatibility Mode AES-128 のセキュアな鍵注入
- RSIP-E50D Protected Mode ECC secp256r1 公開鍵および秘密鍵のセキュアな鍵注入と鍵更新

サンプル鍵は、プロジェクトに付属しています。鍵の再利用は重大なセキュリティリスクを構成するため、**これらの鍵を最終製品で使用しないでください**。このアプリケーションノートでは、カスタム鍵を使用するためにプロジェクトを変更する方法について説明します。

本資料中で使用する用語の説明をいたします。

表 用語説明

用語	説明	詳細が記載されている章
RSIP	Renesas Secure IP ルネサス製 MCU に搭載されているセキュリティエンジン。 RX74M では RSIP-E50D が搭載されている。	1.4
HUK	Hardware Unique Key デバイス内部に存在する、鍵の保護のために使用するデバイス固有の暗号鍵。	1.4
HRK	Hardware Root Key UFPK をラップするための鍵。RSIP 内部と Renesas Key Wrap サーバのみに存在する。	1.4
UFPK	User Factory Programming Key 鍵注入においてユーザ鍵を暗号化するために使用する、ユーザが設定する鍵束。デバイス上では使用しない。	2.2
W-UFPK	Wrapped UFPK UFPK を HRK によりラッピングすることで生成される鍵情報。RSIP 内部にて HRK で UFPK に復号されて使用される。	2.2
KUK	Key Update Key 鍵更新においてユーザ鍵を暗号化するために使用する、ユーザが設定する鍵。 デバイス上で鍵更新を行うにはあらかじめ鍵注入により Wrapped KUK を生成しておく必要がある。	2.2
MRAM	磁気抵抗メモリ 一定以上のプロセスノードにおいてフラッシュに替わり NVRAM として使用される。RX74M にはコード MRAM とデータ MRAM が搭載されている。	5.3
RFP	Renesas Flash Programmer ルネサス製 MCU の NVRAM にプログラムを書き込むためのツール。	5.3
SKMT	Security Key Management Tool セキュアな鍵注入やファームウェアアップデートを支援するツール。	4.2、5.1、5.2、6.1

必要なリソース

対象デバイス

RX74M(RSIP-E50D 搭載)

ソフトウェアと開発ツール

- e² studio IDE v2026-07
- Renesas Flexible Software Package (FSP) v1.0.0
- Renesas Flash Programmer (RFP) v3.24
- Renesas Security Key Management Tool (SKMT) v1.12

FSP および e² studio は、FSP のウェブページ([renesas.com/rx-fsp](https://www.renesas.com/rx-fsp))で入手できるダウンロード可能なプラットフォームインストーラにバンドルされています。RFP は、<https://www.renesas.com/software-tool/renesas-flash-programmer-programming-gui>からダウンロードできます。無償評価版は、本アプリケーションノートで必要とされる機能に使用できます。SKMT は、<https://www.renesas.com/software-tool/security-key-management-tool>からダウンロードできます。

ハードウェア

- EK-RX74M、RX74M グループ用評価キット(<https://www.renesas.com/ek-rx74m>)
- Windows® 11 を実行可能な PC
- USB Type-C®ケーブル(USB Type-C®オス - USB Type-C®オス)1 本

参考情報

本アプリケーションノートは、Renesas の IDE である e² studio を使用した開発モデルについて記載しています。さらに、その開発においては、RX ファミリ MCU のセキュリティ機能についての知識が必要となります。必要に応じて、*RX74M グループユーザズマニュアル ハードウェア編のセキュリティ機能*の章や開発ツールのドキュメントを参照してください。

本アプリケーションノートは、ルネサス RX ファミリ MCU によるセキュアな鍵の注入または更新の段階に携わる製品開発者、製品メーカ、製品サポート、またはエンドユーザが読むことを想定して作成されています。

目次

1. ユーザ鍵の保護	6
1.1 概要	6
1.2 ユーザ鍵保護とRoot of Trust	7
1.3 セキュリティエンジンと関連鍵の概要	8
1.4 ルネサスのセキュアな鍵注入の利点	10
1.4.1 鍵暗号化に対するキーラッピングの利点	10
1.4.2 HUKを用いたキーラッピングの利点	11
1.5 Protected ModeとCompatibility Modeの違い	12
2. セキュアな鍵注入と鍵更新	13
2.1 Protected Modeの鍵注入と鍵更新	13
2.1.1 鍵注入	14
2.1.2 鍵更新	16
2.2 Compatibility Modeの鍵注入	18
2.3 セキュアな鍵注入および鍵更新に使用するツール	20
2.3.1 ルネサスKey Wrapサービス	20
2.3.2 Security Key Management Tool (SKMT)	20
2.3.3 Renesas Flash Programmer (RFP)	20
2.3.4 e ² studio / Flexible Software Package (FSP)	20
3. ルネサスKey Wrapサービスの準備	21
3.1 PGP鍵ペアの生成	21
3.2 ルネサスKey Wrapサーバへの登録	25
3.3 ユーザとルネサスのPGP公開鍵交換	26
3.3.1 ユーザPGP公開鍵の登録	26
3.3.2 ルネサスPGP公開鍵の取得	28
3.3.3 ルネサスPGP公開鍵のインポート	29
4. UFPKの生成とW-UFPKの取得	30
4.1 UFPKの生成	30
4.1.1 SKMTのインストール	30
4.1.2 UFPK生成 (GUI)	30
4.1.3 CLIを使用したUFPKの生成	35
4.2 W-UFPKの取得	36
4.2.1 UFPKのPGP暗号化	36
4.2.2 W-UFPKの取得	40
4.2.3 W-UFPKの復号	43
5. RSIP Protected Modeのセキュアな鍵注入・鍵更新とサンプルプロジェクト	45
5.1 概要	45
5.2 SKMT GUIを使用したUFPKとW-UFPKによる鍵のラップ	45
5.2.1 UFPKを使用した初期ECC鍵ペアのラップ	45
5.2.2 UFPKによるKUKのラップ	49
5.2.3 新しいECC鍵ペアをKUKでラップ	52
5.3 SKMT CLIを使用したUFPKとW-UFPKによる鍵のラップ	55

5.3.1	UFPKを使用した初期ECC鍵ペアのラップ	56
5.3.2	UFPKを使用したKUKの生成とラップ	58
5.3.3	新しいECC鍵ペアをKUKでラップ	59
5.4	MCUのブートファームウェアのインタフェースによるセキュアな鍵注入	61
5.4.1	ハードウェアの設定	61
5.4.2	初期ユーザ鍵とKUKの注入	62
5.5	Protected Modeサンプルプロジェクト	65
5.5.1	サンプルプロジェクトの概要	65
5.5.2	サンプルプロジェクトの鍵の注入	66
5.5.2.1	注入鍵のフォーマット	66
5.5.2.2	注入された鍵と更新された鍵の検証	66
5.5.3	サンプルプロジェクトの実行	67
6.	RSIP Compatibility Modeのセキュアな鍵注入とサンプルプロジェクト	68
6.1	概要	68
6.2	SKMT GUIを使用したUFPKとW-UFPKによる鍵のラップ	68
6.3	SKMT CLIを使用したUFPKとW-UFPKによる鍵のラップ	71
6.4	Compatibility Modeサンプルプロジェクト	72
6.4.1	SKMTで生成されたファイルの使用	72
6.4.2	サンプルプロジェクトの実行	72
7.	References	73
8.	ウェブサイトとサポート	74
	改訂記録	75

1. ユーザ鍵の保護

1.1 概要

暗号システムの安全性は、使用する暗号アルゴリズムそのものの秘匿性ではなく、暗号鍵を安全に生成、保管、使用、および更新できるかに大きく依存します。特に組み込み機器では、製造時に注入したユーザ鍵がプログラムメモリや汎用 RAM 上で平文として露出すると、鍵の漏えいやデバイスの複製につながる可能性があります。

RX74M に搭載されている Renesas Secure IP (RSIP-E50D)は、暗号演算をハードウェアで実行するだけでなく、ユーザ鍵をラップ鍵として扱うことで、CPU から直接参照できない形で鍵を保護します。ラップ鍵は MCU 固有の鍵情報に基づいて保護されるため、ある MCU で使用するために生成されたラップ鍵を別の MCU へコピーしても、そのまま使用することはできません。

本アプリケーションノートでは、RX74M の RSIP-E50D を対象に、ユーザ鍵を安全に注入し、必要に応じて更新する方法を説明します。RSIP-E50D の動作モードには Protected Mode と Compatibility Mode があり、鍵注入に使用するインタフェース、扱える鍵の形式、および鍵更新の方法が異なります。両モードの違いについては、1.5章の「Protected ModeとCompatibility Modeの違い」で説明します。

本資料では、Protected Mode における ECC secp256r1 公開鍵および秘密鍵のセキュアな鍵注入と鍵更新、ならびに Compatibility Mode における AES-128 鍵のセキュアな鍵注入を例として取り上げます。

なお、本資料に含まれるサンプル鍵は動作確認用です。鍵の再利用は重大なセキュリティリスクとなるため、最終製品では使用しないでください。

1.2 ユーザ鍵保護と Root of Trust

IoT 機器や組み込み機器では、デバイスの真正性確認、セキュアブート、ファームウェア更新、通信データの保護など、多くのセキュリティ機能が暗号鍵に依存しています。これらの鍵が漏えい、改ざん、または複製されると、システム全体の信頼性が損なわれる可能性があります。

Root of Trust は、システム内で信頼の基点となるハードウェア、ファームウェア、およびソフトウェアの構成要素です。組み込み機器では、デバイス固有の ID、デバイス鍵、セキュアブート用の検証鍵、通信認証用の鍵などが Root of Trust を構成します。したがって、Root of Trust を維持するためには、これらの鍵を安全に注入し、保管し、使用時にも平文で露出させない仕組みが必要です。

セキュアブートは、多くのセキュリティシステムにおいて Root of Trust を構成する代表的な機能の一つです。アプリケーションは、公開鍵暗号方式を使用して認証されます。このとき使用される検証鍵や、デバイス識別に使用される秘密鍵および証明書は、システムの信頼性を支える重要な資産です。

上記のことから、暗号鍵の漏えいは、安全なシステムを危険な状態に陥らせる要因となります。Root of Trust を保護するには、鍵へのアクセスを暗号境界内に制限し、鍵を安全に保管し、複製が困難な状態に保つ必要があります。また、権限のない者による読み取りおよび書き込みアクセスから保護することも重要です。

ルネサスの RX ファミリ MCU に搭載される RSIP は、ユーザ鍵をラップ鍵として扱うことで、平文鍵を CPU アクセス可能な領域に露出させずに使用するための仕組みを提供します。また、セキュアな鍵注入および鍵更新の仕組みにより、製造時のプロビジョニングからフィールドでの鍵更新まで、製品ライフサイクルを通じた鍵保護を支援します。

1.3 セキュリティエンジンと関連鍵の概要

本節では、RX74M に搭載されるセキュリティエンジン RSIP-E50D の概要と、ユーザ鍵の保護、注入、および更新に関連する鍵情報について説明します。RSIP-E50D は、暗号演算機能だけでなく、ユーザ鍵をラップ鍵として扱うための鍵保護機構を提供します。

セキュリティエンジン(RSIP)は、MCU 内の分離されたサブシステムです。セキュリティエンジンには、対称暗号アルゴリズムおよび非対称暗号アルゴリズム、ならびに各種ハッシュ関数およびメッセージ認証コード用のハードウェアアクセラレータが含まれています。また、暗号処理のエントロピーソースを提供する真性乱数生成回路(True Random Number Generator、TRNG)も含まれています。セキュリティエンジンはアクセス管理回路によって保護されており、不正な外部アクセスが試行された場合にセキュリティエンジンをシャットダウンできます。図1-1は、セキュリティエンジンの概念図を示しています。セキュリティエンジンでサポートされている暗号処理については、表1-1を参照してください。

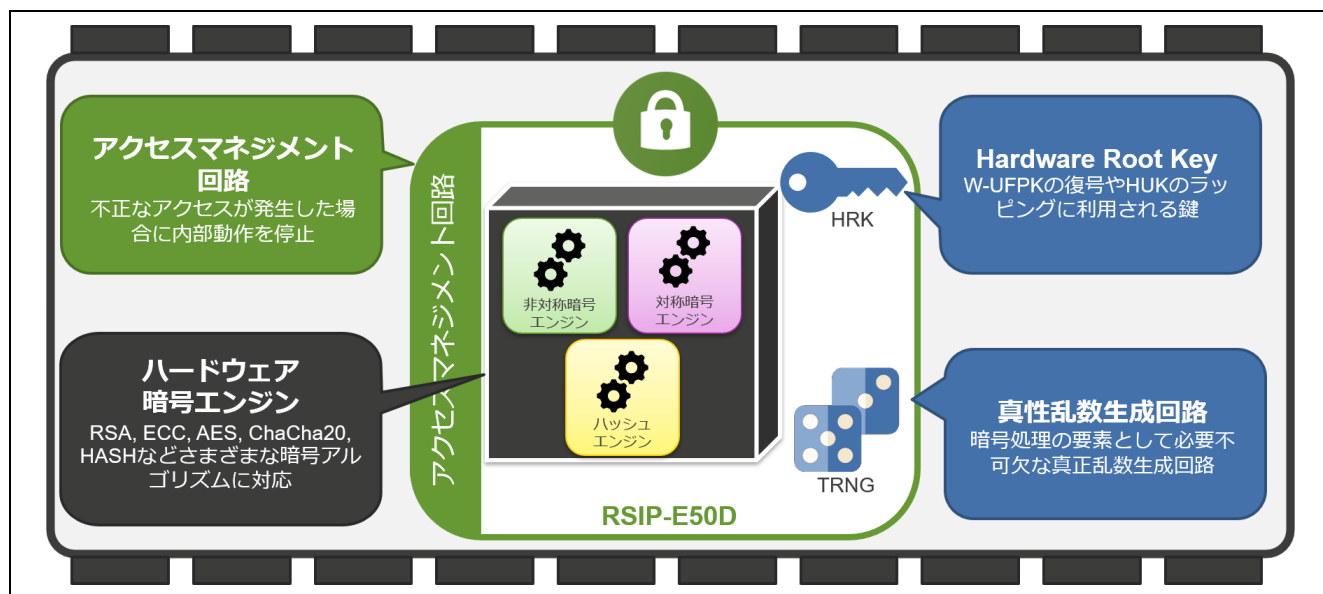


図1-1 セキュリティエンジン RSIP-E50D

HRK は、物理的に格納される単一の鍵ではありません。図1-1では、概念の説明を簡略化するためにそのように表現されています。セキュリティエンジンには、平文鍵などの機密情報を処理する操作の内部 RAM が含まれています。この RAM は、セキュリティエンジンの外部からアクセスできません。

セキュリティエンジンには独自の専用内部 RAM があり、すべての暗号操作をセキュリティエンジン内で物理的に分離できます。これは、高度な鍵処理機能と組み合わせることで、CPU アクセス可能なバス上で平文鍵が一切露出ししないアプリケーションを実装できることを意味します。

ユーザ鍵は、HUK で暗号化され、メッセージ認証コードでタグ付けされたラップ鍵として保存されることにより、安全な形式の鍵として保存および使用できます。ラップ鍵は、ラップした特定 MCU 内のセキュリティエンジンによってのみアンラップできるため、ラッピングメカニズムは、ユーザ鍵のクローン不可能なセキュアストレージを提供します。

RX ファミリは、デバイスを安全にプロビジョニングするためのセキュアな鍵注入メカニズムも提供します。セキュリティエンジンには、上位レベルのソリューションで活用できる暗号化機能が豊富に搭載されており、ハードウェアアクセラレーションを使用して実行時間と消費電力の両方を削減するオプションが提供されます。

表1-1は、セキュリティエンジンとそれに関連する暗号化機能をまとめたものです。

表1-1 Renesas Secure IP およびセキュリティエンジンの暗号化機能

機能			RX74M
搭載された暗号 IP			
	セキュリティエンジン		RSIP-E50D
認証・鍵交換(公開鍵暗号方式)			
	RSA	鍵生成、署名/検証、暗号化/復号	最大 4K
	ECC	鍵生成、署名/検証、暗号化/復号	最大 521 ビット
機密性保護(共通鍵暗号方式)			
	AES	ECB、CBC、CTR	128/192/256 ビット
		XTS	128/256 ビット
		CCM、GCM、CMAC、GMAC	128/192/256 ビット
	ChaCha20	Poly1305	Yes
データの完全性			
		HMAC	SHA-2/SHA-3
		SHA-2	224/256/384/512 ビット
		SHA-3	224/256/384/512 ビット
乱数生成			
	TRNG	128bit	Yes
鍵の処理			
	ラップ鍵	HUK による鍵の保護	Yes
	平文鍵	平文鍵の注入	Yes (Compatibility Mode のみ)

MCU 固有の HUK は、RSIP 用の 256 ビットランダムキーであり、ルネサス工場で注入されます。この鍵は、MCU 固有のキーラッピングメカニズムを使用してラップされた形式で格納されます。

HUK は各 MCU に固有のものであるため、攻撃者が保存されたユーザ鍵を抽出できたとしても、別の MCU はそれを使用できません。

セキュアな鍵注入では、ユーザ鍵を平文のまま MCU に書き込むのではなく、ラップされた形式で MCU に渡し、RSIP 内部で安全にユーザ鍵を生成します。この処理では、UFPK(User Factory Programming Key)、W-UFPK(Wrapped UFPK)などの鍵情報を使用します。UFPK は、ユーザ鍵を注入する際に使用するユーザ指定の鍵です。W-UFPK は、UFPK をルネサスの鍵情報によりラップした鍵情報です。

鍵更新を使用する場合には、KUK(Key Update Key)を使用します。KUK は、フィールド上のデバイスで新しいユーザ鍵を安全に更新するために使用する鍵です。KUK の扱いは RSIP の動作モードに依存します。Protected Mode と Compatibility Mode における鍵注入および鍵更新の違いについては、後述の「Protected Mode と Compatibility Mode の違い」で説明します。

1.4 ルネサスのセキュアな鍵注入の利点

前節で説明した RSIP のラップ鍵機構と、セキュアな鍵注入および鍵更新の仕組みにより、ユーザ鍵を平文のまま保存または転送する場合に比べて、鍵漏えいおよび鍵の複製に対する耐性を高めることができます。

セキュアな鍵注入および更新は、セキュリティエンジンのラップ鍵のサポートと組み合わせることで、平文鍵の使用に関連する多くの脆弱性に対処します。

- 平文鍵はコード MRAM に保存されません。プログラムメモリの侵害が発生した場合でも、秘匿すべき暗号鍵は保護されます。
- 平文鍵は、一般にアクセス可能な RAM に保持されることはありません。システム上で悪意のあるコードが実行された場合でも、秘匿すべき暗号鍵は引き続き保護されます。
- ラップ鍵は、MRAM に安全に保存できるほか、外部メモリにコピーすることもでき、制限なく暗号鍵を格納することが可能です。

さらに、ルネサスのキーラッピング技術は、以下で説明するように、デバイスのクローン作成から保護します。

1.4.1 鍵暗号化に対するキーラッピングの利点

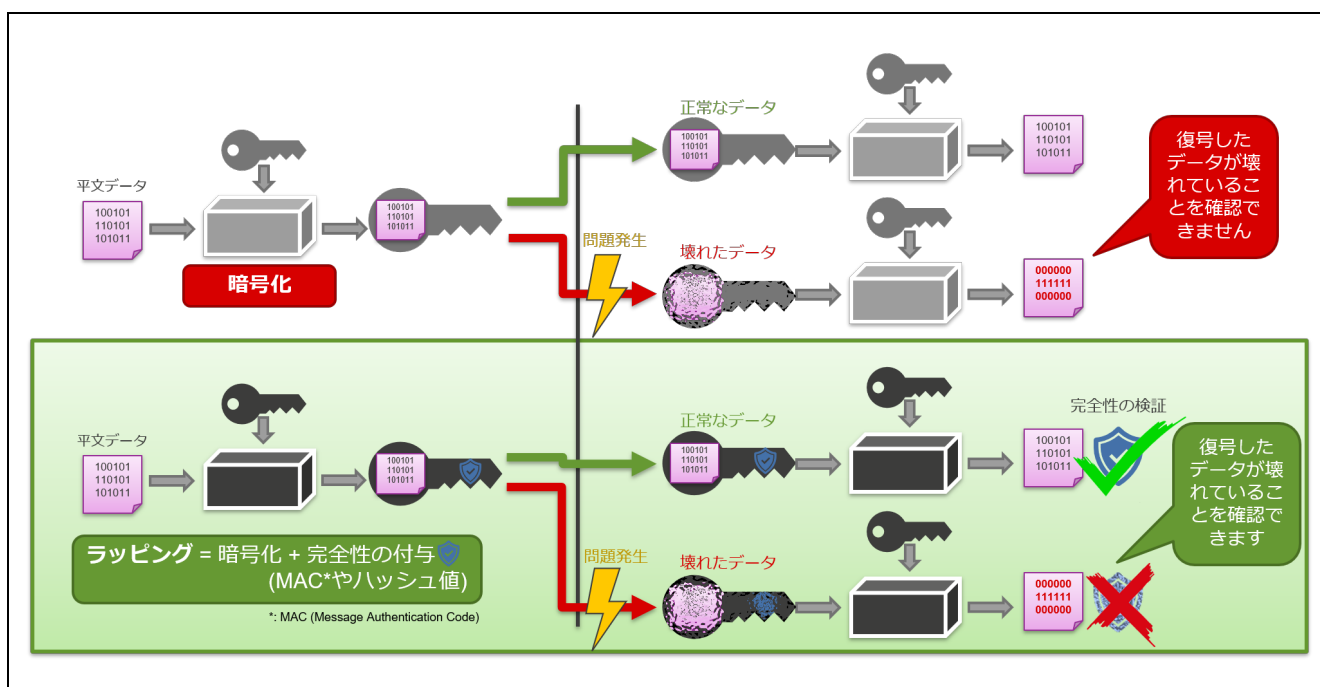


図1-2 データのラッピングと暗号化

セキュアな資産保管のためには、ラッピングと暗号化の違いを理解することが重要です。

データが暗号化されて別の受信者に送信されるとき、その受信者が同じ鍵を持っている場合、その受信者はデータを復号できます。これにより、機密情報の交換が行われます。しかし、暗号化されたデータの送信に問題があった場合はどうなるでしょうか。受信者が無意識のうちに破損した情報を受け取った場合、復号アルゴリズムは、元のデータが破損したことを示すことなく、無意味なデータを生成します。

ラッピングは、整合性チェックのために暗号化された出力にメッセージ認証コードを追加することで、この問題を解決します。

1.4.2 HUK を用いたキーラッピングの利点

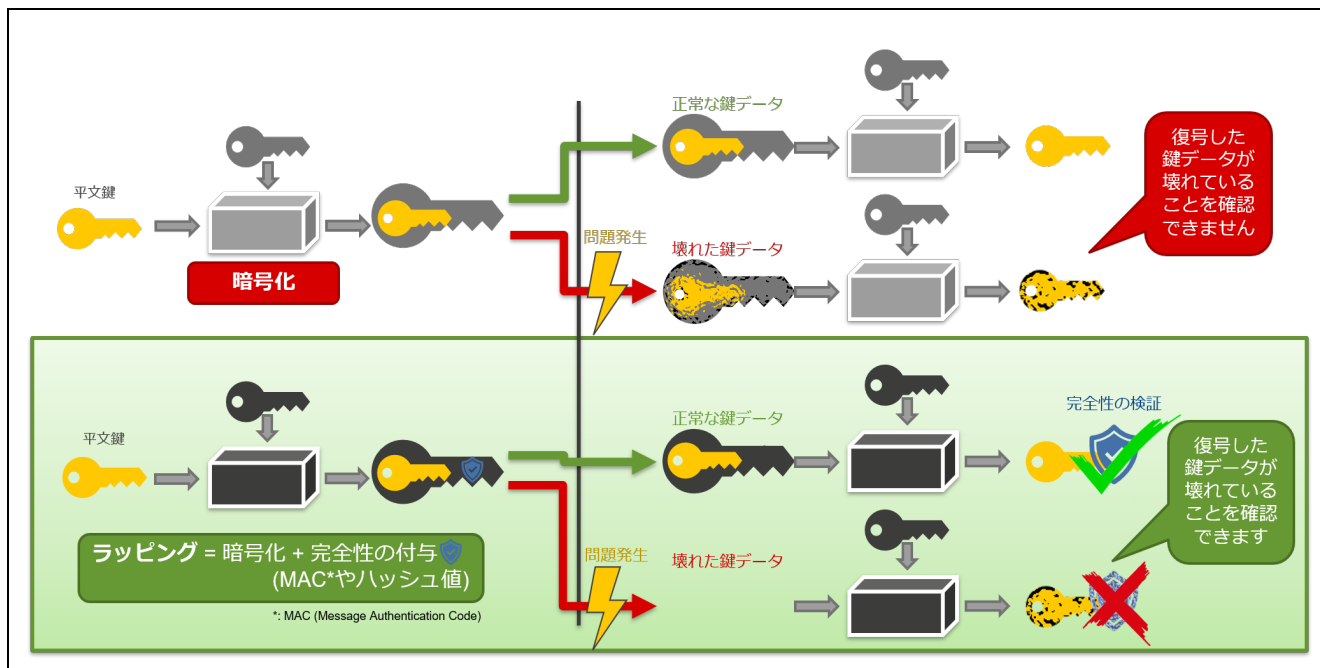


図1-3 キーのラッピングと暗号化

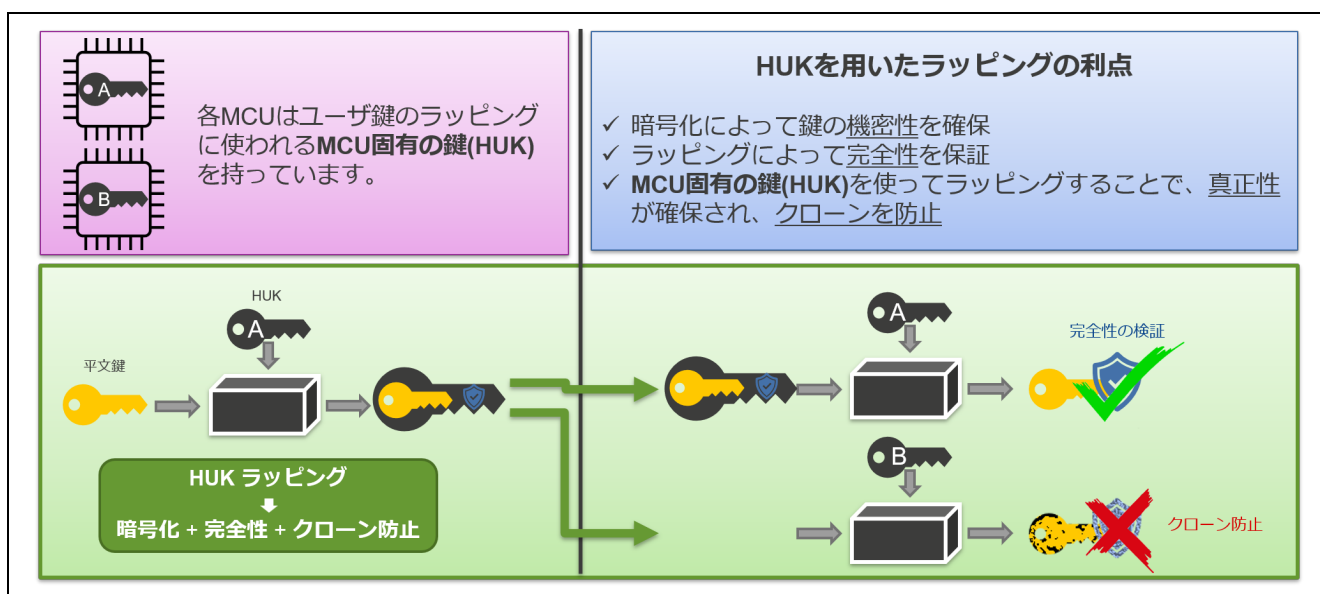


図1-4 HUK を使用したキーラッピング

HUK を使用して保存された鍵をラップすることで、コピー防止という別の保護機能が追加されます。ラップ鍵が別の MCU に送信またはコピーされた場合、その MCU はラップを解除したり、コピーされた鍵を使用したりすることはできません。たとえ MCU の全コンテンツが別のデバイスにコピーされた場合でも、鍵は露出されず、使用することはできません。

