User's Manual

Security Key Management Tool

ユーザーズマニュアル

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、
予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ルネサス エレクトロニクス
www.renesas.com

Rev.1.07　2024.08

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）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またはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
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。

製品ご使用上の注意事項

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

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、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。

このマニュアルの使い方

1.　目的と対象者

　このマニュアルは、Security Key Management Tool の機能をユーザに理解していただくためのマニュアルです。ルネサスエレクトロニクス製マイコンに内蔵されるセキュリティ IP である Security Cryptographic Engine, Trusted Secure IP を使用したシステムを設計開発するユーザを対象にしています。このマニュアルを使用するには、マイクロコントローラと Windows、Linux および macOS に関する基本的な知識が必要です。

ご使用するマイクロコントローラのマニュアルを十分確認の上、本ソフトウェアを使用してください。

2.　凡例

- 注：本文中に付けた”注”の説明
- 数の表記：10 進数　… xxxx
  　　16 進数　… 0xXXXX　または　xxxxH
- “　“：任意の文字、画面内の項目を示します。
- [　]：メニュー名、タブ名、ダイアログ名をしましょう。

3. 用語の説明

略語／略称	説明
CLI	Command Line Interface
DLM Key DLM 鍵	Device Lifecycle Management 用鍵　一部の MCU/MPU でサポート
DOTF	外部フラッシュ ROM などに書き込まれた暗号化されたイメージをリード、実行するときに、直接復号する機能。 MCU/MPUによってはDecryption On-The-Fly(DOTF)もしくはOn-The-Fly Decryption(OTFD)と呼ぶ。本ドキュメント内ではDOTFと記載。
Encrypted KUK	UFPKでラップされたKUK
Encrypted User Key	UFPK もしくは KUK でラップされた User Key
FSBL	First Stage Bootloader Second Stage Bootloader もしくは OEM_BL の正当性検証を行うためのプログラム
GUI	Graphic User Interface
HRK	Hardware Root Key MCU ファミリもしくは MCU ごとに決まる鍵データ
HUK	Hardware Unique Key デバイス毎に設定されている固有の鍵データ
KUK	Key Update Key User Key をアップデートするための鍵
OEM Bootloader OEM_BL	デバイス起動後に実行され、アプリケーションプログラムの正当性検証するためのプログラム
PEM	x509 ASN.1の鍵をBase64でエンコードしたファイル。 本ツールでは Openssl コマンド genrsa もしくは、ecparam -genkey コマンドで生成された鍵の読み込みをサポートしています。
Renesas key file	ルネサス鍵ファイル RFPで使用する鍵フォーマット　フォーマットは付録を参照してください。
Renesas Key Wrap service	UFPK をデバイス固有鍵でラップして W-UFPK を生成するサービス https://dlm.renesas.com/keywrap/
RFP	Renesas Flash Programmer https://www.renesas.com/rfp
RSIP	Renesas Secure IP RA、RX、RZ ファミリなどに実装されているセキュリティ IP
RSU	Renesas Secure Update RX ファミリ Firmware Update FIT(R01AN5824)で定義されている更新するプログラムイメージファイルフォーマット。
SCE	Secure Cryptographic Engine RA ファミリ、Synergy Platform に実装されているセキュリティ IP
SFP	Secure Factory Programming ルネサス製 MCU のブートファームウェアが持つ、暗号化されたイメージを復号しながら、デバイスに書き込む機能

略語／略称	説明
TSIP	Trusted Secure IP RX、RZ ファミリに実装されているセキュリティ IP
UFPK	User Factory Programming Key User Key と DLM Key をラップする際に使用する鍵
User Key ユーザ鍵	ユーザアプリケーションで使用する暗号鍵 AES 鍵や RSA 公開鍵、RSA 秘密鍵など
W-UFPK	Renesas Key Wrapping サービスでラップされた UFPK
Wrapped User Key	User Key を RSIP、SCE もしくは TSIP で使用するため、Encrypted User Key を RSIP、SCE もしくは TSIP でラップし直した User Key

目次

図表目次

1.　ルネサス鍵管理システム

1.1　Root of Trust の概要

Root of Trust は、重要なセキュリティ機能を実装する信頼性の高いハードウェア、ファームウェア、およびソフトウェアによって形成されるコンポーネントです（参照 https://csrc.nist.gov/projects/hardware-roots-of-trust）。通常の IoT システムの Root of Trust は、デバイスのハードウェアに根ざした識別情報と暗号鍵で構成されます。IoT ネットワーク内にデバイスが存在するためには、ユニークで不変かつ複製できない識別情報を使用する必要があります。

多くのセキュリティシステムで提供されるサービスの一部であるセキュアブートは、アプリケーションの認証に公開鍵暗号を使用します。セキュアブートに使用する暗号鍵はシステムの Root of Trust の一部です。また、デバイス秘密鍵やデバイス証明書で構成されるデバイス識別情報もシステムの Root of Trust の一部です。

上記の Root of Trust の説明から、暗号鍵の漏洩は安全なシステムを危険な状態にする可能性があることがわかります。暗号鍵を安全に保管し、改ざんや複製ができないようにすることが重要です。暗号鍵の保護を実現する際には、暗号境界内でのみ鍵へのアクセスを許可する必要があります。

ルネサスの鍵管理システムを使うことで、このような暗号鍵の保護を実現することができます。

1.2　Renesas セキュリティ IP と関連する鍵の概要

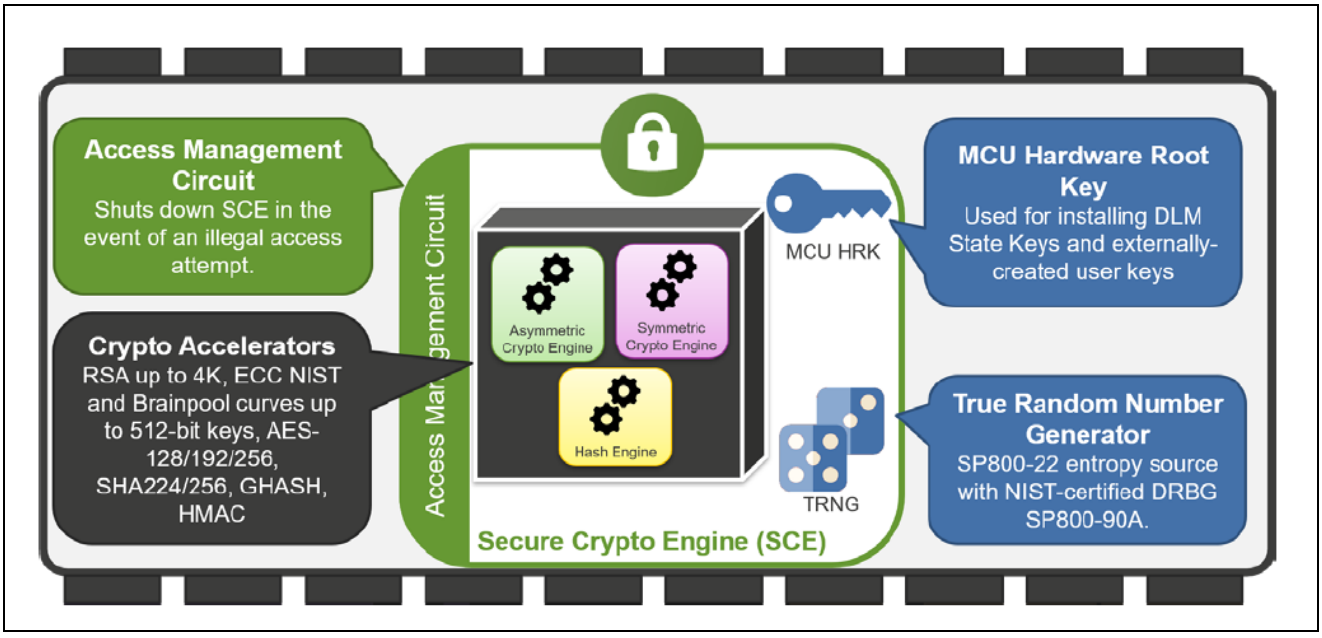

図 1.1　Secure Crypto Engine の概要

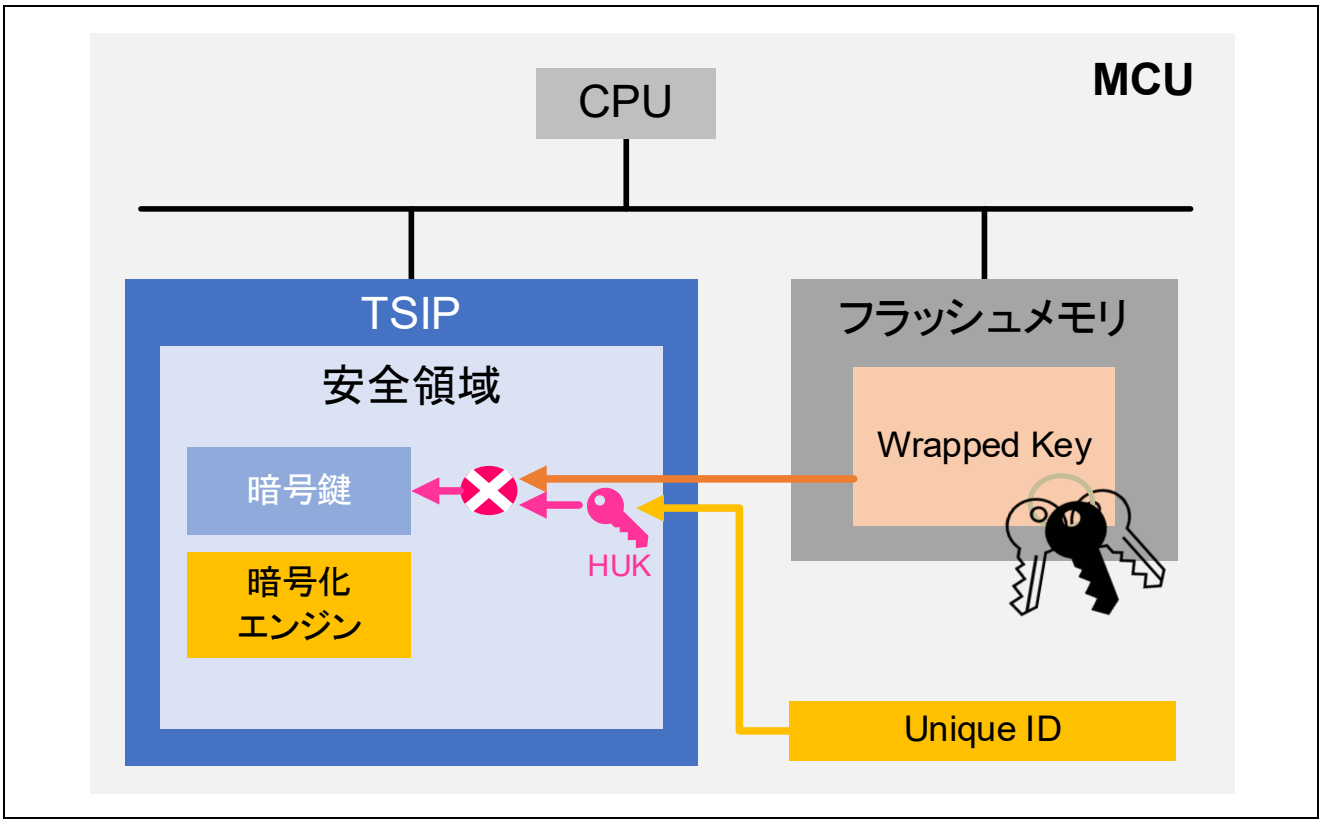

図 1.2　　RX Trusted Secure IP の概要

Renesas のセキュリティ IP である Secure Crypto Engine(SCE)、Trusted Secure IP(TSIP)ならびに RSIP-Exxx セキュリティエンジン (RSIP)は、MCU 内で、他の IP とは独立したサブシステムになっています。暗号エンジンは、対称暗号化アルゴリズムと非対称暗号化アルゴリズムの両方のハードウェアアクセラレータおよび、さまざまなハッシュとメッセージ認証コードにも使用可能です。また、暗号化操作のエントロピーソースを提供する True Random Number Generator（TRNG）も含まれています。RSIP、SCE ならびに TSIP は、アクセスマネジメント回路によって保護されています。アクセスマネジメント回路は、不正な外部アクセスを検出した場合に暗号エンジンを停止します。
サポートしている暗号アルゴリズム、セキュアな鍵のインジェクション、およびセキュアな鍵のアップデートは、MCU / MPU によって異なります。 詳細は、各 MCU/MPU の Web ページから確認してください。

表 1-1 MCU/MPU 関連サイト

MCU/MPU	Category	URL
RA Family	MCU driver	https://www.renesas.com/eu/en/software-tool/flexible-software-package-fsp
	Application Project	https://www.renesas.com/eu/en/document/apn/installing-and-updating-secure-keys-ra-family
RX Family	MCU drivers and example project	https://www.renesas.com/software-tool/trusted-secure-ip-driver
RZ/T2M, RZ/T2ME, RZ/T2L, RZ/N2L	Security Package	https://www.renesas.com/software-tool/rz-mpu-security-package-rtos-bare-metal
Synergy Platform	MCU driver	https://www.renesas.com/eu/en/products/microcontrollers-microprocessors/renesas-synergy-platform-mcus/renesas-synergy-software-package

1.3　Renesas セキュリティ IP を活用したセキュアな鍵のインジェクション

セキュアな鍵のインジェクションとアップデートは、暗号エンジンによるラップされた鍵のサポートと組み合わせて、平文鍵を使用する場合に発生する多くの脆弱性に対処できます。

- 平文鍵をフラッシュ ROM に格納しません。プログラムが漏洩した場合でも、機密性の高い鍵情報を保護します。
- 平文鍵を RAM に格納しません。システム上で悪意のあるコードが実行された場合でも、機密性の高い鍵情報を保護します。
- 鍵はコードフラッシュ、データフラッシュに安全に保存でき、また外部メモリにも格納することもできるため、制限なく鍵を安全に保管することが可能です。

さらに、以下で説明するように、Renesas の鍵のラッピング技術を使用することで、デバイスの不正な複製から保護することが可能になります。

1.3.1　鍵のラッピングの利点

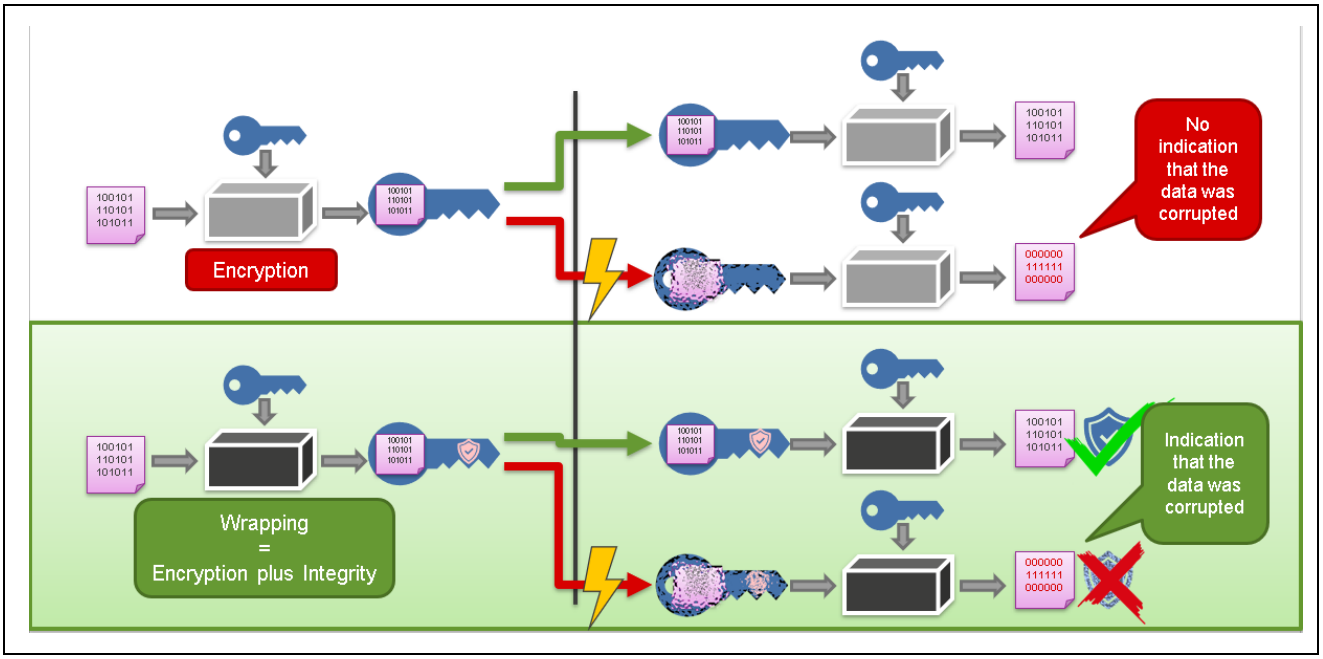

図 1.3　暗号化とラッピングの違い

安全な資産の保管のために、ラッピングと暗号化の違いを説明します。

データが暗号化されて別の受信者に送信されるときに、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、機密情報が交換されます。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でしょうか？ 受信者が無意識のうちに破損した情報を受信した場合、復号アルゴリズムは元のデータが破損していることを示すことなく、そのデータはガベージデータとなってしまいます。ラッピングでは、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。

1.3.2　HUK を使用した鍵のラッピングの利点

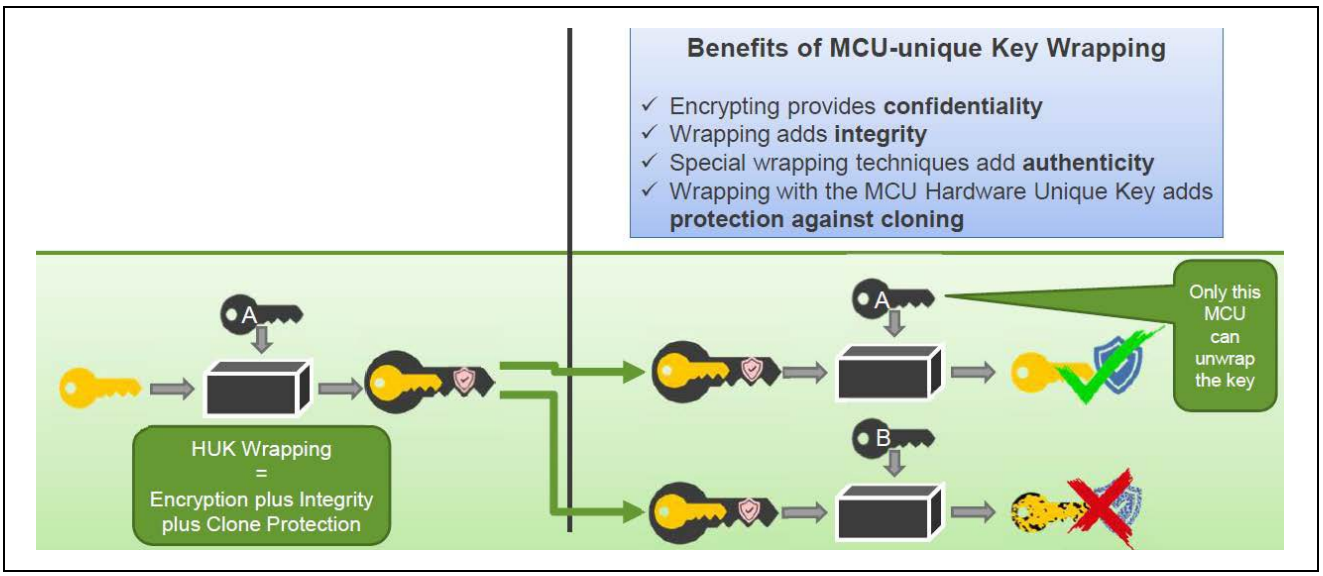

図 1.4　HUK を使用した鍵のラッピング

Hardware Unique Key(HUK)を使用して保存された鍵をラップすることで、もう一つの保護機能であるクローン保護機能が追加されます。

ラップされた鍵が他の MCU に転送またはコピーされた場合、その MCU の HUK はラップを解除することもコピーされた鍵を使用することもできません。たとえ MCU の全コンテンツが他のデバイスにコピーされたとしても、その鍵を使用したり、鍵そのものを取り出したりすることはできません。

1.4　ラップされた鍵のインジェクション手順の概要

ラップされた鍵のインジェクション手順について説明します。 Security Key Management Tool を使用してラップされた鍵を生成します。サポートしている鍵の種類については、MCU/MPU によって異なります。表 1-1MCU/MPU 関連サイトを参考に各デバイスのアプリケーションノートやドライバを参照してください。

1.4.1　セキュアな鍵のインジェクションの手順

セキュアな鍵のインジェクションは、アプリケーション鍵が、プロビジョニングプロセス中に平文鍵が露出しないようにすることを言います。このプロセスの詳細な方法は、MCU/MPU に依存します。デバイスによっては、プログラミングインタフェースを介してセキュアな鍵のインジェクションをサポートしているものもありますし、デバイス上で実行されるファームウェアを使用してセキュアな鍵のインジェクションをサポートしているものもあります。Security Key Management Tool を使用した鍵の準備手順は、平文鍵情報を扱うため、セキュアな環境で実行してください。

セキュアな鍵のインジェクションは、以下の 3 つステップで実施します。

1. セキュアな鍵のインジェクションの最初のステップは、Renesas Device Lifecycle Management（DLM）Server で、Hardware Root Key（HRK）を使用して任意の User Factory Programing Key（UFPK）をラップします。 UFPK は、ユーザが選択した 256 ビットの値です。 同じ UFPK で、任意の数のキーをインジェクションできます。
   Security Key Management Tool を使用して、DLM サーバーファイルフォーマットの UFPK ファイルを出力します。Security Key Management Tool GUI バージョンの場合は[UFPK 生成]タブを使用します。[UFPK 生成]タブの使用方法は 3.4[UFPK 生成]タブを参照してください。

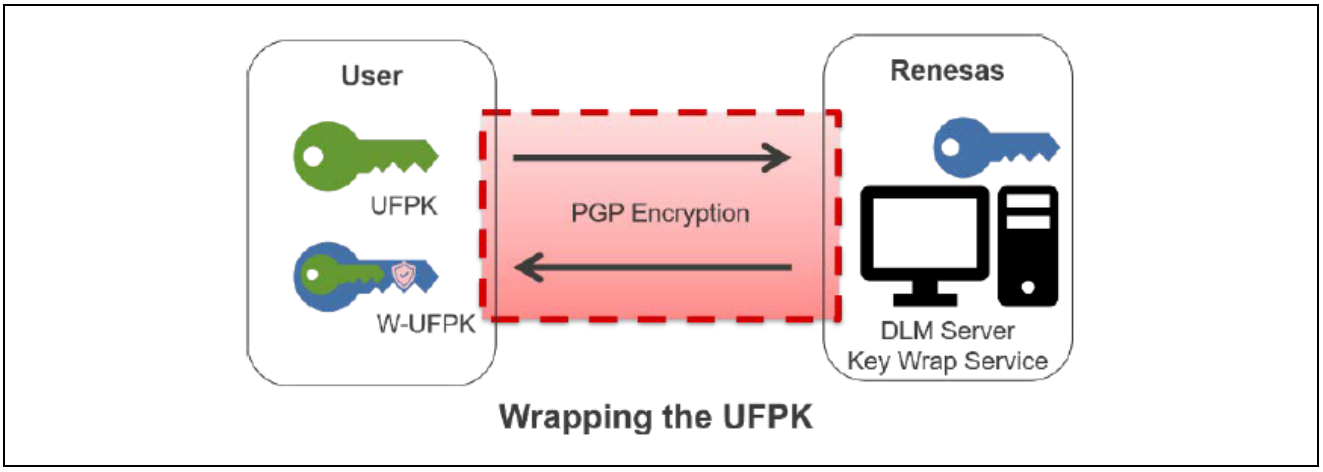

図 1.5　　DLM サーバーを使用した UFPK のラッピング

2. 次は、ユーザ鍵を UFPK でラップします。Security Key Management Tool は、生データまたはバイナリ鍵ファイルで指定されたユーザ鍵をラップし、選択したデバイスによるセキュアな鍵のインジェクションに使用できるファイルを生成することができます。すべての出力ファイルタイプが、すべてのデバイスファミリーでサポートされているわけではありません。例えば、Renesas RA ファミリ SCE9 Protected Mode はプログラミングインタフェース経由でのみセキュアな鍵のインジェクションをサポートしているため、Renesas key file を生成する必要があります。

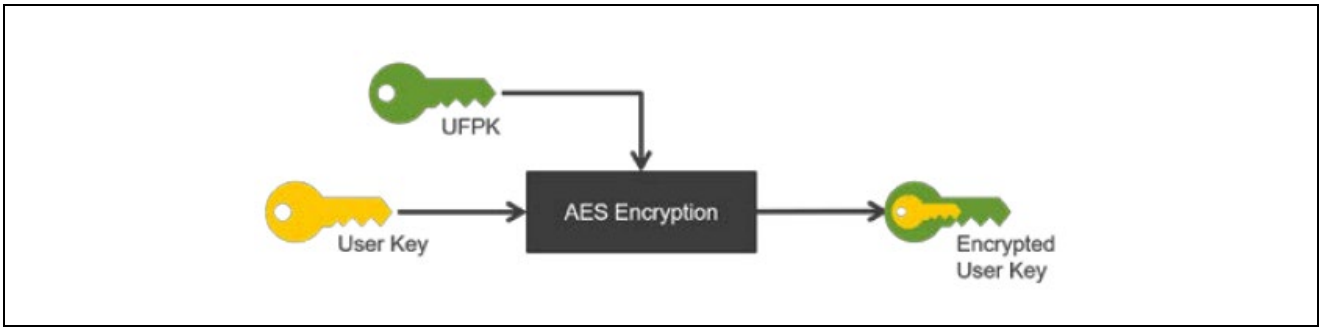

図 1.6　　UFPK を使用したユーザ鍵のラッピング

3. 最後に、ユーザ鍵は、選択したデバイスがサポートするインジェクションプロセスに応じて、シリアルプログラミングインタフェースまたはデバイス上で実行されるファームウェアのいずれかを介してインジェクションされます。このプロセスへの入力には、ラップされた UFPK（W-UFPK）と、前のステップで準備されたラップされたユーザ鍵の両方が含まれます。