RSIP-E50D では、これらの鍵保護機構を使用する際の動作モードとして Protected Mode と Compatibility Mode が用意されています。次節では、両モードにおける鍵注入および鍵更新の違いについて説明します。

1.5 Protected Mode と Compatibility Mode の違い

RSIP-E50D には、Protected Mode と Compatibility Mode の 2 つのモードがあります。両モードでは、鍵注入に使用するインタフェース、扱える鍵の形式、および鍵更新の方法が異なります。使用するモードに応じて、UFPK および W-UFPK の準備、SKMT で生成する鍵データ、MCU への注入方法、ならびにサンプルプロジェクトの構成が異なるため、対象アプリケーションに合わせて適切なモードを選択する必要があります。

Protected Mode では、ユーザ鍵は安全な形式の鍵として注入されます。鍵注入は、MCU のブートファームウェアのインタフェースを使用して実行します。このモードでは、MCU に注入された鍵を RSIP 内部で保護された形式として扱うため、平文鍵を CPU アクセス可能な領域に保持しない構成を実現できます。

また Protected Mode では、KUK を使用して、フィールド上のデバイスに対してユーザ鍵を安全に更新できます。KUK は、製造時のプロビジョニングに、ユーザ鍵と同様に安全な形式の鍵として注入しておく必要があります。フィールド上で新しいユーザ鍵を更新する場合は、あらかじめ注入された KUK を使用して新しいユーザ鍵をラップし、FSP の鍵更新 API を介して安全な形式の鍵として更新します。

プログラミングインタフェースを無効化した後に追加の KUK を注入することはできません。そのため、製品ライフサイクル中に鍵更新を行う可能性がある場合は、製造時またはプロビジョニング時に複数の KUK を注入しておくことを推奨します。これにより、セキュリティポリシーに基づく鍵ローテーション、KUK の切り替え、または鍵漏えい時の対応を行いやすくなります。

Compatibility Mode では、FSP の鍵注入 API を使用して鍵注入を行います。このモードでは、ラップ鍵に加えて平文鍵も扱うことができます。既存ソフトウェアとの互換性や、アプリケーションから FSP の鍵注入 API を使用して鍵注入を行う構成に適しています。

一方で、Compatibility Mode は KUK を使用した鍵更新には対応していません。そのため、Compatibility Mode で鍵更新を行う場合は、Protected Mode とは異なる方法で鍵を取り扱う必要があります。Compatibility Mode での鍵注入および鍵更新の詳細については、FSP のドキュメントを参照してください。

Protected Mode と Compatibility Mode で生成される W-UFPK および鍵注入データには互換性はありません。そのため、ルネサス Key Wrap サービスおよび SKMT を使用する際は、対象 MCU ファミリ、RSIP の動作モード、および使用する鍵の種類を正しく選択してください。

本資料では、Protected Mode の例として ECC secp256r1 公開鍵および秘密鍵のセキュアな鍵注入と、KUK を使用した鍵更新を説明します。また、Compatibility Mode の例として AES-128 鍵のセキュアな鍵注入を説明します。具体的な鍵注入および鍵更新の全体手順は2章で、Protected Mode における鍵注入、鍵更新およびサンプルプロジェクトについては5章を参照してください。Compatibility Mode における鍵注入およびサンプルプロジェクトについては6章を参照してください。

2. セキュアな鍵注入と鍵更新

本章では、RSIP-E50D におけるセキュアな鍵注入および鍵更新の手順について説明します。Protected Mode と Compatibility Mode では、使用するインタフェースや鍵の取り扱い方法が異なります。本章では各モードにおける一般的な手順を説明し、以降の章では具体的な鍵種別を例とした実施手順を説明します。

2.1 Protected Mode の鍵注入と鍵更新

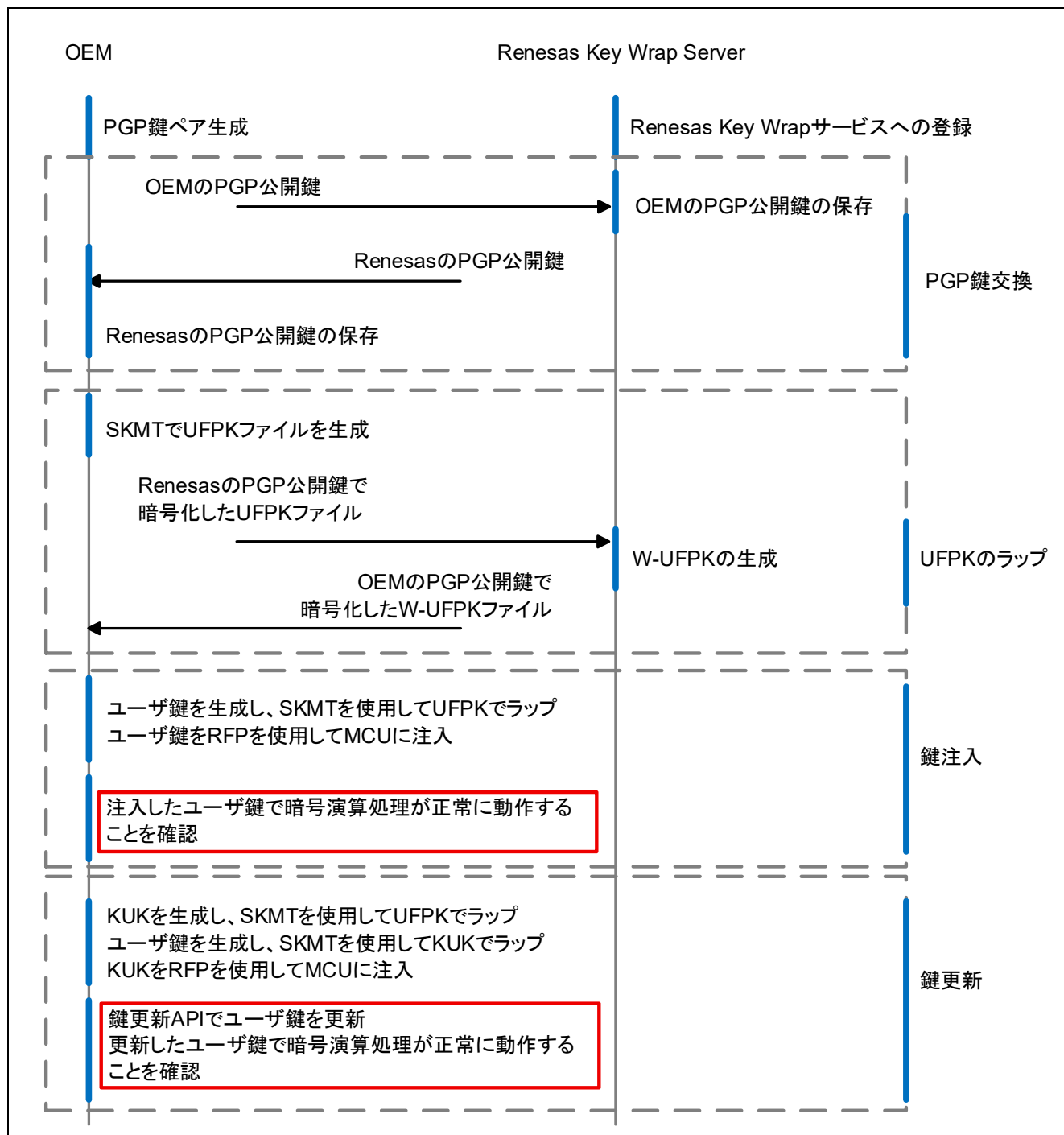


図2-1 Protected Mode における鍵注入および更新の流れ

Protected Mode における鍵注入と鍵更新の流れを図2-1に示します。Protected Mode では、セキュアな鍵注入と鍵更新の両方をサポートしています。KUK を利用することで、フィールド上で鍵更新を行うことができます。詳細な作業の流れは、2.1.1と2.1.2で説明しています。

本アプリケーションノートのサンプルプロジェクトでは、Protected Mode において、サンプルとして用意している鍵を使用して、鍵注入および鍵更新の動作を示します。これは図2-1の赤枠で囲われた部分に相当します。詳細については、3章、4章、および5章を参照してください。

2.1.1 鍵注入

鍵注入には、以下に示す5つの大まかなステップがあります。この中で、暗号鍵が平文で取り扱われる鍵準備の手順は、セキュアな環境で実行する必要があります。

1. ユーザ鍵のラップに使用するための UFPK を生成します。UFPK は SKMT を使用して生成することもできます。詳細については4章を参照してください。
2. HRK(青色)を使用して UFPK(緑色)をラップして W-UFPK を生成します。W-UFPK の生成には、ルネサス Key Wrap サービスを使用します。詳細については3章および4章を参照してください。

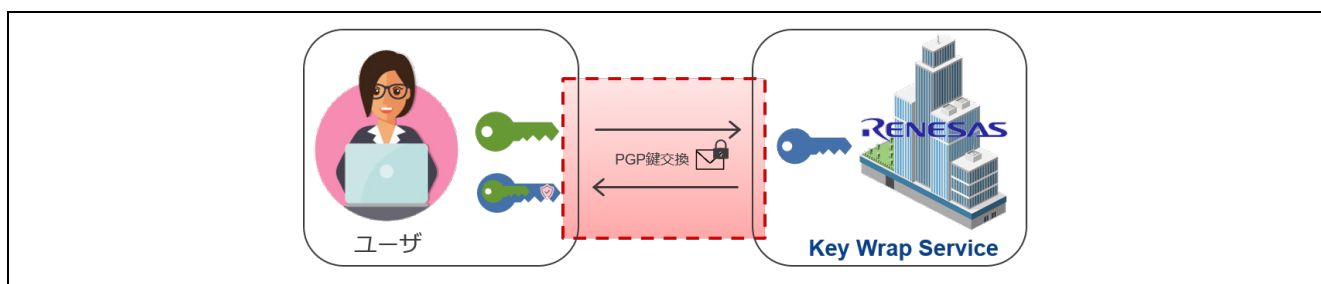


図2-2 ルネサス Key Wrap サーバを使用した UFPK のラップ

3. ユーザ鍵(黄色)を UFPK でラップします。この操作には SKMT を利用します。

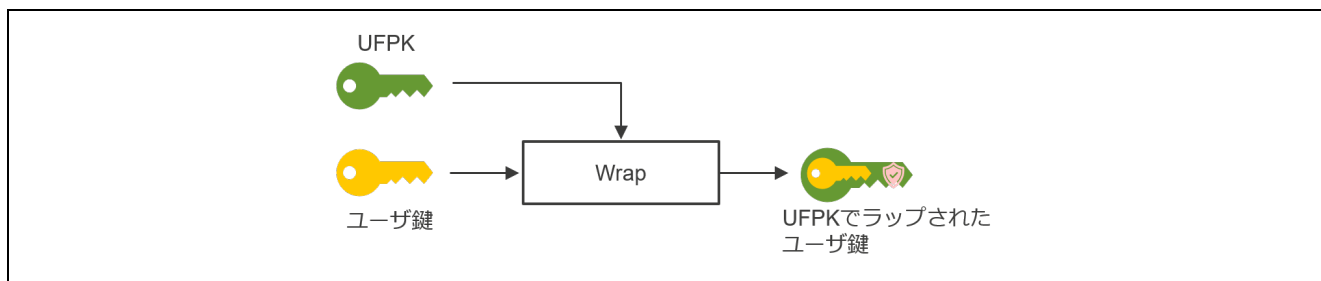


図2-3 ユーザ鍵を UFPK でラップ

HRK でラップされた UFPK と、UFPK でラップされたユーザ鍵を使用することで、平文の UFPK と平文のユーザ鍵を露出させることなく鍵注入を行うことができます。この操作を実現するため、それぞれのラップ鍵を生成します。

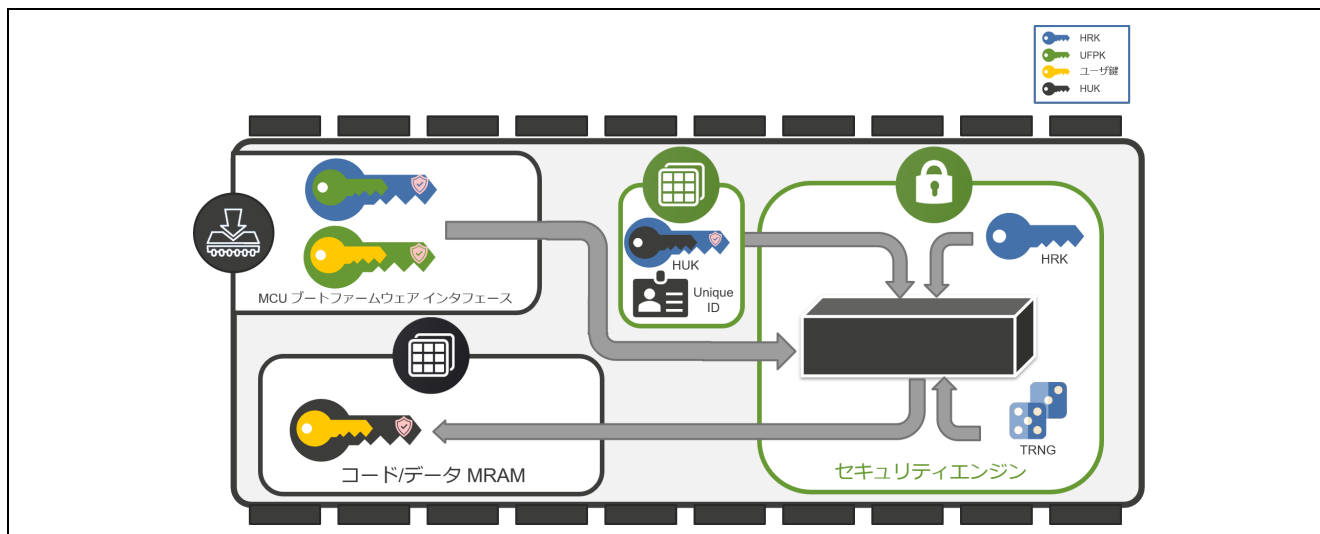


図2-4 シリアルプログラミングインターフェース経由でのユーザ鍵注入(Protected Mode)

4. SKMT を使用して鍵注入データを生成します。鍵注入データは、ラップされたユーザ鍵と W-UFPK を含む 1 つのセキュアな鍵注入のためのファイルとして生成されます。鍵の種別により生成されるデータは異なりますが、基本手順は共通です。
5. 最後に、鍵注入データを RFP で注入します。鍵注入データを注入されたセキュリティエンジンは、図2-4に示すように、セキュリティエンジン内部で HRK を使用して W-UFPK をアンラップし、アンラップされた UFPK を使用してラップされたユーザ鍵をアンラップします。

本手順は AES 鍵、RSA 鍵、ECC 鍵などの対応するユーザ鍵に共通して使用できます。具体的な操作例については5章で説明します。

2.1.2 鍵更新

フィールドへの新しい鍵の注入は通常、古い鍵を置き換える(鍵のローテーションまたは鍵の再入力)ために行われるため、このプロセスは「鍵更新」と呼ばれます。Protected Mode では、フィールドでセキュアな鍵更新を有効にするには、上記のように、製造時のプログラミング/プロビジョニング中に 1 つ以上の KUK を注入する必要があります。

KUK は、他の暗号化鍵と同様に、コード MRAM またはデータ MRAM のいずれかに保存できます。KUK の注入は、2.1.1 で説明されているように、他のユーザ鍵の注入と同じ手順を使用します。KUK は新しい鍵を注入/ラップできる唯一のメカニズムであるため、製造時のプロビジョニング中に複数の KUK を注入することを強くお勧めします。これにより、KUK をローテーションまたは取り消して、インフラストラクチャのセキュリティポリシーへの準拠や、鍵漏洩によるセキュリティ侵害への対応ができます。

プログラミングインタフェースが無効になった後、追加の KUK を注入することは**できません**。プログラミングインタフェースを無効にした状態で製品が市場に展開されると、新しい鍵は既存の KUK を介して**のみ**注入できます。

KUK は、製造工程中に任意のコードまたはデータ MRAM 領域に保存できます。この KUK は、新しいユーザ鍵を注入する鍵更新 API の引数として使用されます。ユーザは複数の KUK を注入し、定期的または鍵漏洩事象に基づいて鍵をローテーションする手段を用意することができます。セキュリティ上の理由から、ユーザは製品を市場に展開する前にプログラミングインタフェースを無効にすることをお勧めします。

鍵更新には、4 つの大まかな手順があります。KUK はすでに MCU に注入されている必要があることに注意してください。

1. KUK を生成します。KUK は2.1.1の手順で事前にデバイスに注入しておく必要があります。
2. 更新する新しいユーザ鍵を生成します。
3. KUK(赤色)を使用して、新しいユーザ鍵(黄色)をラップします。この操作には SKMT を利用します。

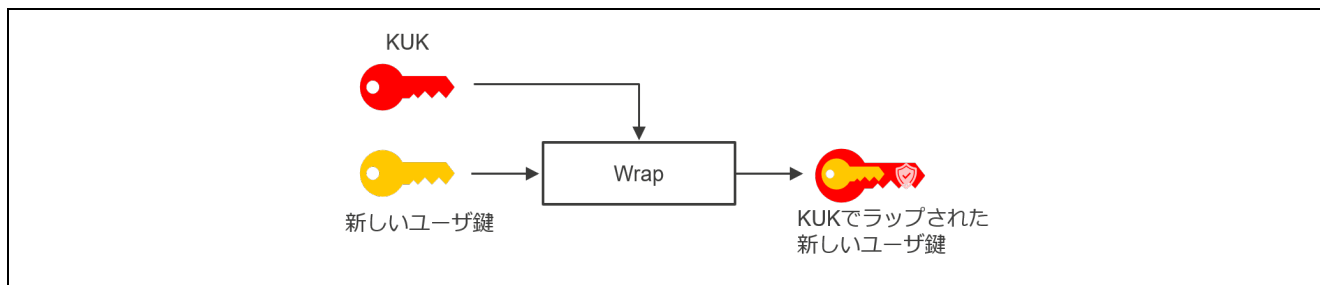


図2-5 新しいユーザ鍵を KUK でラップ

4. FSPの鍵更新APIと以前に注入したKUKを使用して、新しいユーザ鍵を注入します。新しいユーザ鍵は、HUK(黒)によってラップされます。そのため、鍵更新後のユーザ鍵は安全な形式の鍵として保持されます。

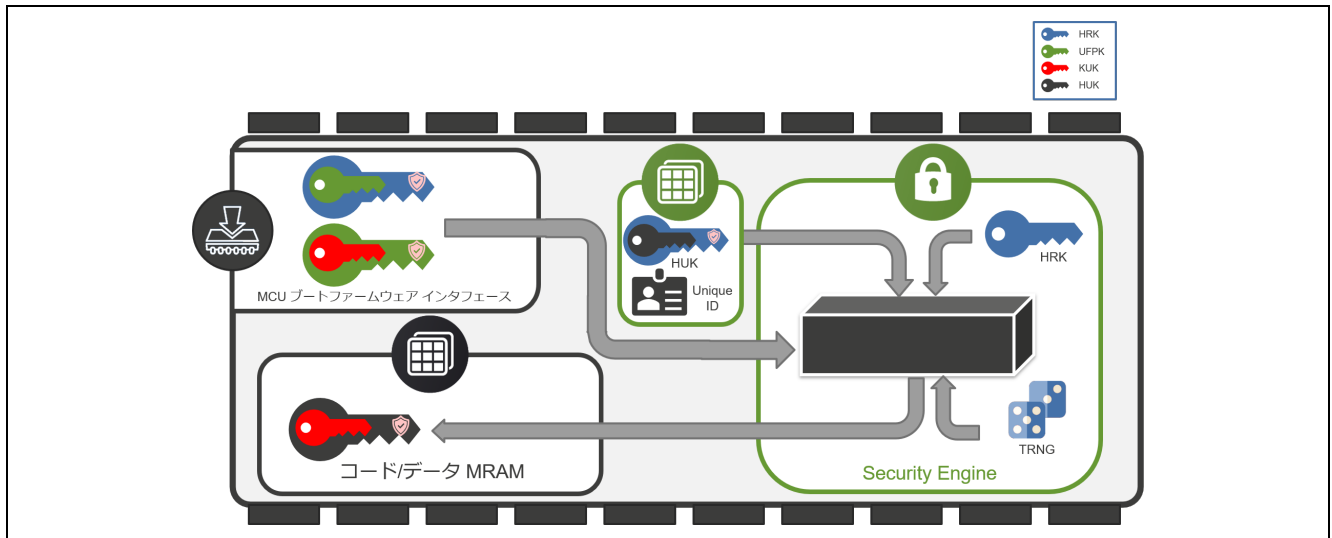


図2-6 KUK の注入

使用する鍵の種別によって生成するデータは異なりますが、鍵更新の基本的な流れは共通です。具体例については5章で説明します。

2.2 Compatibility Mode の鍵注入

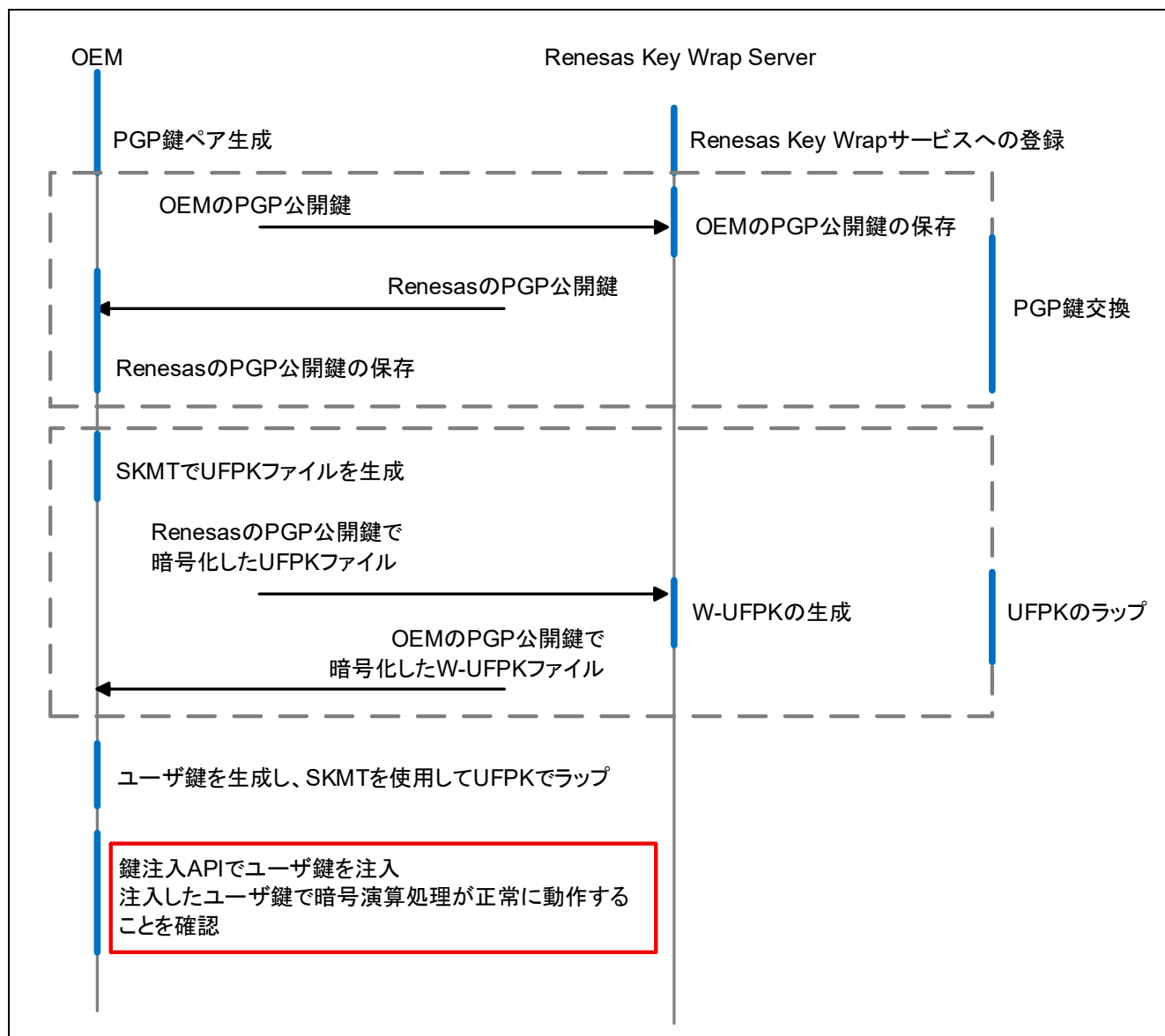


図2-7 Compatibility Mode における鍵注入の流れ

Compatibility Mode における鍵注入の流れを図2-7に示します。本アプリケーションノートのサンプルプロジェクトでは、Compatibility Mode において、サンプルとして用意している鍵を使用して鍵注入の動作を示します。これは、図2-7の赤枠で囲われた部分に相当します。詳細については、3章、4章、および6章を参照してください。

Compatibility Mode では、FSP の鍵注入 API を使用して鍵注入を行います。Protected Mode とは鍵注入インタフェースが異なることに注意してください。

鍵注入には、以下に示す4つの大きなステップがあります。この中で、1から3番目のステップはProtected Mode での鍵注入のステップと共通です。また、暗号鍵が平文で取り扱われる鍵準備の手順は、セキュアな環境で実行する必要があります。

1. ユーザ鍵のラップに使用するための UFPK を生成します。UFPK は SKMT を使用して生成することもできます。詳細については4章を参照してください。
 2. HRK を使用して UFPK をラップして W-UFPK を生成します。W-UFPK の生成には、ルネサス Key Wrap サービスを使用します。詳細については3章および4章を参照してください。
 3. ユーザ鍵を UFPK でラップします。
 4. SKMT を使用して鍵注入データを生成します。鍵注入データは、ラップされたユーザ鍵と W-UFPK を含む C ソースファイルとして生成されます。この C ソースファイルのデータを FSP の鍵注入 API に渡すように実装します。鍵の種別により生成されるデータは異なりますが、基本手順は共通です。
- 鍵注入データを注入されたセキュリティエンジンは、図 2-8に示すように、セキュリティエンジン内部で HRK を使用して W-UFPK をアンラップし、アンラップされた UFPK を使用してラップされたユーザ鍵をアンラップします。