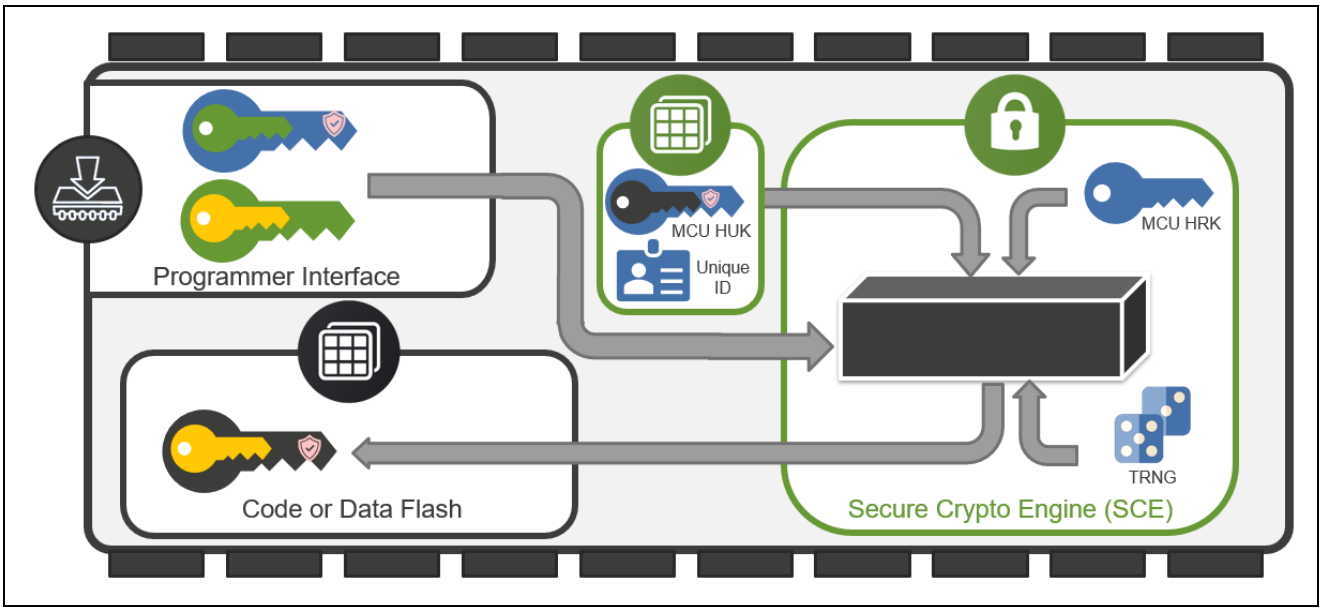

図 1.7　　シリアルプログラミングインタフェースを使用したユーザ鍵のインジェクション

1.4.2　セキュアな鍵のアップデートの手順

フィールドでのセキュアな鍵のインジェクションを行うためには、プロダクションプログラミング/プロビジョニング中に 1 つ以上の鍵アップデートのための鍵：Key Update Key（KUK）をインジェクションする必要があります。フィールドにおいて新しい鍵のインジェクションは通常、古い鍵を置き換えるために行われるため（鍵のローテーションまたは鍵の再生成）、このプロセスは「鍵のアップデート：Key Update」とも呼ばれます。

KUK は、他のユーザ鍵と同様に、コードフラッシュまたはデータフラッシュ（MCU で使用可能な場合）のいずれかに保存できます。 KUK は、新しい鍵をインジェクション/ラップできる唯一のメカニズムであるため、工場出荷書き込み時に複数の KUK をインジェクション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消去することができ、インフラのセキュリティポリシーを準拠したり、鍵の露出によるセキュリティ侵害に対応したりできます。

工場出荷時にプログラミングインタフェースを無効にした後は、追加の KUK をインジェクションできないことに注意してください。プログラミングインタフェースが搭載された製品がフィールドに投入されると、新しい鍵は既存の KUK を介してのみインジェクションできます。

KUK は、生産時に任意のコードまたはデータフラッシュに保存することができます。新しいユーザ鍵をインジェクションするために各デバイスで用意されている暗号ドライバの鍵アップデート用 API はこのデータを使用して、フィールドで新しい鍵をインジェクションできます

鍵のアップデートには、3 つのステップがあります。

1. 最初のステップは、1.4.1 セキュアな鍵のインジェクションの手順 に記載されているように、KUK を生成し、インジェクションすることです。KUK は Key Type “KUK”としてインジェクションする必要があります。

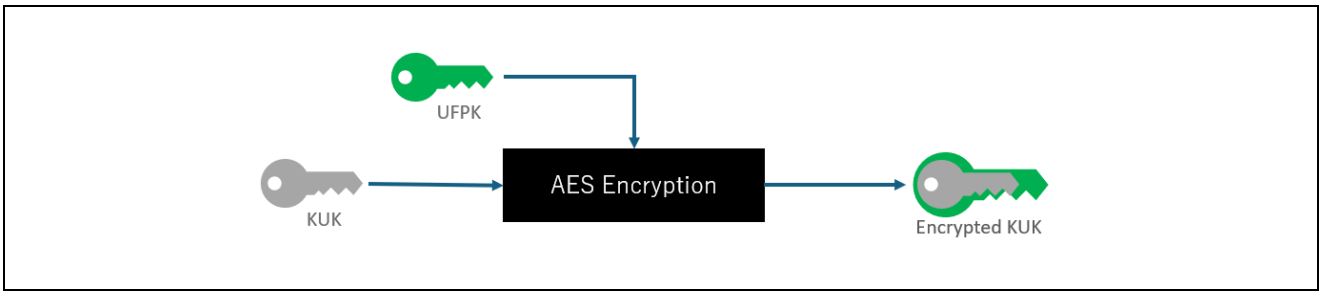

図 1.8　UFPK を使用した KUK のラッピング

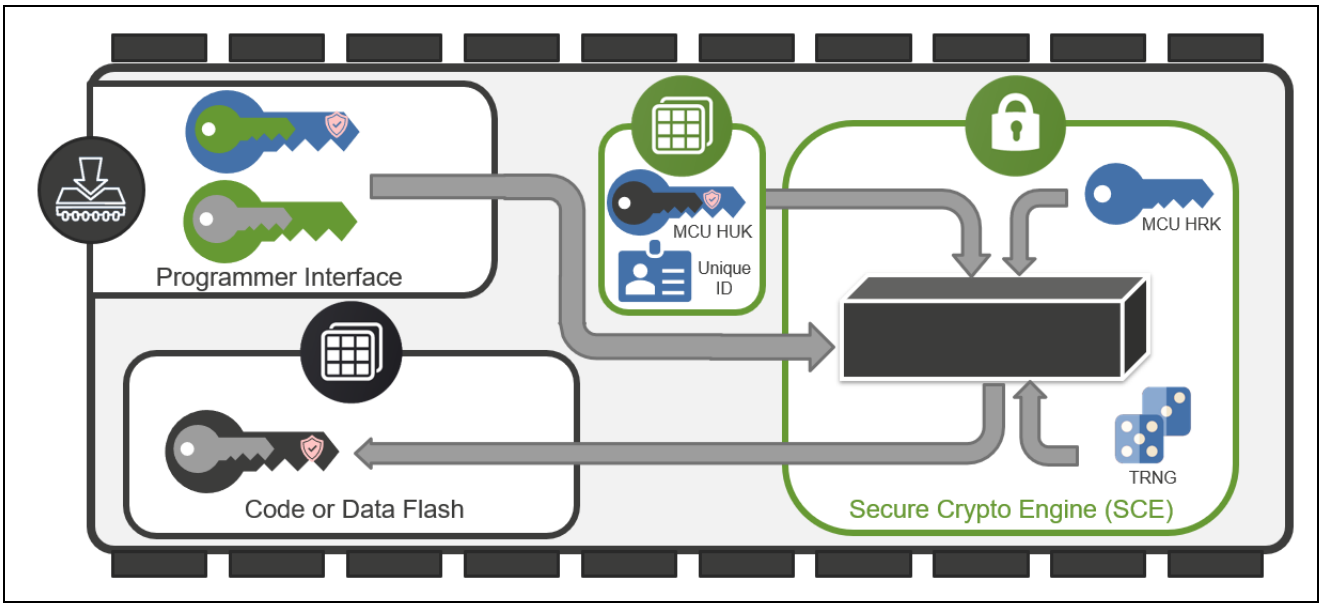

図 1.9　　KUK のインジェクション

2. 2 番目のステップは、KUK を使用して新しいユーザ鍵を KUK でラップします。

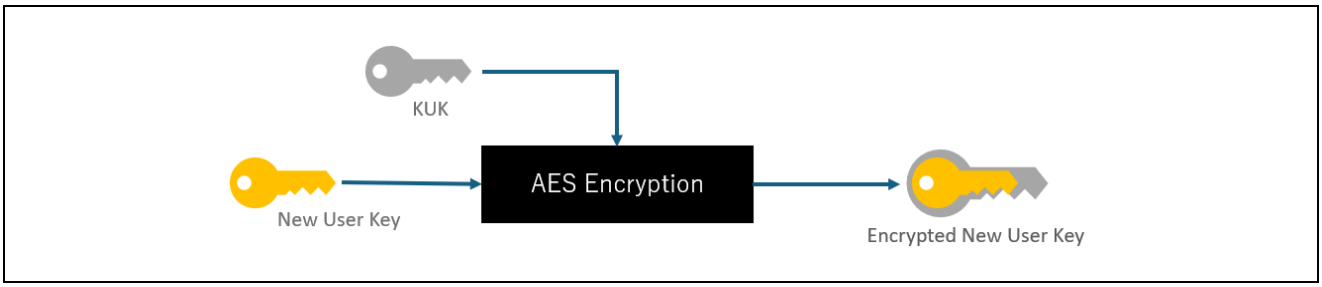

図 1.10　　KUK を使用した新しいユーザ鍵のラップ

3. 最後のステップは、各デバイスドライバとすでにインジェクションされた KUK を使用して、新しいユーザ鍵をインジェクションすることです。新しい鍵のインジェクション API とその使用方法は、各 MCU/MPU のデバイスドライバのマニュアルをご参照ください。

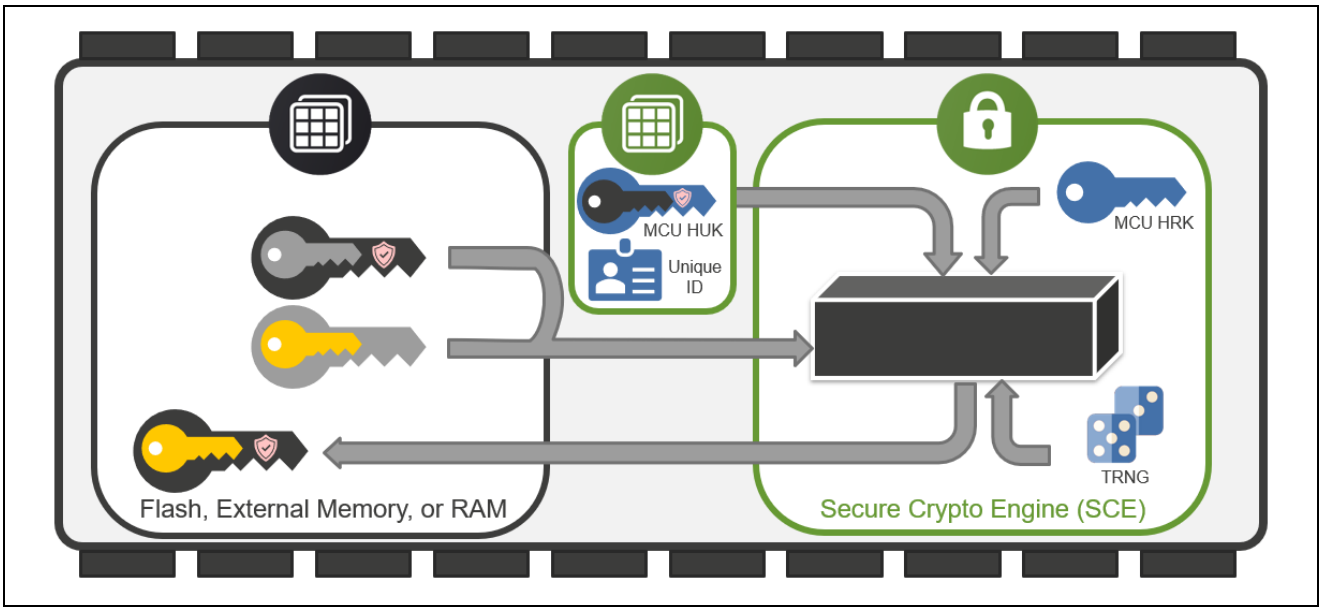

図 1.11　ユーザ鍵のアップデート

1.5　Renesas セキュリティ機能

Renesas 製品では、鍵のインジェクションならびにアップデート以外に、Renesas セキュリティ IP を活用した様々なセキュリティ機能を提供しています。Security Key Management Tool では、そのセキュリティ機能を使用するための機能をサポートしています。サポートしている機能については、MCU/MPU によって異なります。表 1-1MCU/MPU 関連サイトを参考に各デバイスのアプリケーションノートやドライバを参照してください。

1.5.1　First Stage Bootloader

セキュアなシステムでは、実行するアプリケーションプログラムの不正な書き換えが発生していないことを確認するため、アプリケーションプログラムの正当性検証を Bootloader が実施後、アプリケーションプログラムが実行される必要があります。さらに、この Bootloader 自身の正当性を保証する必要があり、Bootloader は書き換えできないメモリに配置するか、書き換えできないメモリに配置した別の Bootloader で、正当性検証をする必要があります。

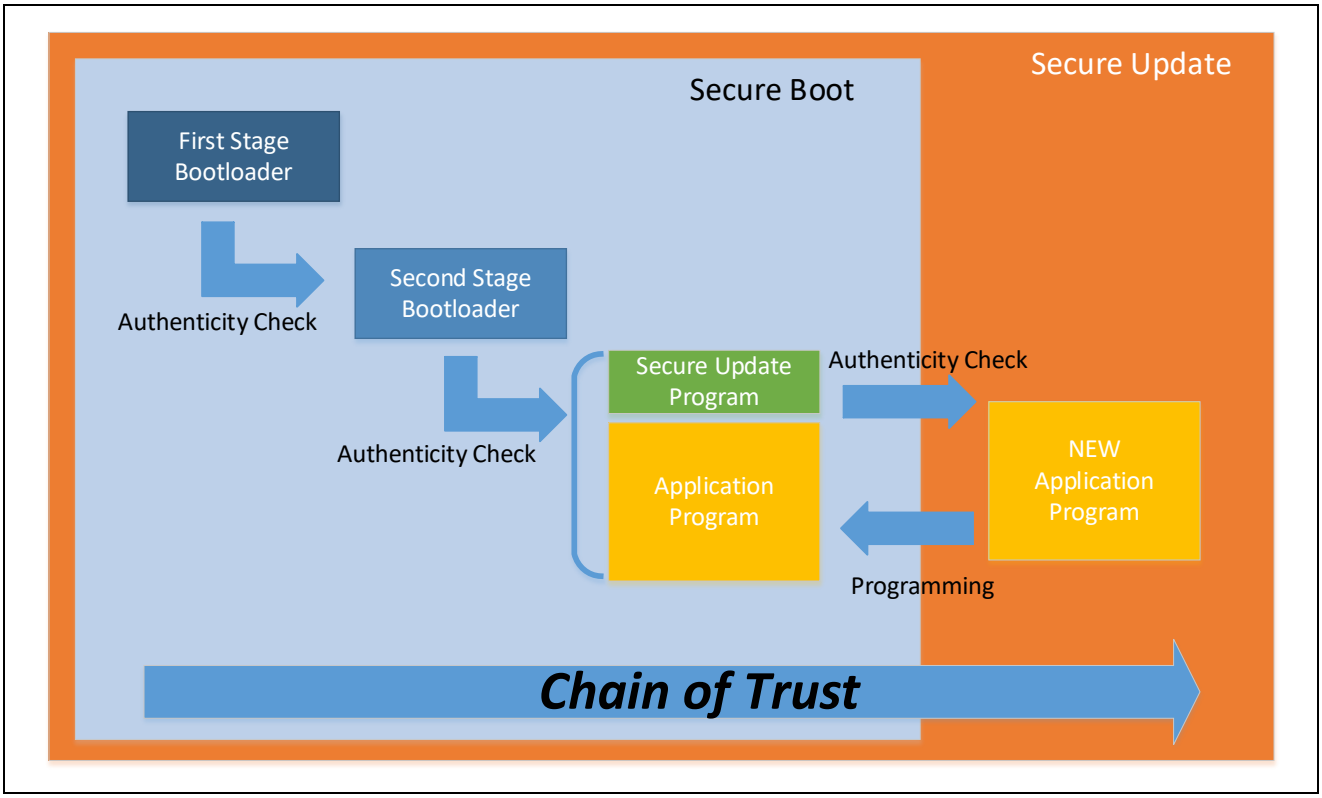

図 1.12　セキュアなシステムの Chain of Trust

Renesas の一部の MCU 製品では、デバイス内の Mask ROM もしくは OTP(One Time Programmable) ROM 領域に First Stage Bootloader(FSBL)を搭載しています。OEM_BL 検証ルート公開鍵は、製造時のプログラミング中にデバイスに登録されます。FSBL は OEM_BL 検証ルート公開鍵を使用して、Second Stage Bootloader の公開鍵を使った検証を行います。

次の図は、OEM_BL 検証ルート公開鍵と秘密鍵(OEM_ROOT_PK および OEM_ROOT_SK)、OEM_BL 検証公開鍵と秘密鍵(OEM_BL_PK および OEM_BL_SK）、OEM(セカンド・ステージ）Bootloader (OEM_BL)を使用した、初期生産プログラミングと通常のアプリケーション実行の両方におけるセキュア・ブート・プロセスのフローを示しています。

Security Key Management Tool は、この検証時に使用する鍵証明書(Key Certificate)ならびにコード証明書(Code Certificate)を生成します。詳細は、MCU/MPU のアプリケーションノート及びドライバをご参照ください。

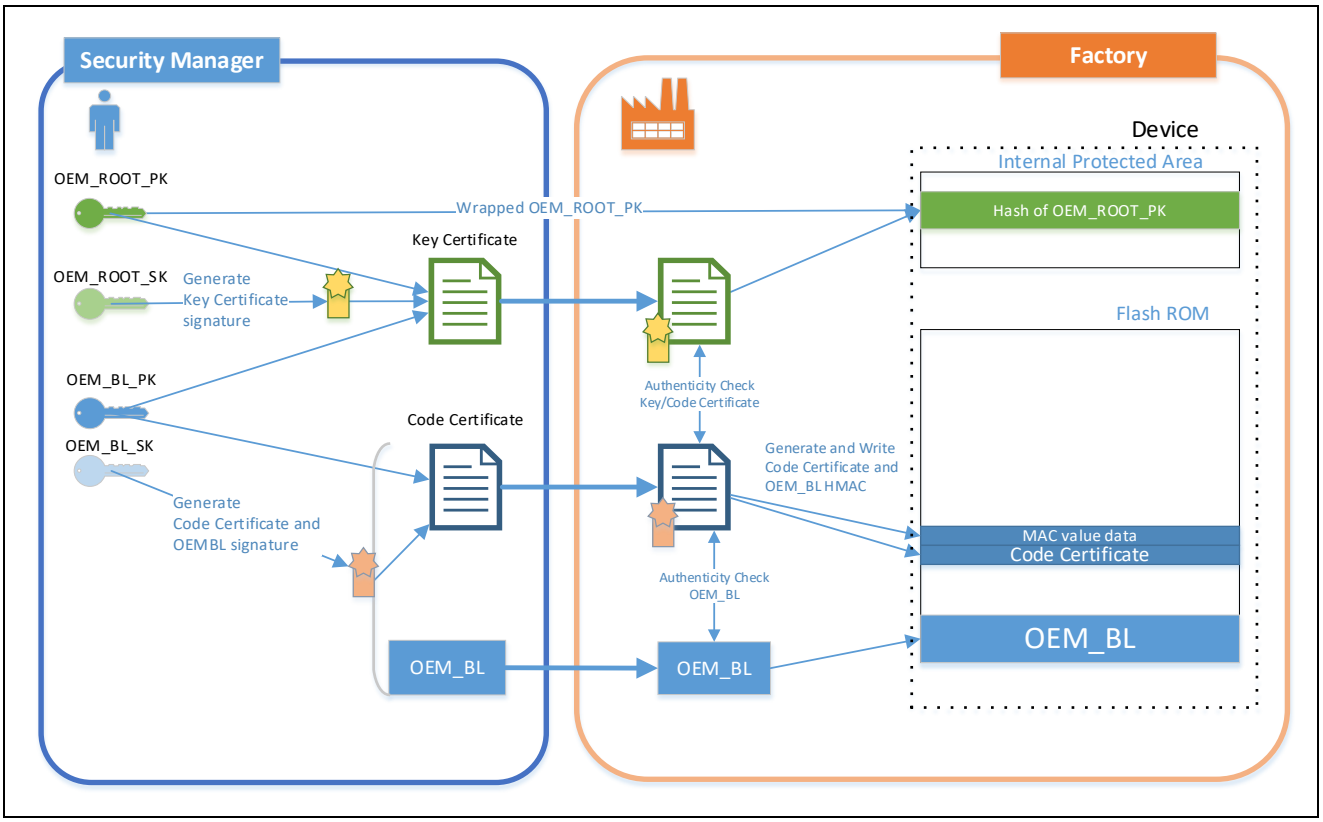

図 1.13　OEM_BL 工場書き込み時の Authenticity Check

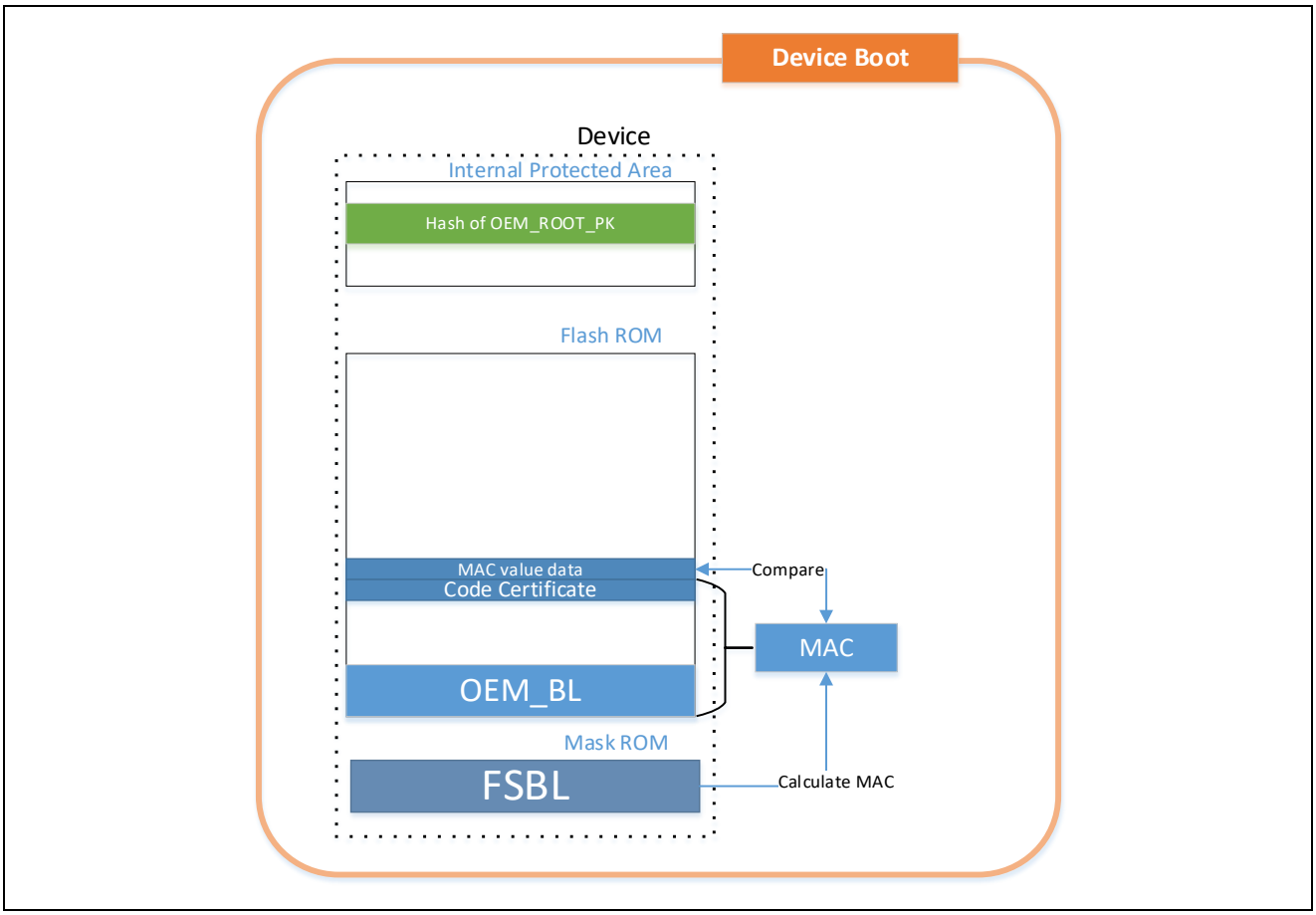

図 1.14　デバイス起動時の FSBL 動作

1.5.2　Decryption On-The-Fly/On-The-Fly Decryption

外部メモリに格納するアプリケーションプログラムや機密データを秘匿するためには、暗号化したアプリケーションプログラムを外部メモリに格納する必要があります。Decryption On-The-Fly もしくは On-The-Fly Decryption 機能(以降 DOTF)は、外部 ROM に格納された暗号化されたアプリケーションプログラムを eXecute in Place で実行する際に On-the-Fly で復号します。