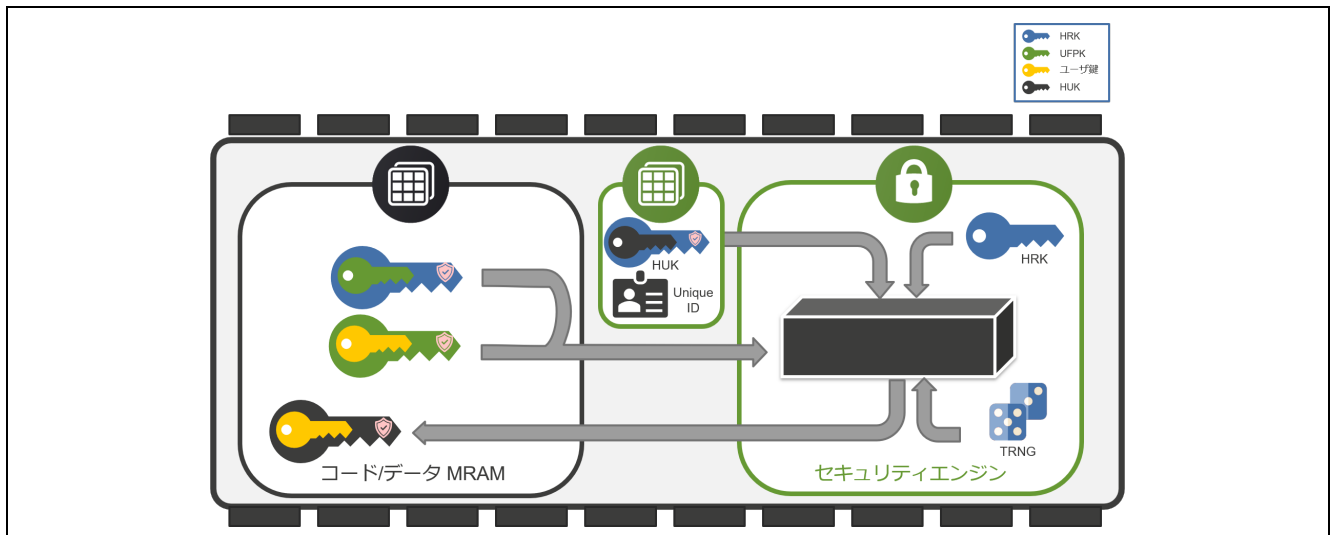


図 2-8 API 経由でのユーザ鍵注入(Compatibility Mode)

Compatibility Mode は KUK を使用した鍵更新をサポートしていません。

本手順は対応する各種ユーザ鍵に適用できます。具体的な操作例については6章で説明します。

2.3 セキュアな鍵注入および鍵更新に使用するツール

セキュアな鍵注入および鍵更新に使用するツールを示します。これらのツールの入手、設定、および使用の詳細については、記載されている章を参照してください。

2.3.1 ルネサス Key Wrap サービス

W-UFPK を取得するために使用します。PGP によりユーザが登録した UFPK をルネサスのサーバにおいてラップし、生成された W-UFPK をメール経由で取得します。詳細は3章を参照してください。

2.3.2 Security Key Management Tool (SKMT)

ユーザ鍵と KUK をラップするために使用します。ラップする鍵のデータに加えて、UFPK と W-UFPK を入力として、ラップされた鍵を注入するためのデータが含まれるファイルを出力します。詳細は4章、5章、および6章を参照してください。

2.3.3 Renesas Flash Programmer (RFP)

Protected Mode を使用する際に、デバイスの初期化、および鍵データを MRAM に書き込むために使用します。SKMT により生成した鍵データファイルを使用し、MRAM の指定したアドレスに書き込みます。詳細は5章を参照してください。

2.3.4 e² studio / Flexible Software Package (FSP)

サンプルプロジェクトのビルド、およびコード生成のために使用します。本アプリケーションノートのサンプルプロジェクトは、FSP によりコード生成されるソースコードを使用しています。また、統合開発環境として e² studio を使用し、ビルドして動作させます。詳細は5章、および6章を参照してください。

3. ルネサス Key Wrap サービスの準備

セキュアな鍵注入で使用する W-UFPK を取得するためには、ルネサス Key Wrap サービスを利用する必要があります。ルネサス Key Wrap サービスでは、鍵情報を安全に交換するために PGP を使用します。そのため、サービス利用開始前にユーザとルネサス Key Wrap サーバ間で PGP 公開鍵を交換する必要があります。本章では、PGP 暗号化通信チャネルを確立するための手順について説明します。この作業は初回のみ必要であり、異なる MCU に対して繰り返し実施する必要はありません。

3.1 PGP 鍵ペアの生成

PGP 鍵ペアが既にある場合は、その鍵を鍵交換プロセスに使用できます。そうでない場合は、以下の手順で PGP 鍵ペアを生成することができます。

ここでデモンストレーションする PGP ソフトウェアは Gpg4win で、この URL: <https://www.gpg4win.org/> からダウンロードできます。

このアプリケーションノートに含まれるスクリーンショットは、gpg4win-4.0.0.exe に基づいています。それ以降のバージョンでは、グラフィックインタフェースのマイナーな更新がある場合があります。ただし、このアプリケーションノートで使用されている機能は維持される必要があります。

Kleopatra インストーラをダウンロードしてインストールします。インストール完了後、**Next** および **Finish** ボタンをクリックしてインストーラを終了します。

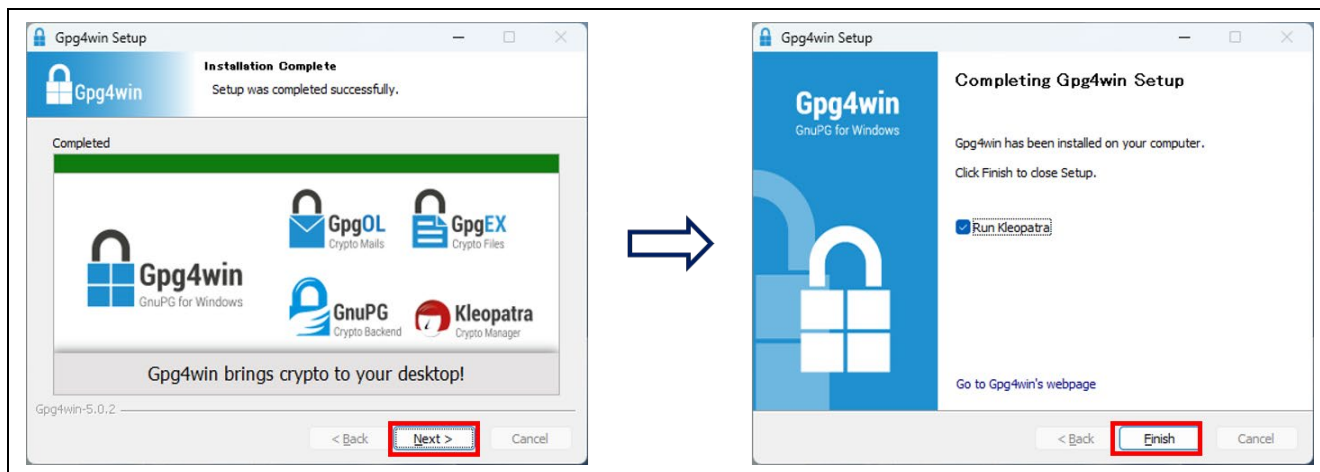


図3-1 Kleopatra インストーラのダウンロードおよびインストール

Kleopatra を起動し、PGP 鍵ペアを生成します。

1. **ファイル > 新しい OpenPGP 鍵ペアを作成** をクリックします。



図3-2 個人用の OpenPGP 鍵ペアを作成

2. 名前およびメールアドレスを指定します。これらはオプションとしてマークされていますが、次のステージに移動するには、少なくとも1つのエンティティを指定する必要があることに注意してください。生成する鍵はパスフレーズで保護されます。 をチェックします。

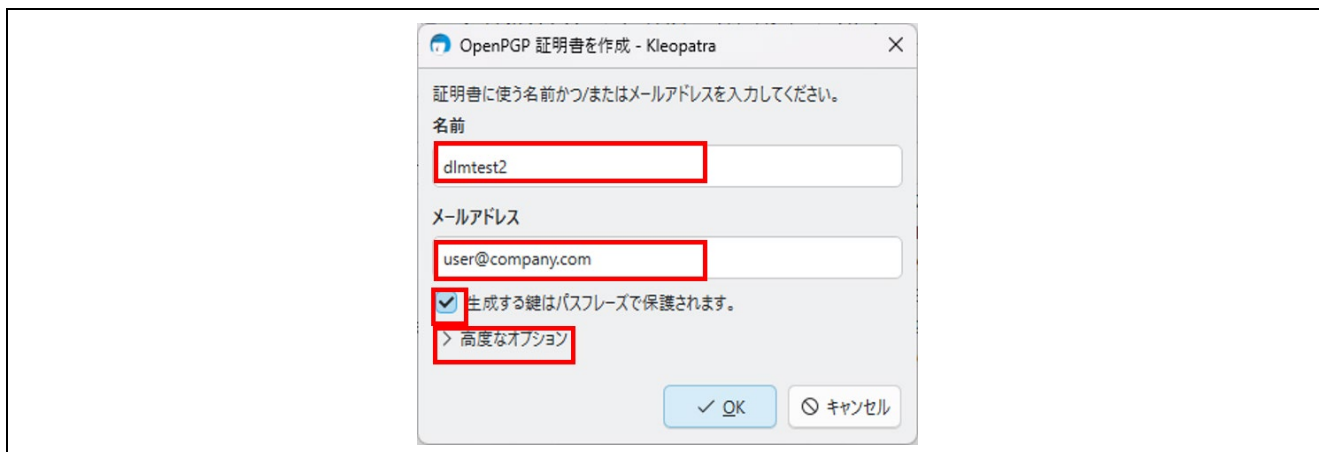


図3-3 名前とメールアドレスを記入

3. 高度なオプションをクリックし、鍵素材の種類として **rsa3072** を選択します。OK をクリックします。

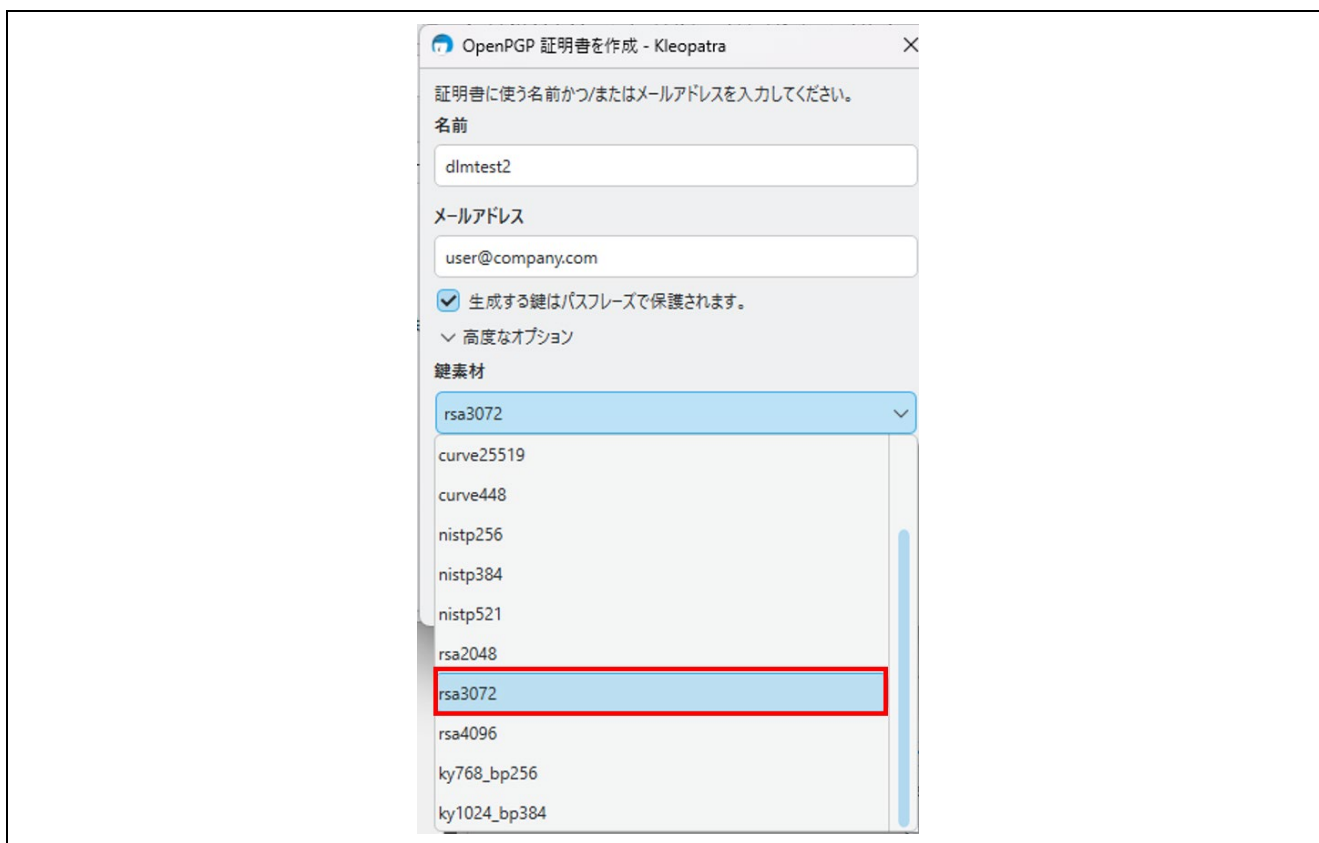


図3-4 RSA 暗号化を選択

4. 秘密鍵を保護するためにパスフレーズを2回指定します。次に **OK** をクリックします。**パスフレーズを必ず保存してください。**

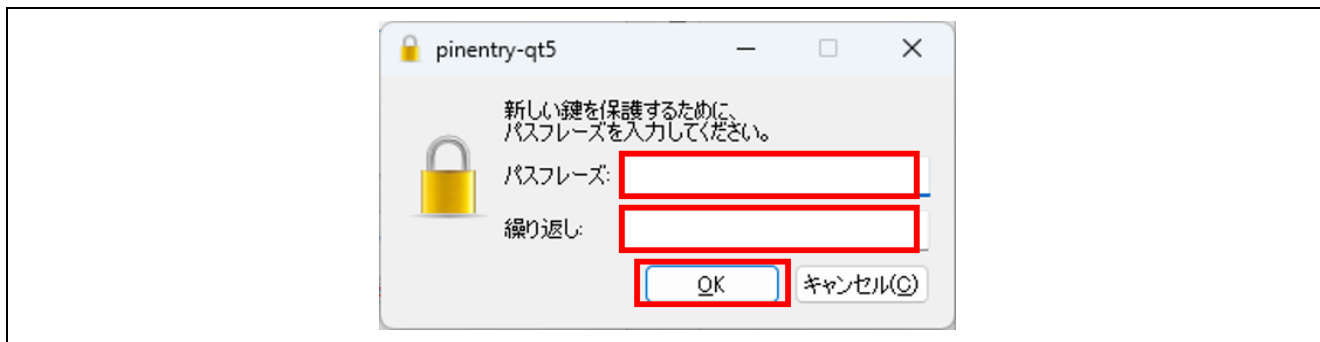


図3-5 パスフレーズを設定

5. PGP 鍵ペアは正常に生成されるはずです。**OK** をクリックします。

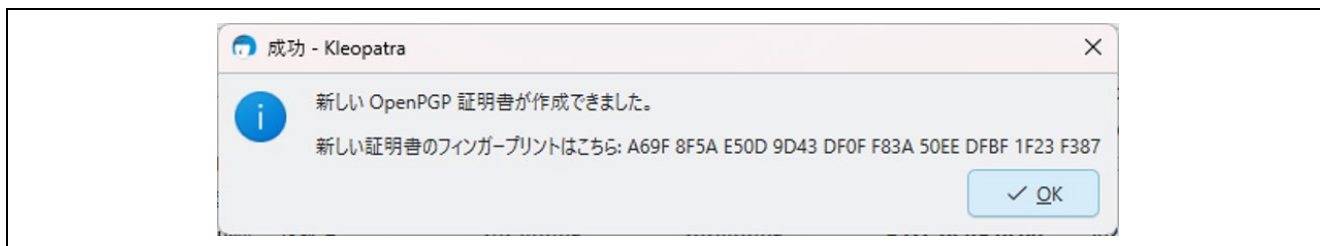


図3-6 PGP 鍵ペア作成

6. Kleopatra に新しいアイテムが生成されます。生成した鍵ペアを右クリックし、**エクスポート**を選択します。



図3-7 ユーザ PGP 公開鍵のエクスポート

7. 公開鍵を*.asc 拡張子付きのファイルに保存します。この例では、このファイルを public_key.asc にリネームしています。保存をクリックします。

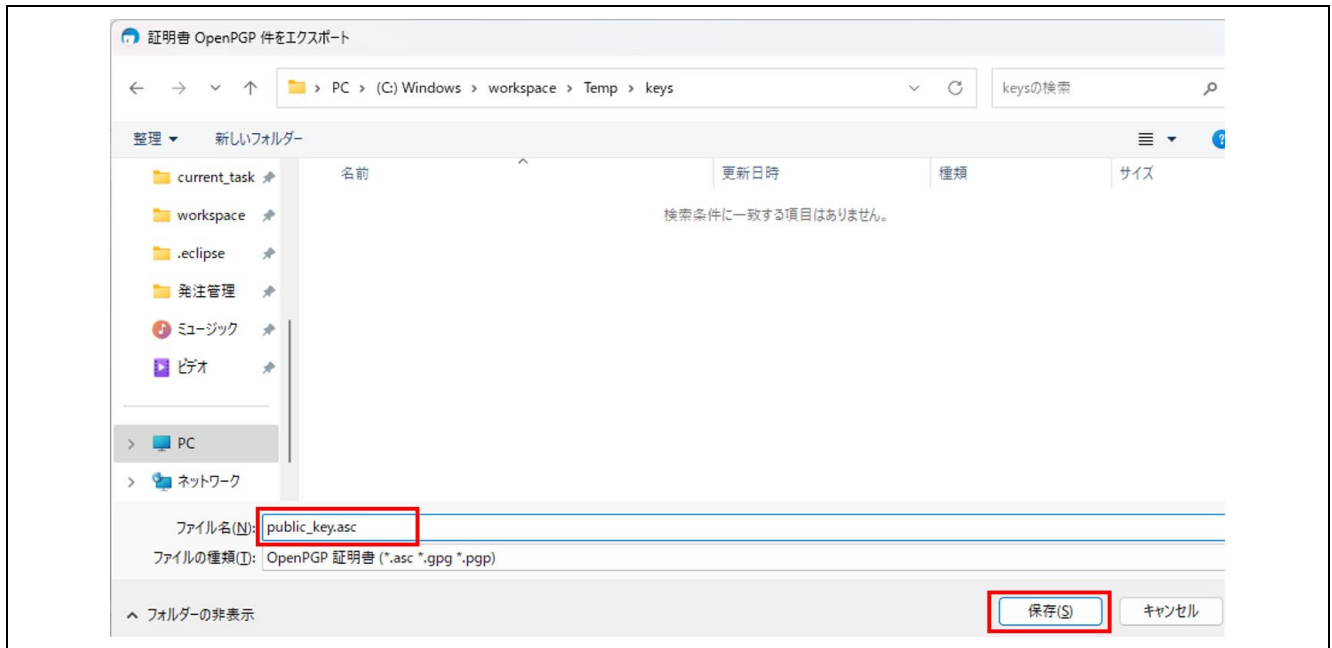


図3-8 PGP 公開鍵をフォルダに保存

3.2 ルネサス Key Wrap サーバへの登録

ルネサスKey Wrapサービスを初めてご利用になる場合は、ルネサスKey Wrapサーバへの登録が必要です。

1. ブラウザで URL <https://dlm.renesas.com/keywrap> を開き、**新規登録**をクリックします。

図3-9 ルネサス KeyWrap サーバへの登録開始

2. プロンプトに従って**有効なメールアドレス**を入力し、**メール送付**をクリックします。

図3-10 ユーザのメールアドレスを登録

メール送付をクリックすると、以下の画面が表示されます。**戻る**をクリックします。

3. 以下のようなメールが届きます。提供された URL をクリックして登録を確認します。

図3-11 登録確認メール

4. プロンプトに従って名前と会社名を入力し、パスワードを作成します。**次へ(確認)**ボタンをクリックします。パスワードは8文字から32文字の英数字で構成され、記号「!」と「@」を含めることができることに注意してください。



お客様情報の登録を行います。以下の項目を全て入力してください。
パスワードは8文字以上32文字以下の半角英数と半角記号「!@」で指定してください。

メールアドレス: usxxxx@*****.co.jp
氏名:
会社名:
パスワード:
パスワード再入力:

次へ (確認)

図3-12 登録の確認

3.3 ユーザとルネサスの PGP 公開鍵交換

ルネサス Key Wrap サーバと PGP 鍵をまだ交換していない場合は、以下の手順に従ってください。

3.3.1 ユーザ PGP 公開鍵の登録

1. ユーザ情報の登録に成功すると、以下の画面が開きます。**サービス開始**ボタンをクリックして、ルネサス Key Wrap サービスの使用を開始します。



登録しました。

メールアドレス: xxxx@*****.co.jp
氏名: ルネサス 太郎
会社名: ルネサスエレクトロニクス

サービス開始

図3-13 ルネサス Key Wrap サービスの開始

2. 以下に示すように、利用規約を確認してから、**トラステッドセキュア IP Key Wrap サービス利用規約に同意します**をクリックし、**トラステッドセキュア IP Key Wrap サービス利用許諾**を受諾します。この規約は、ルネサス Key Wrap サービスにログインするたびに表示されます。

重要-以下の利用規約を注意してお読みください。

以下のトラステッドセキュアIP Key Wrapサービス利用規約（以下「本利用規約」といいます。）は、お客様およびお客様が所属する企業等の団体を代表して当社とのトラステッドセキュアIP Key Wrapサービス利用契約（以下「本契約」といいます。）を締結する場合のお客様が所属する企業等の団体（以下総称して「お客様および所属企業等」といいます。）がトラステッドセキュアIP鍵暗号化に係るサービス（以下「本サービス」といいます。）を利用する際に適用される条件を定めたものです。慎重に内容をお読みください。お客様および所属企業等は、本利用規約に定める条件に従って、本サービスを利用しなければなりません。本利用規約の内容に同意できない場合は、本サービスを利用しないでください。本サービスを利用した場合、お客様および所属企業等は本契約に同意したものとみなされます。

お客様が「トラステッドセキュアIP Key Wrapサービス利用規約に同意します」ボタンをクリックすることにより、(a)お客様は、お客様が所属する企業等の団体を代表して本契約を締結し、それにより当該団体が本利用規約の定める条件に法的に拘束されることを承諾の上、本契約を締結する意思表示を行ったこととなり、仮に、お客様がこのような団体に属さない場合には、お客様がご自身のために本契約を締結し、それによりお客様が本利用規約の定める条件に法的に拘束されることを承諾の上、本契約を締結する意思表示を行ったこととなります。また、これにより、(b)お客様は当該団体（所属する場合）を代表しまたはお客様自身のために行動し、当該団体または自身を拘束する権利、権能および権限を有することを表明しかつ保証したこととなります。

本契約は、お客様が「トラステッドセキュアIP Key Wrapサービス利用規約に同意します」ボタンをクリックすることにより、お客様と当社の間で有効に成立します。

図3-14 ルネサス Key Wrap サービス利用のアグリーメント

3. ルネサス Key Wrap サービスにログインすると、以下のウィンドウが表示されます。**PGP 鍵交換**をクリックします。



図3-15 PGP 鍵交換を開始

4. **参照**をクリックし、以前に生成した公開鍵(public_key.asc)を選択します。ルネサス PGP 公開鍵のフィンガープリントが表示されていることに注意してください。これは、ルネサス公開鍵を受け取った後に認証するために使用されます。その後 **PGP 鍵交換**をクリックします。

戻る

ログアウト

お客様のPGP公開鍵をエクスポート形式にしたものを選択後、“PGP鍵交換ボタン”を押してください。

お客様のPGP公開鍵がルネサスに送られ、ルネサスのPGP公開鍵がお客様のメールアドレスに送信されます。

注意：PGP鍵交換で利用可能な鍵の種類は『RSA鍵 3K/4K』のみです。

参照

PGP鍵交換

ルネサスPGP公開鍵のFingerprintは以下のとおりです。

FB18 EB66 1F61 20E9 9613
8DF7 F517 189C 1EA5 E55D

図3-16 お客様 PGP 公開鍵を参照

3.3.2 ルネサス PGP 公開鍵の取得

1. 鍵交換が成功すると、ルネサス Key Wrap サーバに登録されているメールアドレスに、ルネサスから以下の内容のメールが届きます。通常、このメールが届くまでに約 1～2 分かかります。
- PGP 公開鍵は何度でも登録できることに注意してください。正常に登録された最新の PGP 公開鍵が暗号化に使用されます。以前に登録されたすべての PGP 公開鍵は破棄されます。

ルネサス 太郎 様

KeyWrapサービスをご利用いただき、ありがとうございます。
登録いただいたPGP公開鍵のインポート処理を行いました。
弊社PGP公開鍵を添付ファイルにて送信いたします。
暗号化処理のご依頼は添付のルネサスPGP公開鍵を使用して暗号化処理後、
Webから送付してください。

このメールに心あたりがない場合、当メールを破棄していただきますようお願いいたします。

※このメールは送信専用メールアドレスからお送りしています。
返信いただいても回答できませんので、ご了承ください。

以上、よろしくお願いいたします。

ルネサスエレクトロニクス株式会社

添付ファイル:

keywrap-pub.key	2.3 k	[application/octet-stream]
---------------------------------	-------	------------------------------

図3-17 ルネサス PGP 公開鍵の受信

ルネサス PGP 公開鍵ファイル(keywrap-pub.key)を保存します。

3.3.3 ルネサス PGP 公開鍵のインポート

1. Kleopatra アプリケーションに戻り、以下に示すようにルネサス PGP 公開鍵を Kleopatra にインポートします。

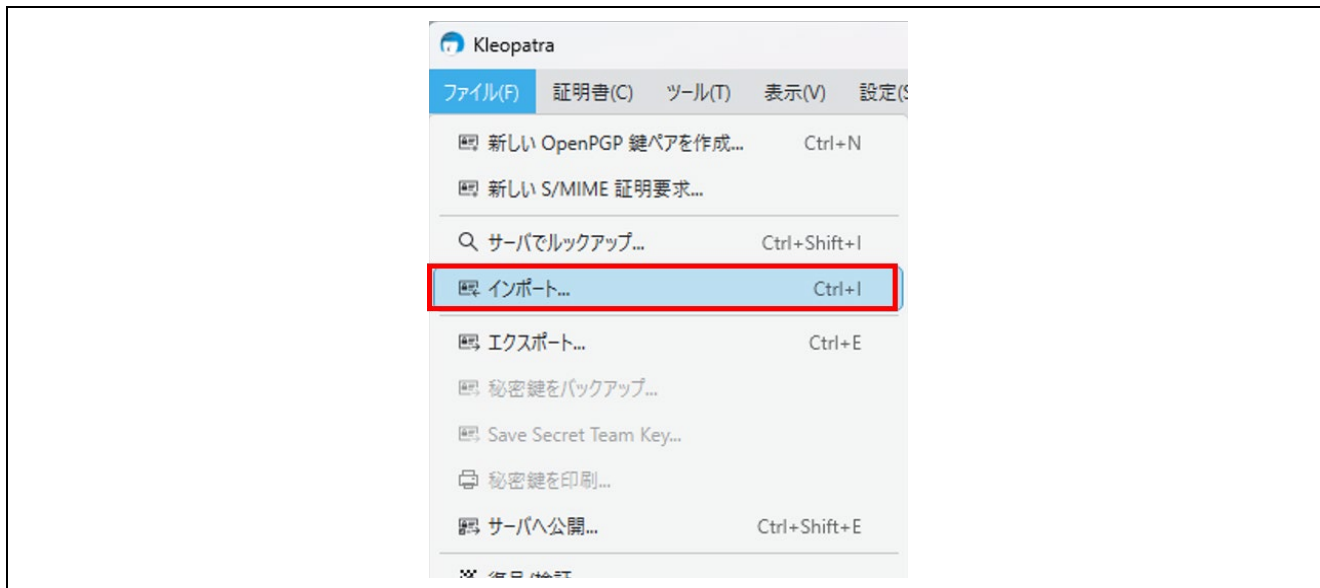


図3-18 ルネサス PGP 公開鍵のインポート

2. **Open** をクリックすると、Kleopatra のインポートされた証明書の中に次の項目が追加されます。

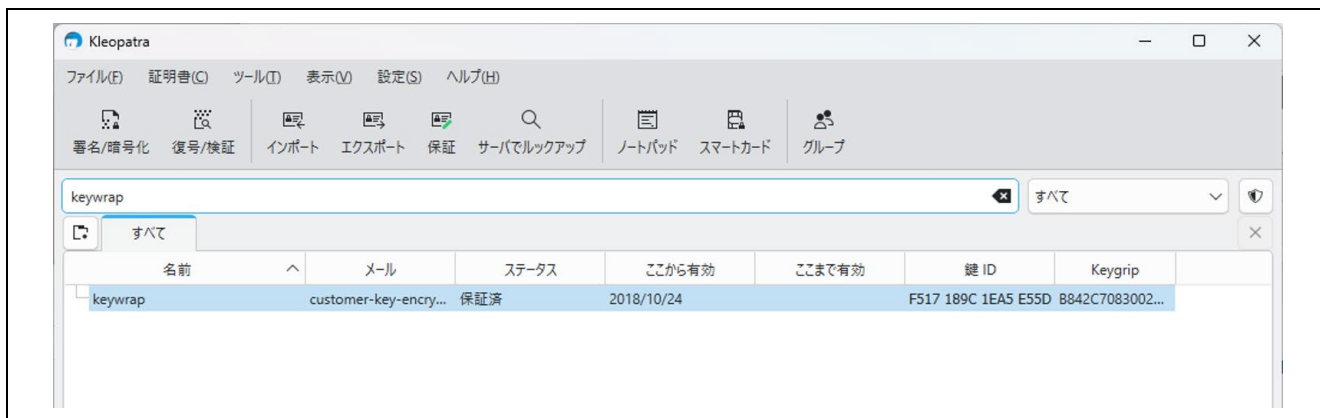


図3-19 ルネサス PGP 公開鍵

4. UFPK の生成と W-UFPK の取得

セキュアな鍵注入で使用する W-UFPK を取得するためには、ルネサス Key Wrap サービスを利用する必要があります。本章では、SKMT を使用した UFPK の生成方法と、ルネサス Key Wrap サービスを使用した W-UFPK の取得方法について説明します。