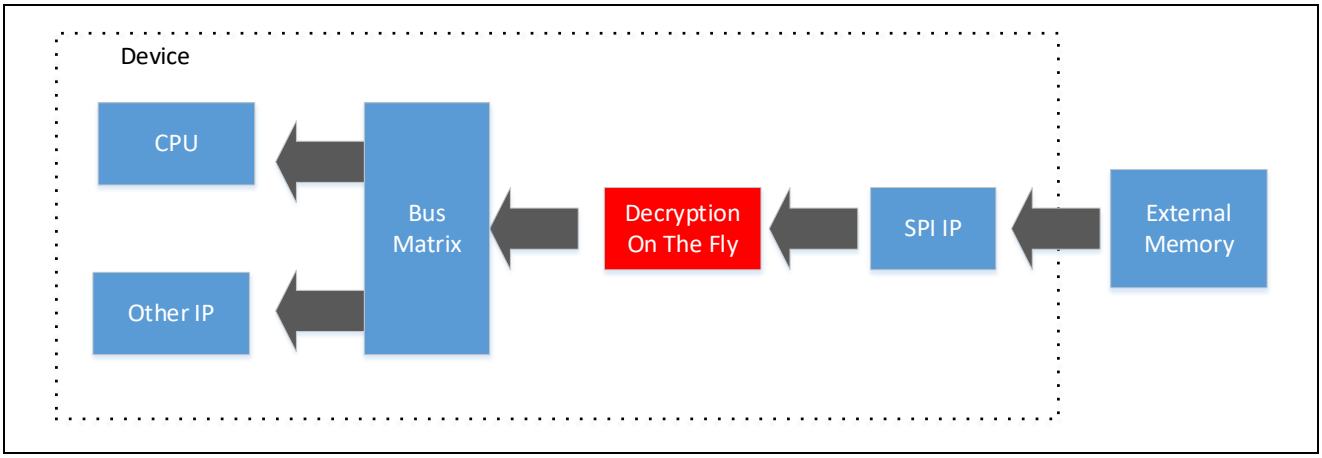

図 1.15　DOTF 実装 MCU/MPU の内部システムバス例

Security Key Management Tool は、Renesas DOTF 機能で使用するアプリケーション情報（実行可能コードまたは機密データ）の生成に対応しています。

1.5.3　Secure Factory Programming

Secure Factory Programming 機能は、暗号化したアプリケーションプログラムを書き込む機能です。工場書き込み時に工場からアプリケーションプログラムファイルが漏洩した場合でも、ファイルのプログラムもしくはデータの漏洩を防ぐことができます。

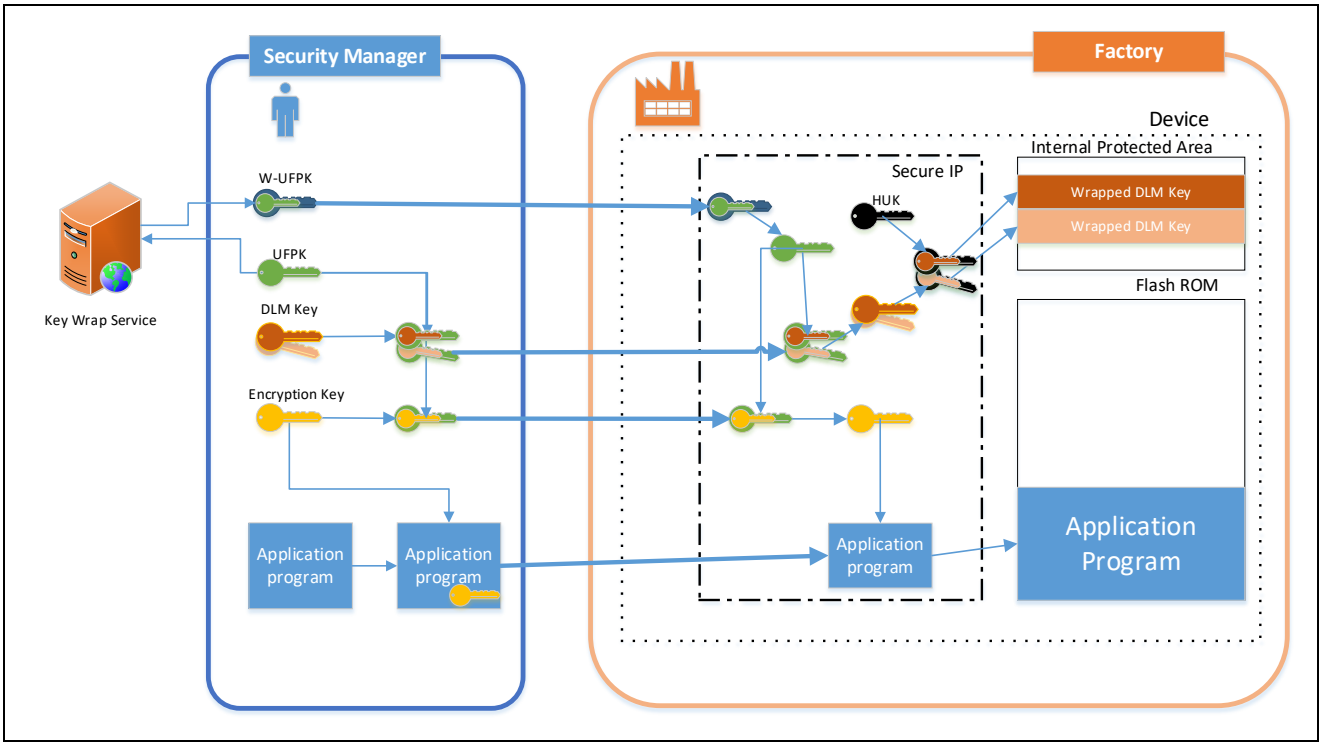

図 1.16　Secure Factory Programming 機能実装 MPU/MPU のシステム例

Security Key Management Tool では、Secure Factory Programming 機能を実現するため以下の機能をサポートします。

・ユーザアプリケーションプログラムの暗号化
・暗号化に使用した鍵のラッピング
・同時に注入可能なデバイスライフサイクルの鍵のラッピング
・First Stage Bootloader の設定
・最終 DLM ステートへの遷移

特定のデバイスでサポートされている機能のリストや、内部メモリのみのプログラミングをサポートするなど、Secure Factory Programming の制限事項については、MCU/MPU のマニュアルを参照してください。

2.　概要

Security key Management Tool は、アプリケーションや DLM（Device Lifecycle Management）の鍵を、セキュアにインジェクションもしくはアップデートを行うためのツールです。
本ツールは 3 つのバージョンを用意しています。

・コマンドライン版　– Windows のコマンドプロンプトや Linux のターミナルソフトから、単一コマンドを実行したり、バッチ処理で複数コマンドを実行したりすることが可能です。キーマネージャーやグループ開発をサポートするためのツールです。
・スタンドアロン GUI 版 – ファームウェア開発者を支援するために設計された直感的な GUI ツールです。サードパーティ製 IDE で開発する場合に便利です。
・e²studio プラグイン版 – Renesas 製 IDE e²studio で使用可能な GUI インターフェースツールで、ファームウェア開発者の開発をサポートします。

2.1　特長

Security Key Management Tool でサポートしている機能と MCU/MPU を以下に示します。:

2.1.1　セキュアな鍵のインジェクション/アップデート

セキュアな鍵のインジェクション、アップデートに関連する以下の機能をサポートしています。
サポートしている MCU/MPU は表 2-1 を参照ください。

① UFPK 生成と Renesas Key Wrap Service へ送付可能なフォーマットのファイル生成
② セキュアな鍵のインジェクションのための UFPK を使った DLM Key のラッピング(DLM Key は一部の MCU/MPU でのみサポートされています)
③ セキュアな鍵のインジェクションションのための UFPK を使ったユーザ鍵のラッピング
④ セキュアな鍵のアップデートのための KUK を使ったユーザ鍵のラッピング

表 2-1 MCU/MPU ファミリ 鍵のインジェクション/アップデート サポート機能

ファミリ	サポート機能			
	①	②	③	④
RA Family				
RSIP-E51A Security Function and Protected Mode	✓	✓	✓	✓
RSIP-E51A Compatibility Mode	✓	-	✓	✓
SCE9 Security Function and Protected Mode	✓	✓	✓	✓
SCE9 Compatibility Mode	✓	-	✓	✓
SCE7	✓	-	✓	✓
SCE5_B	✓	✓	✓	✓
SCE5	✓	-	✓	✓
RX Family				
TSIP	✓	-	✓	✓
TSIP-Lite	✓	-	✓	✓
RSIP-E11A	✓	-	✓	✓
RZ Family				
RZ/T2M	✓	-	✓	✓
RZ/T2ME	✓	-	✓	✓
RZ/T2L	✓	-	✓	✓
RZ/N2L	✓	-	✓	✓
TSIP	✓	-	✓	✓
Renesas Synergy Platform				
SCE7	✓	-	✓	✓
SCE5	✓	-	✓	✓

以下のファイルフォーマットの出力をサポート(Renesas key file はすべての MCU/MPU でサポートされていません):

- Renesas key file
- C source file
- Binary data
- Motorola hex file

2.1.2　Security Key Management Tool がサポートするセキュア機能

Security Key Management Tool では、セキュアな鍵のインジェクション、アップデート以外に、デバイスが持つ以下の機能で使用するデータを生成することが可能です。

サポートしている MCU/MPU は表 2-2 を参照ください。

① FSBL のための鍵証明書ならびにコード証明書の生成
② DOTF で使用可能なユーザプログラム/データ暗号化
③ Secure Factory Programming 機能で使用可能なファイル生成
④ TSIP を使用したセキュアアップデートで使用可能なファイル生成

表 2-2 MCU/MPU ファミリ セキュアな機能のサポート

ファミリ	サポート機能			
	①	②	③	④
RA Family				
RSIP-E51A Security Function and Protected Mode	✓	✓	✓	-
RSIP-E51A Compatibility Mode	-	✓	-	-
SCE9 Security Function and Protected Mode	-	-	-	-
SCE9 Compatibility Mode	-	-	-	-
SCE7	-	-	-	-
SCE5_B	-	-	-	-
SCE5	-	-	-	-
RX Family				
TSIP	-	-	-	✓
TSIP-Lite	-	-	-	✓
RSIP-E11A	-	-	-	✓
RZ Family				
RZ/T2M	-	-	-	-
RZ/T2ME	-	✓	-	-
RZ/T2L	-	-	-	-
RZ/N2L	-	-	-	-
TSIP	-	-	-	-
Renesas Synergy Platform				
SCE7	-	-	-	-
SCE5	-	-	-	-

2.2 動作環境

2.2.1 ハードウェア環境

(1) ホスト PC

- プロセッサ ：1GHz 以上
- メインメモリ：1G バイト以上
- ディスプレイ設定：解像度 1366 × 768 以上
  表示スケール 100%(推奨)

注：解像度ならびに表示スケールが上記以外の場合、すべてのオプションが表示できない場合があります。

2.2.2 ソフトウェア環境

(1) 対応 OS

- Windows 10 (64bit 版)
- Linux (Ubuntu 20.04 LTS、Ubuntu 22.04 LTS)
- macOS 14 Sonoma (Apple Silicon のみ対応)

(2) e^{2}studio plugin 版 e^{2}studio 動作確認バージョン

- e^{2}studio 2024-07

3. GUI 機能説明

この章では、Security Key Management Tool GUI の画面構成と機能について解説します。
Standalone 版と e²studio plugin 版は同じ GUI の構成です。説明は Standalone 版で行います。

3.1 メインウィンドウ

起動後のメインウィンドウは次のような構成です。

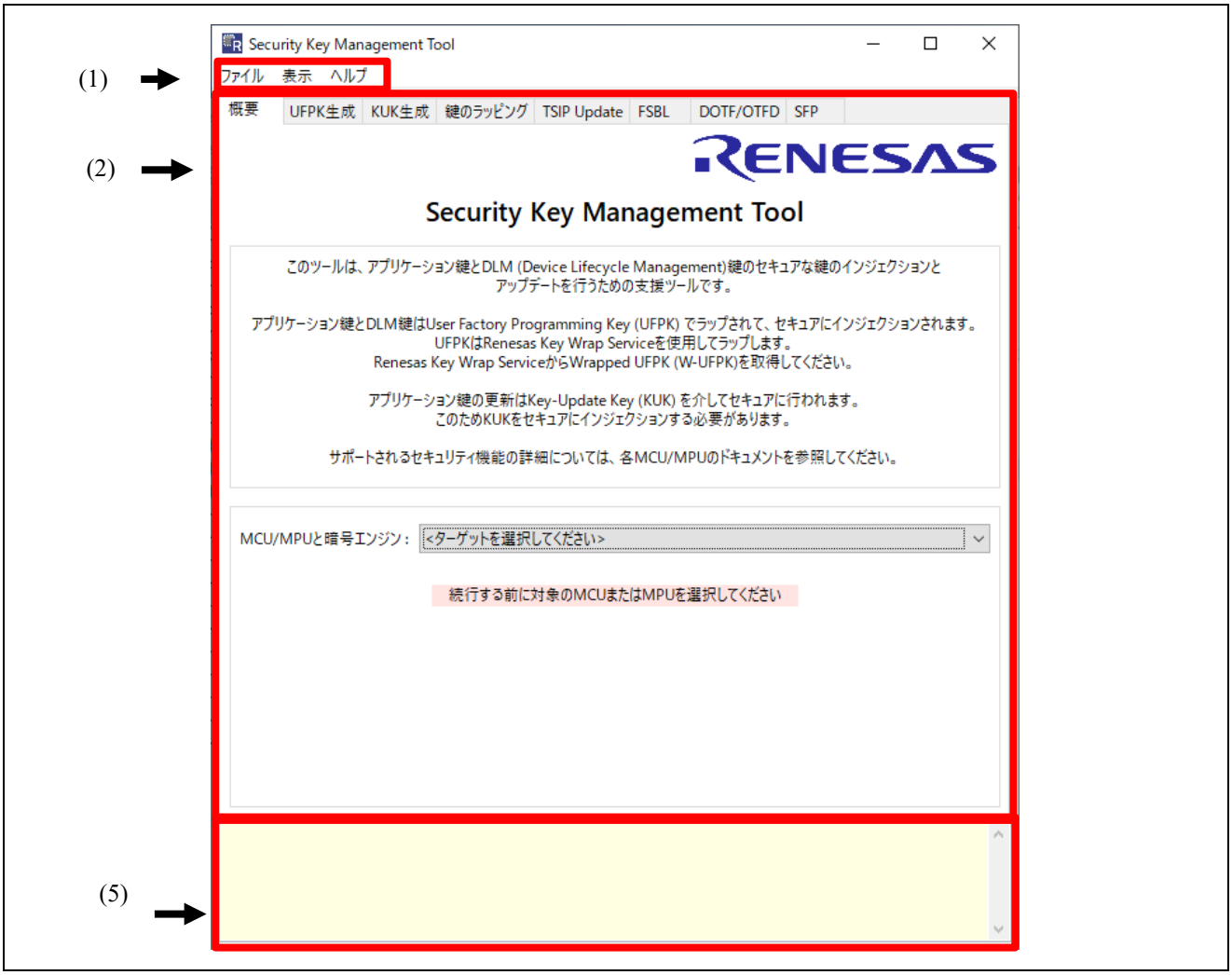

図 3.1 スタンドアロン版 メインウィンドウ

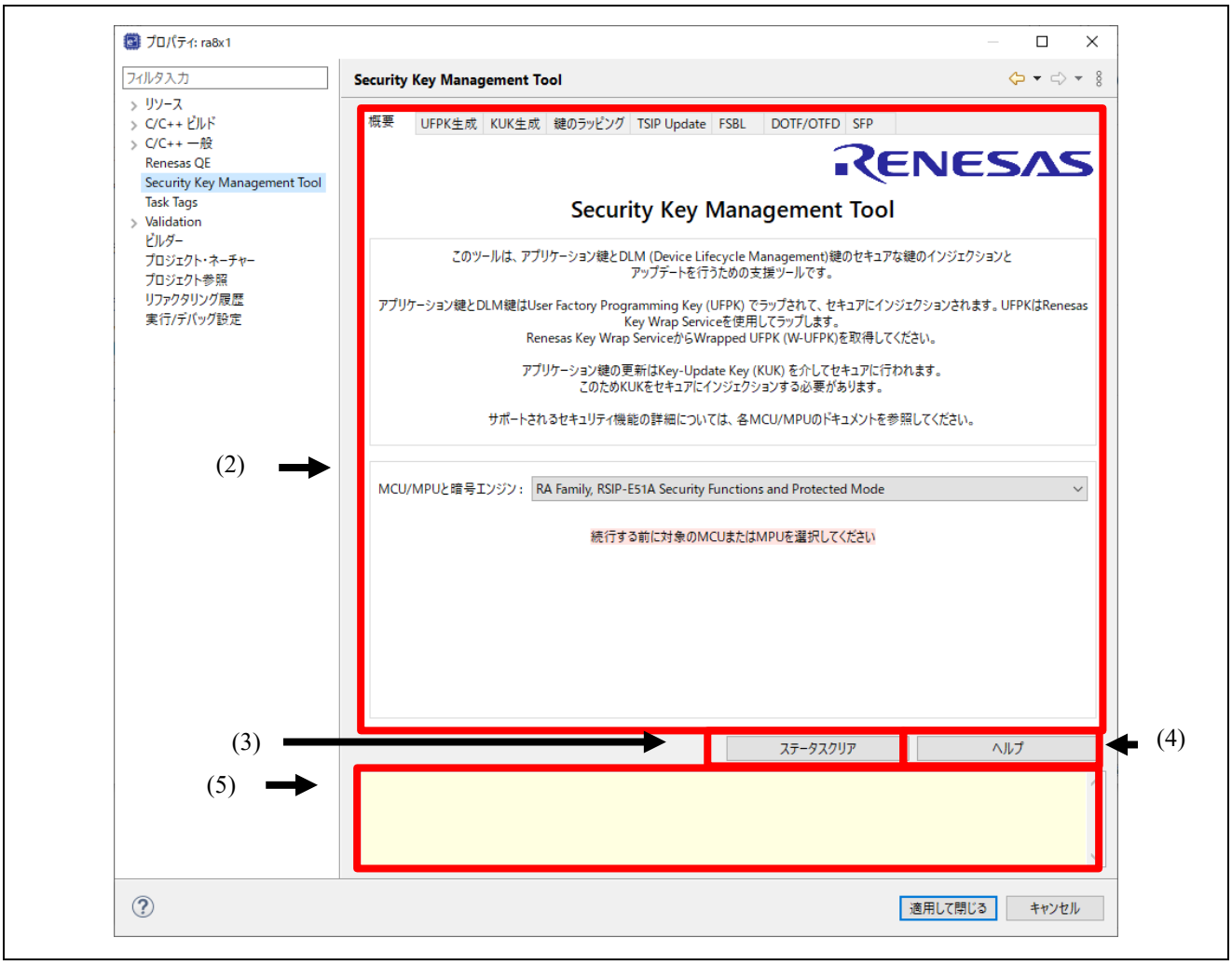

図 3.2 e²studio プラグイン版 メインウィンドウ

No	Item	Description
(1)	メニューバー	スタンドアロン版のみの機能。 詳細は 3.2.1[ファイル]メニュー を参照してください。
(2)	タブウィンドウ	各タブは、セキュアな鍵のインジェクションおよびアップデートプロセスの一部として使用されるファイルを生成するためのインタフェースを提供します。
(3)	**ステータスクリア**	ステータスウィンドウに表示している実行結果をクリアします。 スタンドアロン版はメニューの 表示 > ステータスクリア で実施します。
(4)	**ヘルプ**	Security Key Management ToolのUser's Manualや Renesasの鍵管理システムを理解するのに有益なサイト情報を表示します。 スタンドアロン版はメニューの ヘルプ > Security Key Management Toolについて… で表示します。
(5)	ステータスウィンドウ	各タブで実行した結果を表示します。

3.2　メニューバー

スタンドアロン版のみの機能になります。

3.2.1　[ファイル]メニュー

設定の保存に関する処理メニューを選択することができます。

- [**設定を保存する…**]

XML ファイル形式のファイルに、各タブで設定した入出力ファイル情報を保存します。
鍵情報は保存しません。

- [**設定を読み込む…**]

[**設定を保存する…**]で保存した設定ファイルを読み込んで、各タブに反映します。

e²studio plugin 版の場合、[**Apply and Close**]ボタンを押すことで各タブの設定情報を e²studio のプロジェクトに保存します。

3.2.2　[表示]メニュー

GUI の表示に関するメニューです。

- [**ステータスクリア**]

ステータスウィンドウの実行結果をクリアします。

3.2.3　[ヘルプ]メニュー

- [Security Key Management Tool について]

Security Key Management ToolのUser's Manualや Renesasの鍵管理システムを理解するのに有益なサイト情報を表示します。

3.3　[概要]タブ

このタブでは、使用するターゲットMCU/MPUと暗号エンジンを選択します。他のタブで操作する前に、必ずターゲットデバイスを選択してください。他のタブの機能は、このタブでのデバイス選択によって決まります。

図 3.3　[概要]タブ

No	項目	説明
(1)	MCU/MPUと暗号エンジン	使用するターゲットMCU/MPUと暗号エンジンの選択してください。 設定を保存すると、MCU/MPUと暗号エンジン情報が保存されます。

3.4 [UFPK 生成]タブ

このタブでは、User Factory Programming Key (UFPK) ファイルをバイナリ*.keyファイルとして生成します。このファイルをRenesas Key Wrap Serviceに送信して、ラップされたUFPK（W-UFPK）を取得する必要があります。 UFPKファイルは、セキュアな鍵のインジェクション用の鍵を準備するのに使用します。

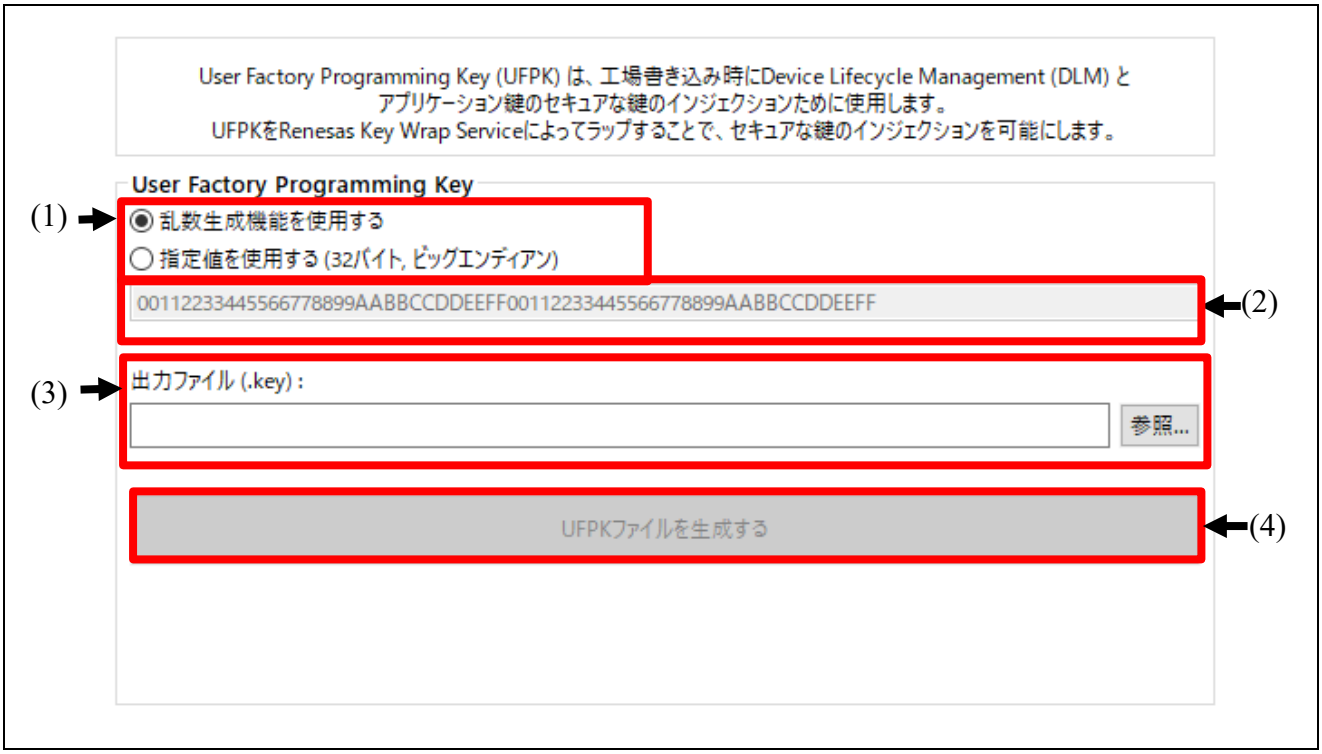

図 3.4 **[UFPK 生成]タブ**

No	項目	説明
(1)	入力するUFPKのフォーマット	UFPK値に(2)の入力値を使用するか、ツール内の乱数生成機能を使用するか選択します。 **乱数生成機能を使用する** を選択時： 　ツール内の乱数生成機能を使用してUFPKを生成します。【注】 **指定値を使用する** を選択時： 　UFPKファイルを生成するときに、(2)に入力された値を使用します。
(2)	**指定値を使用する**	(1)で**指定値を使用する**選択時に、テキストボックスに入力された値をUFPK値として、UFPKファイルを生成します。 データはヘキサデータでビックエンディアンの順序で入力してください。
(3)	**出力ファイル** (.key)	出力するUFPKファイルのパスとファイル名とパスを設定します。 出力ファイルの拡張子は`*.key`と設定してください。 設定を保存すると、出力ファイルパスが保存されます。
(4)	**UFPKファイルを生成する**	UFPKファイルを生成します。 [概要]タブでMCUと暗号エンジンを選択すると有効になります。

注 :ツール内で生成するランダム値は十分な乱数性を保証するものではありません。

　このタブで生成したUFPKファイルをRenesas Key Wrap Service(https://dlm.renesas.com/keywrap)に送信し、W-UFPKを生成します。Renesas Key Wrap Service については、Renesas Key Wrap ServiceのFAQもしくは、各MCU/MPUのアプリケーションノートもしくはドライバ、サンプルプロジェクト (表 1-1MCU/MPU関連サイト) をご参照ください。

3.5　[KUK 生成]タブ

このタブでは、Key-Update Key (KUK) ファイルをバイナリ*.keyファイルとして生成します。このファイルは、KUKのセキュアな鍵のインジェクションだけでなく、セキュアな鍵のアップデートのために他の鍵を準備するためにも使用されます。

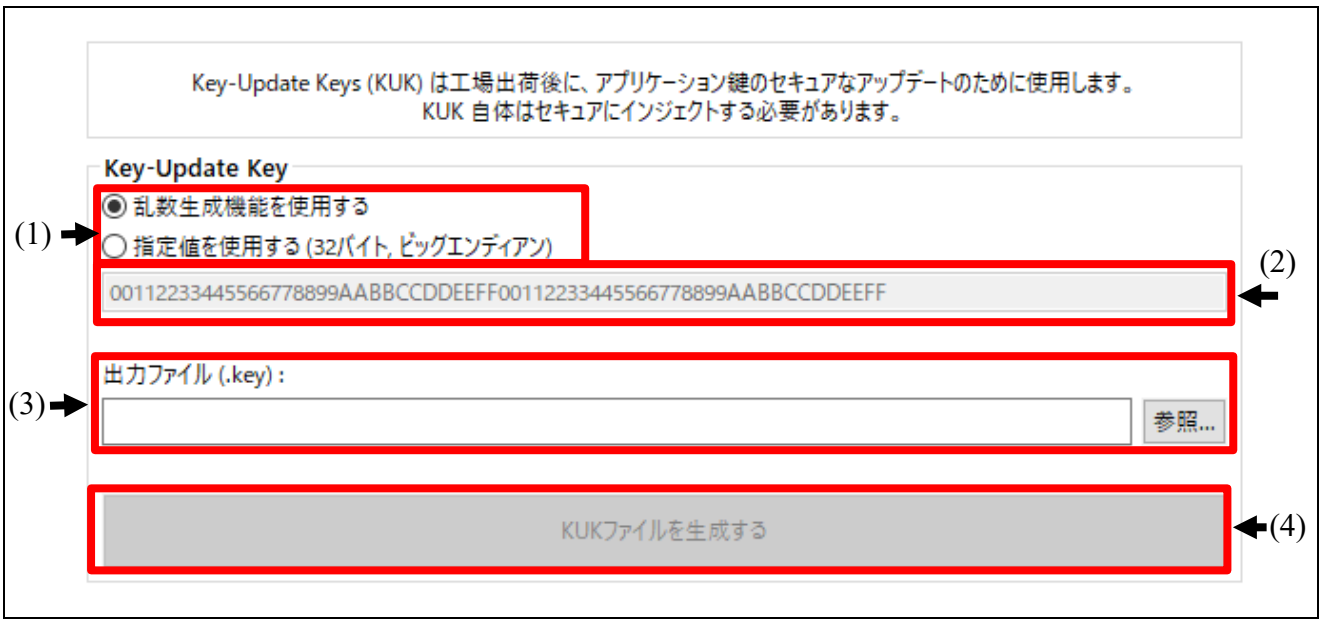

図 3.5　[**KUK 生成**]タブ

No	項目	説明
(1)	入力するKUKのフォーマット	KUK値に(2)の入力値を使用するか、ツール内の乱数生成機能を使用するか選択します。 **乱数生成機能を使用する** を選択時： 　ツール内の乱数生成機能を使用してKUKを生成します。【注】 **指定値を使用する** を選択時： 　KUKファイルを生成するときに、(2)に入力された値を使用します。
(2)	**指定値を使用する**	(1)で**指定値を使用する**選択時に、テキストボックスに入力された値をKUK値として、KUKファイルを生成します。 データはヘキサデータでビックエンディアンの順序で入力してください。
(3)	出力ファイル (.key)	出力するKUKファイルのパスとファイル名とパスを設定します。 出力ファイルの拡張子は.keyと設定してください。 設定を保存すると、出力ファイルパスが保存されます。
(4)	**KUKファイルを生成する**	KUKファイルを生成します。

注 :ツール内で生成するランダム値は十分な乱数性を保証するものではありません。

3.6 [鍵のラッピング]タブ

このタブで、ユーザ鍵もしくはDLM鍵を暗号化し、セキュアな鍵のインジェクションやアップデートに必要なファイルを生成します。

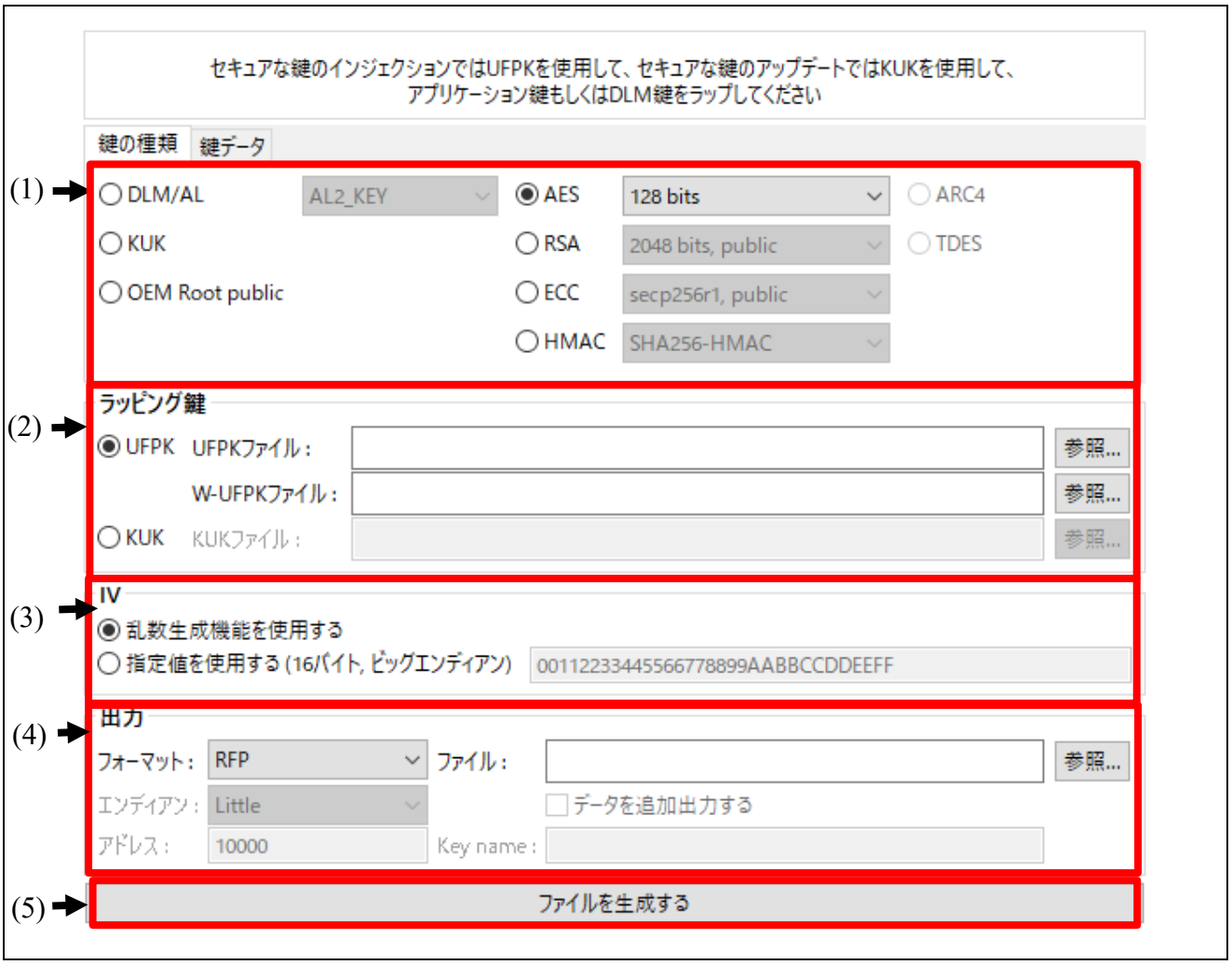

図 3.6 [鍵のラッピング]タブ

No	項目	説明
(1)	[**鍵の種類**] [**鍵データ**] タブ	[**鍵の種類**]タブで暗号化するユーザ鍵の種類を選択後、[**鍵データ**]タブで鍵データを入力します。 [**鍵の種類**]タブの詳細は3.6.1［鍵の種類］タブを参照してください [**鍵データ**]タブの詳細は3.6.2［鍵データ］タブを参照してください。
(2)	ラッピング鍵	暗号化時に使用する鍵の設定をします。 設定の詳細は3.6.3 ラッピング鍵参照してください。 設定したUFPK/W-UFPKファイルもしくはKUKファイルパスは設定保存で保存されます。
(3)	IV	暗号化時に使用するInitial Vector(IV)の設定をします。 設定の詳細は3.6.4 IVを参照してください。
(4)	出力	出力するEncrypted User Keyファイルの設定をします。 設定の詳細は3.6.5 出力を参照してください。 設定したフォーマット、ファイルパス、アドレス、key name は設定保存で保存されます。
(5)	**ファイルを生成する**	(3)で指定したフォーマットのEncrypted User Keyファイルを生成します。

3.6.1 [鍵の種類] タブ

このタブで、セキュアな鍵のインジェクションまたはアップデートのための鍵の種類を指定します。 すべての鍵の種類とオプションが、すべてのデバイスファミリーでサポートされているわけではないことをご注意ください。

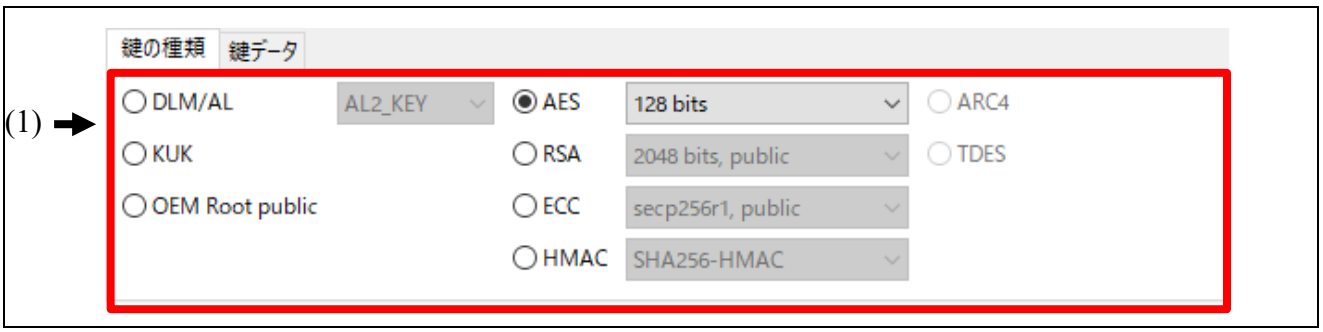

図 3.7 [**鍵の種類**]タブ

No	項目	説明
(1)	鍵の種類と鍵長選択	暗号化する鍵のアルゴリズムならびに鍵長を選択します。

鍵の種類と鍵長は以下が選択可能です。サポートしている鍵の種類と鍵長はMCU/MPUによって異なります。どの鍵並びに鍵長をサポートしているかは、各MCU/MPUのアプリケーションノートならびにドライバのマニュアル(表 1-1MCU/MPU関連サイト)をご参照ください。

表 3-1 DLM 選択時の表示項目

選択項目	説明
DLM-SSD	DLMのSSDステートの認証鍵
DLM-NSECSD	DLMのNSECSDステートの認証鍵
DLM-RMA-REQ	DLMのRMA-REQステートの認証鍵
AL2_KEY	DLM 認証レベルAL2遷移用の鍵
AL1_KEY	DLM 認証レベルAL1遷移用の鍵
RMA_KEY	DLM RMA_REQステート認証用の鍵

表 3-2 AES 選択時の表示項目

選択項目	説明
128 bits	AES-128bitの鍵
192 bits	AES-192bitの鍵
256 bits	AES-256bitの鍵
128 bits, XTS	AES-128bit XTSの鍵
256 bits, XTS	AES-256bit XTSの鍵

表 3-3 RSA 選択時の表示項目

選択項目	説明
1024 bits, public	RSA 1024bitの公開鍵
1024 bits, private	RSA 1024bitの秘密鍵
2048 bits, public	RSA 2048bitの公開鍵
2048 bits, private	RSA 2048bitの秘密鍵
3072 bits, public	RSA 3072bitの公開鍵
3072 bits, private	RSA 3072bitの秘密鍵
4096 bits, public	RSA 4096bitの公開鍵
4096 bits, private	RSA 4096bitの秘密鍵
RSA-2048-public-TLS	TLS API用のRSA 2048bitの公開鍵

表 3-4 ECC 選択時の表示項目

選択項目	説明
secp192r1, public	ECC NIST P-192の公開鍵
secp192r1, private	ECC NIST P-192の秘密鍵
secp224r1, public	ECC NIST P-224の公開鍵
secp224r1, private	ECC NIST P-224の秘密鍵
secp256r1, public	ECC NIST P-256の公開鍵
secp256r1, private	ECC NIST P-256の秘密鍵
secp384r1, public	ECC NIST P-384の公開鍵
secp384r1, private	ECC NIST P-384の秘密鍵
secp521r1, public	ECC NIST P-521の公開鍵
secp521r1, private	ECC NIST P-521の秘密鍵
brainpool P256, public	ECC brainpoolP256r1の公開鍵
brainpool P256, private	ECC brainpoolP256r1の秘密鍵
brainpool P384, public	ECC brainpoolP384r1の公開鍵
brainpool P384, private	ECC brainpoolP384r1の秘密鍵
brainpool P512, public	ECC brainpoolP512r1の公開鍵
brainpool P512, private	ECC brainpoolP512r1の秘密鍵
secp256k1, public	ECC Koblitz curve secp256k1の公開鍵
secp256k1, private	ECC Koblitz curve secp256k1の秘密鍵
Ed25519, public	EdDSA Ed25519の公開鍵
Ed25519, private	EdDSA Ed25519の秘密鍵

表 3-5 HMAC 選択時の表示項目

選択項目	説明
SHA1-HMAC	HMAC-SHA1の鍵
SHA224-HMAC	HMAC-SHA224の鍵
SHA256-HMAC	HMAC-SHA256の鍵
SHA384-HMAC	HMAC-SHA384の鍵
SHA512-HMAC	HMAC-SHA512の鍵
SHA512/224-HMAC	HMAC-SHA512-224の鍵
SHA512/256-HMAC	HMAC-SHA512-256の鍵

各鍵の設定例は6.使用例をご参照ください。

3.6.2　[鍵データ] タブ

暗号化される鍵データを入力する[**鍵データ**]タブの説明をします。

[**鍵データ**]タブで、セキュアな鍵のインジェクションやアップデートのために用意する平文の鍵データを指定します。 このタブの表示は、[**鍵の種類**]タブで指定した鍵の種類に依存します。

3.6.2.1　[鍵の種類]タブで DLM/KUK/AES/TDES/ARC4/ECC private 選択時

図 3.8　DLM / KUK / AES / TDES / ARC4 / ECC private 鍵を選択時の[**鍵データ**]タブ

No	項目	説明
(1)	入力値の選択	暗号化する鍵のデータを選択 「ファイル」選択時：(2)に鍵ファイルを入力します。 「平文データ」選択時：(3)に鍵の平文データを入力します。 「乱数を使用」選択時：(4)に生成された鍵の出力ファイル名を指定します。
(2)	ファイル	(1)で「ファイル」を選択時、暗号化する鍵ファイルを設定します。 鍵ファイルは表 4-53keyオプション ファイル入力のフォーマットの鍵を指定可能です。
(3)	平文データ	(1)で「平文データ」を選択した場合、平文の鍵データを16進数で入力します。 データサイズは、[**鍵の種類**]タブで指定した鍵の種類に依存します。
(4)	出力ファイル	(1)で「乱数を使用」を選択した場合、ツール内で鍵データを生成します。ツール内で生成した平文鍵データの出力ファイルを指定します。 出力する平文鍵ファイルは、テキストファイルもしくはバイナリファイルを指定可能です。 ECC privateの鍵生成時は、公開鍵も同時に生成し、指定したファイル名に公開鍵は_public、秘密鍵は_privateを付けたファイルを出力します。 ECC private鍵は本機能をサポートしていない鍵の種類があります。サポートしているアルゴリズムは表 3-6 鍵生成をサポートしているECC private鍵をご参照ください。 注：本ツールで生成される乱数値はランダム性を保証しません。本ツールで生成した鍵は試作もしくはテスト目的でのみ使用してください。

表 3-6 鍵生成をサポートしている ECC private 鍵の種類

鍵の種類/鍵長
secp256r1, secp384r1, secp521r1 brainpool P256, brainpool P384, brainpool P512 【注】

注: macOS 版では、brainpool の鍵生成は非サポートです。

3.6.2.2 [鍵の種類]タブで RSA 公開鍵選択時

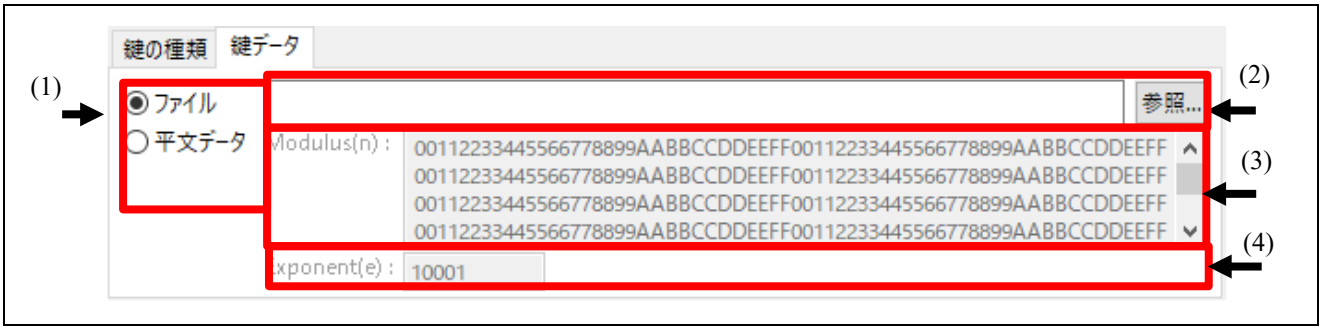

図 3.9　RSA 公開鍵を選択時の[**鍵データ**]タブ

No	項目	説明
(1)	入力値の選択	暗号化する鍵のデータを選択 「ファイル」選択時：(2)に鍵ファイルを入力します。 「平文データ」選択時：(3),(4)に鍵の平文データを入力します。
(2)	ファイル	暗号化する鍵ファイルを設定します。 鍵ファイルはバイナリファイルで、拡張子が*.keyのファイルが指定可能です。
(3)	Modulus(n)	(1)で「平文データ」選択時、RSA modulus nの平文データを16進数で入力します。
(4)	Exponent (e)	(1)で「平文データ」選択時、RSA exponent eの平文データを16進数で入力します。

3.6.2.3　[鍵の種類]タブで RSA 秘密鍵選択時

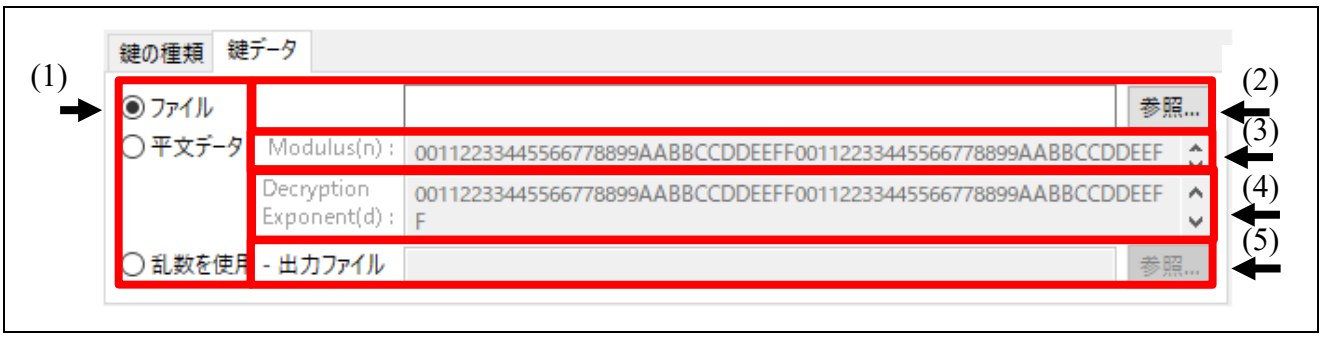

図 3.10　RSA 秘密鍵を選択時の[**鍵データ**]タブ

No	項目	説明
(1)	入力値の選択	暗号化する鍵のデータを選択 「ファイル」選択時：(2)に鍵ファイルを入力します。 「平文データ」選択時：(3),(4)に鍵の平文データを入力します。 「乱数を使用」選択時：(5)にツール内で生成する鍵の出力ファイル名を指定します。
(2)	ファイル	暗号化する鍵ファイルを設定します。 鍵ファイルはバイナリファイルで、拡張子が*.keyのファイルが指定可能です。
(3)	Modulus(n)	(1)で「平文データ」選択時、RSA modulus nの平文データを16進数で入力します。
(4)	Decryption Exponent (d)	(1)で「平文データ」選択時、RSA decryption exponent dの平文データを16進数で入力します。
(5)	**出力ファイル**	(1)でRandomを選択した場合、ツール内で鍵データを生成します。ツール内で生成した平文鍵データの出力ファイルを指定します。 出力する平文鍵ファイルは、テキストファイルもしくはバイナリファイルを指定可能です。 鍵生成時は、公開鍵も同時に生成し、指定したファイル名に公開鍵は_public、秘密鍵は_privateを付けたファイルを出力します。 注：本ツールで生成される乱数値はランダム性を保証しません。本ツールで生成した鍵は試作もしくはテスト目的でのみ使用してください。

3.6.2.4　[鍵の種類]タブで ECC public 選択時

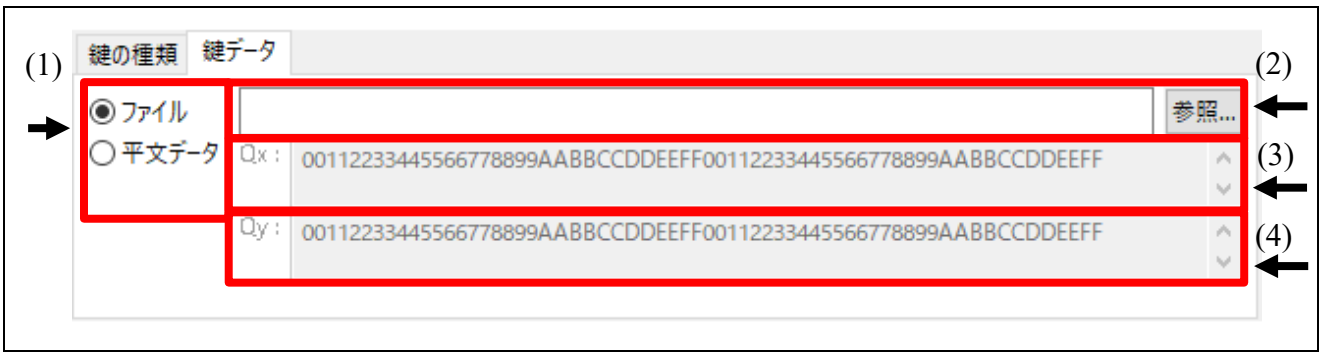

図 3.11　ECC 公開鍵を選択時の[**鍵データ**]タブ

No	項目	説明
(1)	入力値の選択	暗号化する鍵のデータを選択 「ファイル」選択時：(2)に鍵ファイルを入力します。 「平文データ」選択時：(3),(4)にQx、Qyにヘキサデータを入力します。
(2)	ファイル	暗号化する鍵ファイルを設定します。 鍵ファイルはバイナリファイルで、拡張子が*.keyのファイルが指定可能です。
(3)	Qx	(1)で「平文データ」選択時、ECC 公開鍵のQxの平文データを16進数で入力します。
(4)	Qy	(1)で「平文データ」選択時、ECC 公開鍵のQyの平文データを16進数で入力します。

3.6.2.5　[鍵の種類]タブで OEM Root public 選択時

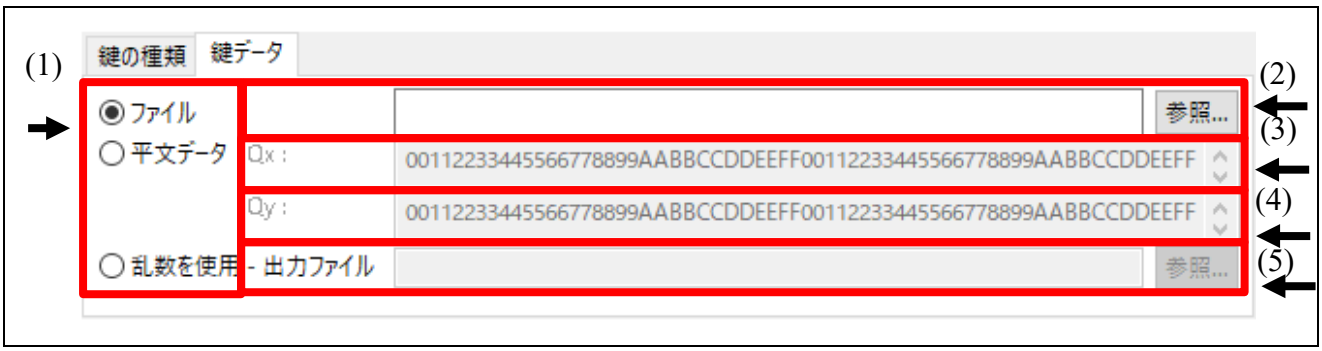

図 3.12　OEM Root public を選択時の[**鍵データ**]タブ

No	項目	説明
(1)	入力値の選択	暗号化する鍵のデータを選択 「ファイル」選択時：(2)に鍵ファイルを入力します。 「平文データ」選択時：(3),(4)にQx、Qyにヘキサデータを入力します。 「乱数を使用」選択時：(5)に生成された鍵の出力ファイル名を指定します。
(2)	ファイル	暗号化する鍵ファイルを設定します。 鍵ファイルはバイナリファイルで、拡張子が*.keyのファイルが指定可能です。
(3)	Qx	(1)で「平文データ」選択時、ECC 公開鍵のQxの平文データを16進数で入力します。
(4)	Qy	(1)で「平文データ」選択時、ECC 公開鍵のQyの平文データを16進数で入力します。
(5)	出力ファイル	(1)で「乱数を使用」を選択した場合、ツール内で鍵データを生成します。ツール内で生成した平文鍵データの出力ファイルを指定します。 出力する平文鍵ファイルは、テキストファイルもしくはバイナリファイルを指定可能です。 注：本ツールで生成される乱数値はランダム性を保証しません。本ツールで生成した鍵は試作もしくはテスト目的でのみ使用してください。