4.1 UFPK の生成

4.1.1 SKMT のインストール

SKMT は、セキュアな鍵注入プロセス中に複数の機能を実行します。最新の SKMT にアクセスするには、次のリンクを開きます。

<https://www.renesas.com/software-tool/security-key-management-tool>

リンク先から、最新の SKMT インストーラをダウンロードします。このツールは、Windows、Linux、および macOS をサポートしています。

インストーラの実行ファイルをダウンロードしたら、インストーラを右クリックし、**管理者として実行**を選択してこのツールをインストールします。プロンプトに従って、**セットアップに使用する言語**を選択します。次に、インストールフォルダを選択します。デフォルトでは、`C:\Renesas\SecurityKeyManagementTool\`です。以前のバージョンがインストールされている場合、古いバージョンは上書きされます。

このツールのユーザズマニュアルは、`\doc` フォルダにあります。事前にユーザズマニュアルを参照してください。

SKMT は、Command Line Interface(CLI)と Graphic User Interface(GUI)の 2 つのインタフェースをユーザに提供します。CLI は通常、製造サポートに使用され、GUI は主に開発用です。このアプリケーションノートでは、両方のインタフェースを使用して鍵注入および更新を実行する方法について説明します。

4.1.2 UFPK 生成 (GUI)

UFPK を定義し、ルネサス Key Wrap サービスと互換性のあるバイナリ形式に変換します。これは、SKMT を使用して行うことができます。

UFPK の生成単位および運用方法は、システムのセキュリティポリシーに依存します。W-UFPK は、対象の MCU ファミリおよびセキュリティエンジンモードに対応して生成されるため、W-UFPK を取得する際は対象 MCU ファミリおよびセキュリティエンジンモードを正しく選択してください。

SecurityKeyManagementTool.exe をダブルクリックして、GUI を起動します。

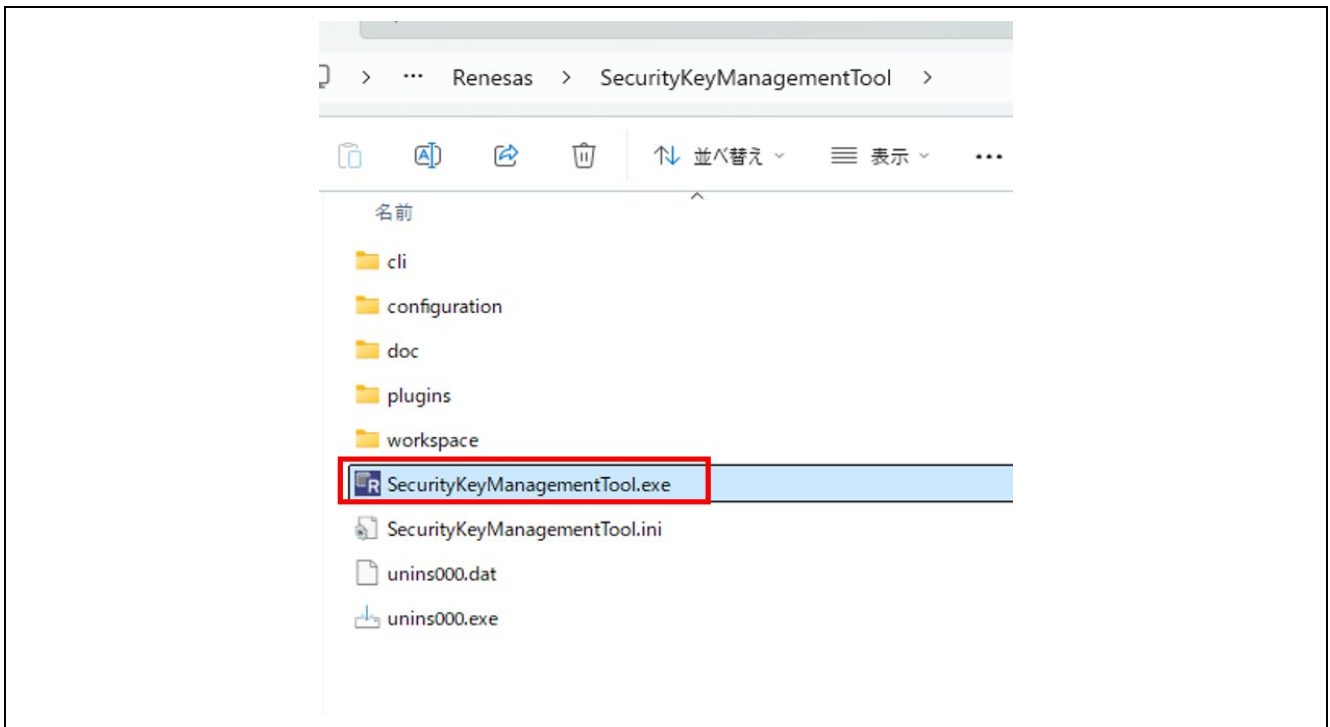


図4-1 SKMT GUI の起動

このアプリケーションプロジェクトに含まれるサンプルプロジェクトを使用するには、UFPK を 000102030405060708090a0b0c0d0e0f000102030405060708090a0b0c0d0e0f に設定します。

32 バイトの UFPK はビッグエンディアン形式で提供する必要があることに注意してください。

SKMT を使用するときは、正しい MCU ファミリとセキュリティエンジンモードを選択することが重要です。

- RX74M は RSIP-E50D を搭載しています。RX74M Protected Mode のサンプルプロジェクト向けに使用する場合は、**概要**タブで **RX Family, RSIP-E50D Security Functions and Protected Mode** を選択します。



図4-2 RXファミリ、RSIP-E50D Protected Modeを選択

- RX74M Compatibility Mode のサンプルプロジェクト向けに使用する場合は、概要タブで **RX Family, RSIP-E50D Compatibility Mode** を選択します。

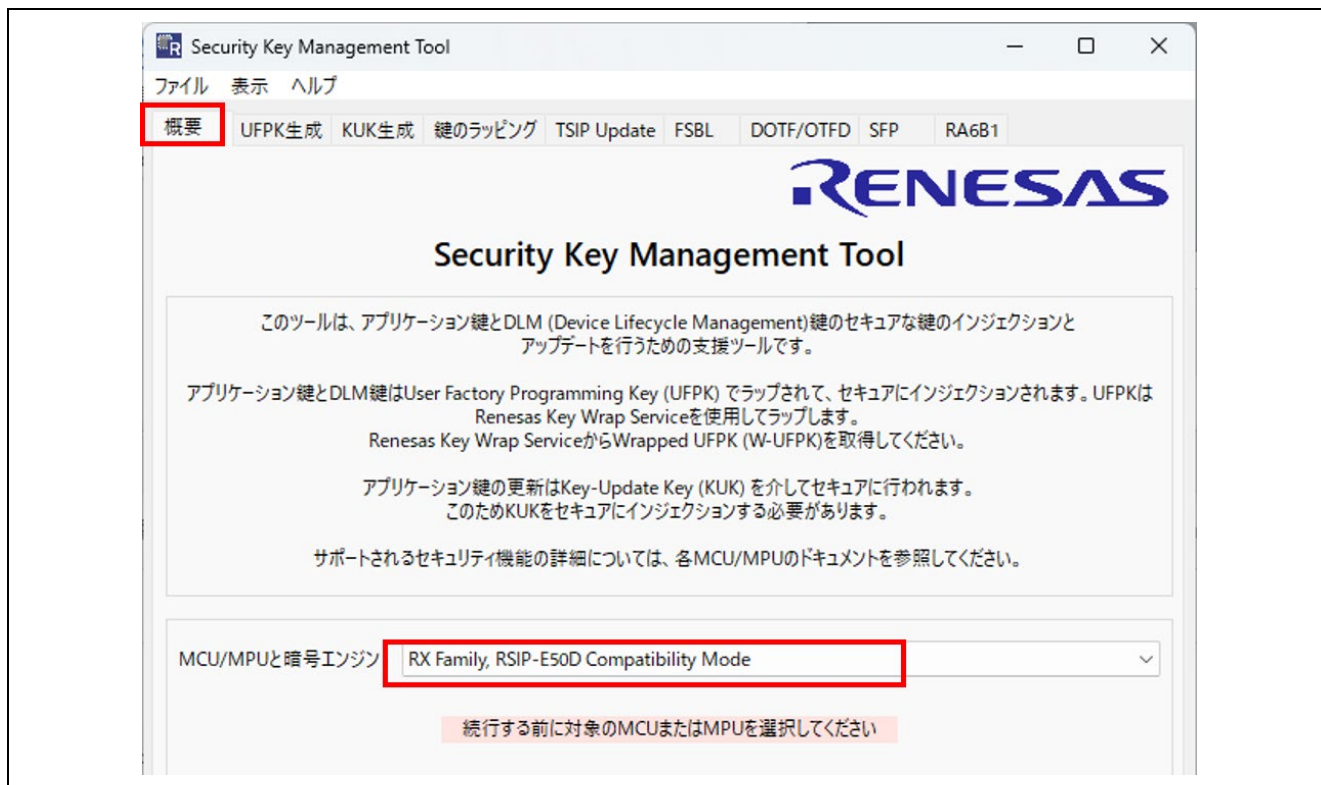


図4-3 RX ファミリ、RSIP-E50D Compatibility Mode を選択

正しいMCU ファミリ、セキュリティエンジン、およびモードを選択したら、**UFPK 生成**タブに移動します。

- **User Factory Programming Key** で、**指定値を使用する** を選択し、値を入力します。ここでは、以下の値を例に使用しています。

000102030405060708090a0b0c0d0e0f000102030405060708090a0b0c0d0e0f

- **参照ボタン**をクリックして、鍵を格納するフォルダを選択し、ファイル名を入力します。
UFPK ラッピング段階での混乱を避けるために、MCU ファミリごとに異なるファイル名を選択することをお勧めします。この例では、ファイル名を rx74m_ufpk.key とします。

UFPK を生成する をクリックします。rx74m_ufpk.key ファイルが生成されます。

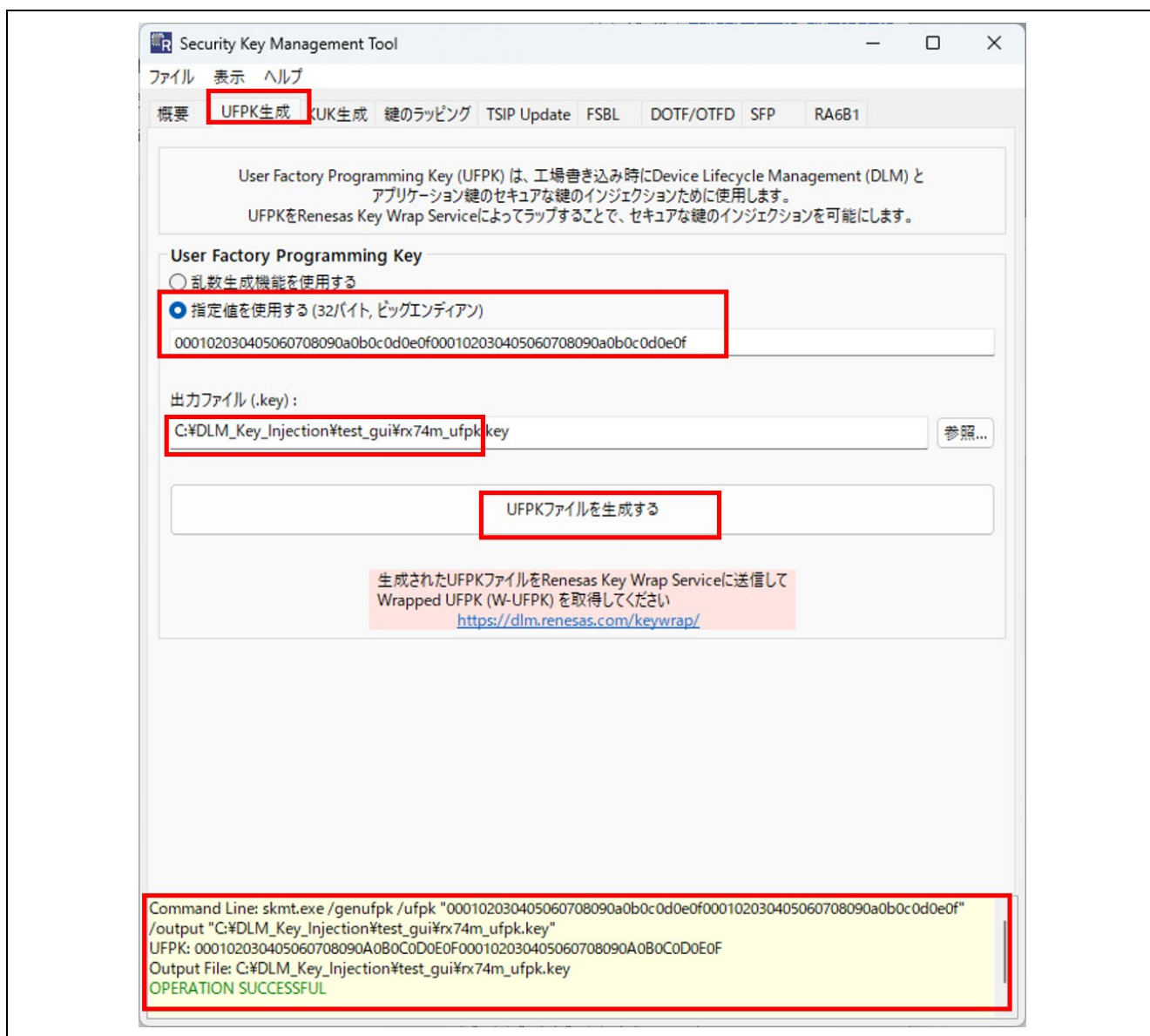


図4-4 RSIP-E50D の GUI を使用した固定 UFPK の生成

上記の操作の代わりに、ユーザは**乱数生成機能を使用する** を選択して UFPK を生成することもできます。

4.1.3 CLI を使用した UFPK の生成

コマンドプロンプトウィンドウを開き、skmt.exe が存在するフォルダに移動します。本 APN で使用している SKMT のバージョンでは、\Renesas\SecurityKeyManagementTool\cli\配下にあります。

次のコマンドを使用して、ランダムな UFPK を生成し、鍵ファイル(rx74m_ufpk.key)に配置します。必要に応じて、パス付きの完全なファイル名を指定できます。/genufpk オプションの用法については、SKMT のユーザズマニュアルを参照してください。

```
skmt.exe /genufpk /output  
"C:\Secure_Key_Injection\rx74m_protected_mode_key_info\rx74m_ufpk.key"
```

このコマンドは、次に示すようにランダムな 256 ビット UFPK を生成します。

```
UFPK: E8AB23E99C9AD42823DA4215549A41496720F7243680A4715F4B944ACC94B691  
Output File:  
C:\Secire_Key_Injection\rx74m_protected_mode_key_info\rx74m_ufpk.key
```

図4-5 SKMT CLI を使用したランダム UFPK の生成

次のコマンドに示すように、特定の UFPK を指定することもできます。

```
skmt.exe /genufpk /ufpk  
"000102030405060708090a0b0c0d0e0f000102030405060708090a0b0c0d0e0f" /output  
"C:\Secure_Key_Injection\rx74m_protected_mode_key_info\rx74m_ufpk.key"
```

```
UFPK: 000102030405060708090a0b0c0d0e0f000102030405060708090a0b0c0d0e0f  
Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\rx74m_ufpk.key
```

図4-6 SKMT CLI を使用した固定 UFPK の生成

4.2 W-UFPK の取得

次のステップは、選択した UFPK に基づいて、ルネサス Key Wrap サービスから W-UFPK を取得することです。UFPK を変更する場合は、新しい W-UFPK を取得する必要があることに注意してください。

4.2.1 UFPK の PGP 暗号化

1. Kleopatra プログラムを起動します。
2. ルネサス公開鍵で UFPK を暗号化します。この鍵は先に Kleopatra にインポート済みです。Kleopatra を使用して、**署名/暗号化**を選択し、UFPK ファイルを選択します。このスクリーンショットでは、デモンストレーションの目的で `rx74m_ufpk.key` ファイルという名前のサンプルファイルが使用されています。次に、**開く**をクリックします。

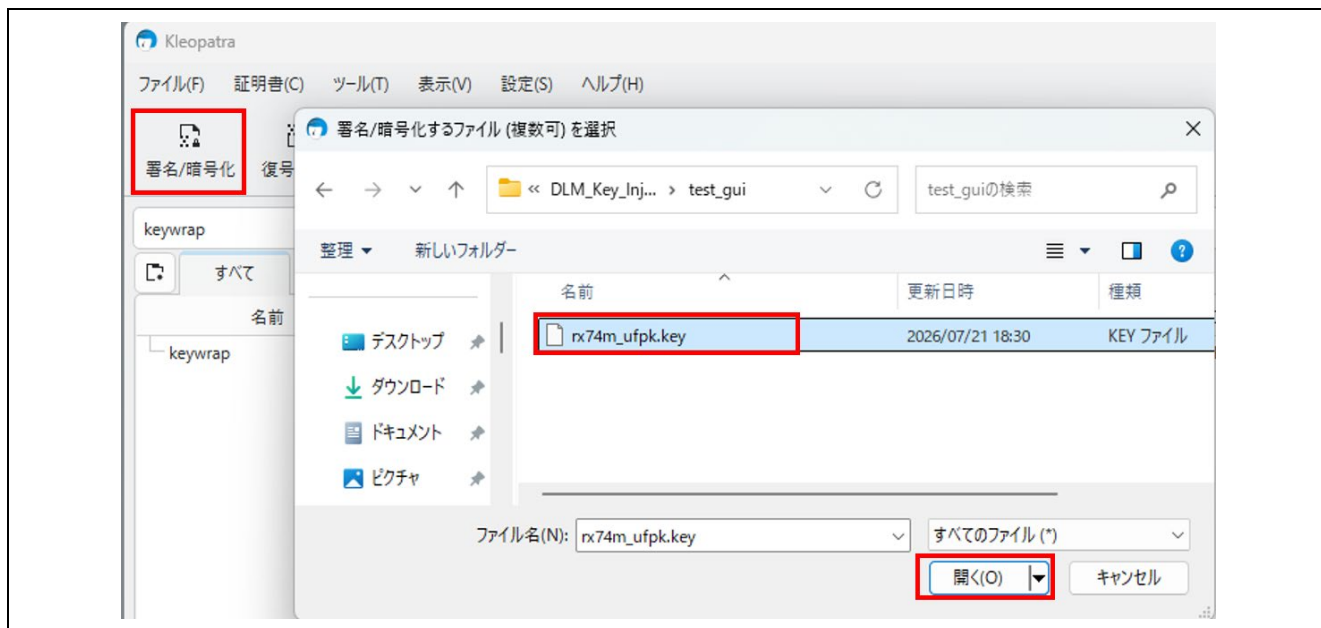


図4-7 PGP 転送用の UFPK ファイルの暗号化

3. このファイルを誰向けに暗号化するかを尋ねられたら、必要に応じて自分自身のために暗号化のチェックを外し、**Sign as**、**他の方のために暗号化**、および **Encrypt / Sign each file separately** をチェックします。

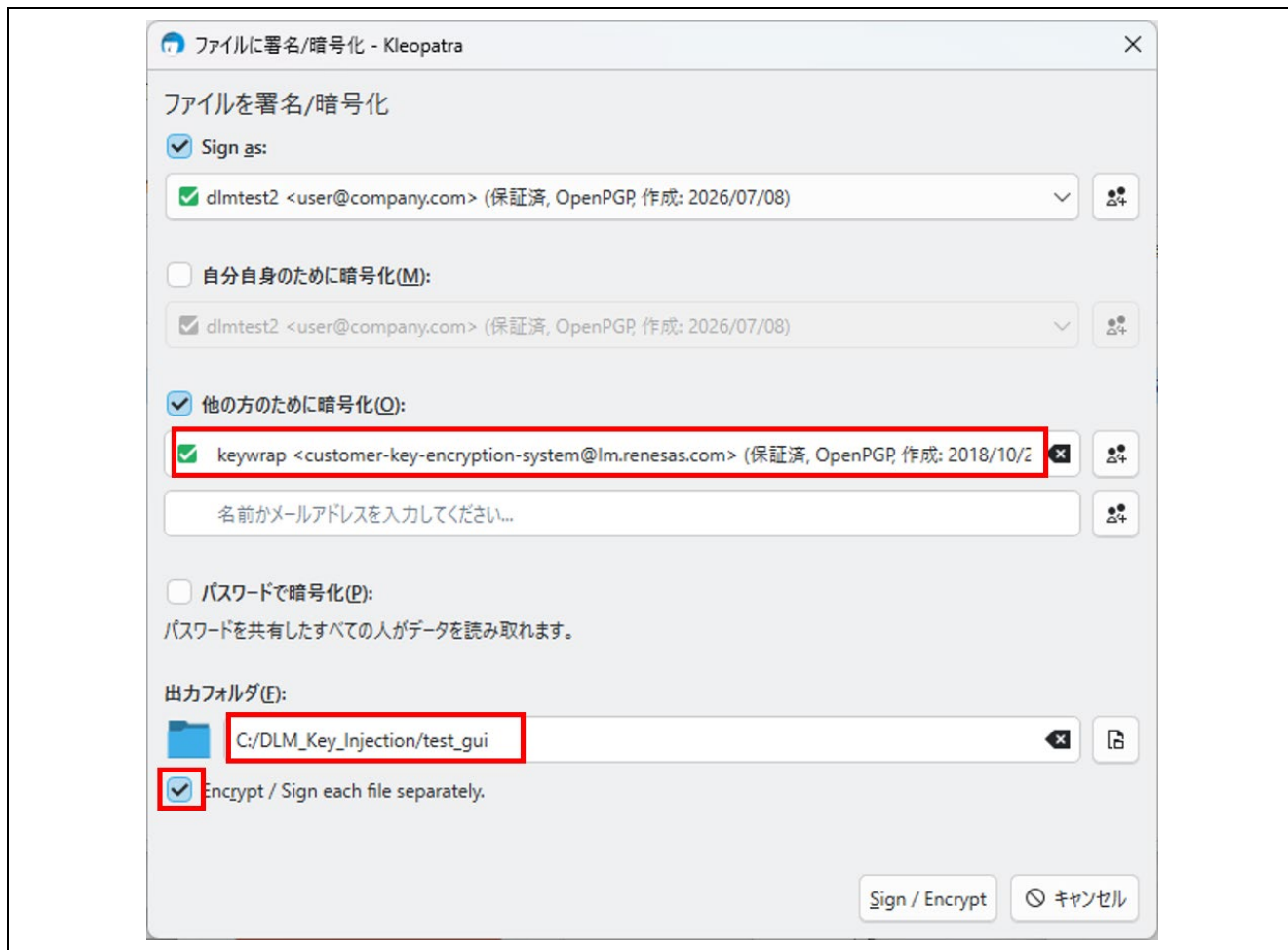


図4-8 PGP 暗号化オプションの選択

4. 他の方のために暗号化の証明書リストを表示 (👤 アイコン)をクリックします。これにより、証明書の選択ダイアログボックスを開きます。

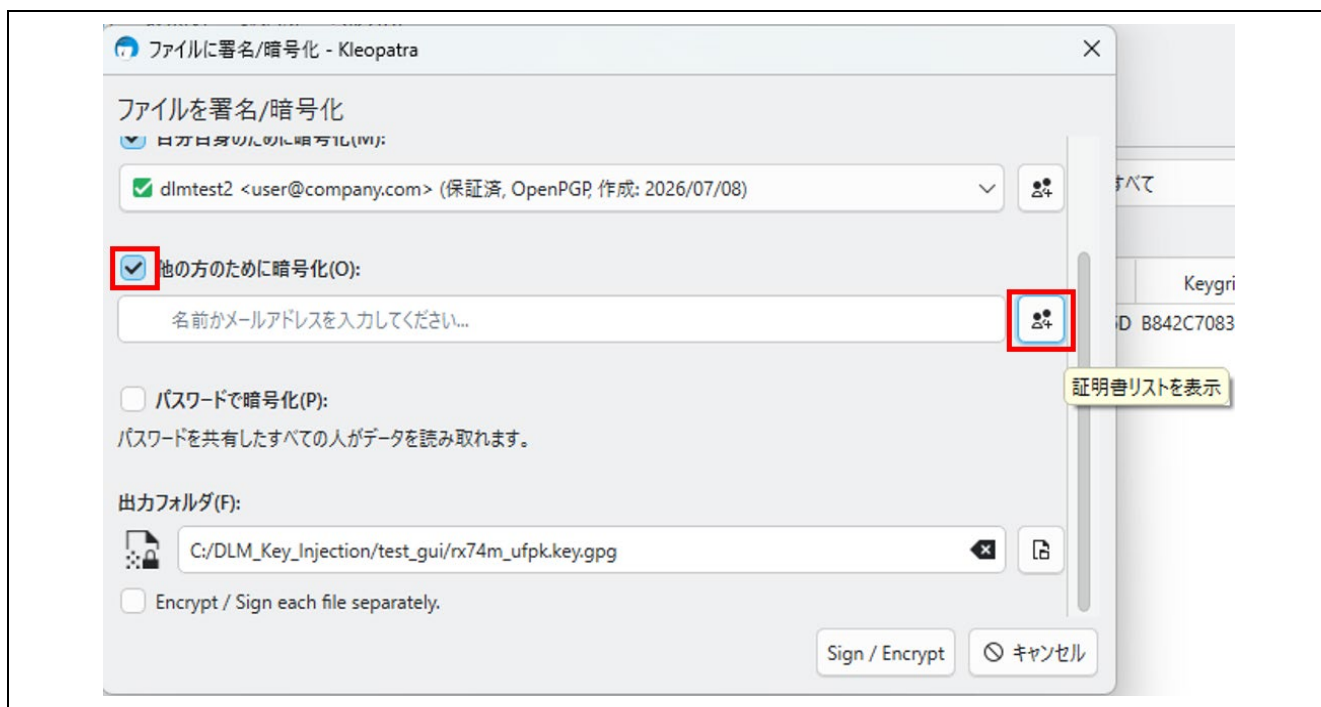


図4-9 選択ダイアログを開く

5. このウィンドウで、**keywrap** を選択してルネサス公開鍵を選択し、**OK** をクリックします。

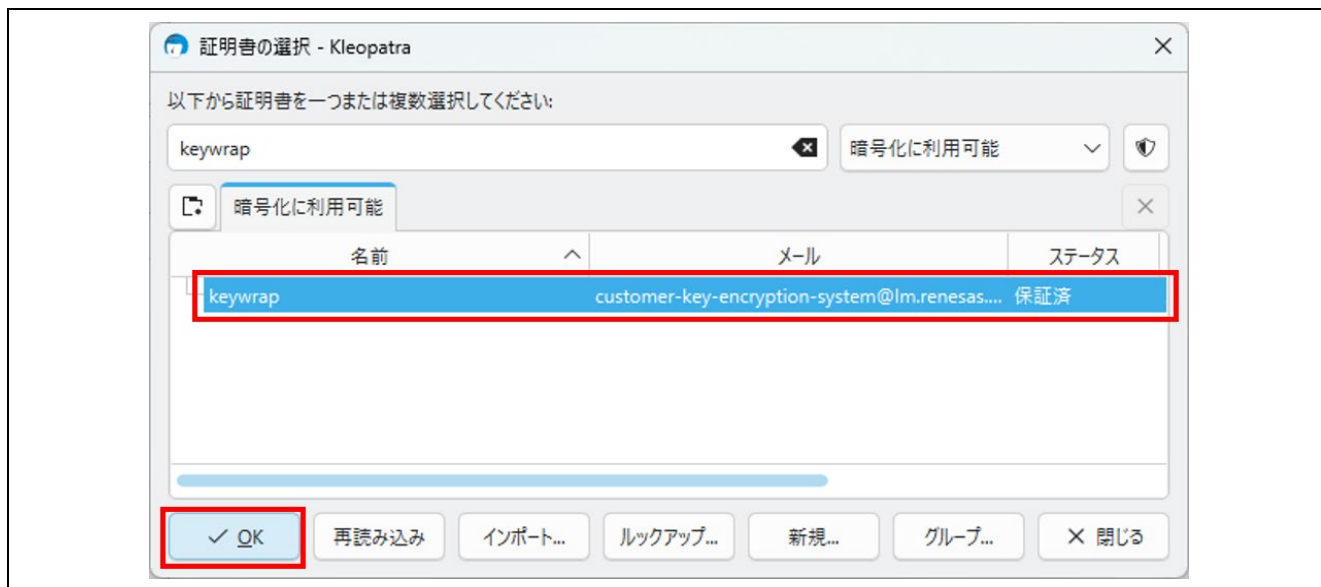


図4-10 ルネサス PGP 公開鍵を選択

6. **Output** で、暗号化された鍵の正しい保存先フォルダが選択されていることを確認します。最後に、**Sign/Encrypt** をクリックします。異なる MCU ファミリの UFPK と W-UFPK は、別々のフォルダと別々の名前で保存することをお勧めします。