3.6.3　ラッピング鍵

セキュアな鍵のインジェクションで新しい鍵をデバイスにセットする場合、UFPKでラップされ、W-UFPKを用意する必要があります。セキュアな鍵のアップデートで新しい鍵をインジェクションする場合、KUKでラップする必要があります。

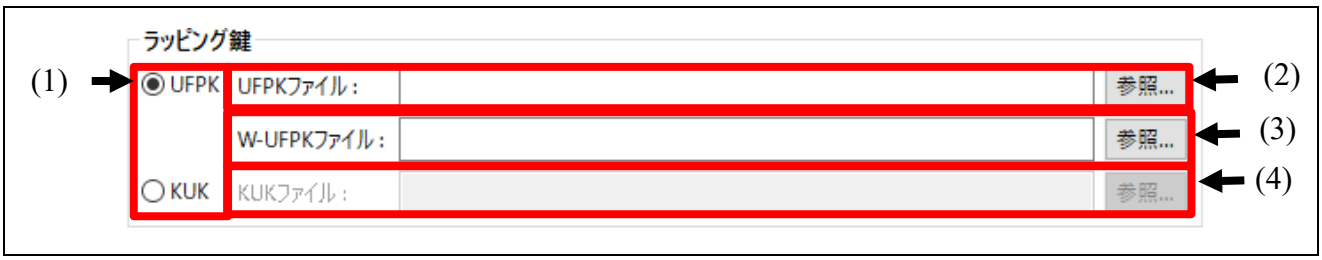

図 3.13　ラッピング鍵

No	項目	説明
(1)	Wrapping Keyの種類	Wrappingする鍵の選択 「UFPK」選択時：(2)にUFPKファイル、(3) W-UFPKファイルを選択 「KUK」選択時：(4) KUKファイルを選択
(2)	UFPK ファイル	(1)で「UFPK」選択時に、[UFPK生成]タブで生成したUFPKファイルを入力します。
(3)	W-UFPK ファイル	(1)で「UFPK」選択時に、[UFPK生成]タブで生成したUFPKファイルをRenesas Key Wrapサービスでラップした W-UFPKファイルを入力します。
(4)	KUK ファイル	(1)で「KUK」選択時に、[KUK生成]タブで生成したKUKファイルを入力します。

3.6.4 IV

Wrapping keyでUFPKもしくはKUKどちらを選択した場合も、Initial Vector(IV)が必要です。IVは指定することも、ツールで生成することも可能です。

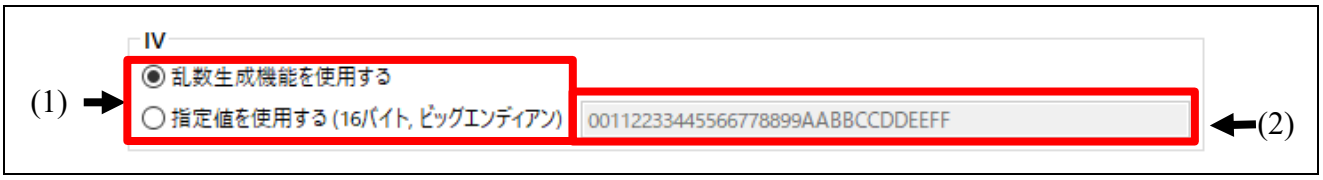

図 3.14 IV

No	項目	説明
(1)	IVのフォーマット	鍵をラッピングする時に使用するInitial Vector値を選択 「乱数生成機能を使用する」選択時：ツールの乱数生成機能からIVを生成します。 「指定値を使用する」選択時：(2)に入力されたヘキサデータを使用します。
(2)	指定値を使用する	(1) で“指定値を使用する”を選択時、ここに入力された値を暗号化時にInitial Vectorとして使用します。16進数、ビッグエンディアンフォーマットで入力してください。

3.6.5　出力

このセクションでは、セキュアな鍵のインジェクションまたはアップデートのために生成する出力ファイルの種類を指定します。すべてのオプションが、すべてのMCU/MPUファミリに対応しているわけではないことに注意してください。例えば、RAファミリのSCE9 Protected Modeの鍵のインジェクションはデバイスプログラマ経由でのみサポートされており、RFP出力フォーマットのみがサポートされています。

図 3.15　出力

No	項目	説明
(1)	フォーマット	出力するファイルフォーマットの選択 選択できるフォーマットは表 3-7を参照してください。
(2)	ファイル	出力するファイル名とパスを設定します。 非対称秘密鍵を指定し、鍵ペアの生成する場合、出力される暗号化鍵のファイル名に秘密鍵ファイルには"_private "、公開鍵ファイルには"_public"が付加されます。
(3)	エンディアン	(1)で"Motorola Hex"もしくは"Binary"選択時に有効になります。 出力するデータのデータ並びを決定します。詳細は4.5.6.**bswap** オプションを参照ください。
(4)	データを追加する	(1)で"C Source"、"Motorola Hex"もしくは"Binary"選択時に有効になります。 出力するファイル名がすでにあるファイル名の場合、そのファイルの後ろに鍵の暗号化情報を付加します。詳細は4.5.5.**fileadd** オプションを参照ください
(5)	アドレス	(1)で"Motorola Hex"選択時に有効になります。 Motorola Hexファイルに設定するアドレスを入力してください。
(6)	Key name	(1)で"C Source"を選択時に有効になります。Cソースファイルやヘッダファイルで定義されている構造体、変数、データサイズ値の<keyname>部分を指定します。<keyname>の使い方の詳細は、4.5.4.2 filetypeオプション **csource指定時** を参照してください。

表 3-7 選択可能なフォーマット

選択項目	内容
C source	Cソースファイルとヘッダを出力します。
Binary	バイナリデータを出力します。
Motorola Hex	モトローラヘキサファイルを出力します。
RFP	Renesas Key Fileフォーマットの鍵データを出力します。

表 3-8 エンディアン選択項目

選択項目	内容
Little	モトローラヘキサファイルもしくはバイナリデータを、”Big”選択時に出力するデータを4バイトスワップしたデータを出力します。
Big	モトローラヘキサファイルもしくはバイナリデータを、表 4-60、表 4-61のフォーマットに従い、データをバイト並びで出力します。

3.7 [TSIP Update]タブ

このタブで、TSIPのSecure Updateソリューションのユーザプログラムの暗号化を行います。

図 3.16 **[TSIP Update]**タブ

No	項目	説明
(1)	**出力イメージ**	出力するデータタイプの選択と入力するファイルを指定します。 「Factory Programming」 ：工場書き込み時のイメージ作成の動作 **ファームウェアイメージ**と**セキュアブートイメージ**を指定してください。 「Secure Update」　　：ファームアップデート時のイメージ作成 **ファームウェアイメージ**を指定してください。 詳細は3.7.1 出力を参照してください。
(2)	**ファームウェアイメージ/** **セキュアブートイメージ**	「ファームウェアイメージ」には暗号化するプログラムのmotファイルを入力します。 「セキュアブートイメージ」には、出力データで「Factory Programming」選択時に、暗号化したファームアップデートイメージにつけるセキュアブートプログラムのmotファイルを入力します。
(3)	設定タブ	暗号化ならびに出力データに関する設定を行います。 各タブの設定については、 3.7.3 [暗号化アドレス範囲]タブ 3.7.4 [Image Encryption Key]タブ 3.7.5 [IV]タブ 3.7.6 [RSUヘッダ]タブ をご参照ください。
(4)	**出力**	出力するファイルの設定をします。 設定の詳細は3.7.7 出力を参照してください。
(5)	**ファイルを生成する**	(4)で指定したフォーマットのファイルを生成します。

3.7.1　出力イメージ

このタブで出力するデータのフォーマットを指定します。

図 3.17　出力イメージ

No	項目	説明
(1)	**出力イメージ**	本タブで作成するファイルのタイプを選択します。

表 3-9 出力イメージの表示項目

選択項目	説明
Factory Programming	工場書き込みを想定し、セキュアブートで指定したファイルに、ファームウェアイメージを暗号化したデータを合わせたmotファイルを生成します。 セキュアブート部分は暗号化されません。書き込みの際は、セキュアな環境下で書き込むことを推奨します。
Secure Update	ファームアップデートを想定し、ファームウェアイメージに入力されたmotファイルの指定エリアを暗号化したmotファイルもしくはRSUヘッダを付属したバイナリファイルを生成します。

3.7.2　ファームウェアイメージ/セキュアブートイメージ

暗号化するファームウェアイメージならびに、暗号化したデータにつけるセキュアブートイメージを指定します。

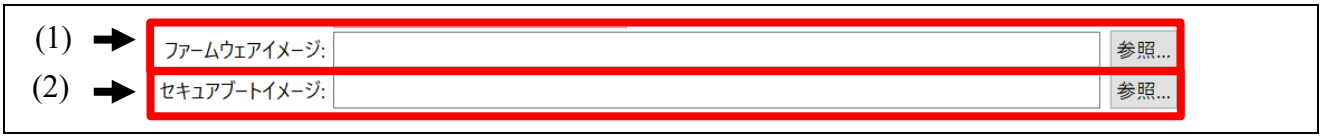

図 3.18　ファームウェアイメージ/セキュアブートイメージ

No	項目	説明
(1)	**ファームウェアイメージ**	暗号化するプログラムのmotファイルを指定します。 指定したmotファイル内の「暗号化アドレス範囲」で指定した範囲のデータをTSIP Firm Updateソリューション用の暗号化を行います。暗号式はTSIPのApplication Noteを参照してください。
(2)	**セキュアブートイメージ**	出力データで”Factory Programming”指定時に、暗号化したファームウェアイメージと合わせるセキュアブートイメージを指定します。 セキュアブートイメージは暗号化されません。

3.7.3　[暗号化アドレス範囲]タブ

ファームウェアイメージの暗号化する範囲を指定します。

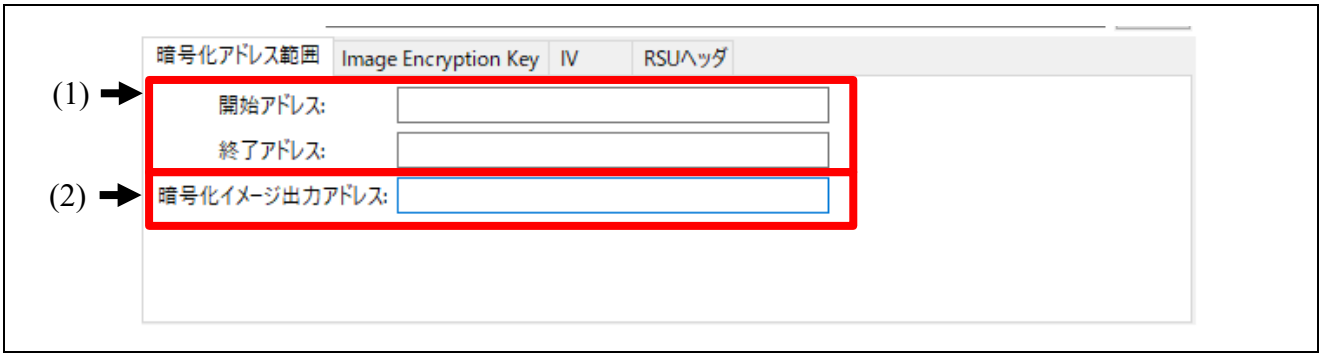

図 3.19　**[暗号化アドレス範囲]**タブ

No	項目	説明
(1)	**暗号化開始アドレス/暗号化終了アドレス**	ファームウェアイメージで指定したmotファイル内の暗号化する範囲を指定します。暗号化する範囲の上限は8MBです。
(2)	**暗号化イメージ出力アドレス**	出力ファイルにMOTを指定時に、暗号化したファームウェアイメージを出力するアドレスを指定します。

3.7.4　[Image Encryption Key]タブ

ファームウェアイメージを暗号化するときに使用する鍵データを指定します。

図 3.20　**[Image Encryption Key]**タブ

No	項目	説明
(1)	**Key Encryption Key**	暗号化する鍵(Image Encryption Key)を暗号化する鍵を指定します。16進数、ビッグエンディアンフォーマットで入力してください。
(2)	Image Encryption Key のフォーマット	ファームウェアイメージを暗号化する鍵値を選択 「乱数生成機能を使用する」選択時：ツールの乱数生成機能から鍵を生成します。 「指定値を使用する」選択時：(3)に入力されたヘキサデータを使用します。
(3)	**指定値を使用する**	(2) で“指定値を使用する”を選択時、ここに入力された値を暗号化する鍵として使用します。16進数、ビッグエンディアンフォーマットで入力してください。

3.7.5 [IV]タブ

図 3.21 **[IV]**タブ

No	項目	説明
(1)	IVのフォーマット	ファームウェアイメージを暗号化時に使用するInitial Vector値を選択 「乱数生成機能を使用する」選択時：ツールの乱数生成機能からIVを生成します。 「指定値を使用する」選択時：(2)に入力されたヘキサデータを使用します。
(2)	**指定値を使用する**	(1) で“指定値を使用する”を選択時、ここに入力された値をファームウェアイメージの暗号化時にInitial Vectorとして使用します。16進数、ビッグエンディアンフォーマットで入力してください。

3.7.6 [RSU ヘッダ]タブ

このセクションでは、RSUヘッダの出力するファイルフォーマットを指定します。

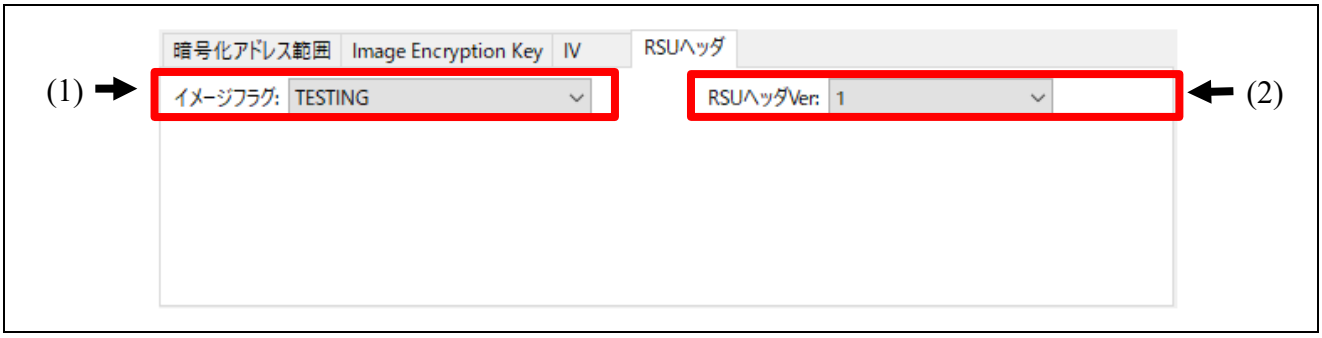

図 3.22 **[RSU ヘッダ]**タブ

No	項目	説明
(1)	**イメージフラグ**	RSUヘッダのImage Flagsに設定する値を選択します。 (2)で"2"を選択した場合、イメージフラグは指定不要のため、指定できません。 選択できる値は 表 3-10 を参照してください。
(2)	**RSUヘッダVer**	暗号化したファームウェアイメージにつけるRSUヘッダのバージョンを指定します。Versionは1もしくは2を指定可能です。 RSUヘッダVerについては、4.6.2 **ver** オプションを参照してください。

表 3-10 RSU ヘッダ イメージフラグ

選択項目	説明
BLANK	ファームアップイメージがない。
TESTING	ファームアップイメージがアップデート中、検証中。
INSTALLING	ファームアップイメージが初期イメージをインストール中であり、検証中。
VALID	ファームアップイメージが有効。
INVALID	ファームアップイメージが無効。
END_OF_LIFE	ファームアップイメージは End of Life。

3.7.7 出力

このセクションでは、出力するファイルフォーマットを指定します。

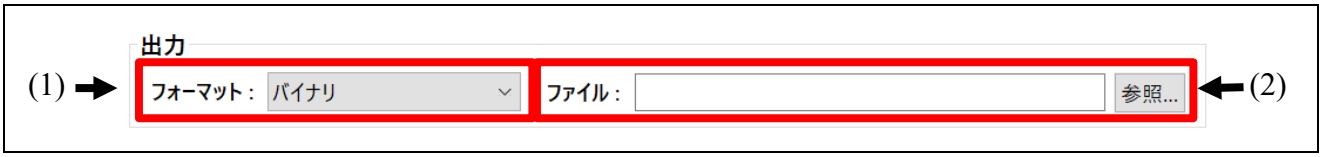

図 3.23 出力

No	項目	説明
(1)	**フォーマット**	出力するファイルフォーマットの選択 選択できるフォーマットは表 3-11 を参照してください。
(2)	**ファイル**	出力するファイル名とパスを設定します。

表 3-11 選択可能なフォーマット

選択項目	内容
バイナリ	RSU ファイルヘッダを付けたバイナリデータを出力します。 RSU は、出力データで”Secure Update”指定時のみ指定可能。
モトローラヘキサ	モトローラヘキサファイルを出力します。

3.8 [FSBL]タブ

このタブで、First Stage Bootloader(FSBL)機能を搭載したルネサスデバイスで使用可能な鍵証明書とコード証明書の生成を行います。

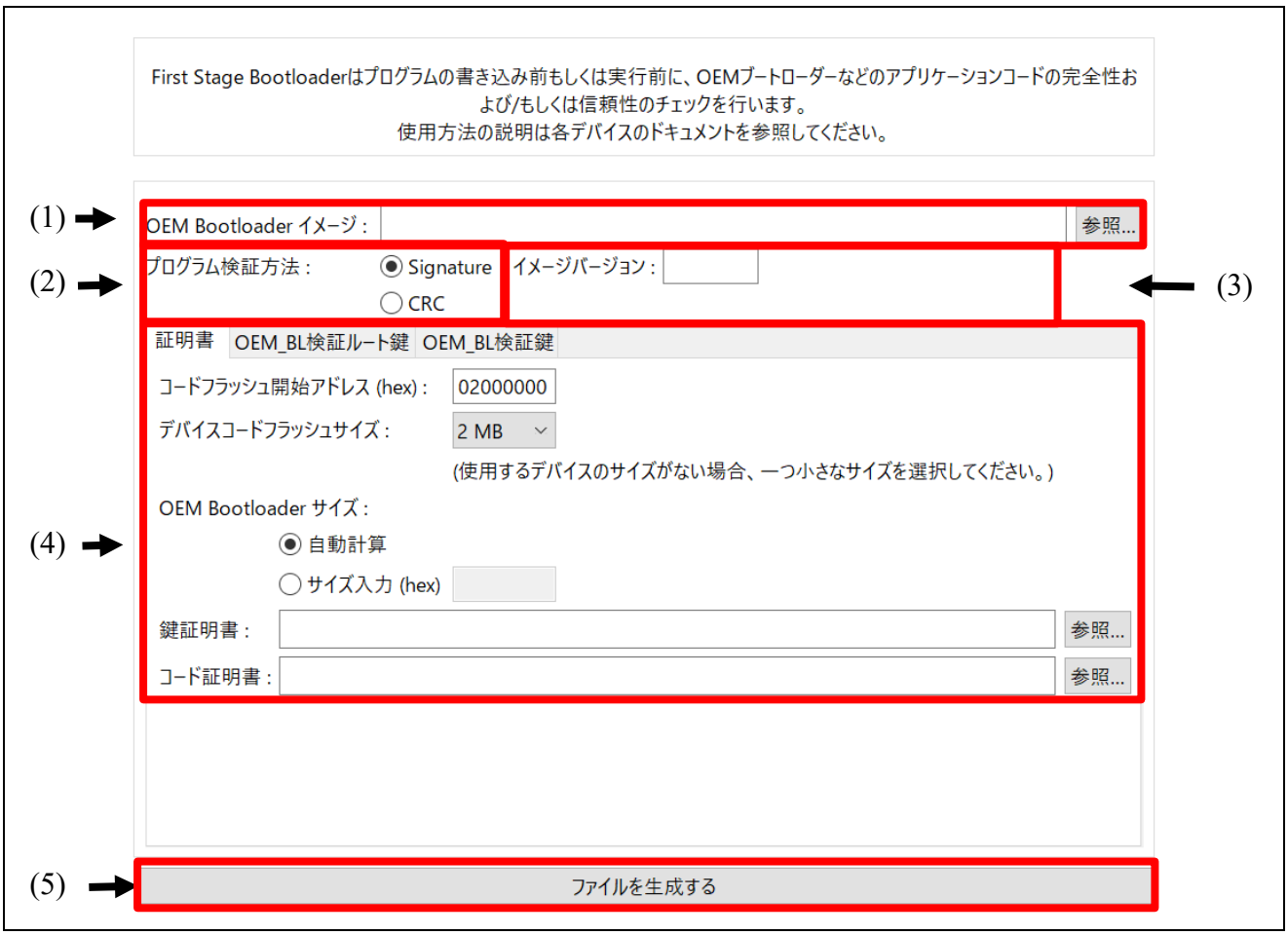

図 3.24 **[FSBL]タブ**

No	項目	説明
(1)	OEM Bootloaderイメージ	FSBLが検証対象とするOEM_BLのモトローラヘキサファイルを指定します。
(2)	プログラム検証方法	「Signature」: ECDSA署名を使用したコード証明書と鍵証明書を生成します。 「CRC」　　: CRCを使用した簡易的な検証を行うためのコード証明書のみを作成します。 詳細は3.8.1 プログラム検証方法を参照してください。
(3)	イメージバージョン	プログラム検証方法で”Signature”選択時に、コード証明書に記載するOEM Bootloaderのバージョンを指定します。
(4)	[**証明書**] [**OEM_BL検証ルート鍵**] [**OEM_BL検証鍵**] タブ	[**証明書**]タブ：OEM_BL情報と鍵証明書とコード証明書の出力ファイル名を指定します。 [**OEM_BL検証ルート鍵**]タブと[**OEM_BL検証鍵**]タブ： プログラム検証方法で「Signature」選択時に、鍵情報を入力します。 [**証明書**]タブの詳細は3.8.2 [証明書]タブを参照してください [**OEM_BL検証ルート鍵**]タブの詳細は3.8.3 [**OEM_BL検証ルート鍵**]タブを参照してください。 [**OEM_BL検証鍵**]タブの詳細は3.8.4 [**OEM_BL検証鍵**]タブを参照してください。
(5)	**ファイルを生成する**	(4)で指定した鍵証明書ならびにコード証明書を生成します。

3.8.1　　プログラム検証方法

OEM Bootloaderの検証方法を指定します。

図 3.25　プログラム検証方法

No	項目	説明
(1)	**Signature**	OEM_BLの署名をするための鍵(OEM_BL検証鍵)と、OEM_BL検証鍵を証明する鍵(OEM_BL検証ルート鍵)を使用して、鍵証明書とコード証明書を生成します。 「Signature」を選択時は、[**OEM_BL検証ルート鍵**]タブからOEM_BL検証ルート鍵、[**OEM_BL検証鍵**]タブからOEM_BL検証鍵の入力が必要です。 証明書のChain Of Trust構造は、FSBL機能を実装するデバイスのUser Manualをご参照ください。
(2)	**CRC**	OEM_BLのCRC値を計算し、コード証明書のみを生成します。【注】

注：「**CRC**」を指定してコード証明書を生成した場合、Signer ID にダミー値として CRC 値を出力します。FSBL 動作モードで”CRC + report measurement”選択し、OEM_BL 検証鍵を用いた measurement report を出力したい場合は、「**Signature**」を選んでコード証明書を生成してください。

3.8.2　[証明書]タブ

OEM Bootloaderの情報ならびに、鍵証明書とコード証明書のファイル名を指定します。

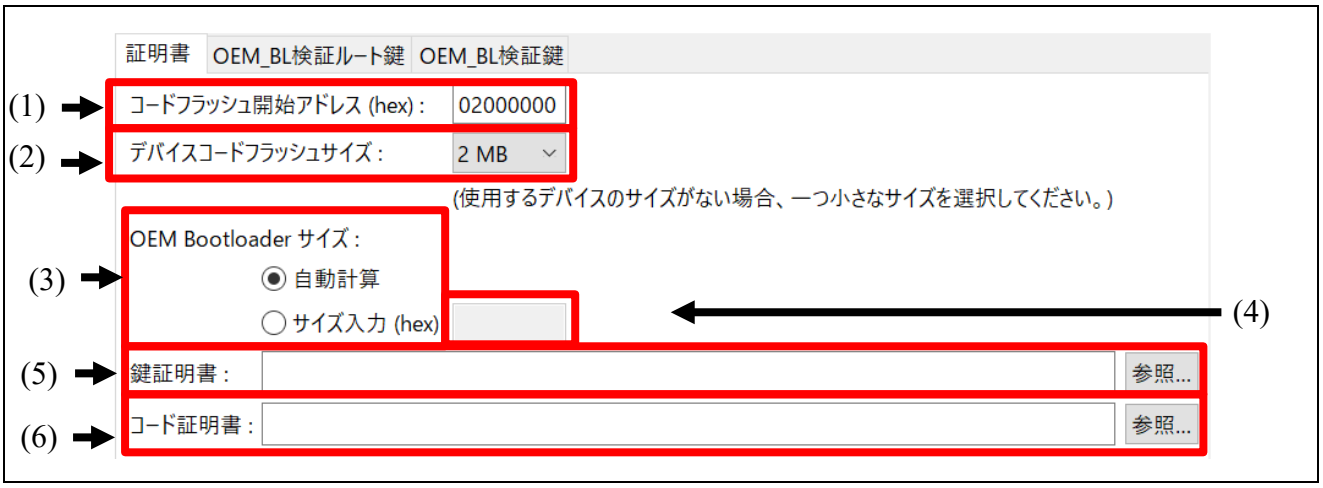

図 3.26　[**証明書**]タブ

No	項目	説明
(1)	コードフラッシュ開始アドレス	OEM_BLの開始アドレスを指定します。
(2)	デバイスコードフラッシュサイズ	使用するデバイスのコードフラッシュのサイズを指定してください。 該当サイズがない場合は、近いサイズを指定してください。
(3)	OEM Bootloaderサイズ	OEM_BLサイズの算出方法を設定します。 「自動計算」 選択時: OEM_BLイメージで入力されたmotファイルと**デバイスコードフラッシュサイズ**からOEM_BLのサイズを算出します。 OEM_BLのサイズ算出方法は4.7.2 OEM_BLの署名もしくはCRC演算対象領域を参照してください。 「サイズ入力」 選択時 : (4)に入力された値がOEM_BLサイズになります。
(4)	OEM Bootloaderサイズ	(3)で「サイズ入力」を選択時、OEM_BLのサイズを入力します。 サイズは16の倍数になるサイズ-1の値を入力してください。
(5)	鍵証明書	鍵証明書の出力ファイル名を指定します。 **プログラム検証方法**で「Signature」を選択時に入力してください。
(6)	コード証明書	コード証明書の出力ファイル名を指定します。