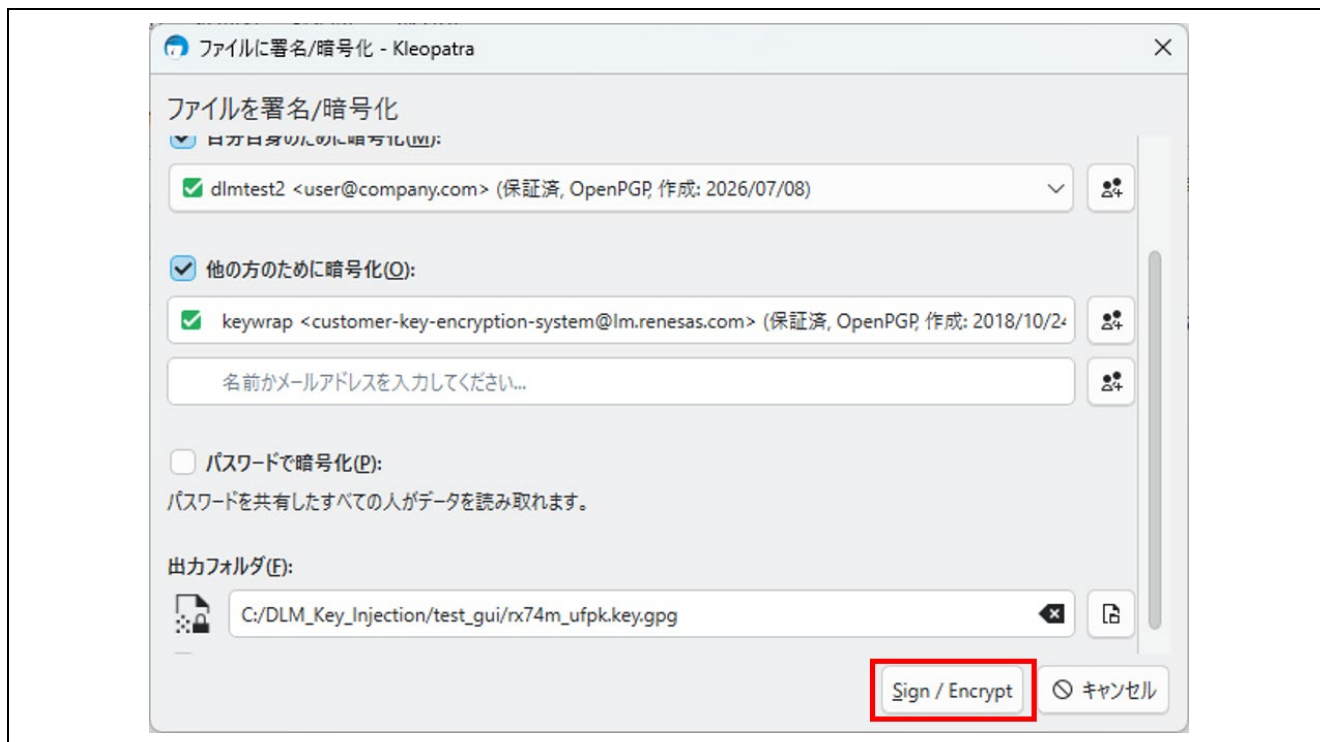


図4-11 ルネサス PGP 公開鍵を使用した UFPK の暗号化

7. **Encrypt for me** をチェックしないと、データを復号できないという **Encrypt-To-Self** の警告が表示されます。データを復号する必要が無ければ、警告を無視して**続行**をクリックします。

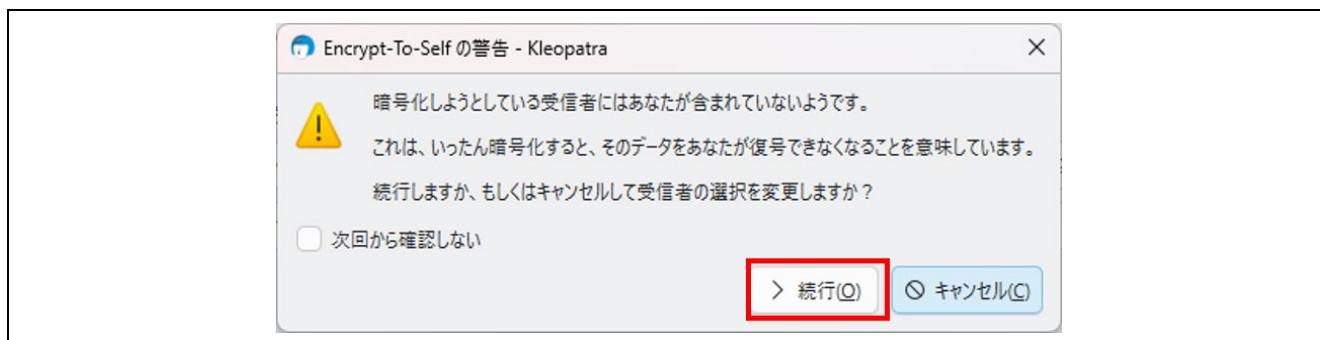


図4-12 UFPK 暗号化プロセスを開始

- ルネサスの公開鍵で暗号化された UFPK が生成され、鍵ファイルの拡張子に .gpg が追加されます。この場合、ファイル `ufpk_rx74m.key.gpg` が生成されます。終了をクリックします。



図4-13 暗号化された鍵が生成

4.2.2 W-UFPK の取得

- これで、ルネサス公開鍵で暗号化された UFPK を、ラッピングのためにルネサス Key Wrap サーバへ送信できます。ルネサス Key Wrap サーバのウェブページに戻り、Renesas RX を選択します。

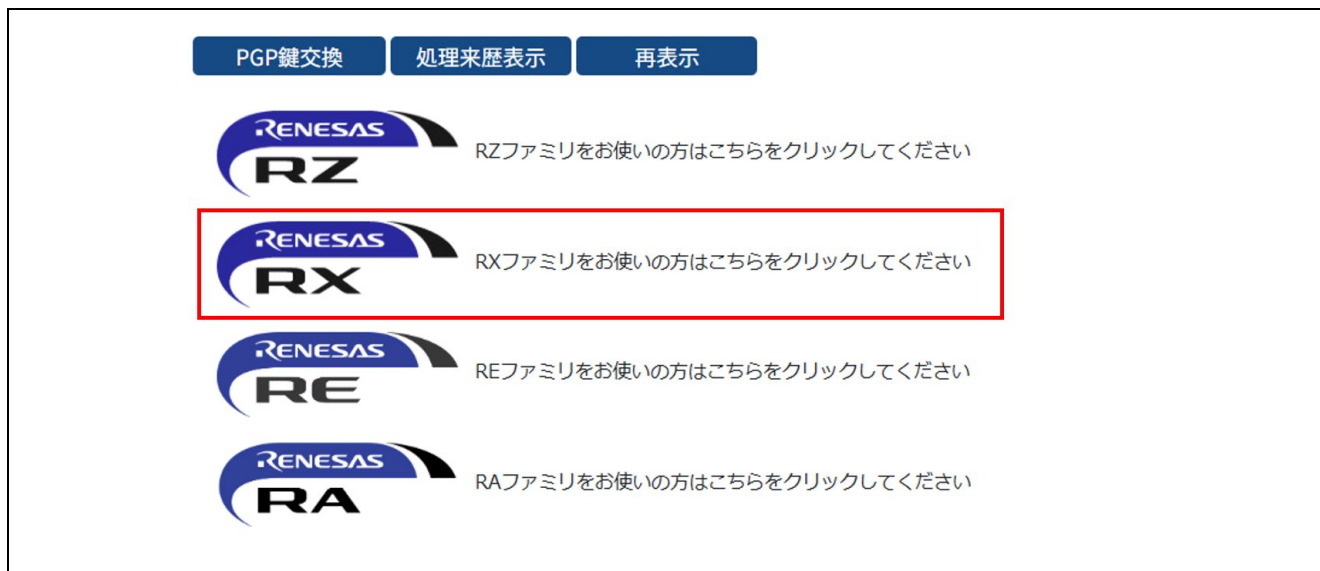


図4-14 MCU ファミリを選択

W-UFPK を生成するときは、正しい MCU ファミリとセキュリティエンジンモードを選択することが重要です。

注: Compatibility Mode 用に生成された W-UFPK は、DLM および Protected Mode での鍵注入に使用できず、その逆も同様です。

- RX74M Protected Mode のセキュアな鍵注入のサンプルプロジェクト用の W-UFPK を生成するには、**Renesas RX** ファミリを選択し、**RSIP Protected Mode** の項目の **RX74M Wrap UFPK** をクリックします。

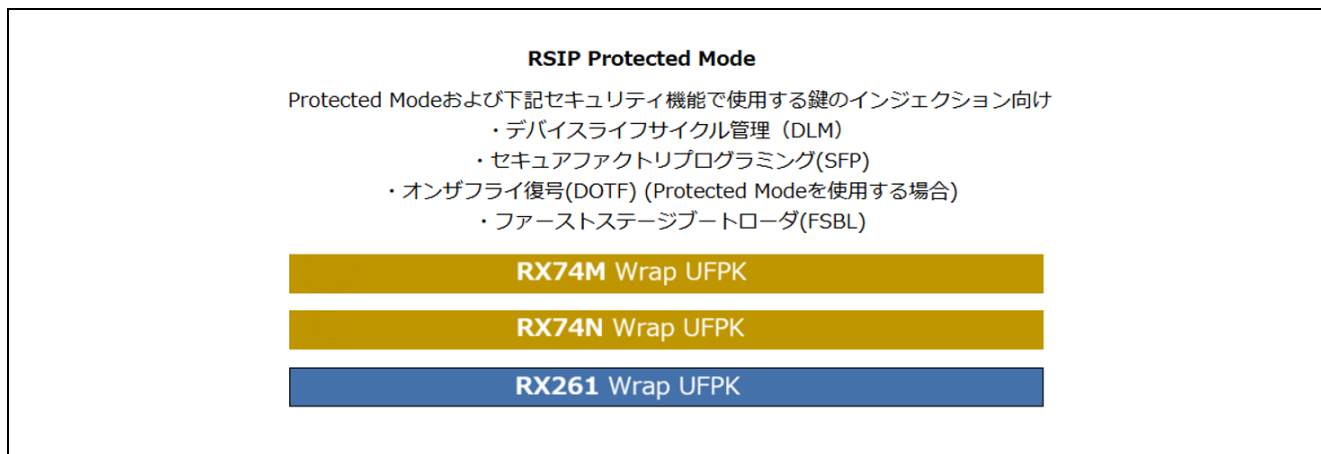


図4-15 RX74M グループ DLM and Protected Mode を選択

- RX74M Compatibility Mode のセキュアな鍵注入のサンプルプロジェクト用の W-UFPK を生成するには、**Renesas RX** ファミリを選択し、**RSIP Compatibility Mode** の項目の **RX74M Wrap UFPK** をクリックします。

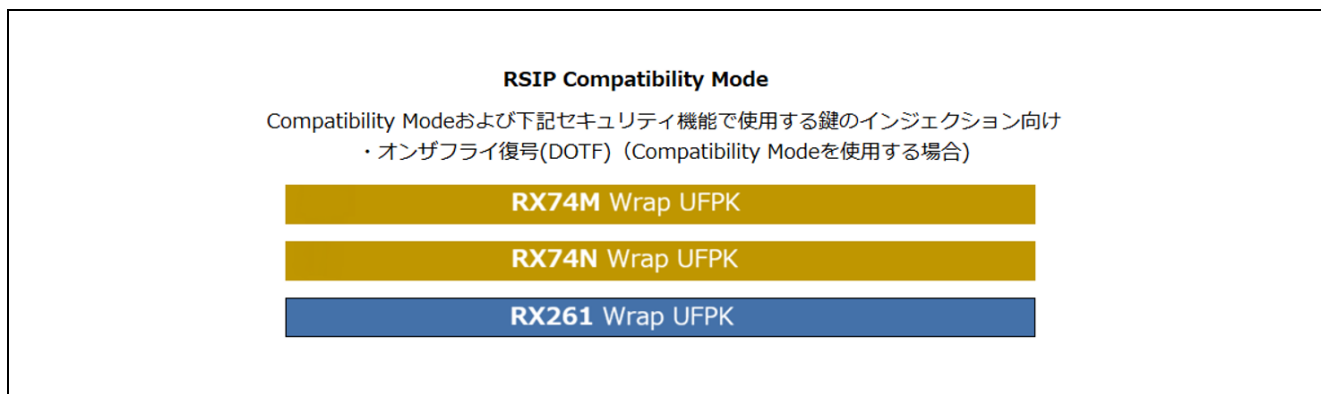


図4-16 RX74M グループ Compatibility Mode を選択

2. 次の画面で**製品用暗号化サービス**をクリックします。

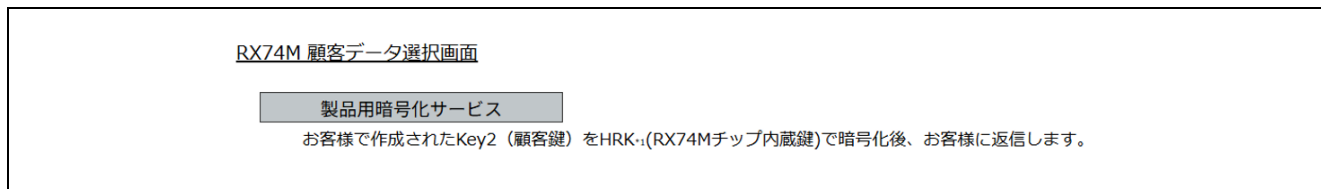


図4-17 製品用暗号化サービスを選択

3. **Reference** をクリックし、ルネサス公開鍵で暗号化された UFPK を選択します。例えば、以前に生成した rx74m_ufpk.key.gpg を選択し、**Open** をクリックします。ルネサス Key Wrap サービスの画面上に表示される説明では、**Key2** は UFPK を参照していることに注意してください。

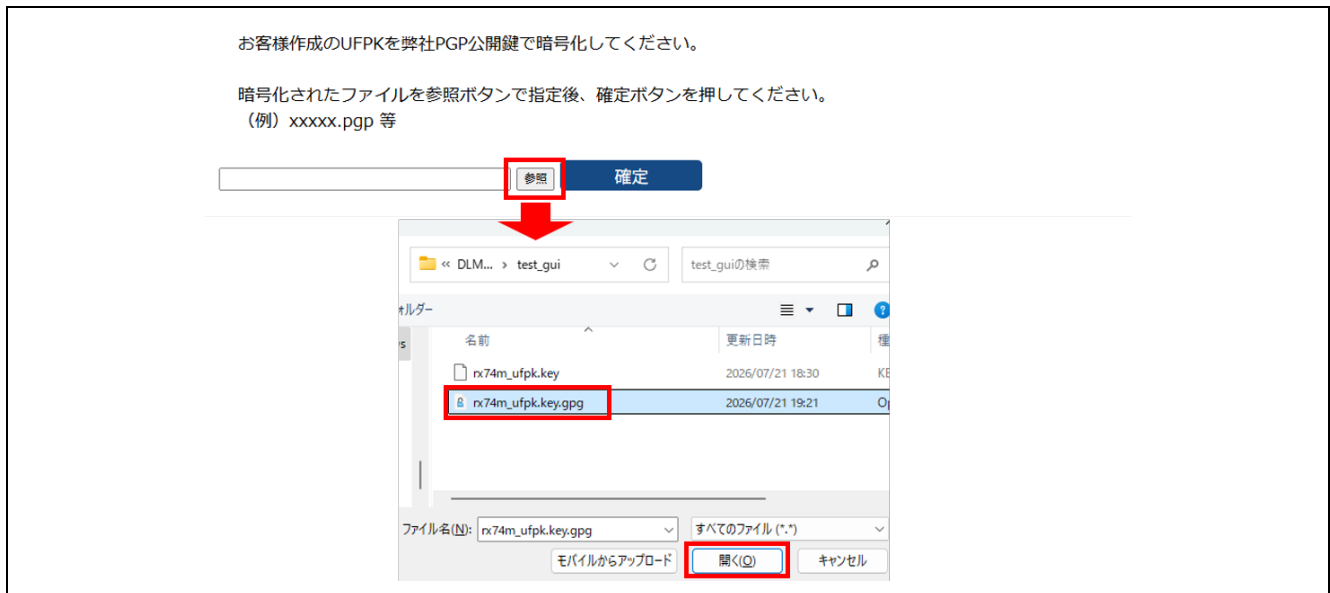


図4-18 PGP 暗号化された UFPK ファイルを選択

4. **確定** をクリックします。次のメッセージが出力されます。次に、**メニューに戻る** をクリックします。これで、ルネサス Key Wrap サービスからログアウトできます。

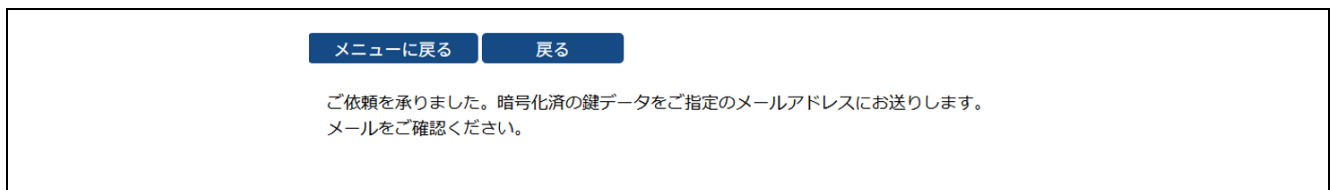


図4-19 ルネサス Key Wrap サーバのメインメニューに戻る

5. PGP 公開鍵で暗号化された W-UFPK は、通常約 1～2 分でメールに届きます。添付ファイルを保存します。

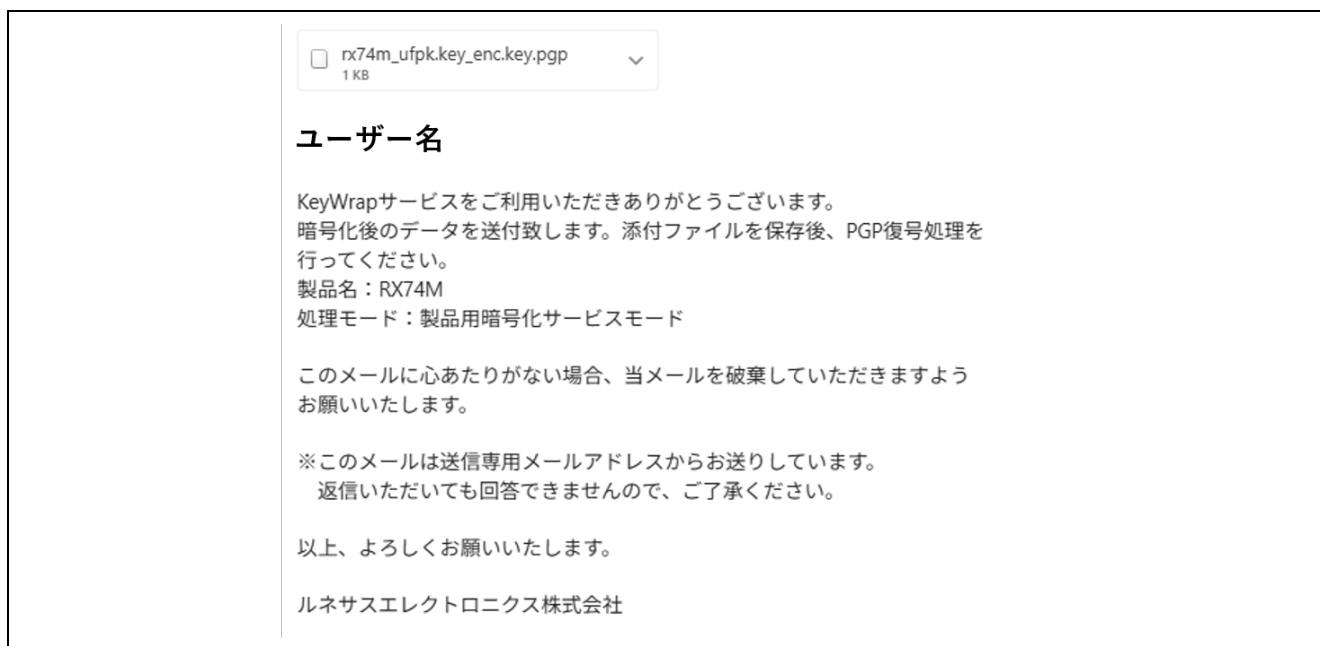


図4-20 W-UFPK をメールで受信

注意：W-UFPK は対象の MCU ファミリおよびセキュリティエンジンモードに依存します。Compatibility Mode 用に取得した W-UFPK は Protected Mode では使用できません。また、Protected Mode 用に取得した W-UFPK は Compatibility Mode では使用できません。

4.2.3 W-UFPK の復号

1. Kleopatra プログラムで、**復号/検証**をクリックし、W-UFPK ファイルを選択して、**開く**をクリックします。

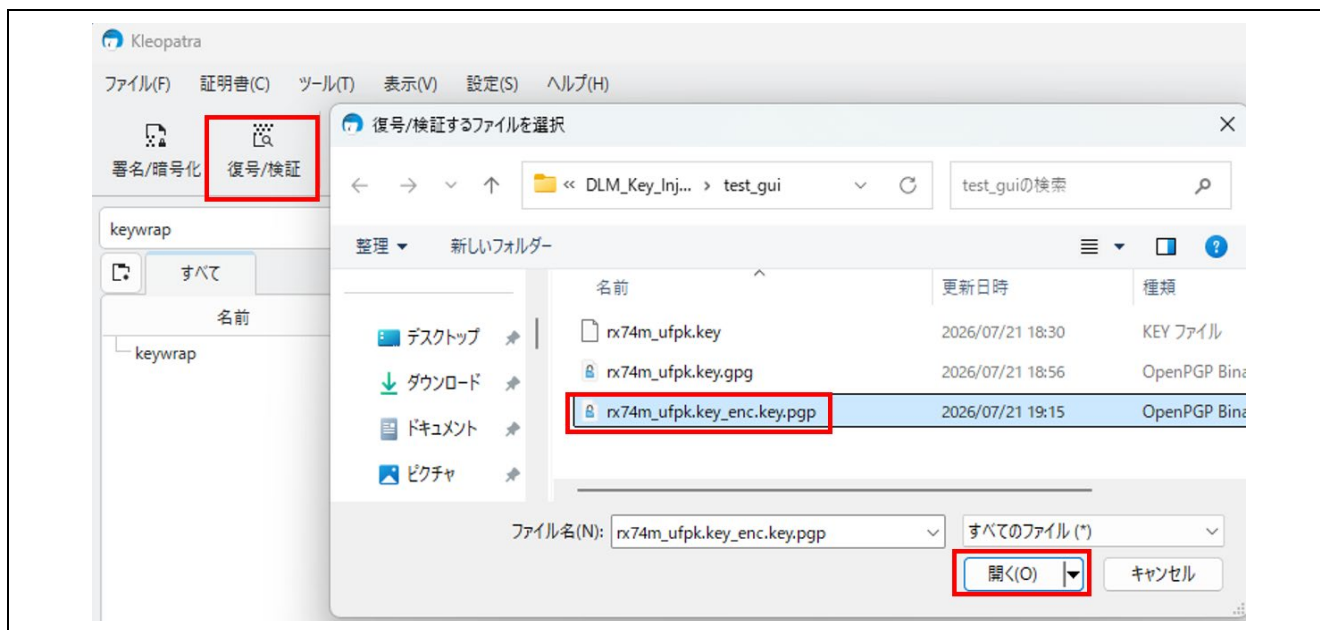


図4-21 W-UFPK の復号

2. プロンプトに従って PGP 秘密鍵パスフレーズを入力し、**OK** をクリックします。復号された W-UFPK は、指定されたフォルダに生成されます。

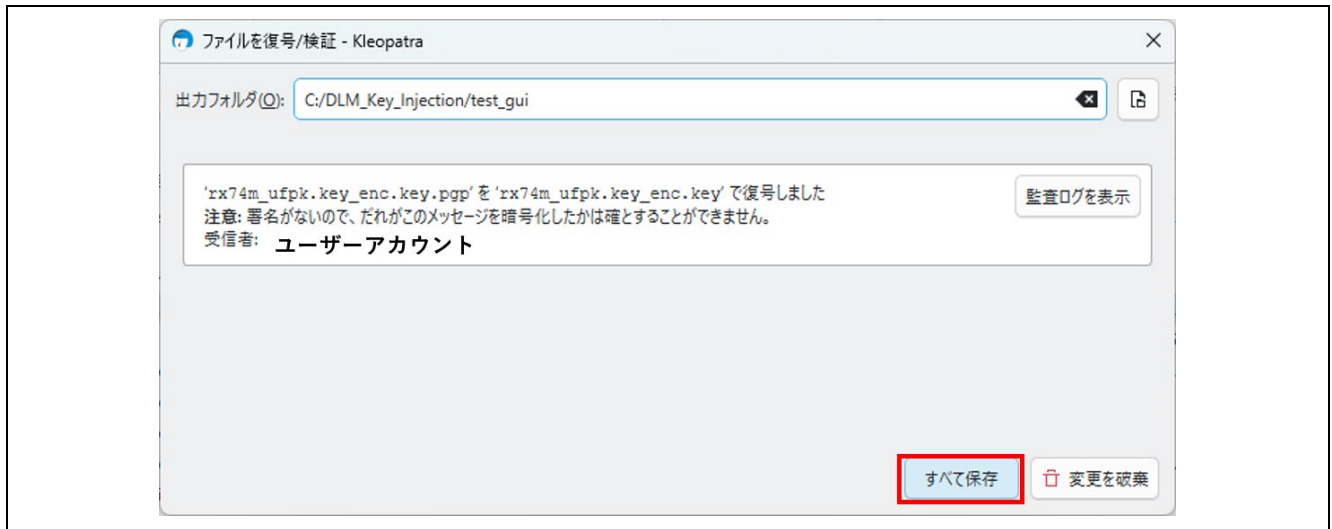


図4-22 PGP 暗号化された W-UFPK の復号

3. **すべて保存**をクリックして、復号した W-UFPK ファイル `rx74m_ufpk.key_enc.key` を保存します。後続の鍵注入データ生成では、UFPK ファイルと W-UFPK ファイルの両方を使用します。