3.8.3　[OEM_BL 検証ルート鍵]タブ

プログラム検証方法で「Signature」選択時に、OEM_BL検証ルート鍵を入力します。

図 3.27　**[OEM_BL 検証ルート鍵]**タブ

No	項目	説明
(1)	OEM_BL 検証ルート秘密鍵 入力方法選択	OEM_BL検証ルート秘密鍵の入力方法を選択 「ファイル」選択時：(2)に鍵ファイルを入力します。 「平文データ」選択時：(3)に鍵の平文データを入力します。
(2)	ファイル	OEM_BL検証ルート秘密鍵の鍵ファイルを設定します。 鍵ファイルは公開鍵情報を含むPEMファイル、テキストファイルで拡張子が*.txtのファイル、もしくはバイナリファイルで、拡張子が*.keyのファイルが指定可能です。 PEMファイルを指定時は、**OEM_BL検証ルート公開鍵**の入力は不要です。
(3)	平文データ	(1)で「平文データ」を選択した場合、平文の鍵データを16進数で入力します。 OEM_BL検証ルート秘密鍵がsecp256r1の場合は、秘密鍵のサイズ、32バイトのデータを入力してください。
(4)	OEM_BL検証ルート公開鍵 入力方法選択	OEM_BL検証ルート公開鍵の入力方法を選択 「ファイル」選択時：(5)に鍵ファイルを入力します。 「平文データ」選択時：(6)、(7)に鍵の平文データを入力します。
(5)	ファイル	OEM_BL検証ルート公開鍵の鍵ファイルを設定します。 鍵ファイルは、テキストファイルで拡張子が*.txtのファイルもしくは、バイナリファイルで拡張子が*.keyのファイルが指定可能です。
(6)	Qx	(4)で「平文データ」を選択した場合、OEM_BL検証ルート公開鍵Qxの平文鍵データを16進数で入力します。
(7)	Qy	(4)で「平文データ」を選択した場合、OEM_BL検証ルート公開鍵Qyの平文鍵データを16進数で入力します。

3.8.4　[OEM_BL 検証鍵]タブ

プログラム検証方法で「Signature」選択時に、OEM Bootloader Keyを入力します。

図 3.28　[**OEM_BL 検証鍵**]タブ

No	項目	説明
(1)	OEM_BL検証秘密鍵 入力方法選択	OEM_BL検証秘密鍵の入力方法を選択 「ファイル」選択時：(2)に鍵ファイルを入力します。 「平文データ」選択時：(3)に鍵の平文データを入力します。 「乱数を使用」選択時：(4)に生成された鍵の出力ファイル名を指定します。
(2)	ファイル	OEM_BL検証秘密鍵の鍵ファイルを設定します。 鍵ファイルは公開鍵情報を含むPEMファイル、テキストファイルで拡張子が*.txtのファイル、もしくは、バイナリファイルで、拡張子が*.keyのファイルが指定可能です。 PEMファイルを指定時は、**OEM_BL検証公開鍵**の入力は不要です。
(3)	平文データ	(1)で「平文データ」を選択した場合、平文の鍵データを16進数で入力します。 OEM_BL検証秘密鍵がsecp256r1の場合は、秘密鍵のサイズ、32バイトのデータを入力してください。
(4)	出力ファイル	(1)で「乱数を使用」を選択した場合、ツール内で鍵データを生成します。ツール内で生成した平文鍵データの出力ファイルを指定します。 出力する平文鍵ファイルは、テキストファイルもしくはバイナリファイルを指定可能です。 公開鍵も同時に生成し、指定したファイル名に公開鍵は_public、秘密鍵は_privateを付けたファイルを出力します。

No	項目	説明
(5)	OEM_BL検証公開鍵 入力方法選択	OEM_BL検証公開鍵の入力方法を選択 「ファイル」選択時：(6)に鍵ファイルを入力します。 「平文データ」選択時：(7)と(8)に鍵の平文データを入力します。
(6)	ファイル	OEM_BL検証公開鍵の鍵ファイルを設定します。 鍵ファイルは、バイナリファイルで拡張子が.keyのファイル、もしくはテキストファイルで拡張子が*.txtのファイルが指定可能です。
(7)	Qx	(5)で「平文データ」を選択した場合、OEM_BL公開鍵Qxの平文鍵データを16進数で入力します。
(8)	Qy	(5)で「平文データ」を選択した場合、OEM_BL公開鍵Qyの平文鍵データを16進数で入力します。

3.9　[DOTF/OTFD]タブ

このタブは、DOTF機能を搭載したルネサスデバイスで使用可能な暗号化データユーザ(実行可能なバイナリデータまたはアプリケーションデータ)を暗号化するために使用します。

図 3.29　**[DOTF/OTFD]タブ**

No	項目	説明
(1)	平文イメージ	暗号化するイメージを含むモトローラヘキサファイルを指定します。
(2)	暗号化範囲	(1)で指定されたファイル内の暗号化する範囲を指定します。 詳細は3.9.1 暗号化範囲を参照してください。
(3)	転送先アドレス	実際にデータを配置するもしくは実行をするアドレスを指定します。 詳細は3.9.2 転送先アドレスを参照してください。
(4)	イメージ暗号化鍵	暗号化で使用する鍵を指定します。 詳細は3.9.3 イメージ暗号化鍵を参照してください。
(5)	IV	暗号化で使用するnonce値を指定します。 詳細は3.9.4 IVを参照してください。
(6)	暗号化イメージ	暗号化されたデータの出力ファイル名を指定します。
(7)	出力イメージのアドレスとコンテンツ	出力するファイルのオプションを指定します。 詳細は3.9.5 出力イメージアドレスとコンテンツを参照してください。
(8)	暗号化イメージファイル生成	(6)で指定した暗号化イメージファイルを生成します。

3.9.1　暗号化範囲

入力された平文イメージファイル内で暗号化する範囲を指定します。

図 3.30　暗号化範囲

No	項目	説明
(1)	暗号化範囲選択	「全データ暗号化」選択時：平文イメージで入力されたファイルの全データを暗号化します。(*注1) 「暗号化アドレス」選択時：平文イメージで入力されたファイル内で(2)で指定された範囲を暗号化します。(*注2)
(2)	暗号化範囲	「暗号化開始アドレス」：暗号化を開始するアドレスを指定します。 「暗号化終了アドレス」：暗号化を終了するアドレスを指定します。 開始アドレスは16バイトアラインアドレス、終了アドレスは16バイトアラインアドレス-1のアドレスを入力してください。

注1：入力ファイルの開始アドレス、終了アドレスが16バイトアラインを取れていない場合、もしくは入力イメージ内にデータがないエリアがある場合、暗号化を実施する範囲内にデータがない箇所は、00のデータでデータを補完します。

注2：指定されたエリア内に入力されたイメージのデータがないエリアがある場合、00のデータでデータを補完します。

3.9.2　転送先アドレス

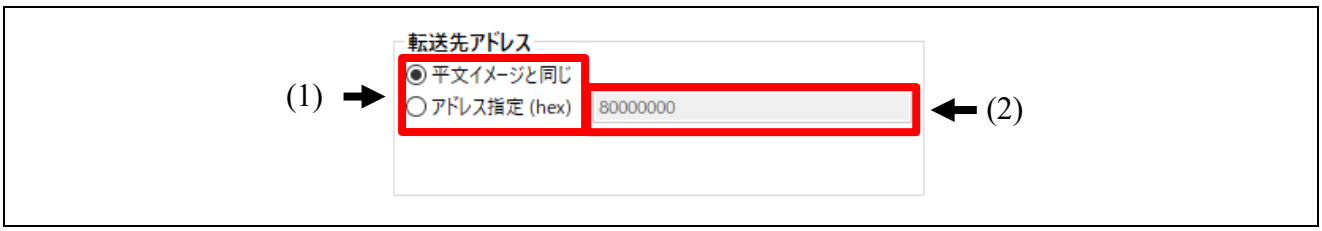

図 3.31　転送先アドレス

No	項目	説明
(1)	転送先アドレス選択	「平文イメージと同じ」選択時：平文イメージで入力されたアドレスを転送先アドレスとして指定します。 「アドレス指定」選択時：指定されたアドレスを転送先アドレスとして指定します
(2)	転送先アドレス	(1)で「アドレス指定」を選択時に、暗号化するプログラムが転送されるアドレスを指定します。

3.9.3　　イメージ暗号化鍵

ファームウェアイメージを暗号化するときに使用する鍵データを指定します。

図 3.32　イメージ暗号化鍵

No	項目	説明
(1)	鍵長指定	暗号化で使用する鍵長を指定します。 「AES128」、「AES192」、「AES256」から指定可能です。
(2)	鍵のフォーマット	イメージ暗号化鍵の入力方法を選択 「ファイル」選択時：(3)に鍵ファイルを入力します。 「平文データ」選択時：(4)に鍵の平文データを入力します。
(3)	ファイル	(2)で「ファイル」を選択時、イメージ暗号化鍵の鍵ファイルを設定します。バイナリファイルで、拡張子が*.key、もしくはテキストファイルで、拡張子が*.txtのファイルが指定可能です。
(4)	平文データ	(2)で「平文データ」を選択した場合、平文の鍵データを16進数で入力します。 (1)で選択した鍵長に合わせたサイズの鍵データを入力してください。

3.9.4　　IV

ファームウェアイメージを暗号化するときに使用するIV(Counter値)を指定します。

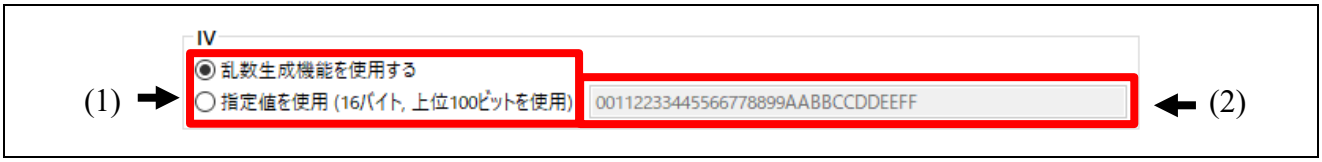

図 3.33　IV

No	項目	説明
(1)	IVのフォーマット	ファームウェアイメージを暗号化するときに使用するInitial Vector値を選択 「乱数生成機能を使用する」選択時：ツールの乱数生成機能からIVを生成します。 「指定値を使用」選択時：(2)に入力されたヘキサデータを使用します。
(2)	指定値を使用	(1) で「指定値を使用」を選択時、ここに入力された値の上位100bitを暗号化時にInitial Vectorとして使用します。16進数、ビッグエンディアンフォーマットで入力してください。

3.9.5　出力イメージアドレスとコンテンツ

図 3.34　出力イメージアドレスとコンテンツ

No	項目	説明
(1)	アドレスの指定	出力するイメージファイルのアドレスを指定します。 「入力ファイルのアドレスを使用」選択時：出力ファイルの開始アドレスは入力されたファイルと同じアドレスを設定します。 「開始アドレス 0」選択時：出力ファイルの開始アドレスは0番地に設定します。
(2)	暗号化したデータ以外の領域に、平文データを含める	(1) で「入力ファイルのアドレスと同じ」を選択時、本機能を選択可能です。 チェックボックスにチェックを入れた場合、入力ファイルで、暗号化対象範囲外のデータを平文データのまま、出力ファイルに含めることができます。

3.10 [SFP]タブ

このタブでは、Secure Factory Programming機能で書き込むためのプログラムとパラメータ情報を暗号化して、ファイル出力します。

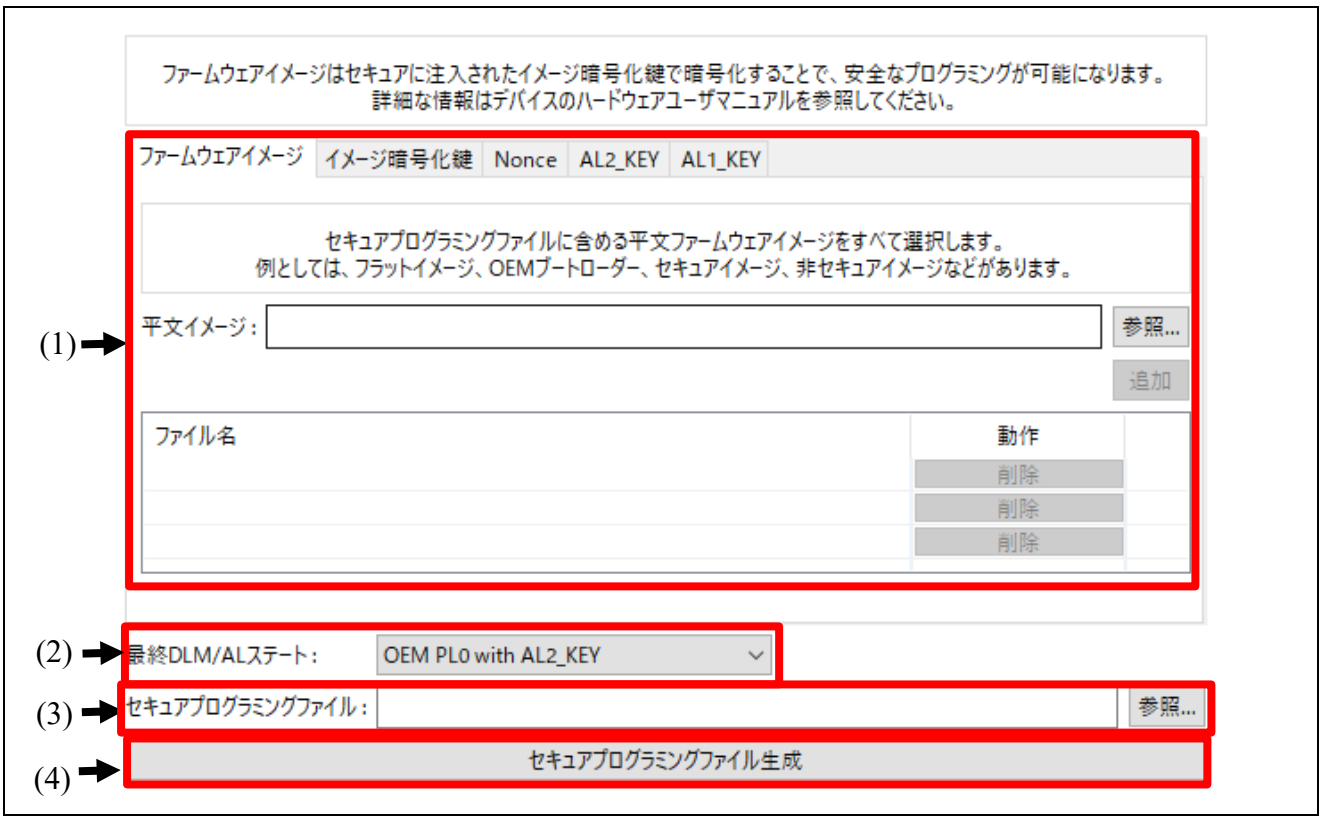

図 3.35 [**SFP**]タブ

No	項目	説明
(1)	[**ファームウェアイメージ**] [**イメージ暗号化鍵**] [**Nonce**] [**AL2_KEY**] [**AL1_KEY**] タブ	Secure Factory Programmingファイル生成に使用する各種データを入力します。 [**ファームウェアイメージ**]タブの詳細は3.10.1 [ファームウェアイメージ]タブを参照してください。 [**イメージ暗号化鍵**]タブの詳細は3.10.2[イメージ暗号化鍵]タブを参照してください。 [**Nonce**]タブの詳細は3.10.3 [Nonce]タブを参照してください。 [**AL2_KEY**] タブの詳細は3.10.4 [AL2_KEY]タブを参照してください。 [**AL1_KEY**] タブの詳細は3.10.5 [AL1_KEY]タブを参照してください。
(2)	最終 DLM/AL ステート	DLM遷移先情報を指定します。 選択可能な設定の詳細は表 3-12を参照してください。
(3)	セキュアプログラミングファイル	Secure Factory Programmingファイルの出力パスと名前を指定します。
(4)	**セキュアプログラミングファイル生成**	(1)~(3)で指定した情報でSecure Factory Programming ファイルを生成します。

表 3-12 最終DLM/ALステート

選択項目	説明
OEM PL0 with AL2_KEY	PL0への転送かつAL2鍵書き込み AL1_KEYタブは使用不可になります。
LCK_BOOT	LCK_BOOT AL2_KEY, AL1_KEYタブは使用不可になります。

注: Secure Factory Programming 機能をサポートする MCU/MPU で、すべてのオプションをサポートしているわけではありません。どのオプションをサポートしているかは、MCU/MPU の Hardware User Manual をご参照ください。

3.10.1　[ファームウェアイメージ]タブ

暗号化するファイルを指定します。

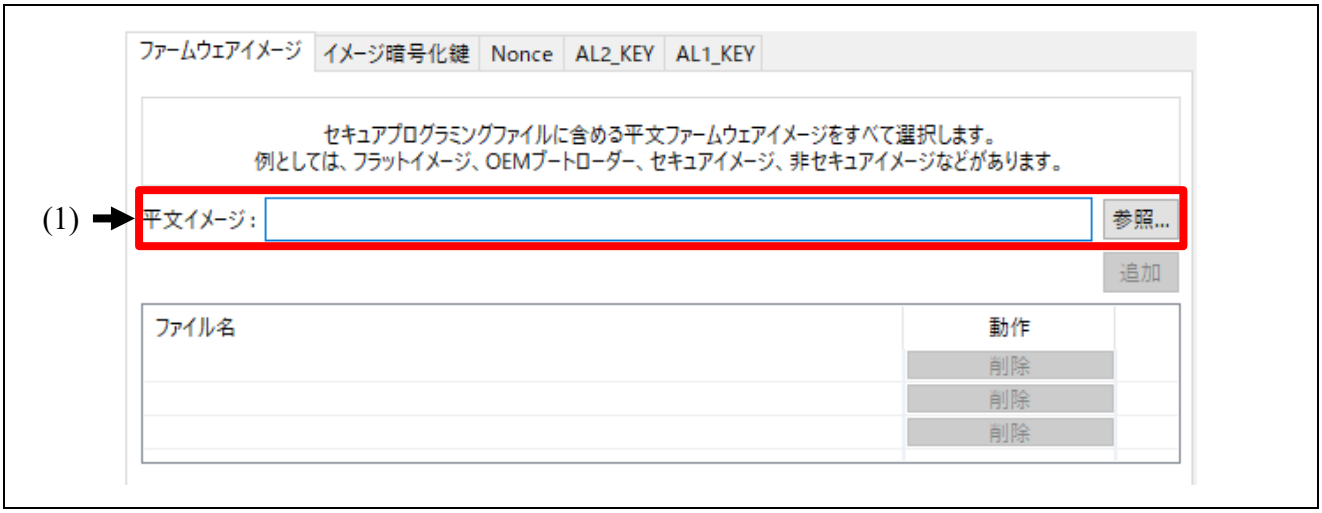

図 3.36　[**ファームウェアイメージ**]タブ

No	項目	説明
(1)	平文イメージ	暗号化するユーザプログラムを指定します。 **参照**ボタンでファイルを選択し、**追加**ボタンで一覧にファイルを追加します。一覧の**削除**ボタンで追加したファイルを削除できます。 最大3ファイルまで指定可能です。

3.10.2　[イメージ暗号化鍵]タブ

ファームウェアイメージの暗号化で使用する鍵ならびに、その鍵の暗号化で使用するIV、UFPK、W-UFPKを指定します。

図 3.37　[**イメージ暗号化鍵**]タブ

No	項目	説明
(1)	鍵データ	ユーザプログラム、パラメータ情報の暗号化で使用するAES128bit鍵データを指定します。 「ファイル」を選択時 : (2)で鍵ファイルを指定します。 「平文データ」を選択時 : (3)で入力したヘキサデータを鍵として使用します。 「乱数を使用」を選択時 : 鍵データを自動生成します。(4)で自動生成する鍵データのファイル名と出力パスを設定します。
(2)	鍵データ ファイル	(1)で「ファイル」を選択時、AES128bit鍵データファイルを指定します。 バイナリ(`*.key`)、もしくはテキストファイル(`*.txt`)の指定が可能です。
(3)	鍵データ 平文データ	(1)で「平文データ」を選択時、AES128bit鍵をヘキサデータで入力します。
(4)	鍵データ 乱数を使用	(1)で「乱数生成機能を使用する」を選択時、自動生成する鍵データのファイル名と出力パスを設定します。
(5)	IV	鍵データを暗号化する時の初期化ベクタを選択します。 「乱数生成機能を使用する」選択時 : ツールの乱数からIVを生成します。 「指定値を使用する」選択時 : (6)に入力されたヘキサデータを使用します。
(6)	IV 指定値を使用する	(5)で「指定値を使用する」選択時、ここに入力された値を鍵データ暗号化時に初期化ベクタとして使用します。
(7)	UFPK File	暗号化に使用するUFPKファイルを選択します。
(8)	W-UFPK File	W-UFPKファイルを選択します。

3.10.3　[Nonce]タブ

パラメータ情報やユーザプログラムの暗号化に使用する初期化ベクタ値を指定します。

図 3.38　**[Nonce]**タブ

No	項目	説明
(1)	IV(プログラミングパラメータ)	パラメータ情報の暗号化に使用する初期化ベクタを選択します。 「乱数生成機能を使用する」選択時：ツールの乱数からIVを生成します。 「指定値を使用する」選択時：(2)に入力されたヘキサデータを使用します。
(2)	IV(プログラミングパラメータ) 指定値を使用する	(1)で「指定値を使用する」選択時、ここに入力された値を、パラメータ情報を暗号化する時の初期化ベクタとして使用します。
(3)	IV(ファームウェアイメージ)	ユーザプログラムの暗号化に使用する初期化ベクタを選択します。 「乱数生成機能を使用する」選択時：ツールの乱数からIVを生成します。 「指定値を使用する」選択時：(4)に入力されたヘキサデータを使用します。
(4)	IV(ファームウェアイメージ) 指定値を使用する	(3)で「指定値を使用する」選択時、ここに入力された値を、ユーザプログラムを暗号化する時の初期化ベクタとして使用します。

3.10.4　[AL2_KEY]タブ

AL2_KEYで暗号化するときに使用する鍵データを指定します。

図 3.39　**[AL2_KEY]タブ**

No	項目	説明
(1)	鍵データ	AL2_KEYの暗号化で使用するAES128bit鍵データを指定します。 「ファイル」を選択時 : (2)で鍵ファイルを指定します。 「平文データ」を選択時 : (3)で入力したヘキサデータを鍵として使用します。 「乱数を使用」を選択時 : 鍵データを自動生成します。(4)で自動生成する鍵データのファイル名と出力パスを設定します。
(2)	鍵データ ファイル	(1)で「ファイル」を選択時、AES128bit鍵データファイルを指定します。 バイナリ(`*.key`)、もしくはテキストファイル(`*.txt`)の指定が可能です。
(3)	鍵データ 平文データ	(1)で「平文データ」を選択時、AES128bit鍵をヘキサデータで入力します。
(4)	鍵データ 乱数を使用	(1)で「乱数を使用」を選択時、自動生成する鍵データのファイル名と出力パスを設定します。
(5)	IV	AL2_KEYの暗号化に使用する初期化ベクタを選択します。 「乱数生成機能を使用する」選択時 : ツールの乱数からIVを生成 「指定値を使用する」選択時 : (6)に入力されたヘキサデータを使用
(6)	IV 指定値を使用する	(5)で「指定値を使用する」選択時、ここに入力された値をAL2_KEYを暗号化する時の初期化ベクタとして使用します。

3.10.5　[AL1_KEY]タブ

AL1_KEY で暗号化するときに使用する鍵データを指定します。

注: この機能は、すべての MCU/MPU でサポートされていません。サポートしている機能のリストは、MCU/MPU の Hardware User Manual をご参照ください。

図 3.40　**[AL1_KEY]**タブ

No	項目	説明
(1)	鍵データ	AL1_KEYの暗号化で使用するAES128bit鍵データを指定します。 「ファイル」を選択時 :(2)で鍵ファイルを指定します。 「平文データ」を選択時 :(3)で入力したヘキサデータを鍵として使用します。 「乱数を使用」を選択時 : 鍵データを自動生成します。(4)で自動生成する鍵データのファイル名と出力パスを設定します。
(2)	鍵データ ファイル	(1)で「ファイル」を選択時、AES128bit鍵データファイルを指定します。 ファイル入力の場合、バイナリ(`*.key`)、もしくはテキストファイル(`*.txt`)の指定が可能です。
(3)	鍵データ 平文データ	(1)で「平文データ」を選択時、AES128bit鍵をヘキサデータで入力します。
(4)	鍵データ 乱数を使用	(1)で「乱数を使用」を選択時、自動生成する鍵データのファイル名と出力パスを設定します。
(5)	IV	AL1_KEYの暗号化に使用する初期化ベクタを選択します。 「乱数生成機能を使用する」選択時：ツールの乱数からIVを生成 「指定値を使用する」選択時：(6)に入力されたヘキサデータを使用
(6)	IV 指定値を使用する	(5)で「指定値を使用する」選択時、ここに入力された値を、AL1_KEYを暗号化する時の初期化ベクタとして使用します。

4. CLI 機能説明

4.1 コマンドライン構文

skmt.exe [command] [options..] 【注】

注：コマンド、オプション、パラメータの大文字小文字の区別はしません。

表 4-1 コマンドライン構文

項目	説明
skmt.exe	実行ファイル名
command	"/" または"-"から始まるコマンドです。
option…	"/" または"-"から始まるオプションです。 ・必要に応じてパラメータを指定してください。 ・ファイル名は絶対パスと相対パスに対応しています。