5. RSIP Protected Mode のセキュアな鍵注入・鍵更新とサンプルプロジェクト

5.1 概要

本章では、RSIP-E50D Protected Mode におけるセキュアな鍵注入および鍵更新の手順と、これらの鍵を使用したサンプルプロジェクトについて説明します。

Protected Mode では、安全な形式の鍵のみを注入できます。初期ユーザ鍵および KUK の注入は、MCU のブートファームウェアのインタフェースを使用して実行します。また、フィールドでユーザ鍵を更新する場合は、事前に注入した KUK を使用して新しいユーザ鍵をラップし、FSP Crypto API から鍵更新を実行します。

本章では、以下の内容を順に説明します。

- SKMT GUI を使用した ECC 鍵ペアおよび KUK のラップ
- SKMT CLI を使用した ECC 鍵ペアおよび KUK のラップ
- MCU のブートファームウェアインタフェースを使用した初期ユーザ鍵および KUK の注入
- Protected Mode サンプルプロジェクトの実行
- カスタム鍵を使用する場合のソースファイル差し替え手順

このアプリケーションノートは、RX74M の RSIP-E50D Protected Mode 向けに ECC secp256r1 鍵ペア(秘密鍵と公開鍵)のユーザ鍵ラッピングの例を提供します。

このアプリケーションノートのサンプルプロジェクトでは、NIST テストベクタの値を例として使用しています。元のテストベクタを確認される場合は、以下のリンクからダウンロードしてください。

[Cryptographic Algorithm Validation Program | CSRC \(nist.gov\)](https://csrc.nist.gov/Cryptographic-Algorithm-Validation-Program)

5.2 SKMT GUI を使用した UFPK と W-UFPK による鍵のラップ

5.2.1 UFPK を使用した初期 ECC 鍵ペアのラップ

ECC 公開鍵をラップするには、SKMT GUI の概要タブで **RX Family, RSIP-E50D Security Functions, and Protected Mode** が選択されていることを確認します。次に、**鍵のラッピング**タブで、図5-1に示すように、**鍵の種類**を **ECC** と **secp256r1, public** として選択します。

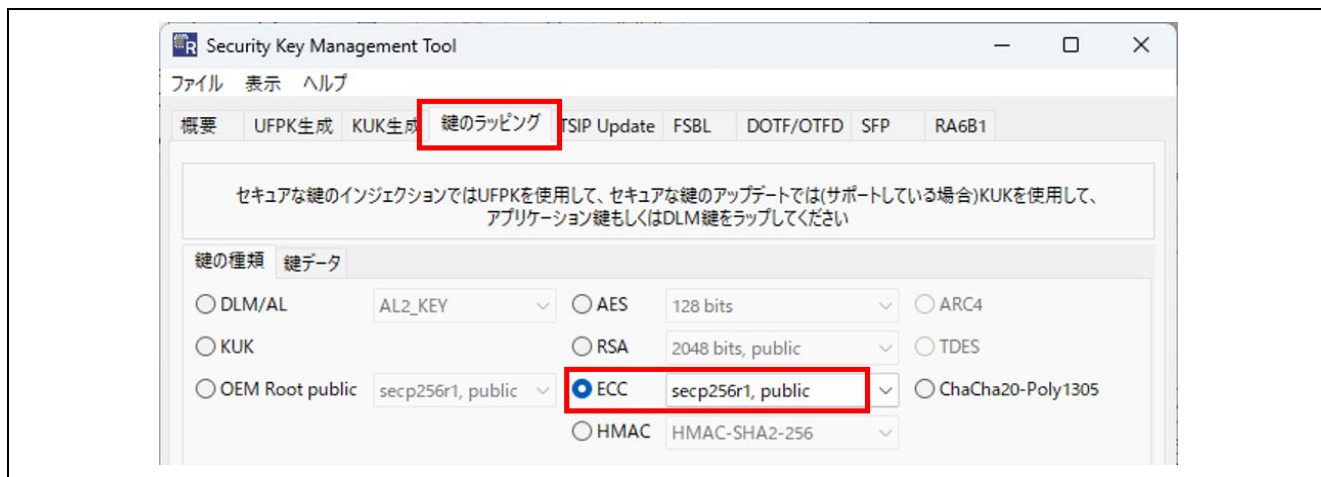


図5-1 secp256r1 公開鍵の選択

次に、**鍵データ**を構成します。**鍵データ**タブで**平文データ**を選択し、次に示すように**Qx**と**Qy**を指定します。鍵データは、GUI へのコピーおよび貼り付けを容易にするため、以下にも同じ内容を掲載しています。

Qx = 1ccbe91c075fc7f4f033bfa248db8fccd3565de94bbfb12f3c59ff46c271bf83

Qy = ce4014c68811f9a21a1fdb2c0e6113e06db7ca93b7404e78dc7ccd5ca89a4ca9



図5-2 ECC 公開鍵データの提供

次に、**鍵のラッピング**セクションで、対応する**参照**ボタンをクリックして、4.1.2および4.2で生成した**UFPK**と**W-UFPK**のペアを選択します。**IV**については、**乱数生成機能を使用する**を選択します。**フォーマット**オプションで**RFP**を選択します。次に**参照**ボタンをクリックし、出力フォルダを選択して、出力ファイルに名前を付けます。**ファイル**を生成するをクリックすると、*.rkey ファイルが生成されます。



図5-3 GUI を使用した ECC 公開鍵の RFP 鍵注入ファイルの生成

平文の ECC 鍵と UFPK は*.rkey ファイルに含まれていないため、鍵注入ファイルの内容を機密に転送できます。

続いて上記の手順と同様に、ラップされた公開鍵に対応する秘密鍵を UFPK でラップしていきます。鍵のラッピングタブで、図5-4に示すように、**鍵の種類**を ECC と **secp256r1, private** として選択します。

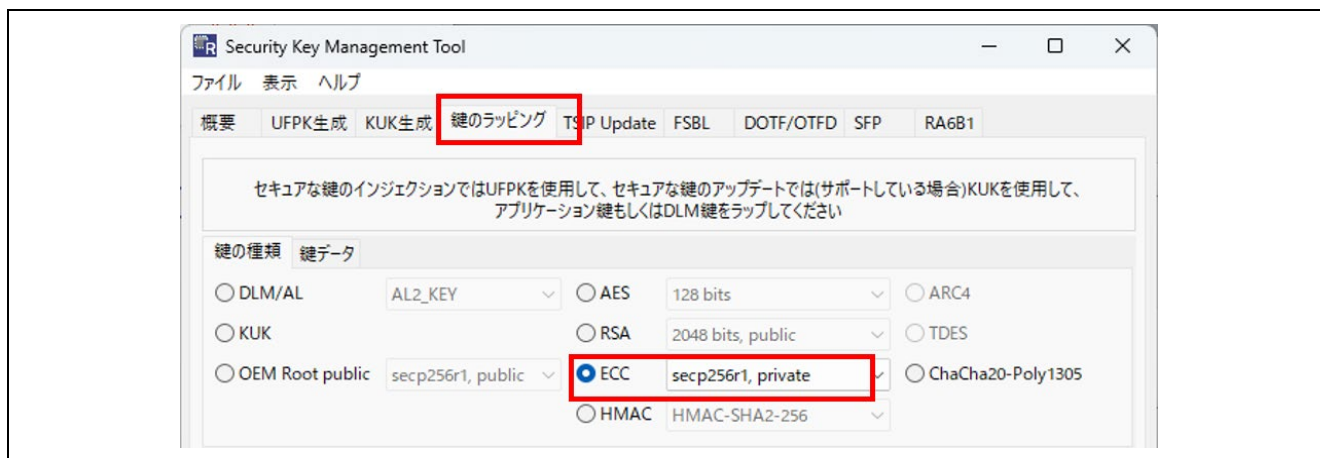


図5-4 secp256r1 秘密鍵の選択

次に、**鍵データ**を構成します。**鍵データ**タブで**平文データ**を選択し、次に示すように **d** を指定します。鍵データは、GUI へのコピーおよび貼り付けを容易にするため、以下にも同じ内容を掲載しています。

d = 519b423d715f8b581f4fa8ee59f4771a5b44c8130b4e3eacca54a56dda72b464



図5-5 ECC 秘密鍵データの提供

次に、ラッピング鍵セクションで、対応する参照ボタンをクリックして、4.1.2および4.2で生成した UFPK と W-UFPK のペアを選択します。IV については、乱数生成機能を使用する を選択します。フォーマットオプションで RFP を選択します。次に参照ボタンをクリックし、出力フォルダを選択して、出力ファイルに名前を付けます。ファイルを生成する をクリックすると、*.rkey ファイルが生成されます。

ラッピング鍵

☒ UFPK UFPKファイル: C:\DLM_Key_Injection\test_gui\rx74m_ufpk.key 参照...

W-UFPKファイル: C:\DLM_Key_Injection\test_gui\rx74m_ufpk.key_enc.key 参照...

☐ KUK KUKファイル: 参照...

IV

☒ 乱数生成機能を使用する

☐ 指定値を使用する (16バイト, ビッグエンディアン) 00112233445566778899AABBCCDDEEFF

出力

フォーマット: RFP ファイル: on%rx74m_protected_mode_key_info%ECC_Private_Key.rkey 参照...

エンディアン: Little ☐ データを追加出力する

アドレス: 10000 Key name:

ファイルを生成する

Command Line: skmt.exe /genkey /ufpk "C:\DLM_Key_Injection\test_gui\rx74m_ufpk.key" /wufpk "C:\DLM_Key_Injection\test_gui\rx74m_ufpk.key_enc.key" /mcu "RX-RSIP-E50D" /keytype "secp256r1-private" /key "519b423d715f8b581f4fa8ee59f4771a5b44c8130b4e3eace54e56dda72b464" /filetype "rfp" /output "C:\Secure_Key_Injection\rx74m_protected_mode_key_info%ECC_Private_Key.rkey"

Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info%ECC_Private_Key.rkey

UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F

W-UFPK: 00000000F77A67AB713D174EB9D7C6639DAADB1F77A67AB713D174EB9D7C6639DAADB1

IV: 01B315DD36CA2C044BAF8C021F3B4A8A

Encrypted key:

9889C7A14B27DC8ADAE3E66ABF6461A0F149FFE07DCE4872C75F385FE94FAAC6DB0DD31C846726B7E87EFEA5D5AAD7FA

OPERATION SUCCESSFUL

図5-6 GUIを使用した ECC 秘密鍵の RFP 鍵注入ファイルの生成

平文の ECC 鍵と UFPK は*.rkey ファイルに含まれていないため、鍵注入ファイルの内容を機密に転送できます。

5.2.2 UFPK による KUK のラップ

SKMT はサンプル KUK を生成するために使用できます。KUK ファイルを生成するには、**KUK 生成**タブに移動し、**指定値を使用する** を選択して以下の値を入力します。

000102030405060708090a0b0c0d0e0f000102030405060708090a0b0c0d0e0f

参照ボタンをクリックし、生成される鍵ファイルの出力されるフォルダと出力ファイル名を選択します。ここでは `kuk_for_new_key.key` と指定されています。次に、**KUK ファイルを生成する** をクリックすると、選択したフォルダに `kuk_for_new_key.key` ファイルが生成されます。

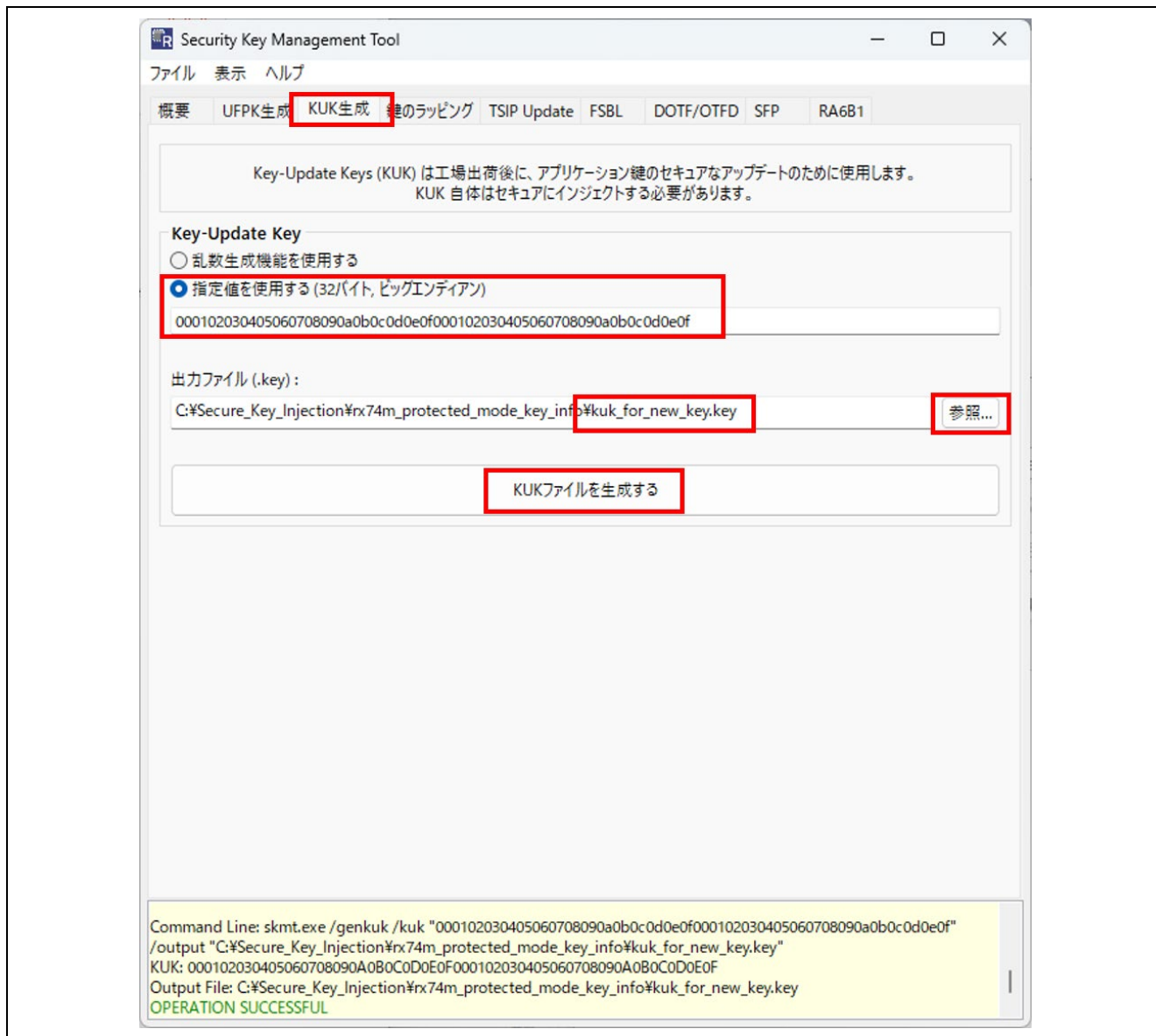


図5-7 RSIP-E50D のユーザ鍵暗号化に使用する KUK ファイルの生成

次に、KUK をラッピングして MCU に注入できるようにします。鍵のラッピングタブに移動し、鍵の種類タブから **KUK** を選択します。

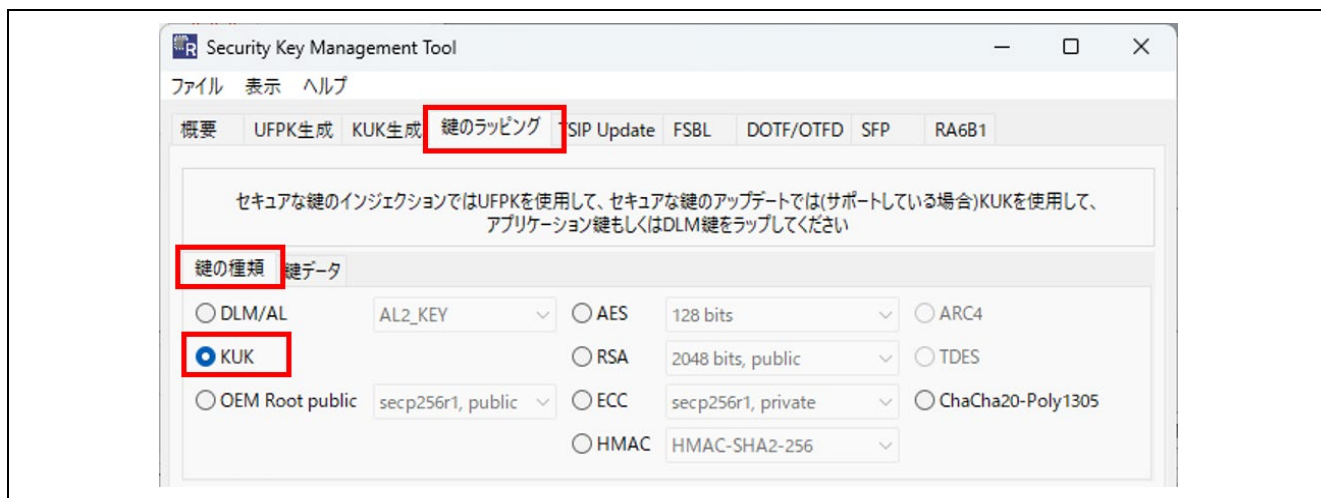


図5-8 ラップする KUK を選択

鍵データタブに移動し、ファイルオプションを選択して、図5-7で生成された `kuk_for_new_key.key` ファイルを参照します。

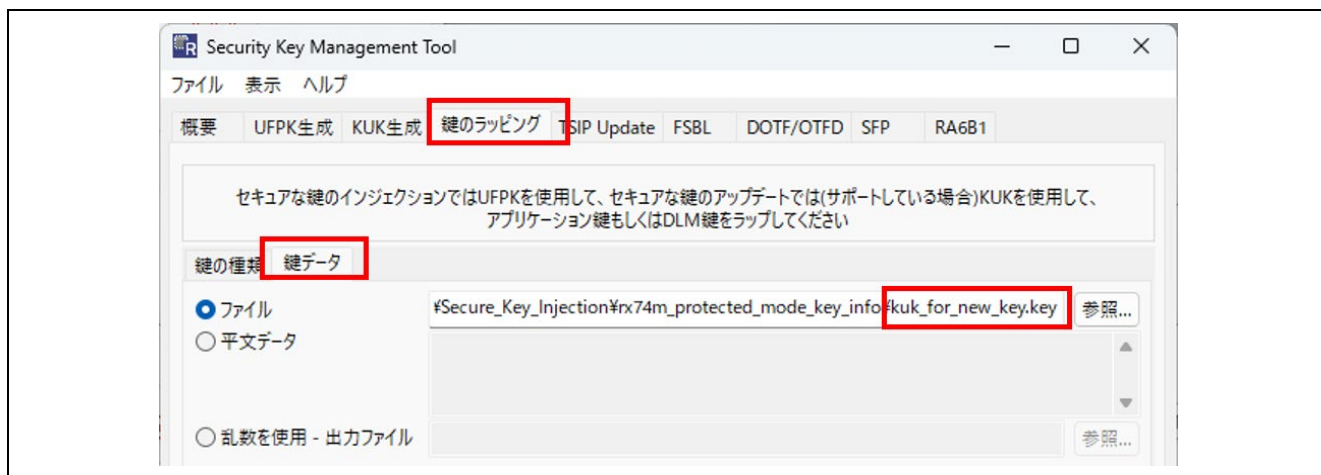


図5-9 KUK .key ファイルを提供

次に、ラッピング鍵セクションで、対応する参照ボタンをクリックして、4.1.2および4.2で生成した UFPK と W-UFPK のペアを選択します。IV については、乱数生成機能を使用する を選択します。フォーマットオプションで RFP を選択します。次に参照ボタンをクリックし、出力フォルダを選択して、出力ファイルに名前を付けます。

次に、ファイルを生成するボタンをクリックします。KUK.rkey ファイルが生成されます。

ラッピング鍵

☒ UFPK UFPKファイル: C:\DLM_Key_Injection\test_gui\rx74m_ufpk.key 参照...

W-UFPKファイル: C:\DLM_Key_Injection\test_gui\rx74m_ufpk.key_enc.key 参照...

☐ KUK KUKファイル: 参照...

IV

☒ 乱数生成機能を使用する

☐ 指定値を使用する (16バイト, ビッグエンディアン) 00112233445566778899AABBCCDDEEFF

出力

フォーマット: RFP ファイル: _Key_Injection\rx74m_protected_mode_key_info\KUK.rkey 参照...

エンディアン: Little ☐ データを追加出力する

アドレス: 10000 Key name:

ファイルを生成する

Command Line: skmt.exe /genkey /ufpk "C:\DLM_Key_Injection\test_gui\rx74m_ufpk.key" /wufpk "C:\DLM_Key_Injection\test_gui\rx74m_ufpk.key_enc.key" /mcu "RX-RSIP-E50D" /keytype "key-update-key" /key file="C:\Secure_Key_Injection\rx74m_protected_mode_key_info\KUK_for_new_key.key" /filetype "rfp" /output "C:\Secure_Key_Injection\rx74m_protected_mode_key_info\KUK.rkey"

Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\KUK.rkey

UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F

W-UFPK: 00000000F77A67AB713D174EB9D7C6639DAADB1F77A67AB713D174EB9D7C6639DAADB1

IV: A6CC8D23808B4F47CD4A72FF80225CEE

Encrypted key:
908C08C50447EBC91A9B84C714D4FEC1B97A571543F220498E6784CE524A306593DF5871155BD2A1FF3CEEFFCAA1854A

OPERATION SUCCESSFUL

図5-10 RSIP-E50D の GUI を使用した KUK 注入ファイルの生成

平文の KUK と UFPK は*.rkey ファイルに含まれていないため、鍵注入ファイルの内容を機密に転送できません。

5.2.3 新しいECC 鍵ペアを KUK でラップ

本項では、図5-7で生成された `kuk_for_new_key.key` を使用して、新しいECC 鍵ペアをラップします。

SKMT GUIの概要タブで **RX Family, RSIP-E50D Security Functions, and Protected Mode** を選択します。次に、**Wrap Key** タブに移動します。図5-1に示すように、**Key Type** を **ECC** と **secp256r1, public** として選択します。

鍵データ タブで **平文データ** を選択し、次に示すように **Qx** と **Qy** を指定します。鍵データはここに複製されているため、ユーザはそれをコピーして GUI に貼り付けることができます。

Qx = e266ddfdc12668db30d4ca3e8f7749432c416044f2d2b8c10bf3d4012aeffa8a

Qy = bfa86404a2e9ffe67d47c587ef7a97a7f456b863b4d02cfc6928973ab5b1cb39



図5-11 新しいECC 公開鍵データの提供

次に、ラッピング鍵セクションで、対応する参照ボタンをクリックして、5.2.2で生成された KUK を選択します。IV については、乱数生成機能を使用する を選択します。フォーマットオプションで C ソースを選択し、出力に new_ecc_public_key.c という名前を付けます。Key name を NEW_ECC_PUB に設定します。

最後に、ファイルを生成する をクリックします。new_ecc_public_key.c ファイルと new_ecc_public_key.h ファイルの両方が生成されます

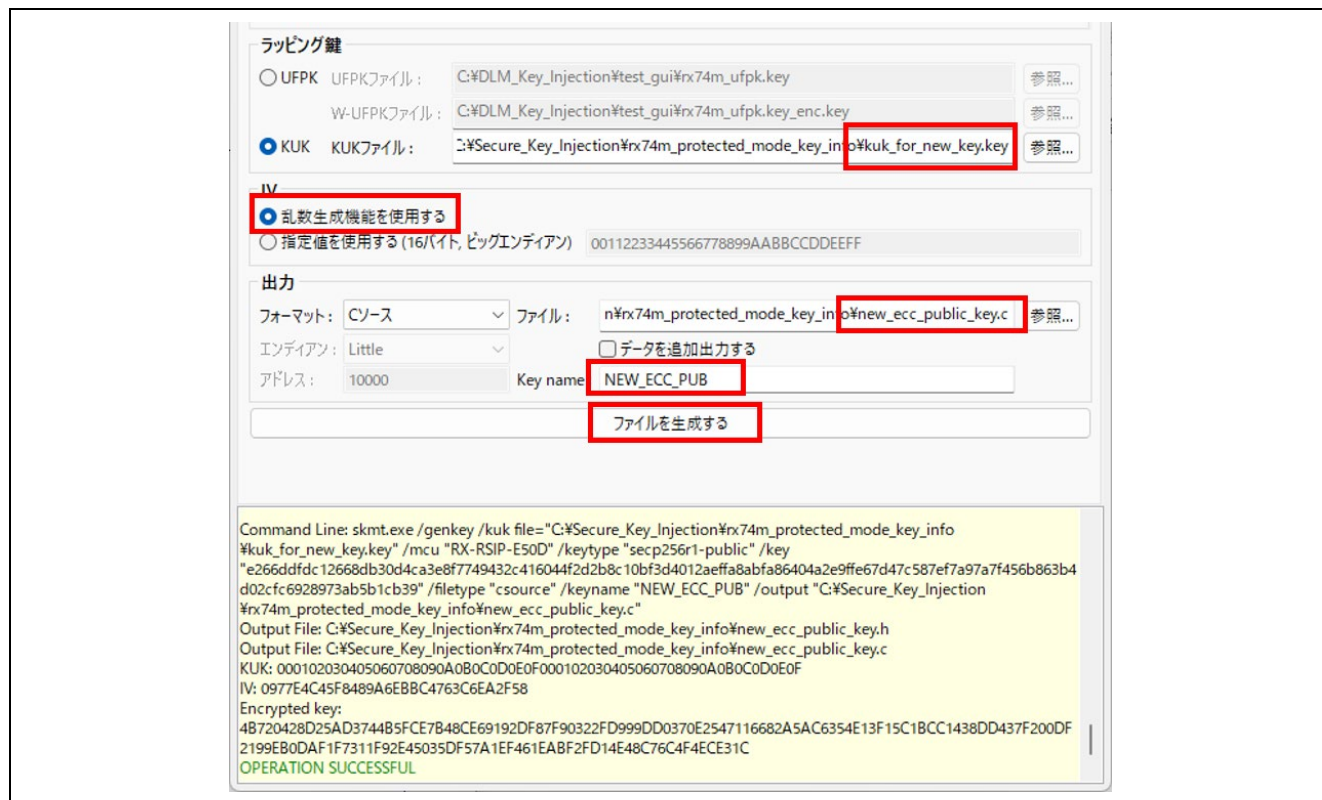


図5-12 KUK でラップされた ECC 公開鍵の生成

上記の手順に従い、新しいラップされた公開鍵を生成し、対応する秘密鍵を KUK でラップします。鍵のラッピングタブで、図5-4に示すように、鍵の種類を ECC と secp256r1, private として選択します。

次に、鍵データを構成します。鍵データタブで平文データを選択し、次に示すように d を指定します。鍵データは、GUI へのコピーおよび貼り付けを容易にするため、ここに複製されています。

d = 0f56db78ca460b055c500064824bed999a25aaf48ebb519ac201537b85479813

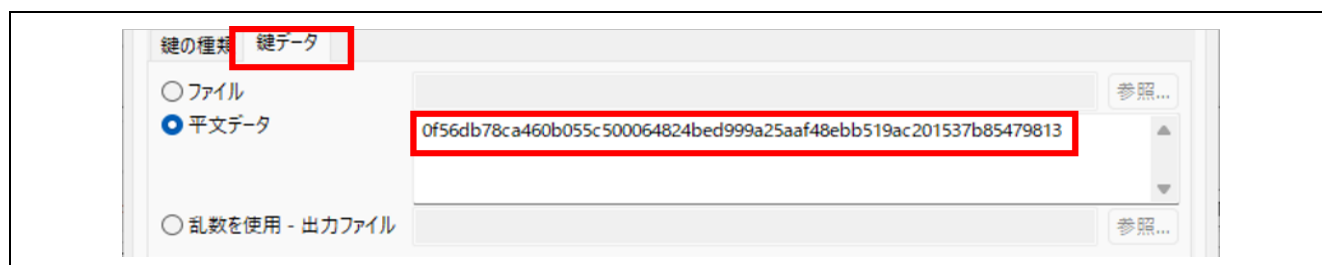


図5-13 新しい ECC 秘密鍵データの提供

次に、ラッピング鍵セクションで、対応する参照ボタンをクリックして、5.2.2で生成された KUK を選択します。IV については、乱数生成機能を使用する を選択します。フォーマットオプションで C ソースを選択し、出力に new_ecc_private_key.c という名前を付けます。Key name を NEW_ECC_PRI に設定します。

最後に、ファイルを生成する をクリックします。new_ecc_private_key.c ファイルと new_ecc_private_key.h ファイルの両方が生成されます。

ラッピング鍵

☐ UFPK UFPKファイル: C:\DLM_Key_Injection\test_gui\rx74m_ufrpk.key 参照...

W-UFPKファイル: C:\DLM_Key_Injection\test_gui\rx74m_ufrpk.key_enc.key 参照...

☒ KUK KUKファイル: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\kuk_for_new_key.key 参照...

IV

☒ 乱数生成機能を使用する

☐ 指定値を使用する (16バイト、ビッグエンディアン) 00112233445566778899AABBCCDDEEFF

出力

フォーマット: Cソース ▼ ファイル: rx74m_protected_mode_key_info\new_ecc_private_key.c 参照...

エンディアン: Little ▼ ☐ データを追加出力する

アドレス: 10000 Key name: NEW_ECC_PRI

ファイルを生成する

Command Line: skmt.exe /genkey /kuk file="C:\Secure_Key_Injection\rx74m_protected_mode_key_info\kuk_for_new_key.key" /mcu "RX-RSIP-E50D" /keytype "secp256r1-private" /key "0f56db78ca460b055c500064824bed999a25aaf48ebb519ac201537b85479813" /filetype "csource" /keyname "NEW_ECC_PRI" /output "C:\Secure_Key_Injection\rx74m_protected_mode_key_info\new_ecc_private_key.c"

Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\new_ecc_private_key.h

Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\new_ecc_private_key.c

KUK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F

IV: BFD21C7CDF39229C429C3BECBD4F8B4C

Encrypted key: F98CE9B8BF9CBF66E816F3A095FC1D089096FB173035E4AF59FC327C374DF8243A314EDD342BDC4F38D8D1116FA536CE9

OPERATION SUCCESSFUL

図5-14 KUK でラップされた ECC 秘密鍵の生成

5.3 SKMT CLI を使用した UFPK と W-UFPK による鍵のラップ

本節では、SKMT CLI を使用して上記の操作を実行する方法について説明します。

SKMT コマンドラインツール `skmt.exe` の `/genkey` コマンドは、セキュアな注入および更新の鍵を準備するために使用されます。このコマンドのオプションは次のとおりです。

- `/keytype` - この入力、鍵の種類を示す ASCII または 1 バイトの 16 進数入力パラメータのいずれかを取ることができます。
- `/ufpk` - UFPK。
- `/wufpk` - ルネサスの HRK でラップされた UFPK。
- `/kuk` - セキュアな鍵更新用の KUK。
- `/mcu` - ターゲット MCU およびセキュリティエンジン。
- `/output` - コマンドの出力。

各パラメータの有効な値を含む、これらのコマンドの詳細については、SKMT のユーザーズマニュアルを参照してください。

このアプリケーションノートでは、ECC `secp256r1` 鍵ペアを使用して、セキュアな鍵注入および鍵更新のプロセスを示します。

これらの例では、以前に生成した UFPK と W-UFPK を使用します。

5.3.1 UFPK を使用した初期 ECC 鍵ペアのラップ

本項では、セキュアな鍵注入用の ECC 鍵ペアを準備する例として、図5-2と図5-5と同じ ECC 鍵ペアを使用します。

前(4.1.3)に開いたコマンドプロンプトウィンドウで、次のコマンドを使用して ECC 公開鍵注入ファイル (ECC_Public_Key_CLI.rkey) を生成します。コマンドの生成方法の詳細については、SKMT のユーザズマニュアルを参照してください。

```
skmt.exe /genkey /ufpk
file="C:\Secure_Key_Injection\rx74m_protected_mode_key_info\rx74m_ufpk.key"
/wufpk
file="C:\Secure_Key_Injection\rx74m_protected_mode_key_info\rx74m_ufpk.key_enc.k
ey" /mcu "RX-RSIP-E50D" /keytype "secp256r1-public" /key
"1ccb91c075fc7f4f033bfa248db8fccc3565de94bbfb12f3c59ff46c271bf83ce4014c68811f9a
21a1fdb2c0e6113e06db7ca93b7404e78dc7ccd5ca89a4ca9" /filetype "rfp" /output
"C:\Secure_Key_Injection\rx74m_protected_mode_key_info\ECC_Public_Key_CLI.rkey"
```

この例では、次のようになります。

- 1ccb91c075fc7f4f033bfa248db8fccc3565de94bbfb12f3c59ff46c271bf83ce4014c68811f9a21a1fdb2c0e6113e06db7ca93b7404e78dc7ccd5ca89a4ca9 は、NIST テストベクタの ECC 公開鍵です。
- 鍵の種類 "secp256r1-public" を指定しました。
- /mcu オプションには "RX-RSIP-E50D" を使用します。
- ランダムに生成された IV を使用しています。IV は、暗号化インスタンスごとに更新されます。
- この例では、UFPK と W-UFPK は C:\Secure_Key_Injection\rx74m_protected_mode_key_info に格納されています。
- コマンドオプション /output は、出力ファイルの場所と名前を定義します。

```
Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\ECC_Public_Key_CLI.rkey
UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F
W-UFPK: 1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF12345678
IV: 0273B7277508F33491F2BA569B092535
Encrypted key:
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
```

図5-15 CLI を使用した ECC 公開鍵注入ファイルの生成

次のコマンドを使用して ECC 秘密鍵注入ファイル(ECC_Private_Key_CLI.rkey)を生成します。

```
skmt.exe /genkey /ufpk  
file="C:\Secure_Key_Injection\rx74m_protected_mode_key_info\rx74m_ufpk.key"  
/wufpk  
file="C:\Secure_Key_Injection\rx74m_protected_mode_key_info\rx74m_ufpk.key_enc.k  
ey" /mcu "RX-RSIP-E50D" /keytype "secp256r1-private" /key  
"519b423d715f8b581f4fa8ee59f4771a5b44c8130b4e3eacca54a56dda72b464" /filetype  
"rfp" /output  
"C:\Secure_Key_Injection\rx74m_protected_mode_key_info\ECC_Private_Key_CLI.rkey"
```

この例では、次のようになります。

- 519b423d715f8b581f4fa8ee59f4771a5b44c8130b4e3eacca54a56dda72b464 は、NIST テストベクタの ECC 秘密鍵です。
- 鍵の種類"secp256r1-private"を指定しました。
- /mcu オプションには"RX-RSIP-E50D"を使用します。
- ランダムに生成された IV を使用しています。IV は、暗号化インスタンスごとに更新されます。
- コマンドオプション/output は、出力ファイルの場所と名前を定義します。

```
Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\ECC_Private_Key_CLI.rkey  
UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F  
W-UFPK: 1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF12345678  
IV: 0273B7277508F33491F2BA569B092535  
Encrypted key:  
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
```

図5-16 CLI を使用した ECC 秘密鍵注入ファイルの生成

5.3.2 UFPK を使用した KUK の生成とラップ

SKMT を使用して、KUK の鍵ファイルを生成できます。これは、次のコマンドで行います。

```
skmt.exe /genkuk /kuk  
"000102030405060708090a0b0c0d0e0f000102030405060708090a0b0c0d0e0f" /output  
"C:\Secure_Key_Injection\rx74m_protected_mode_key_info\kuk_for_new_key_cli.key"
```

この例では、次のようになります。

- 鍵ファイルの完全なファイルパスを指定しました。
- 5.2.2で使用されたものと同じ KUK を使用する必要があります。

```
KUK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F  
Output File:  
C:\Secure_Key_Injection\rx74m_protected_mode_key_info\kuk_for_new_key_cli.key
```

図5-17 KUK ファイルの生成

生成された鍵ファイル `kuk_for_new_key_cli.key` に KUK が含まれるようになりました。この鍵ファイルを保持して、セキュアな鍵更新のために新しいユーザ鍵をラップするために使用します。

セキュアな鍵更新を有効にするには、まず KUK を安全に注入する必要があります。SKMT を使用して KUK を UFPK でラップし、次のコマンドで RFP で使用する鍵注入ファイルを生成します。

```
skmt.exe /genkey /ufpk  
file="C:\Secure_Key_Injection\rx74m_protected_mode_key_info\rx74m_ufpk.key"  
/wufpk  
file="C:\Secure_Key_Injection\rx74m_protected_mode_key_info\rx74m_ufpk.key_enc.k  
ey" /mcu "RX-RSIP-E50D" /keytype "key-update-key" /key  
file="C:\Secure_Key_Injection\rx74m_protected_mode_key_info\kuk_for_new_key_cli.  
key" /filetype "rfp" /output  
"C:\Secure_Key_Injection\rx74m_protected_mode_key_info\KUK_CLI.rkey"
```

この例では、次のようになります。

- 上記で生成した KUK ファイルを使用しています。
- 鍵の種類 "key-update-key" を指定しました。
- ランダムに生成された IV を使用しています。IV は、このコマンドが実行されるたびに変更されます。
- この例では、鍵ファイル(`KUK_CLI.rkey`)の完全なファイルパスを指定しました。

```
Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\KUK_CLI.rkey  
UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F  
W-UFPK: 1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF12345678  
IV: 1234567890ABCDEF1234567890ABCDEF  
Encrypted key:  
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
```

図5-18 CLI を使用した KUK 注入ファイルの生成

生成された鍵ファイル `KUK_CLI.rkey` には、ラップされた KUK と W-UFPK が含まれるようになりました。平文の KUK と UFPK は `*.rkey` ファイルに含まれていないため、鍵注入ファイルの内容を機密に転送できます。

5.3.3 新しいECC 鍵ペアを KUK でラップ

次のコマンドを使用して、図5-11に示す新しい ECC 公開鍵をラップします。

```
skmt.exe /genkey /kuk  
file="C:\Secure_Key_Injection\rx74m_protected_mode_key_info\kuk_for_new_key_cli.  
key" /mcu "RX-RSIP-E50D" /keytype "secp256r1-public" /key  
"e266ddfdcl2668db30d4ca3e8f7749432c416044f2d2b8c10bf3d4012aeffa8abfa86404a2e9ffe  
67d47c587ef7a97a7f456b863b4d02cfc6928973ab5b1cb39" /filetype "csource" /keyname  
"NEW_ECC_PUB" /output  
"C:\Secure_Key_Injection\rx74m_protected_mode_key_info\new_ecc_public_key_cli.c"
```

この例では、次のようになります。

- e266ddfdcl2668db30d4ca3e8f7749432c416044f2d2b8c10bf3d4012aeffa8abfa86404a2e9ffe67d47c587ef7a97a7f456b863b4d02cfc6928973ab5b1cb39 は、NIST テストベクタの ECC 公開鍵です。
- 鍵の種類"secp256r1-public"は、SKMT のユーザーズマニュアルで指定されているオプションの 1 つです。
- /mcu オプションには"RX-RSIP-E50D"を使用します。
- ランダムに生成された IV を使用しています。IV は、このコマンドが実行されるたびに変更されます。
- コマンドオプション/output は、出力ファイルの場所と名前を定義します。
- /keyname を使用して、ソフトウェアプロジェクト内でユニークの識別可能な鍵構造体名を生成します。これにより、複数の新規ユーザ鍵セットを生成する場合の混乱を解消します。このオプションが指定されていない場合は、encrypted_user_key_data という名前の鍵構造体が生成されます。
- 生成された new_ecc_public_key_cli.c ファイルと new_ecc_public_key_cli.h ファイルには、鍵構造体出力情報が含まれています。これは、サンプルプロジェクトに含まれるファイルで確認することができます。

```
Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\new_ecc_public_key_cli.h  
Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\new_ecc_public_key_cli.c  
KUK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F  
IV: 36E763D5A82924B4888732D50C93B602  
Encrypted key:  
9B0A7F8C91C038704A4F2C758EAC3DDD1372B4DC6AA4F22667D7D0E41218A1DEDBB8337E557B59B91100225BC8BBE280  
72214FF3C729D953AEFA9E997C3989967C831DC6501E9528715ADA30FA0D0402
```

図5-19 KUK を使用した新しい ECC 公開鍵の暗号化

次のコマンドを使用して、図5-13に示す新しい ECC 秘密鍵をラップします。

```
skmt.exe /genkey /kuk  
file="C:\Secure_Key_Injection\rx74m_protected_mode_key_info\kuk_for_new_key_cli.  
key" /mcu "RX-RSIP-E50D" /keytype "secp256r1-private" /key  
"0f56db78ca460b055c500064824bed999a25aaf48ebb519ac201537b85479813" /filetype  
"csource" /keyname "NEW_ECC_PRI" /output  
"C:\Secure_Key_Injection\rx74m_protected_mode_key_info\new_ecc_private_key_cli.c  
"
```

この例では、次のようになります。

- 0f56db78ca460b055c500064824bed999a25aaf48ebb519ac201537b85479813 は、NIST テストベクタの ECC 秘密鍵です。
- 鍵の種類"secp256r1-private"は、SKMT のユーザーズマニュアルで指定されているオプションの 1 つです。
- /mcu オプションには"RX-RSIP-E50D"を使用します。
- ランダムに生成された IV を使用しています。IV は、このコマンドが実行されるたびに変更されます。
- コマンドオプション/output は、出力ファイルの場所と名前を定義します。
- /keyname を使用して、ソフトウェアプロジェクト内でユニークの識別可能な鍵構造体名を生成します。これにより、複数の新規ユーザ鍵セットを生成する場合の混乱を解消します。このオプションが指定されていない場合は、鍵構造体に対して encrypted_user_key_data という名前の鍵構造体が生成されます。
- 生成された new_ecc_private_key_cli.c ファイルと new_ecc_private_key_cli.h ファイルには、データ構造に出力情報が含まれています。これは、含まれているサンプルプロジェクトで示されています。

```
Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\new_ecc_private_key_cli.h  
Output File: C:\Secure_Key_Injection\rx74m_protected_mode_key_info\new_ecc_private_key_cli.c  
KUK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F  
IV: 36E763D5A82924B4888732D50C93B602  
Encrypted key:  
9B0A7F8C91C038704A4F2C758EAC3DDD1372B4DC6AA4F22667D7D0E41218A1DEDBB8337E557B59B91100225BC8BBE280
```

図5-20 KUK を使用した新しい ECC 秘密鍵の暗号化

5.4 MCU のブートファームウェアのインタフェースによるセキュアな鍵注入

本節に従って、5.2または5.3で準備した ECC 公開鍵、ECC 秘密鍵、および KUK を注入します。この機能は、RSIP(Protected Mode)セキュリティエンジンを組み込んだ RX ファミリ MCU でサポートされています。

5.4.1 ハードウェアの設定

EK-RX74M 評価ボードを使用する場合、セットアップは以下のとおりです。

- EK-RX74M の SW6-Pin1 および SW6-Pin2 を OFF に設定して、MCU 動作モードを Debug on-board モードにします。詳細は、EK-RX74M ユーザーズマニュアルを参照してください。
- USB Type-C®ケーブルを使用して、EK-RX74M の J8 を開発用 PC に接続し、電源供給とデバッグ機能を利用できるようにします。

MRAM 全体を消去し、MCU が OEM デバイスライフサイクル状態であることを確認します。これは、次に示すように、RFP を使用して実行できます。

1. rfp_resources_rx74m.zip を解凍します。
2. RFP GUI を起動します。
3. 図5-21に示すように、**ファイル-> プロジェクトを開く** を選択し、rx74m_secure_key_inject.rpj を選択します。
4. 図5-21に示すように、**ターゲットデバイス -> デバイスを初期化する** を選択します。

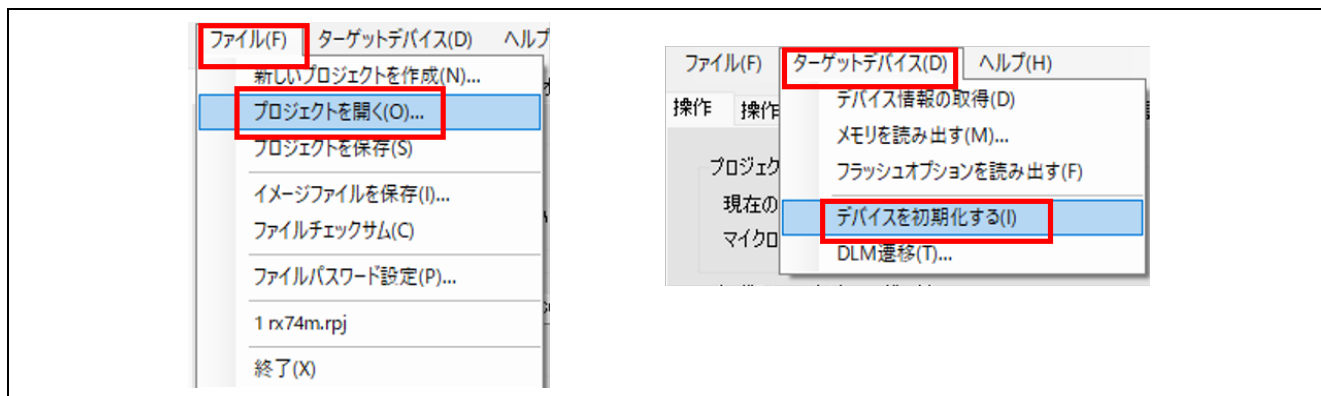


図5-21 RFP プロジェクトを開きデバイスを初期化

初期化に成功すると、図5-22のような次のメッセージが出力されます。

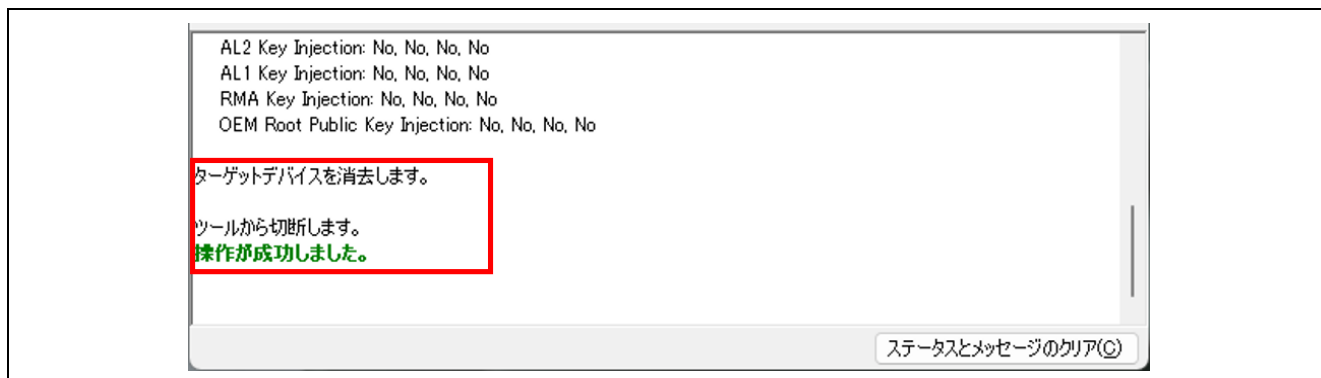


図5-22 RX74M 初期化

5.4.2 初期ユーザ鍵と KUK の注入

ハードウェアを設定した後、ボードの電源を入れ直し、以下の手順に従って、ECC 鍵ペアと KUK を注入します。本項では、GUI から生成された注入鍵のセットを使用します。

この手順を簡単に実施できるように、サンプルプロジェクトに対応する .rkey ファイルは rfp_resources_rx74m.zip ファイルに含まれています。本項に含まれる画面キャプチャは、デモンストレーション目的でこれらのファイルを使用しています。異なる鍵を使用する場合は、対応する .rkey ファイルをそれらの鍵に一致するように更新する必要があります。

次の手順では、EK-RX74M サンプルプロジェクトで鍵を注入する方法について説明します。

操作タブで、**ファイルの追加と削除**をクリックします。次に、**Add Files** をクリックし、ECC 秘密鍵を含む .rkey ファイルを追加します、この例では ECC_Private_Key.rkey(図5-6)です。**Address** プロパティを、特定のアプリケーションに適用できるデータ MRAM またはコード MRAM アドレスに設定します。この例では、ECC 秘密鍵がデータ MRAM の 0xFFFD0000 に注入されます。

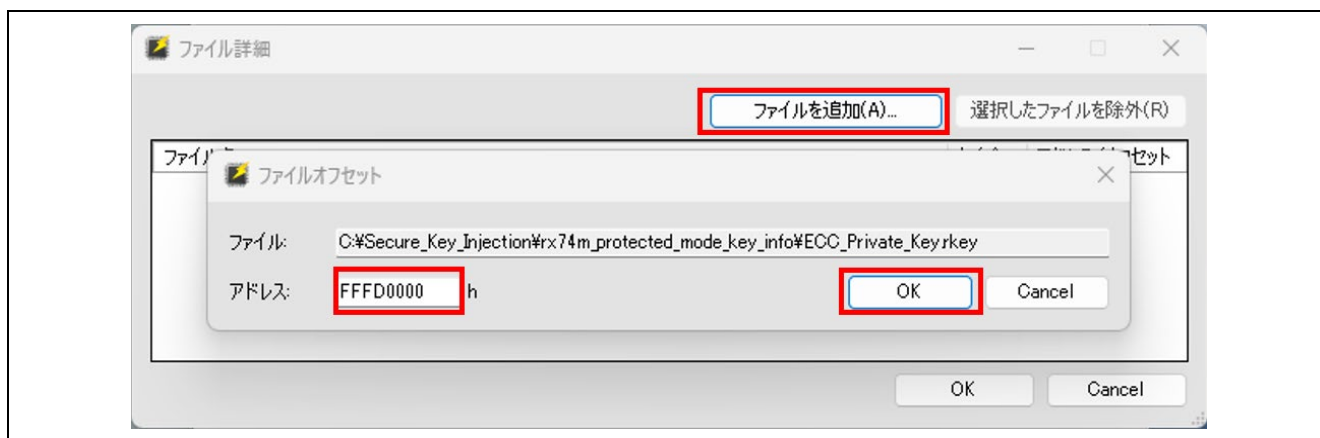


図5-23 RFP 構成に ECC_Private_Key.rkey を追加

OK をクリックすると、ECC_Private_Key.rkey ファイルが対応するロードアドレスに構成されます。

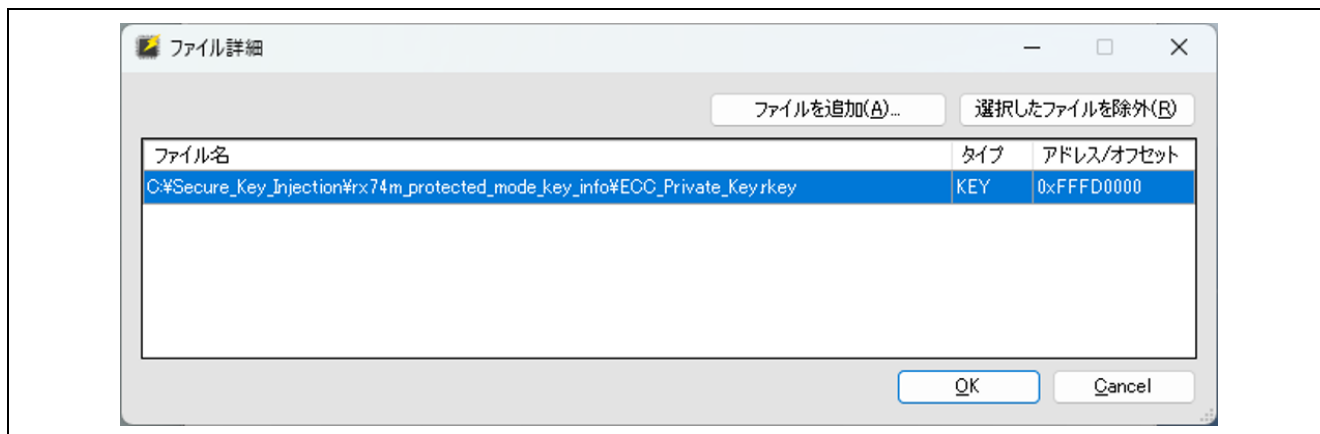


図5-24 ECC_Private_Key.rkey が RFP 構成に追加

再度**ファイルを追加**をクリックし、ECC_Public_Key.rkeyを追加します。ECC_Public_Key.rkey(図5-3)を参照します。アドレスプロパティを、特定のアプリケーションに適用できるデータ MRAM またはコード MRAM アドレスに設定します。この例では、ECC 公開鍵がデータ MRAM の 0xFFFFD0100 に注入されます。

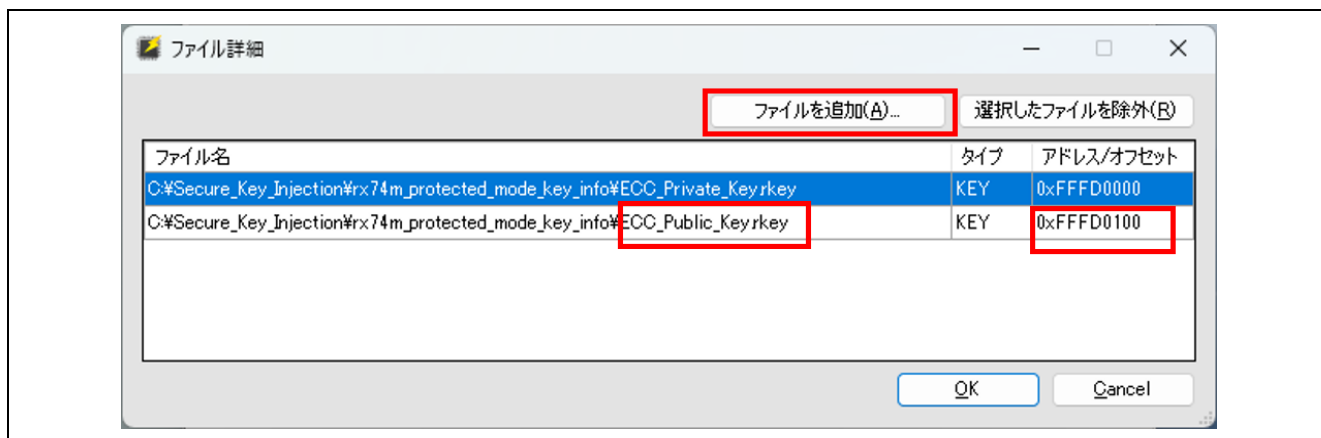


図5-25 ECC 公開鍵の選択と注入アドレスの設定

再度**ファイルを追加**をクリックし、KUK.rkeyを追加します。KUK.rkey(図5-10)を参照します。アドレスプロパティを、特定のアプリケーションに適用できるデータ MRAM またはコード MRAM アドレスに設定します。この例では、KUK がコード MRAM の 0xFFFFD0200 に注入されます。