4.2 コマンド

以下表にコマンドを示します。

表 4-2 コマンド一覧

コマンド	説明
genufpk	UFPKファイルを生成します。 成功時にはコンソールに生成したUFPKを表示します。
genkuk	KUKファイルを生成します。 成功時にはコンソールにKUKを表示します。
genkey	ユーザ鍵およびDLM鍵を暗号化して、ファイル出力します。 成功時にはコンソールにW-UFPK, IVおよびEncrypted key(MACを含む)を表示します。
gencert	First Stage Bootloader(FSBL)機能を搭載したルネサスデバイスで使用可能な鍵証明書とコード証明書の生成を行います。
encdotf	DOTF機能を搭載したルネサスデバイスで使用可能なユーザプログラムの暗号化を行います。
encsfp	Secure Factory Programming(SFP)機能を搭載したルネサスデバイスで使用可能なユーザプログラムの暗号化を行います。
enctsip	TSIPを使用したセキュアアップデート、ファクトリープログラミングで使用するプログラムの暗号化を行います。
calcresponse	Challenge & Response 認証のレスポンス値を計算してコンソールに出力します。
H	ヘルプ

本ツールは、複数のファイルタイプに対応しています。ファイル名の拡張子で目的のファイル形式を指定します(表 4-3参照)。絶対パスと相対パスの両方に対応しています。

表 4-3 ファイルの種類と拡張子

ファイルの種類	拡張子	説明
バイナリ鍵データ	*.key	本ツールから出力されるUFPKや暗号鍵データなどのバイナリデータファイル
Renesas key file	*.rkey	RFPがユーザ鍵もしくはDLM鍵のセキュアな鍵のインジェクションに使用するファイル(一部のMCU/MPUのみサポート)
モトローラヘキサファイル	*.mot, *.srec	モトローラヘキサファイル
バイナリデータ	*.bin	バイナリデータファイル
Cソース、ヘッダファイル	*.c, *.h	Cソース、ヘッダファイル
テキストファイル	*.txt	Ascii文字で書かれるファイル
PEMファイル	*.pem	x509 ASN.1キーをBase64でエンコードしたファイル。
RSUファイル	*.rsu	TSIP Firmware Updateで使用されるイメージファイル。
Secure Factory Programming file	*.sfp	Secure Factory Programming機能で使用されるイメージファイル。

4.3 genufpk コマンド オプション

表 4-4 genufpk オプション

オプション	パラメータ	説明
ufpk	Hex data	UFPKで使用する、32バイトのバイナリデータを指定します。 本オプションは省略可能です。本オプション省略した場合、ツール内で生成したランダム値をUFPKとして使用します。【注】
output	File Path	出力ファイル名を指定します。 本オプションは省略可能です。本オプションを省略した場合、実行結果をコンソールに出力します。
nooverwrite	なし	本オプションは省略可能です。本オプション指定時、出力ファイルが存在する場合、エラーになります。

注 :ツール内で生成するランダム値は十分な乱数性を保証するものではありません。

ufpkオプション使用時の使用例：

```
> skmt.exe /genufpk
          /ufpk "00112233445566778899aabbccddeeff00112233445566778899aabbccddeeff"
          /output "D:¥example¥ufpk.key"
```

ufpkオプション省略時の使用例：

```
> skmt.exe /genufpk /output "D:¥example¥ufpk.key"
```

4.4 genkuk コマンド オプション

表 4-5 **genkuk** オプション

オプション	パラメータ	説明
kuk	Hex data	KUKで使用する、32バイトのバイナリデータを指定します。 本オプションは省略可能です。本オプション省略した場合、ツール内で生成したランダム値をKUKとして使用します。【注】
output	File Path	出力ファイル名を指定します。 本オプションは省略可能です。本オプションを省略した場合、実行結果をコンソールに出力します。
nooverwrite	なし	本オプションは省略可能です。本オプション指定時、出力ファイルが存在する場合、エラーになります。

注 :ツール内で生成するランダム値は十分な乱数性を保証するものではありません。

ufpkオプション使用時の使用例：

```
> skmt.exe /genkuk "ffeeddccbbaa99887766554433221100ffeeddccbbaa99887766554433221100"
        /output "D:¥example¥kuk.key"
```

ufpkオプション省略時の使用例：

```
> skmt.exe /genkuk /output "D:¥example¥kuk.key"
```

4.5 genkey コマンド オプション

genkeyコマンドでは、以下のオプションが使用可能です。データ入力は16進データまたはバイナリファイルで指定します。ファイルを指定する場合、ファイルパスの先頭に"file="をつけてください。

表 4-6 **genkey** オプション (1)

オプション	パラメータ	説明
iv	Hex data / File Path	ユーザ鍵もしくはDLM鍵の暗号化時に使用するInitialization Vector (IV)です(16バイト)。 本オプションは省略可能です。本オプション省略した場合、ツール内で生成したランダム値をIVとして使用します。【注】
ufpk	File Path	ユーザ鍵もしくはDLM鍵の暗号化時に使用するUFPK値が書かれたファイルを指定します。 **kuk**オプション指定時に、**ufpk**オプションは指定不可です。
wufpk	File Path	Renesas Key Wrap サービスでラップされたUFPKを指定します。 **kuk**オプション指定時に、**wufpk**オプションは指定不可です。
kuk	Hex data / File Path	セキュアな鍵のアップデート時に使用するKUKを指定します。 **ufpk**オプション指定時に、**kuk**オプションは指定不可です。
mcu	ASCII	使用するMCU/MPUを指定します。4.5.1 mcu オプションに記載したASCII文字を指定します。
keytype	ASCII / Hex data	4.5.2 keytype オプションに記載したASCII文字もしくは1バイトのHexデータを指定します。
key	Hex data / File Path	DLM鍵データもしくはユーザ鍵データを指定します。 入力する鍵データのフォーマットは4.5.3.1Hexデータ直接入力を参照してください。 本オプションは省略可能です。本オプションを省略時は、ツール内で生成した鍵データを使用します。ただし、公開鍵暗号の場合は生成できない鍵が存在します。詳細は 4.5.3.3 keyオプションの省略 を参照してください。
filetype	ASCII	出力するファイルの種類を指定します。4.5.4 filetype オプションを参照してください。 本オプションは省略可能です。本オプションを省略かつ**output**オプション指定時は`*.bin`ファイルで出力します。 本オプションとoutputオプションを省略時はコンソールに出力します。

注: ツール内で生成するランダム値は十分な乱数性を保証するものではありません。

表 4-7 **genkey** オプション (2)

オプション	パラメータ	説明
address	Hex data	鍵をインジェクションするアドレスを指定します。**filetype**で”**mot**”を指定時に設定が必要です。
fileadd	-	**filetype**で、”**csource**”、”**bin**”、”**mot**”を指定時に設定が可能です。本オプション指定時、出力ファイルが存在する場合、出力ファイルにデータを付加します。
bswap	ASCII	**filetype**で、”**bin**”、”**mot**”を指定時に設定が可能です。本オプション指定時、出力データをスワップします。
keyname	ASCII	**filetype**で”**csource**”を指定時に設定が可能です。Cソースもしくはヘッダファイルで定義している、構造体名、変数名、定義値名を指定することが可能です。
keyfileoutput	File Path	本オプションは省略可能です。**key**オプションを省略時、ツール内で生成した鍵データの出力パスを指定します。
output	File Path	出力ファイル名を指定します。 本オプションは省略可能です。本オプションを省略した場合、実行結果をコンソールに出力します。
nooverwrite	なし	本オプションは省略可能です。本オプション指定時、出力ファイルが存在する場合、エラーになります。

注: ツール内で生成するランダム値は十分な乱数性を保証するものではありません。

ufpk使用例：

```
> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
  /mcu "RA-SCE9" /keytype "AES-128" /key "000102030405060708090A0B0C0D0E0F"
  /filetype "rfp" /output "D:¥example¥aes128.key"
```

kuk使用例：

```
> skmt.exe /genkey /kuk file="D:¥example¥kuk.key" /mcu "RA-SCE9" /keytype "AES-128" /key
"000102030405060708090A0B0C0D0E0F" /filetype "csource" /output "D:¥example¥aes128.c"
```

4.5.1 mcu オプション

mcuオプションは、ASCII文字でMCU/MPUの種類と暗号エンジンを指定します。

表 4-8 mcu オプション

ASCII	説明
RA-RSIP-E51A	RA Family RSIP-E51A セキュリティ機能もしくはProtected Mode使用時に指定
RA-RSIP-E51A-CM	RA Family RSIP-E51A Compatibility Mode使用時に指定
RA-SCE9	RAファミリ SCE9 セキュリティ機能もしくはProtected Mode使用時に指定
RA-SCE9-CM	RAファミリ SCE9 Compatibility Mode使用時に指定
RA-SCE7	RAファミリ SCE7使用時に指定
RA-SCE5_B	RAファミリ SCE5_B使用時に指定
RA-SCE5	RAファミリ SCE5使用時に指定
RX-TSIP	RXファミリ TSIP使用時に指定
RX-TSIPLite	RXファミリ TSIP-Lite使用時に指定
RX-RSIP-E11A	RXファミリ RSIP-E11A使用時に指定
RZ-RSIP-T2M	RZファミリ RZ/T2M RSIP使用時に指定
RZ-RSIP-T2ME	RZファミリ RZ/T2ME RSIP使用時に指定
RZ-RSIP-T2L	RZファミリ RZ/T2L RSIP使用時に指定
RZ-RSIP-N2L	RZファミリ RZ/N2L RSIP使用時に指定
RZ-TSIP	RZファミリ TSIP使用時に指定
Synergy-SCE7	Synergy Platform SCE7使用時に指定
Synergy-SCE5	Synergy Platform SCE5使用時に指定

4.5.2 keytype オプション

keytypeオプションは、セキュアな鍵のインジェクションまたはアップデートを行う鍵の種類を指定します。 このオプションでは、ASCII値または16進数のいずれかを使用できます。 可読性を高めるためにASCIIを推奨します。

すべてのMCU/MPUがすべての鍵の種類をサポートしておりません。ご使用になるMPU/MCUのドライバやアプリケーションノートでご使用可能な暗号アルゴリズムをご確認ください。

表 4-9 DLM 遷移のための認証鍵

ASCII	**keytype** value	説明
DLM-SSD	0x01	DLMのSSDステートの認証鍵
DLM-NSECSD	0x02	DLMのNSECSDステートの認証鍵
DLM-RMA-REQ	0x03	DLMのRMA-REQステートの認証鍵
DLM-AL2	0x01	DLM 認証レベルAL2遷移用の鍵(AL2_Key)
DLM-AL1	0x02	DLM 認証レベルAL1遷移用の鍵(AL1_Key)
DLM-RMA	0x03	DLM RMA_REQステート認証用の鍵(RMA_Key)

表 4-10 ユーザ鍵 AES

ASCII	**keytype** value	説明
AES-128	0x05	AES-128bitの鍵
AES-192	0x06	AES-192bitの鍵
AES-256	0x07	AES-256bitの鍵
AES-128XTS	0x08	AES-128bit XTSの鍵
AES-256XTS	0x09	AES-256bit XTSの鍵

表 4-11 ユーザ鍵 RSA

ASCII	**keytype** value	説明
RSA-1024-public	0x0A	RSA 1024bitの公開鍵
RSA-1024-private	0x0B	RSA 1024bitの秘密鍵
RSA-2048-public	0x0C	RSA 2048bitの公開鍵
RSA-2048-private	0x0D	RSA 2048bitの秘密鍵
RSA-3072-public	0x0E	RSA 3072bitの公開鍵
RSA-3072-private	0x0F	RSA 3072bitの秘密鍵
RSA-4096-public	0x10	RSA 4096bitの公開鍵
RSA-4096-private	0x11	RSA 4096bitの秘密鍵
RSA-2048-public-TLS	0xfe	TLS API用のRSA 2048bitの公開鍵

注 1 : keytype value による指定はできません。ASCII で指定してください。

表 4-12 ユーザ鍵 ECC

ASCII	**keytype** value	説明
secp192r1-public	0x12	ECC NIST P-192(secp192r1)の公開鍵
secp192r1-private	0x13	ECC NIST P-192(secp192r1)の秘密鍵
secp224r1-public	0x14	ECC NIST P-224(secp224r1)の公開鍵
secp224r1-private	0x15	ECC NIST P-224(secp224r1)の秘密鍵
secp256r1-public	0x16	ECC NIST P-256(secp256r1)の公開鍵
secp256r1-private	0x17	ECC NIST P-256(secp256r1)の秘密鍵
secp384r1-public	0x18	ECC NIST P-384(secp384r1)の公開鍵
secp384r1-private	0x19	ECC NIST P-384(secp384r1)の秘密鍵
secp521r1-public	0x24	ECC NIST P-521(secp521r1)の公開鍵
secp521r1-private	0x25	ECC NIST P-521 (secp521r1)の秘密鍵
brainpoolP256r1-public	0x1C	ECC brainpoolP256r1の公開鍵
brainpoolP256r1-private	0x1D	ECC brainpoolP256r1の秘密鍵
brainpoolP384r1-public	0x1E	ECC brainpoolP384r1の公開鍵
brainpoolP384r1-private	0x1F	ECC brainpoolP384r1の秘密鍵
brainpoolP512r1-public	0x20	ECC brainpoolP512r1の公開鍵
brainpoolP512r1-private	0x21	ECC brainpoolP512r1の秘密鍵
secp256k1-public	0x22	ECC Koblitz curve secp256k1の公開鍵
secp256k1-private	0x23	ECC Koblitz curve secp256k1の秘密鍵
Ed25519-public	0x26	EdDSA Ed25519 の公開鍵
Ed25519-private	0x27	EdDSA Ed25519 の秘密鍵

表 4-13 ユーザ鍵 SHA-HMAC

ASCII	**keytype** value	説明
HMAC-SHA1	0x00【注】	HMAC-SHA1の鍵
HMAC-SHA224	0x1A	HMAC-SHA224の鍵
HMAC-SHA256	0x1B	HMAC-SHA256の鍵
HMAC-SHA384	0x28	HMAC-SHA384の鍵
HMAC-SHA512	0x29	HMAC-SHA512の鍵
HMAC-SHA512-224	0x2a	HMAC-SHA512-224の鍵
HMAC-SHA512-256	0x2b	HMAC-SHA512-256の鍵

注 : keytype value による指定はできません。Ascii で指定してください。

表 4-14 ユーザ鍵 ARC4

ASCII	**keytype** value	説明
ARC4	0x00【注】	ARC4 の鍵

注 : keytype value による指定はできません。Ascii で指定してください。

表 4-15 ユーザ鍵 DES

ASCII	**keytype** value	説明
TDES	0x00【注】	Triple DESの鍵

注 : keytype value による指定はできません。Ascii で指定してください。

表 4-16 OEM Root 公開鍵

ASCII	**Keytype** value	説明
OEM_ROOT_PK	0xFD	FSBL使用時にデバイスに注入しておくOEM Root 公開鍵

表 4-17 Key Update Key

ASCII	**Keytype** value	説明
key-update-key	0xFF	Key Update Key

4.5.3 key オプション

keyオプションは、セキュアな鍵のインジェクションやアップデートのために、必要な平文のDLM鍵またはユーザ鍵を指定するために使用します。 平文は、16進数のデータを直接入力するか、バイナリの `*.key` ファイルで指定できます。 鍵の種類によっては、ツールで生成することもできます。

4.5.3.1 Hex データ直接入力

平文鍵を16進データで直接入力する場合は、以下の表のように、指定された**keytype**オプションに応じた必要バイト数を入力する必要があります。

表 4-18 DLM-SSD, DLM-NSECSD, DLM-RMA-REQ, DLM-AL2, DLM-AL1, DLM-RMA

Byte	Data
0-15	DLM key data

表 4-19 AES-128

Byte	Data
0-15	AES-128bit key data

表 4-20 AES-192

Byte	Data
0-23	AES-192bit key data

表 4-21 AES-256

Byte	Data
0-31	AES-256bit key data

表 4-22 AES-128XTS

Byte	Data
0-15	AES-128bit key1
16-31	AES-128bit key2

表 4-23 AES-256XTS

Byte	Data
0-31	AES-256bit key1
32-63	AES-256bit key2

表 4-24 RSA-1024-public

Byte	Data
0-127	RSA 1024bit Modulus n
128-131	RSA 1024bit Exponent e

表 4-25 RSA-1024-private

Byte	Data
0-127	RSA 1024bit Modulus n
128-255	RSA 1024bit Decryption Exponent d

表 4-26 RSA-2048-public / RSA-2048-public-TLS

Byte	Data
0-255	RSA 2048bit Modulus n
256-259	RSA 2048bit Exponent e

表 4-27 RSA-2048-private

Byte	Data
0-255	RSA 2048bit Modulus n
256-511	RSA 2048bit Decryption Exponent d

表 4-28 RSA-3072-public

Byte	Data
0-383	RSA 3072bit Modulus n
384-387	RSA 3072bit Exponent e

表 4-29 RSA-3072-private

Byte	Data
0-383	RSA 3072bit Modulus n
384-767	RSA 3072bit Decryption Exponent d

表 4-30 RSA-4096-public

Byte	Data
0-511	RSA 4096bit Modulus n
512-515	RSA 4096bit Exponent e

表 4-31 RSA-4096-private

Byte	Data
0-511	RSA 4096bit Modulus n
512-1023	RSA 4096bit Decryption Exponent d

表 4-32 secp192r1-public

Byte	Data
0-23	ECC Public key Qx
24-47	ECC Public key Qy

表 4-33 secp192r1-private

Byte	Data
0-23	ECC Private key d

表 4-34 secp224r1-public

Byte	Data
0-27	ECC Public key Qx
28-55	ECC Public key Qy

表 4-35 secp224r1-private

Byte	Data
0-27	ECC Public key d

表 4-36 secp256r1-public / brainpoolP256r1-public / secp256k1-public / OEM_ROOT_PK

Byte	Data
0-31	ECC Public key Qx
32-63	ECC Public key Qy

表 4-37 secp256r1-private / brainpoolP256r1-private / secp256k1-private / Ed25519-private

Byte	Data
0-31	ECC Private key d

表 4-38 secp384r1-public / brainpoolP384r1-public

Byte	Data
0-47	ECC Public key Qx
48-95	ECC Public key Qy

表 4-39 secp384r1-private / brainpoolP384r1-private

Byte	Data
0-47	ECC Private d

表 4-40 brainpoolP512r1-public

Byte	Data
0-63	ECC Public key Qx
64-127	ECC Public key Qy

表 4-41 brainpoolP512r1-private

Byte	Data
0-63	ECC Private key d

表 4-42 secp521r1-public

Byte	Data
0-65	0 Padding(7bit) || ECC Public key Qx【注】
66-131	0 Padding(7bit) || ECC Public key Qy【注】

注： || はビット連結を表します

表 4-43 secp521r1-private

Byte	Data
0-65	0 Padding(7bit) || ECC Private key d【注】

注： || はビット連結を表します

表 4-44 Ed25519-public

Byte	Data
0-31	sign(1bit) || ECC Public key Qy(255bit) 【注】

注： || はビット連結を表します

表 4-45 HMAC-SHA1

Byte	Data
0-19	HMAC-SHA1 key

表 4-46 HMAC-SHA224

Byte	Data
0-27	HMAC-SHA224 key

表 4-47 HMAC-SHA256

Byte	Data
0-31	HMAC-SHA256 key

表 4-48 HMAC-SHA384

Byte	Data
0-47	HMAC-SHA384 key

表 4-49 HMAC-SHA512 / HMAC-SHA512-224 / HMAC-SHA512-256

Byte	Data
0-63	HMAC-SHA512 / 512-224 / 512-256 key

表 4-50 ARC4

Byte	Data
0-255	ARC4 key

表 4-51 TDES

Byte	Data
0-7	56bit DES key with odd parity 1 【注】
8-15	56bit DES key with odd parity 2 【注】
16-23	56bit DES key with odd parity 3 【注】

注： 鍵データ 7bit 毎に奇数パリティをつけてください。

例 DES key data = 0x00000000000000 → 0x0101010101010101

DES key data = 0xFFFFFFFFFFFFFF → 0xFEFEFEFEFEFEFEFE

表 4-52 key-update-key

Byte	Data
0-31	Key update key

使用例：

> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
/mcu "RA-SCE9" /keytype "AES-128" /key "000102030405060708090A0B0C0D0E0F"
/filetype "rfp" /output "D:¥example¥aes128.rkey"

4.5.3.2 ファイル入力

keyオプションでは、以下のファイルフォーマットのファイル入力をサポートしています。

表 4-53 **key** オプション ファイル入力

ファイル	拡張子	フォーマット
バイナリ	*.key	4.5.3.1Hexデータ直接入力 で示すフォーマットのバイナリデータファイル
テキスト	*.txt	4.5.3.1Hexデータ直接入力 で示すフォーマットのバイトデータを16進Ascii文字でテキストデータファイル
PEM	*.pem	OpenSSLで生成された非対称鍵のPEMファイルフォーマットの鍵ファイルを読み込み可能です。 表 4-54 PEMファイルサポート 非対称鍵PEMファイルサポート 非対称鍵 で示すkeytypeの非対称鍵のみ指定可能

表 4-54 PEM ファイルサポート 非対称鍵

Algorithm	keytype
RSA	RSA-1024-private, RSA-1024-public, RSA-2048-private, RSA-2048-public, RSA-3072-private, RSA-3072-public, RSA-4096-private, RSA-4096-public RSA-2048-public-TLS
ECC	secp256r1-private, secp256r1-public, secp384r1-private, secp384r1-public, secp521r1-private, secp521r1-public brainpoolP256r1-private, brainpoolP256r1-public, brainpoolP384r1-private, brainpoolP384r1-public, brainpoolP512r1-private, brainpoolP512r1-public, OEM_ROOT_PK 【注】

注: macOS 版では、brainpool カーブの PEM ファイル入力は非サポートです。

バイナリファイル使用例：

```
> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
  /mcu "RA-SCE9" /keytype "AES-128" /key file="D:¥example¥aes128.key" /filetype "rfp"
  /output "D:¥example¥aes128.rkey"
```

テキストファイル使用例：

```
> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
  /mcu "RA-SCE9" /keytype "AES-128" /key file="D:¥example¥aes128.txt" /filetype "rfp"
  /output "D:¥example¥aes128.rkey"
```

PEMファイル使用例：

```
> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
  /mcu "RA-SCE9" /keytype "RSA-2048-private" /key file="D:¥example¥rsa2048.pem"
  /filetype "rfp" /output "D:¥example¥rsa2048private.rkey"
```

(1) バイナリファイルの生成方法

手動で鍵ファイルを作成する場合、ヘキサエディタもしくはバイナリエディタを使用して作成します。
データはビッグエンディアンの並びでデータ作成してください。
以下に例を示します。

- AES 128bit の鍵ファイルを生成する例 :
  AES 128bit 鍵データ 00112233445566778899AABBCCDDEEFF の場合

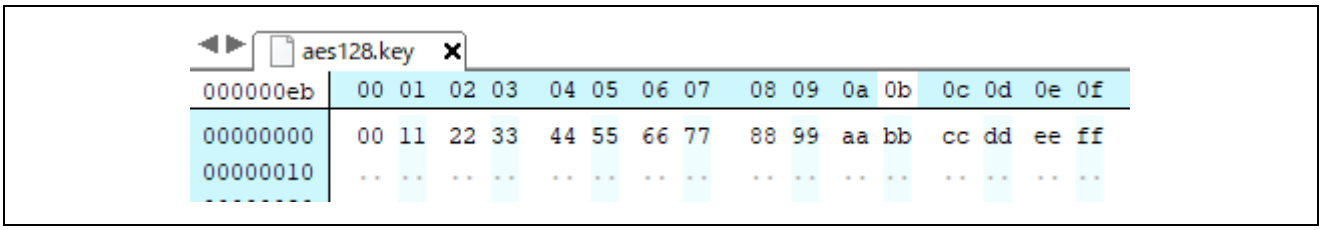

図 4.1 AES 128bit 鍵ファイルを生成する例

- RSA 2048bit 公開鍵ファイルを生成する例:
  Modulus bad47a84c1782e4dbdd913f2a261fc8b
  65838412c6e45a2068ed6d7f16e9cdf4
  462b39119563cafb74b9cbf25cfd544b
  dae23bff0ebe7f6441042b7e109b9a8a
  faa056821ef8efaab219d21d67634847
  85622d918d395a2a31f2ece8385a8131
  e5ff143314a82e21afd713bae817cc0e
  e3514d4839007ccb55d68409c97a18ab
  62fa6f9f89b3f94a2777c47d6136775a
  56a9a0127f682470bef831fbec4bcd7b
  5095a7823fd70745d37d1bf72b63c4b1
  b4a3d0581e74bf9ade93cc4614861755
  3931a79d92e9e488ef47223ee6f6c061
  884b13c9065b591139de13c1ea292749
  1ed00fb793cd68f463f5f64baa53916b
  46c818ab99706557a1c2d50d232577d1
  Exponent 10001