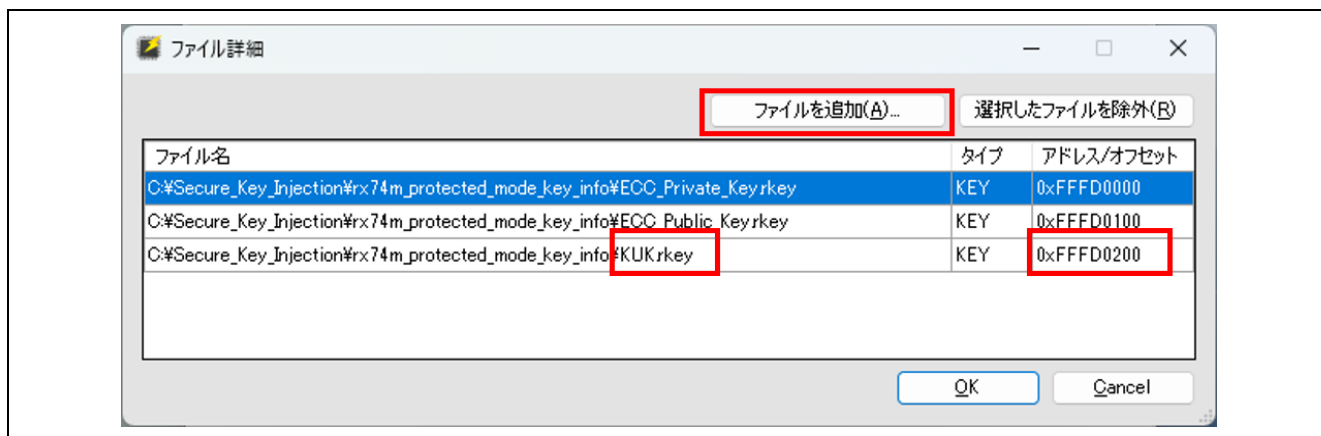


図5-26 KUK の選択と注入アドレスの設定

OK をクリックし、**操作設定**に移動します。消去、書き込み、およびベリファイが選択されていることに注意してください。

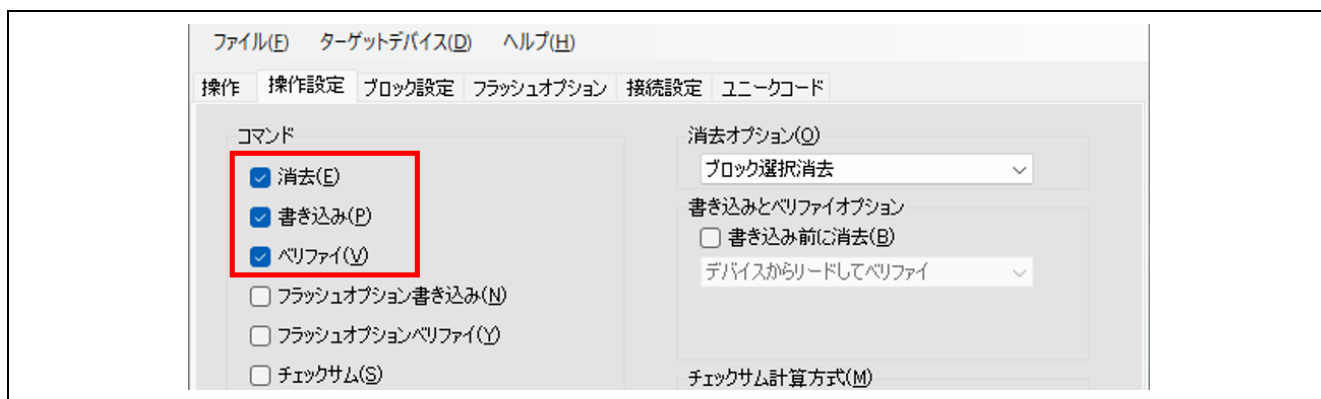


図5-27 MRAM 消去、プログラム、および検証を実行する選択

- **ブロック設定タブ**を参照し、消去で MRAM 領域全体が選択されていることに注意してください。

Region	Start	End	Size	Select
R5K574MCB4AE				☒
Code MRAM 1	0xFFF00000	0xFFFFFFF	1.0 M	☒
Code MRAM 2	0x7FF00000	0x7FFFFFF	1.0 M	☒
Data MRAM 1	0x86900000	0x8690BFFF	48 K	☒
Data MRAM 2	0x06900000	0x0690BFFF	48 K	☒

図5-28 MRAM 領域全体が消去対象として選択

- **操作タブ**を参照します。**スタート**をクリックして、ECC 秘密鍵、ECC 公開鍵、および KUK を注入します。選択された MRAM アドレスにおいて、以下の出力メッセージと同様のメッセージが表示されれば、注入は成功です。

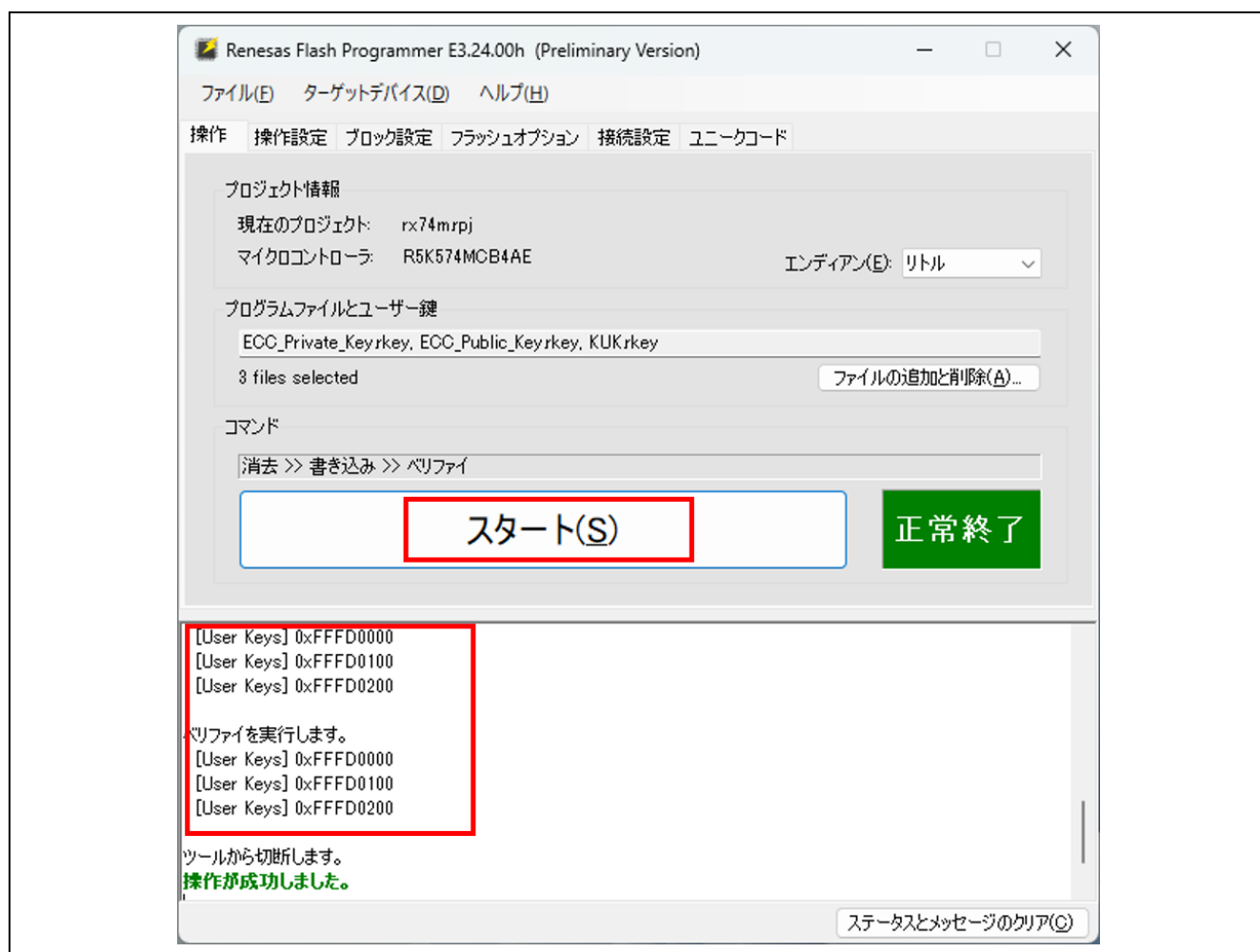


図5-29 セキュアな鍵注入に成功

このサンプルコードでは、鍵注入の方法のみを示しているため、アプリケーションはプログラムされていません。製造工程においては、アプリケーションとユーザ鍵を一緒にプログラムすることができます。この操作は、RFP のコマンドライン機能を使用して実行することもできます。

5.5 Protected Mode サンプルプロジェクト

本章では、ラップされた ECC 鍵ペア注入および更新の例を用いて、RSIP Protected Mode を紹介します。サンプルプロジェクトをそのまま実行するには、以下の手順に従います。

- 5.4.2に従って、付属のサンプル RFP 注入鍵(rfp_resource_rx74m.zip に含まれる KUK.rkey、ECC_Public_Key.rkey、および ECC_Private_Key.rkey)を注入してください。
- サンプル KUK を使用して生成された一連の新しいユーザ鍵(ECC 公開鍵および秘密鍵)は、サンプルプロジェクトですでにプロビジョニングされています。その後、ユーザはサンプルプロジェクトの実行に直接進むことができます。

注:サンプル鍵は、量産用途には使用しないでください。

ユーザによって新たに生成された鍵でサンプルプロジェクトを使用するには、以下の手順に従います。

- 新たに生成された注入鍵と更新鍵(rfp_resources_rx74m.zip に含まれるものを使用するのではなく、次の5.2または5.3によって生成)をテストするには、ユーザは5.4に従って鍵を MCU に注入する必要があります。ユーザはまた、サンプルプロジェクトで使用されている対応するファイルを置き換えるため、同じ鍵名で新しいユーザ鍵ファイル(new_ecc_public_key.c/.h と new_ecc_private_key.c/.h)を生成する必要があります。サンプルプロジェクトが更新されると、ユーザはサンプルプロジェクトの実行に進み、操作を検証できます。
- 新しいユーザ鍵更新手順のみをテストするために、ユーザは付属の KUK.rkey ファイルを使用して、サンプルプロジェクト内の対応するファイルを置き換える新しいソースファイルを生成できます。サンプルプロジェクトが更新されると、ユーザは操作の検証に進むことができます。

5.5.1 サンプルプロジェクトの概要

このサンプルプロジェクトでは、RSIP-E50D の Protected Mode を使用して、次の処理を実施します。

- 注入された ECC 秘密鍵を使用して、NIST テストベクタのメッセージに署名し、注入された ECC 公開鍵を使用して、対応する秘密鍵によって生成された署名を検証します。
- 注入された KUK を使用して、新しくラップされた ECC 鍵ペアを注入し、MRAM にこの新しい ECC 鍵ペアを格納します。
- 新しい ECC 鍵ペアを使用して、署名を生成しその署名を検証します。

5.5.2 サンプルプロジェクトの鍵の注入

5.5.2.1 注入鍵のフォーマット

RFP を使用して MCU MRAM に注入される鍵は、FSP Crypto API で直接使用することはできません。書式設定を少し変更する必要があります。

(1) 注入された ECC 鍵ペアのフォーマット

ソースコード(hal_entry.c)の下記の部分に、ECC 公開鍵と ECC 秘密鍵の情報が格納されます。その後、格納先変数を暗号処理に使用できます。マクロ `ECC_PUB_KEY_ADDRESS` と `ECC_PRI_KEY_ADDRESS` を実際の注入アドレスに置き換え、鍵の種類を設定して、FSP Crypto API で使用するフォーマットとします。

```
static rsip_wrapped_key_t ecc_public_key_injected =
{
    .type = RSIP_KEY_TYPE_ECC_SECP256R1_PUBLIC, .p_value = (void *)ECC_PUB_KEY_ADDRESS
};

static rsip_wrapped_key_t ecc_private_key_injected =
{
    .type = RSIP_KEY_TYPE_ECC_SECP256R1_PRIVATE, .p_value = (void *) ECC_PRI_KEY_ADDRESS
};
```

(2) 注入された KUK のフォーマット

ソースコード(hal_entry.c)の下記の部分に、注入された KUK の情報を格納します。その後、格納先変数をセキュアな鍵更新に使用できます。マクロ `KUK_ADDRESS` を実際の注入アドレスに置き換え、鍵の種類を設定して、FSP Crypto API で使用するフォーマットとします。

```
static rsip_wrapped_key_t kuk_key =
{
    .type = RSIP_KEY_TYPE_KUK, .p_value = (void *) KUK_ADDRESS
};
```

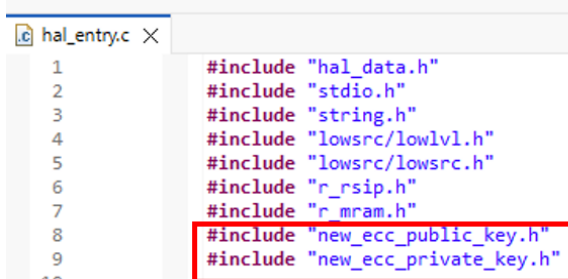
5.5.2.2 注入された鍵と更新された鍵の検証

ECC 鍵ペアのラップ鍵が正しく注入されたことを確認するには、同じ鍵ペアを用いた NIST テストベクタの期待値と比較します。具体的には、ECC 秘密鍵で生成した署名がテストベクタの期待値と一致すること、さらにその署名が対応する ECC 公開鍵で正しく検証できることを確認することで、ラップ鍵が正しく注入されていることを検証できます。

5.5.3 サンプルプロジェクトの実行

以下の手順に従って、サンプルプロジェクトを実行します。プロジェクトのビルドと実行前に、上記で生成されたセキュアな鍵注入の結果を使用して更新する必要があるコードのセクションがあることに注意してください。

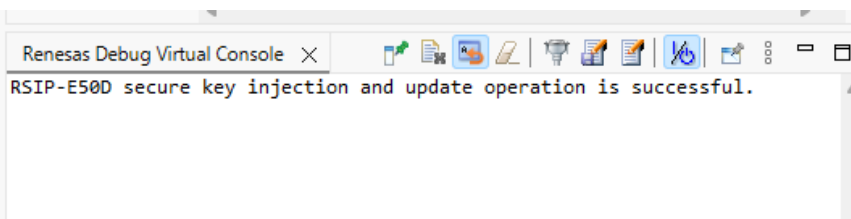
1. e² studio を起動し、secure_key_inject_update_rx74m_pm_ccrx または secure_key_inject_update_rx74m_pm_gcc をワークスペースにインポートします。
2. 新しい ECC 鍵ペアを生成して使用する場合は、5.2.3または5.3の方法で生成した new_ecc_public_key.c および new_ecc_public_key.h を、フォルダ \secure_key_inject_update_rx74m\src\にあるものと置き換えてください。
3. 異なるファイル名が使用されている場合は、hal_entry.c のインクルードファイルを更新して、新しいファイル名を反映させます。



```
1 #include "hal_data.h"
2 #include "stdio.h"
3 #include "string.h"
4 #include "lowsrc/lowlvl.h"
5 #include "lowsrc/lowsrc.h"
6 #include "r_rsip.h"
7 #include "r_mram.h"
8 #include "new_ecc_public_key.h"
9 #include "new_ecc_private_key.h"
10
```

図5-30 ECC 操作用に生成されたヘッダファイルをインクルード

4. 次に、configuration.xml をダブルクリックします。コンフィグレータを開いたら、**Generate Project Content** をクリックし、プロジェクトをビルドします。
5. **Renesas Views > デバッグ > Renesas Debug Virtual Console** を選択しコンソール画面を表示します。
6. デバッグを実行します。
7. コンソール画面において、下記のように表示されることを確認します。



```
Renesas Debug Virtual Console
RSIP-E50D secure key injection and update operation is successful.
```

図 5-31 コンソール画面の表示

6. RSIP Compatibility Mode のセキュアな鍵注入とサンプルプロジェクト

6.1 概要

本章では、RSIP-E50D Compatibility Mode におけるセキュアな鍵注入の準備と、準備した鍵を使用するサンプルプロジェクトについて説明します。

Compatibility Mode では、FSP の鍵注入 API を使用して PSA Crypto API に鍵を注入します。本章では、SKMT を使用して、サンプルプロジェクトで利用できる C ソースファイルおよびヘッダファイルを生成する方法を示します。

このサンプルプロジェクトでは、RSIP-E50D の Compatibility Mode の以下の機能を示します。

- AES-128 鍵の注入
- 注入された AES-128 鍵を使用した AES-128 暗号化および復号処理

このアプリケーションノートのサンプルプロジェクトでは、NIST CAVP テストベクタの値を例として使用しています。元のテストベクタを確認される場合は、以下のリンクからダウンロードしてください。

<https://csrc.nist.gov/Projects/Cryptographic-Algorithm-Validation-Program/Block-Ciphers>

6.2 SKMT GUI を使用した UFPK と W-UFPK による鍵のラップ

SKMT GUI を使用して、概要タブで **RX Family, RSIP-E50D Compatibility Mode** を選択します。



図6-1 RX Family, RSIP-E50D Compatibility Mode を選択

鍵のラッピングタブの鍵の種類タブで、AES と 128 bits を選択します。



図6-2 鍵の種類として AES-128 ビットを選択

鍵データタブを選択し、以下に示すように平文データに鍵データを入力します。ここでは、以下の値を例に使用しています。

e0000000000000000000000000000000

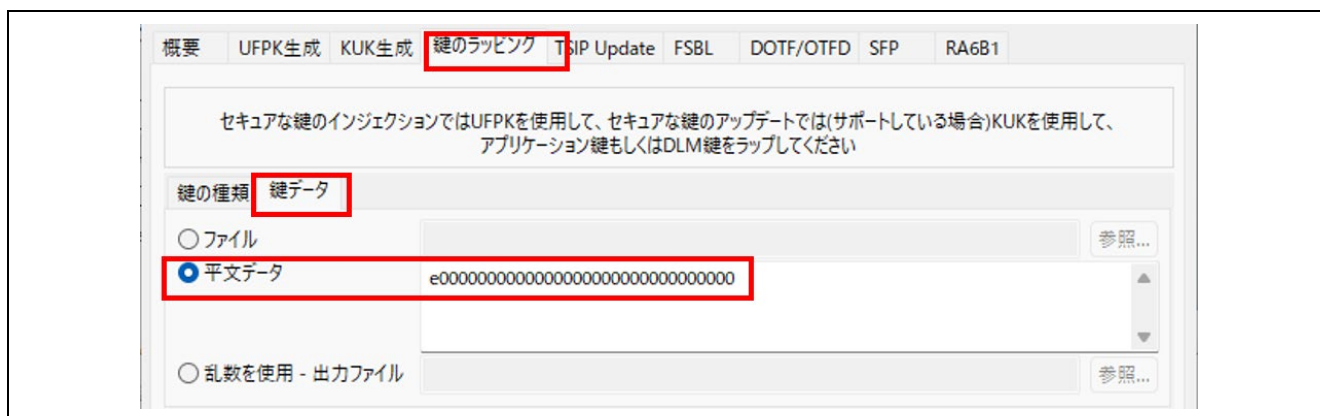


図6-3 初期 AES-128 鍵データを設定

ラッピング鍵セクションで、対応する参照ボタンをクリックして、UFPK と W-UFPK のペアを選択します。IV データの乱数生成機能を使用するオプションを選択します。フォーマットオプションで C ソースを選択します。次に参照ボタンをクリックし、出力フォルダとファイル名を選択し、鍵に名前を付けます。この名前は、C ソースファイル用に生成された定義に反映されます。

次に、**ファイル**を生成する ボタンをクリックします。AES 鍵を注入するソースファイルが生成されます。

ランニング鍵

☒ UFPK UFPKファイル: C:\DLM_Key_Injection\test_gui\%rx74m_ufpk.key [参照...](#)

W-UFPKファイル: C:\DLM_Key_Injection\test_gui\%rx74m_ufpk.key_enc.key [参照...](#)

☐ KUK KUKファイル: [参照...](#)

IV

☒ 乱数生成機能を使用する

☐ 指定値を使用する (16バイト, ビッグエンディアン) 00112233445566778899AABBCDDEEFF

出力

フォーマット: Cソース ▼ ファイル: 74m_protected_mode_key_info\%rx74m_initial_aes128_key.c [参照...](#)

エンディアン: Little ▼ ☐ データを追加出力する

アドレス: 10000 Key name: RX74M_AES128

```

Command Line: skmt.exe /genkey /ufpk "C:\DLM_Key_Injection\test_gui\%rx74m_ufpk.key" /wufpk "C:\DLM_Key_Injection\test_gui\%rx74m_ufpk.key_enc.key" /mcu "RX-RSIP-E50D-CM" /keytype "AES-128" /key "e000000000000000000000000000000000" /filetype "csource" /keyname "RX74M_AES128" /output "C:\Secure_Key_Injection\%rx74m_protected_mode_key_info\%rx74m_initial_aes128_key.c"
Output File: C:\Secure_Key_Injection\%rx74m_protected_mode_key_info\%rx74m_initial_aes128_key.h
Output File: C:\Secure_Key_Injection\%rx74m_protected_mode_key_info\%rx74m_initial_aes128_key.c
UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F
W-UFPK: 00000000F77A67AB713D174EB9D7C6639DAADB1F77A67AB713D174EB9D7C6639DAADB1
IV: 8FF1C49D597A7D797453C1AB446918ED
Encrypted key: 47558ECF0E7892F85599540191A5A446C1922A08FD4D96C6041AC624836941F
OPERATION SUCCESSFUL

```

図6-4 初期 AES-128 暗号化鍵ファイルの生成

生成された rx74m_initial_aes128_key.c に含まれるデータはサンプルプロジェクトで使用するためのサンプルデータです。お客様の最終製品では使用しないでください。

6.3 SKMT CLI を使用した UFPK と W-UFPK による鍵のラップ

SKMT を使用して、ラップ鍵のソースファイルを生成します。これは、次のコマンドで行います。

```
skmt.exe /genkey /ufpk  
file="C:\Secure_Key_Injection\rx74m_compatibility_mode_key_info\rx74m_ufpk.key"  
/wufpk  
file="C:\Secure_Key_Injection\rx74m_compatibility_mode_key_info\rx74m_ufpk.key_e  
nc.key" /mcu "RX-RSIP-E50D-CM" /keytype "AES-128"  
/key "e0000000000000000000000000000000" /filetype "csource" /output  
"C:\Secure_Key_Injection\rx74m_compatibility_mode_key_info\rx74m_initial_aes128_  
key_cli.c"
```

この例では、次のようになります。

- 鍵の種類“AES-128”を指定しました。
- ランダムに生成された IV を使用しています。IV は、このコマンドが実行されるたびに変更されます。
- この例では、UFPK と W-UFPK は C:\Secure_Key_Injection\rx74m_compatibility_mode_key_info に格納されています。
- この例では、ソースファイル(rx74m_initial_aes128_key_cli.c)の完全なファイルパスを指定しました。

```
Output File:  
C:\Secure_Key_Injection\rx74m_compatibility_mode_key_info\rx74m_initial_aes128_key_cli.c  
  
UFPK: 000102030405060708090A0B0C0D0E0F000102030405060708090A0B0C0D0E0F  
  
W-UFPK: 1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF12345678  
  
IV: 1234567890ABCDEF1234567890ABCDEF  
  
Encrypted key:  
  
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
```

図6-5 CLI を使用した KUK 注入ファイルの生成

6.4 Compatibility Mode サンプルプロジェクト

本章では、AES-128 ユーザ鍵注入の例を用いて、RSIP Compatibility Mode を紹介します。

6.4.1 SKMT で生成されたファイルの使用

6.2の方法で生成されるソースファイルは、サンプルプロジェクトに含まれています。新しく生成したソースファイルを使用する場合は、src\にあるものと置き換えてください。

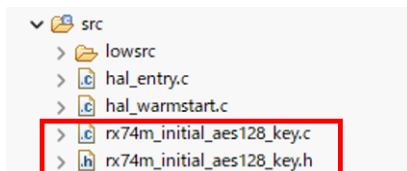


図6-6 RX74M サンプルプロジェクトソースコード

6.4.2 サンプルプロジェクトの実行

以下の手順に従って、サンプルプロジェクトを実行します。

1. e² studio を起動し、secure_key_inject_rx74m_cm_ccrx または secure_key_inject_rx74m_cm_gcc をワークスペースにインポートします。
2. 新しい AES-128 鍵を生成して使用する場合は、6.2の方法で生成した rx74m_initial_aes128_key.h と rx74m_initial_aes128_key.c を、src\にあるものと置き換えてください。
3. 異なるファイル名が使用されている場合は、hal_entry.c のインクルードファイルを更新して、新しいファイル名を反映させます。

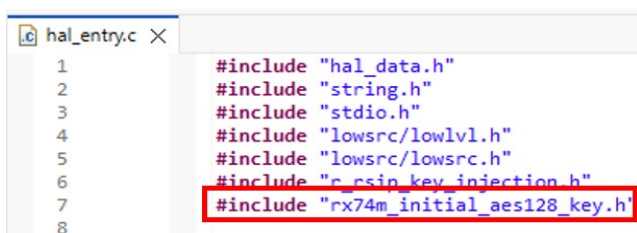


図 6-7 AES 操作用に生成されたヘッダファイルをインクルード

4. 次に、configuration.xml をダブルクリックします。コンフィグレータが開いたら、**Generate Project Content** をクリックし、プロジェクトをビルドします。
5. **Renesas Views > デバッグ > Renesas Debug Virtual Console** を選択しコンソール画面を表示します。
6. デバッグを実行します。
7. コンソール画面において、下記のように表示されることを確認します。

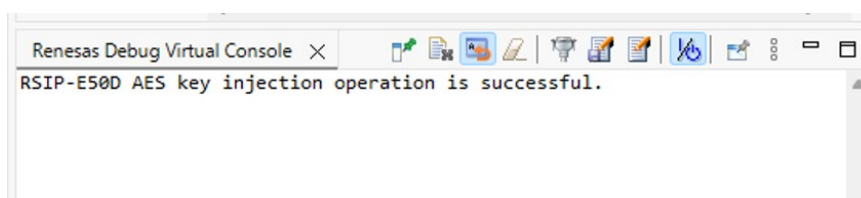


図 6-8 コンソール画面の表示

7. References

1. RX74M グループ ユーザーズマニュアル ハードウェア編 (R01UH1055)
2. Security Key Management Tool ソフトウェアマニュアル (R20UT5349)
3. Renesas Flash Programmer V3.24 ソフトウェアマニュアル (R20UT5820)

8. ウェブサイトとサポート

次の URL にアクセスして、マイクロコントローラの RX ファミリについて学び、ツールとドキュメントをダウンロードし、サポートを受けてください。

EK-RX74M リソース	renesas.com/ek-rx74m
RX 製品情報	renesas.com/rx
Flexible Software Package(FSP)	renesas.com/rx-fsp
RX 製品サポートフォーラム	renesas.com/rx/forum
ルネサスサポート	renesas.com/support

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2026.10.22	-	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策

CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

当該製品の電源がオフ状態のときに、入力信号や入出力ブルアップ電源を入れないでください。入力信号や入出力ブルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、 V_{IL} (Max.) から V_{IH} (Min.) までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、 V_{IL} (Max.) から V_{IH} (Min.) までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

高品質水準： 輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じて、当社は一切その責任を負いません。

7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限られません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因したまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証お客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものいたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1 2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。