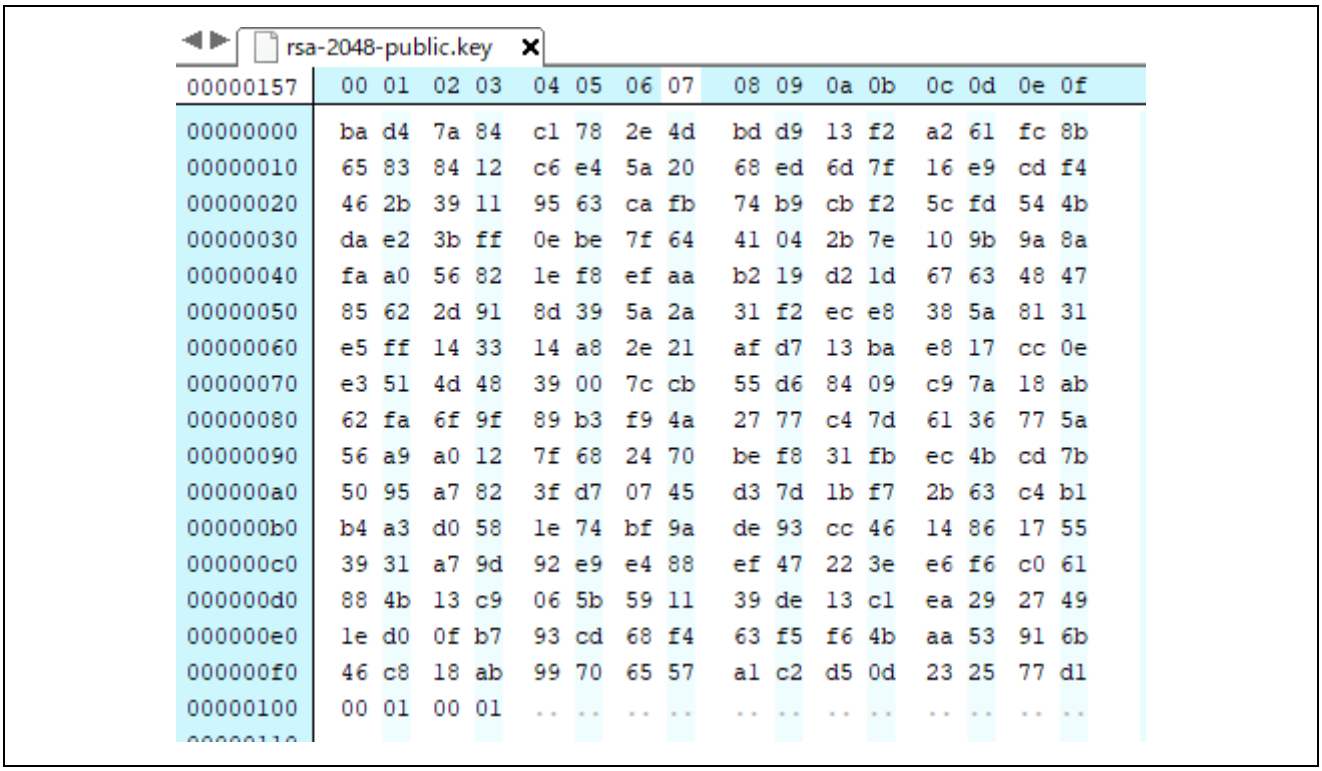

図 4.2 RSA 2048 公開鍵ファイルを生成する例

4.5.3.3 key オプションの省略

コマンドラインから **key** オプションが省略され、指定された **keytype** が対称型アルゴリズムまたは以下の表で指定された非対称型アルゴリズムの場合、ツールは乱数を使用して鍵を生成します。

非対称鍵の場合は、鍵ペアが生成されます。秘密鍵と公開鍵が同時に生成され、指定されたファイル名には、公開鍵の場合は _public、秘密鍵の場合は _private が付加されます。このオプションは、下表に示す非対称鍵タイプのオプションに対応しています。

keyオプションを省略する場合、**keyfileoutput**オプションを使用して、平文鍵の入った鍵ファイルを作成してください。

本ツールが生成する乱数値の具体的なランダム性は保証されません。本ツールで生成した鍵は、プロトタイピングやテスト目的でのみ使用してください。

表 4-55 非対称鍵生成サポート **keytype**

Algorithm	keytype
RSA	RSA-1024-private, RSA-2048-private, RSA-3072-private, RSA-4096-private
ECC	secp256r1-private, secp384r1-private, secp521r1-private, brainpoolP256r1-private, brainpoolP384r1-private, brainpoolP512r1-private 【注】 OEM_ROOT_PK

注: macOS 版では、brainpool カーブの鍵生成は非サポートです。

4.5.4 filetype オプション

filetypeオプションでは出力ファイル形式をASCII文字で指定します。
省略時はバイナリデータを出力します。
指定したfiletypeとファイル名の拡張子が一致しない場合、エラーになります。

表 4-56 filetype option

ASCII	拡張子	説明
rfp	*.rkey	Renesas Flash Programmer(RFP)で読み込めるファイルを出力します。 Renesas Key Fileフォーマットの鍵データを出力します。
csource	*.c	Cソースファイルとヘッダファイルを出力します。 keynameオプションを使用することで構造体名、変数名、定義値名を変更可能です。
bin	*.bin	バイナリデータファイルを出力します。
mot	*.mot	バイナリデータをモトローラヘキサフォーマットにして出力します。 addressオプションで、16進数8桁(32ビット)の開始アドレスを指定してください。

4.5.4.1 filetype オプション rfp 指定時

Renesas Flash Programmerに設定するRenesas Key Fileフォーマットの鍵データを出力します。

rfpファイル出力例：

```
> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
  /mcu "RA-SCE9" /keytype "AES-128" /key file="D:¥example¥aes128.key" /filetype "rfp"
  /output "D:¥example¥abc.rkey"
```

4.5.4.2 filetype オプション csource 指定時

Cソースファイルとヘッダファイルを出力します。

keynameオプションを使用すると、構造体名、グローバル変数名、バイトサイズ定義の一部として、指定した<keyname>が使用されます。**keyname**オプションを使用しない場合、デフォルトのテキストが使用されます。これらのオプションは次の表に示すとおりです。

表 4-57 **keyname** option

	keyname設定時	keyname省略時
構造体名	<keyname>_t	encrypted_user_key_data_t
グローバル変数名	g_<keyname>	g_encrypted_user_key_data
encrypted_user_keyバイトサイズ定義値	<keyname>_SIZE 【注】	ENCRYPTED_KEY_BYTE_SIZE

注：keyname オプションで指定した Ascii を大文字で、設定します。

出力するCソース構造体は以下になります。

表 4-58 ufpk オプション指定時の c ソースファイルの構造体

name	型	説明
g_<keyname>	<keyname>_t	-
keytype	uint32_t	keytype値を使用するセキュリティエンジンの場合、keytype value、keytype値を使用しないセキュリティエンジンの場合0を出力します。
shared_key_number	uint32_t	0を出力します。**mcu**オプションが”RX-TSIP”もしくは”RX-TSIPLite”の場合は使用しません。
wufpk[32]	uint8_t	**wufpk**オプションで指定された値を出力します。
initial_vector[16]	uint8_t	ユーザ鍵もしくはDLM鍵の暗号化で使用したInitial Vectorを出力します。
encrypted_user_key [<KEYNAME>_SIZE]	uint8_t	ユーザ鍵もしくはDLM鍵の暗号化した結果を出力します。<KEY_NAME>_SIZEはEncrypted User Keyのサイズになります。Encrypted User Keyサイズは 表 4-62-表 4-71 を参照してください。
crc[4]	uint8_t	keytypeからencrypted_user_keyのCRC-32値を出力します。

表 4-59 kuk オプション指定時の c ソースファイルの構造体

name	型	説明
g_<keyname>	<keyname>_t	-
keytype	uint32_t	keytype値を使用するセキュリティエンジンの場合、keytype value、keytype値を使用しないセキュリティエンジンの場合0を出力します。
shared_key_number	uint32_t	0を出力します。**mcu**オプションが”RX-TSIP”もしくは”RX-TSIPLite”の場合は使用しません。
initial_vector[16]	uint8_t	ユーザ鍵もしくはDLM鍵の暗号化で使用したInitial Vectorを出力します。
encrypted_user_key [<KEYNAME>_SIZE]	uint8_t	ユーザ鍵もしくはDLM鍵の暗号化した結果を出力します。<KEYNAME>_SIZEはEncrypted User Keyのサイズになります。Encrypted User Keyサイズは 表 4-62-表 4-71 を参照してください。
crc[4]	uint8_t	keytypeからencrypted_user_keyのCRC-32値を出力します。

cソースファイル出力例：

> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key" /mcu "RX-TSIP" /keytype "AES-128" /key file="D:¥example¥aes128.key" /filetype "csource" /keyname "aes128" /output "D:¥example"¥abc.c"

4.5.4.3 filetype オプション bin 指定時

バイナリデータのファイルを出力します。
出力するバイナリデータのデータ配列は以下になります。

表 4-60 **ufpk** オプション指定時のバイナリデータ

Byte	Data
0	keytype keytype値を使用するセキュリティエンジンの場合、keytype value、 keytype値を使用しないセキュリティエンジンの場合0を出力します。
1 - 3	Reserved (0 padding)
4	shared_key number 0を出力します。**mcu**オプションが”RX-TSIP”もしくは”RX-TSIPLite”の場合は使用しません。
5 - 7	Reserved (0 padding)
8 - 39	wufpk wufpkオプションで指定された値を出力します。
40 - 55	initial_vector ユーザ鍵およびDLM鍵の暗号化で使用したInitial Vectorを出力します
56 - (56+N-1) 【注】	encrypted_user_key ユーザ鍵およびDLM鍵を暗号化した結果を出力します。
(56+N) - 56+N +3 【注】	crc keytypeからencrypted_user_keyのCRC-32値を出力します。

注： 「N」は、encrypted user key サイズです。 表 4-62 - 表 4-71 を参照してください。

表 4-61 **kuk** オプション指定時のバイナリデータ

Byte	Data
0	keytype keytype値を使用するセキュリティエンジンの場合、keytype value、 keytype値を使用しないセキュリティエンジンの場合0を出力します。
1 - 3	Reserved (0 padding)
4	shared_key number 0を出力します。**mcu**オプションが”RX-TSIP”もしくは”RX-TSIPLite”の場合は使用しません。
5 - 7	Reserved (0 padding)
8 - 23	initial_vector ユーザ鍵およびDLM鍵の暗号化で使用したInitial Vectorを出力します
24 - (24+N-1) 【注】	encrypted_user_key ユーザ鍵およびDLM鍵を暗号化した結果を出力します。
(24+N) - 24+N +3 【注】	crc keytypeからencrypted_user_keyのCRC-32値を出力します。

注： 「N」は、encrypted user key サイズです。 表 4-62 - 表 4-71 を参照してください。

binaryファイル出力例：

```
> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
  /mcu " RX-TSIP" /keytype "AES-128" /key file="D:¥example¥aes128.key" /filetype "bin"
  /output "D:¥example¥abc.bin"
```

4.5.4.4 filetype オプション mot 指定時

このオプションは、モトローラHexフォーマットファイルを出力します。ファイルのフォーマットは、表 4-60 **ufpkオプション指定時のバイナリデータ**および表 4-61 **kukオプション指定時のバイナリデータ**に規定されています。 データのアドレスは、**address**オプションを使用して、16進数8桁（32ビット）の開始アドレスを指定する必要があります。

motファイル出力例：

```
> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
  /mcu " RX-TSIP" /keytype "AES-128" /key file="D:¥example¥aes128.key" /filetype "mot"
  /address "FFF00000" /output "D:¥example¥abc.mot"
```

以下に、keytypeオプション毎のencrypted_user_key sizeを示します。

表 4-62 encrypted_user_key size DLM Key

keytype オプション	Byte size
DLM-SSD	32
DLM-NSECSD	32
DLM-RMA-REQ	32
DLM-AL2	32
DLM-AL1	32
DLM-RMA	32

表 4-63 encrypted_user_key size AES Key

keytype オプション	Byte size
AES-128	32
AES-192	48
AES-256	48
AES-128XTS	48
AES-256XTS	80

表 4-64 encrypted_user_key size RSA Key

keytype オプション	Byte size
RSA-1024-public	160
RSA-1024-private	272
RSA-2048-public	288
RSA-2048-private	528
RSA-3072-public	416
RSA-3072-private	784
RSA-4096-public	544
RSA-4096-private	1040
RSA 2048bit public key for TLS	288

表 4-65 encrypted_user_key size ECC Key

keytype オプション	Byte size
secp192r1-public	80
secp192r1-private	48
secp224r1-public	80
secp224r1-private	48
secp256r1-public	80
secp256r1-private	48
secp384r1-public	112
secp384r1-private	64
secp521r1-public	132
secp521r1-private	66
brainpoolP256r1-public	80
brainpoolP256r1-private	48
brainpoolP384r1-public	112
brainpoolP384r1-private	64
brainpoolP512r1-public	144
brainpoolP512r1-private	80
secp256k1-public	80
secp256k1-private	48

表 4-66 encrypted_user_key size Ed25519 Key

keytype オプション	Byte size
Ed25519-public	48
Ed25519-private	48

表 4-67 encrypted_user_key size HMAC-SHA Key

keytype オプション	Byte size
HMAC-SHA1	48
HMAC-SHA224	48
HMAC-SHA256	48
HMAC-SHA384	64
HMAC-SHA512	80
HMAC-SHA512-224	80
HMAC-SHA512-256	80

表 4-68 encrypted_user_key size ARC4 Key

keytype オプション	Byte size
ARC4	272

表 4-69 encrypted_user_key size TDES Key

keytype オプション	Byte size
TDES	48

表 4-70 encrypted_user_key size OEM_ROOT_PK Key

keytype オプション	Byte size
OEM_ROOT_PK	80

表 4-71 encrypted_user_key size Key Update Key

keytype オプション	Byte size
key-update-key	48

4.5.5 fileadd オプション

fileaddオプションは、**filetype**オプションに”**csource**”、”**bin**”または”**mot**”を指定時に設定可能です。

出力ファイルと同一ファイル名がある場合に、上書きではなく、既存のデータに生成したencrypted key情報を付加します。

注意：**filetype** オプションで”**csource**”を指定時に、すでにあるデータと新規に追加するデータで、同じ定義値が定義されているかの確認を行いません。**keyname** オプションを使用して、同じ定義値にならないようにしてください。

fileadd設定例：

```
> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
  /mcu "RX-TSIP" /keytype "AES-128" /key file="D:¥example¥aes128.key" /filetype "csource"
  /fileadd /keyname "aes128" /output "D:¥example¥abc.c"
```

4.5.6 bswap オプション

bswapオプションは、ASCII文字で出力するデータのデータ並びを指定します。省略時は”**32-big**”の動作になります。

表 4-72 **bswap** オプション

ASCII	説明
32-big	表 4-60、表 4-61の各データをビッグエンディアンデータ並びで、ファイルに出力します。
32-little	表 4-60、表 4-61の各データを4バイトスワップしたデータ並びで、ファイルに出力します。

bswap設定例：

```
> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
  /mcu "RX-TSIP" /keytype "AES-128" /key file="D:¥example¥aes128.key" /filetype "csource"
  /bswap "32-little" /keyname "aes128" /output "D:¥example¥abc.c"
```

4.5.7 keyfileoutput オプション

コマンドラインから **key** オプションを省略した場合、ツールはランダムな鍵を生成します。 このオプションは、生成された鍵をバイナリもしくはテキストファイルに保存するために使用します。 指定するファイルの拡張子がバイナリデータの場合`*.key`、テキストファイルの場合`*.txt`を指定します。

このツールで生成できるのは、対称鍵と一部の非対称鍵ペアのみです。
非対称鍵ペアの生成は以下の**keytype**オプションをサポートしています。

表 4-73 非対称鍵生成サポート **keytype**

Algorithm	keytype
RSA	RSA-1024-private, RSA-2048-private, RSA-3072-private, RSA-4096-private
ECC	secp256r1-private, secp384r1-private, secp521r1-private, brainpoolP256r1-private, brainpoolP384r1-private, brainpoolP512r1-private, OEM_ROOT_PK

本ツールで生成される乱数値の具体的な乱数量は保証されません。本ツールで生成した鍵は、プロトタイピングおよびテスト目的でのみ使用してください。

非対称鍵ペアを生成し、**/keyfileoutput**を指定した場合、秘密鍵のファイル名には"`_private`"、公開鍵のファイル名には"`_public`"が付加されます。例えば、"`/keyfileoutput abc.key /output abc.rkey`"を指定すると、以下のファイルが生成されます。

- `abc_private.key` 平文の秘密鍵が格納されています。
- `abc_public.key` 平文の公開鍵が格納されている。
- `abc_private.rkey` 秘密鍵をインジェクションするための暗号化された鍵のファイル
- `abc_public.rkey` 公開鍵をインジェクションするための暗号化された鍵のファイル

対称鍵暗号ファイル出力例 ：

```
> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
  /mcu "RA-SCE9" /keytype "AES-128" /filetype "rfp" /keyfileoutput "D:¥example¥aes.key"
  /output "D:¥example¥abc.rkey"
```

非対称鍵暗号ファイル出力例 ：

```
> skmt.exe /genkey /ufpk file="D:¥example¥ufpk.key" /wufpk file="D:¥example¥ufpk_enc.key"
  /mcu "RX-TSIP" /keytype "RSA-1024-private" /filetype "bin"
  /keyfileoutput "D:¥example¥rsa1024.key" /output "D:¥example¥rsa1024_private.bin"
```

4.6 enctsip コマンド オプション

enctsipコマンドでは、以下のオプションが使用可能です。データ入力は16進データまたはバイナリファイルで指定します。

表 4-74 **enctsip** オプション

オプション	パラメータ	説明
mode	ASCII	出力するファイルのデータフォーマットを指定します。 詳細は4.6.1 **mode** オプションを参照してください。
ver	Decimal data	出力するファイルに添付するRSUヘッダのバージョンを指定します。 バージョンは1もしくは2が指定可能です。 本オプションは省略可能です。省略時は2を指定します。 詳細は4.6.2 **ver** オプションを参照してください。
prg	File Path	暗号化するmotファイルを指定します。
prg_sb	File Path	modeオプションで**factory**指定時に、暗号化するmotファイルと合わせるセキュアブートプログラムのmotファイルを指定します。
enckey	Hex data	prgオプションで指定したmotファイルのデータを暗号化するSession Keyを暗号化するための鍵を指定します。(16バイト)
session_key	Hex data	prgオプションで指定したmotファイルのデータを暗号化するSession Keyを指定します。(32バイト) 本オプションは省略可能です。本オプションを省略した場合、ツール内で生成したランダム値を鍵データとして使用します。【注】
iv_fw	Hex data	prgオプションで指定したmotファイルのデータを暗号化時に使用するInitialization Vector (IV)です(16バイト)。 本オプションは省略可能です。本オプションを省略した場合、ツール内で生成したランダム値をIVとして使用します。【注】
startaddr	Hex data	prgオプションで指定したmotファイル内で、暗号化する領域の開始アドレスを指定します。アドレスは16バイトアラインを取る必要があります。
endaddr	Hex data	prgオプションで指定したmotファイル内で、暗号化する領域の終了アドレスを指定します。開始アドレスと終了アドレスで指定される暗号化領域のサイズは16バイトアラインを取る必要があり、上限は8MBです。
destaddr	Hex data	filetypeオプションで”**mot**”を指定時に、暗号化したユーザプログラムの出力するアドレスを指定します。
imgflg	ASCII / Hex data	RSUヘッダのImage Flagsに入力する値を指定します。 詳細は4.6.3 **imgflg** オプションを参照してください。
filetype	ASCII	出力するファイルの種類を指定します。 詳細は4.6.4 filetype オプションを参照してください。
output	File Path	出力ファイル名を指定します。
nooverwrite	なし	本オプションは省略可能です。本オプション指定時、出力ファイルが存在する場合、エラーになります。

注: ツール内で生成するランダム値は十分な乱数性を保証するものではありません。

mode factory設定時の使用例：

```
> skmt.exe /enctsip /mode "factory" /ver "1" /prg "D:¥example¥userprog.mot"
 /prg_sb "D:¥example¥secureboot.mot" /enckey "0123456789ABCDEF0123456789ABCDEF"
 /startaddr "FFF80300" /endaddr "FFFEFFFF" /destaddr "FFF00300"
 /filetype "mot" /output "D:¥example¥factory.mot"
```

mode update設定時の使用例：

```
> skmt.exe /enctsip /mode "update" /ver "1" /prg "D:¥example¥userprog.mot"
 /enckey "0123456789ABCDEF0123456789ABCDEF"
 /startaddr "FFF80300" /endaddr "FFFEFFFF" /filetype "rsu" /output "D:¥example¥update.rsu"
```

4.6.1 mode オプション

modeオプションは、ASCII文字で出力するデータのフォーマットを指定します。

表 4-75 **mode** オプション

ASCII	説明
factory	工場書き込みを想定し、**prg_sb**で指定したセキュアブートプログラムと暗号化したプログラムにRSUを付加したmotファイルを出力します。 ファイルのイメージは図 4.3modeオプション factory指定時のファイル生成イメージを参照してください。
update	市場でのFirmware Updateを想定し、暗号化したプログラムにRSUを付加したバイナリファイルもしくはmotファイルを出力します。 ファイルのイメージは図 4.4modeオプション updateかつ filetypeオプションrsu指定時のファイル生成イメージ もしくは 図 4.5modeオプション updateかつ filetypeオプションmot指定時のファイル生成イメージを参照してください。

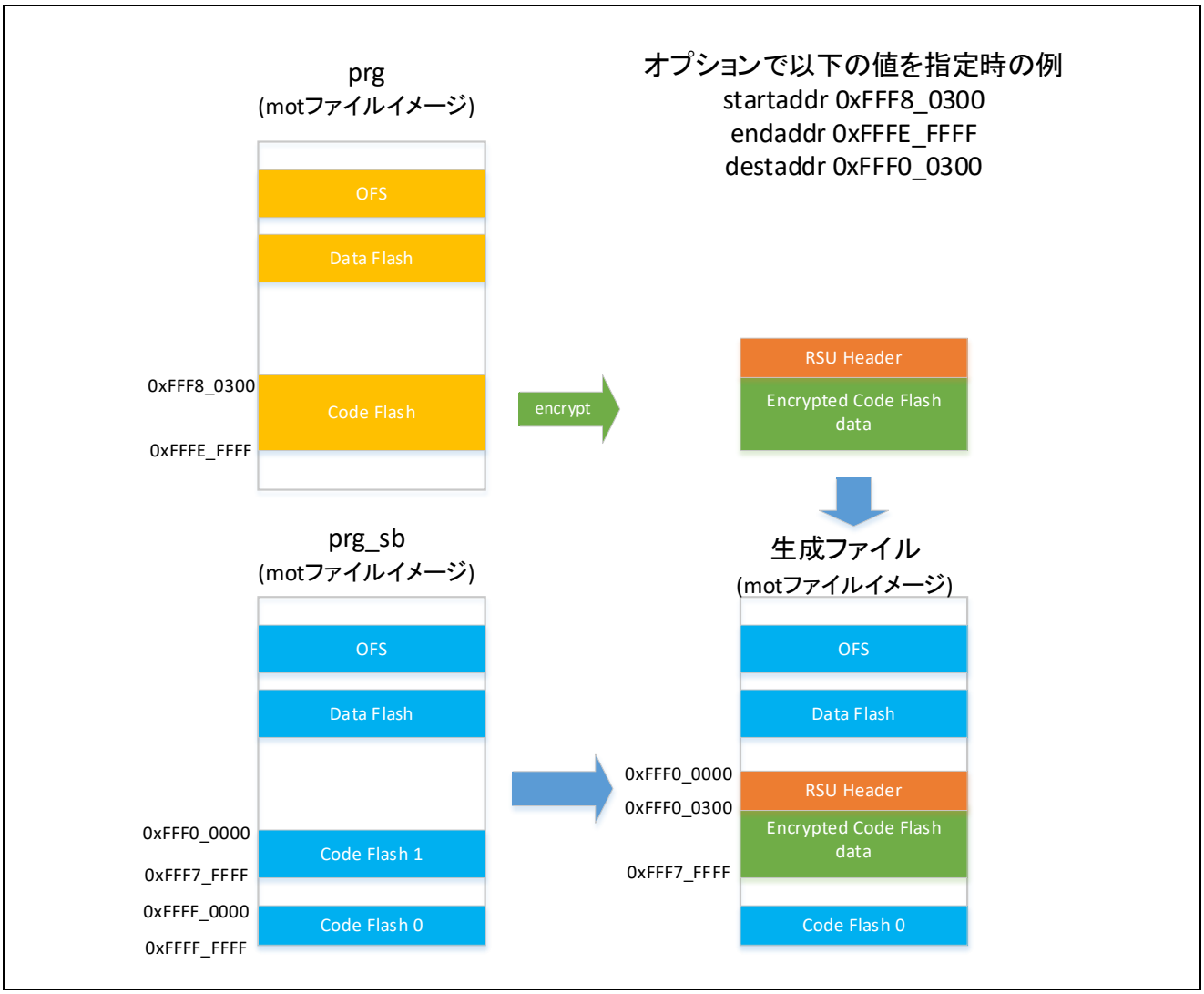

図 4.3 **mode** オプション **factory** 指定時のファイル生成イメージ

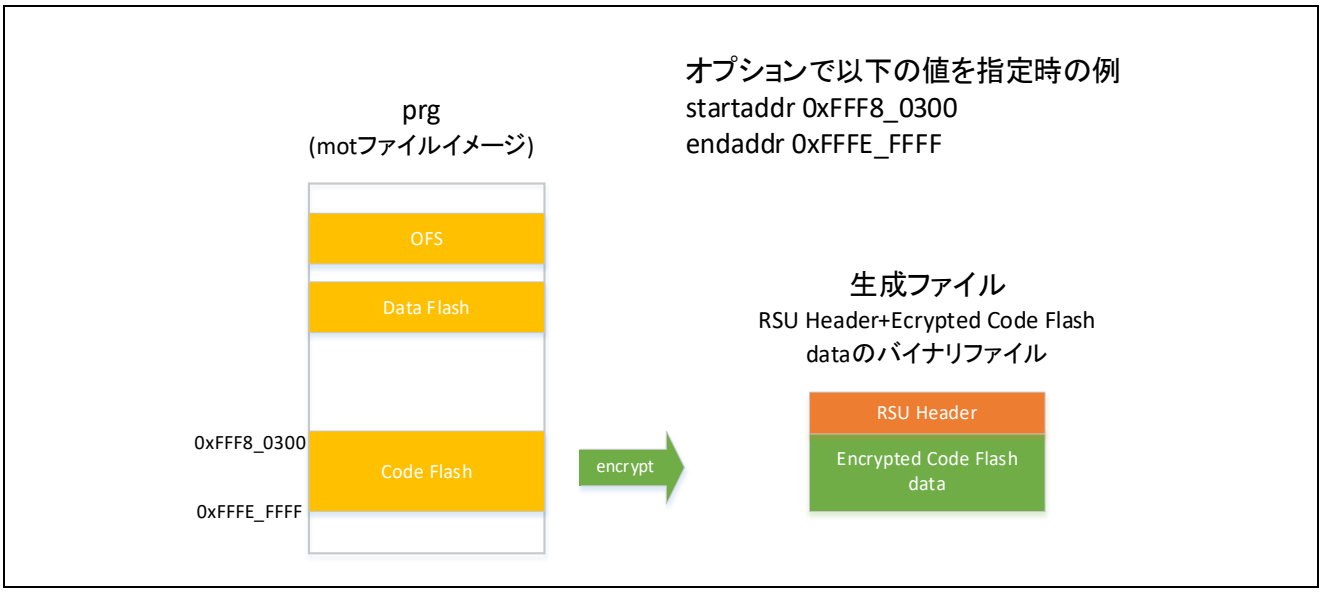

図 4.4　**mode** オプション **update** かつ **filetype** オプション **rsu** 指定時のファイル生成イメージ

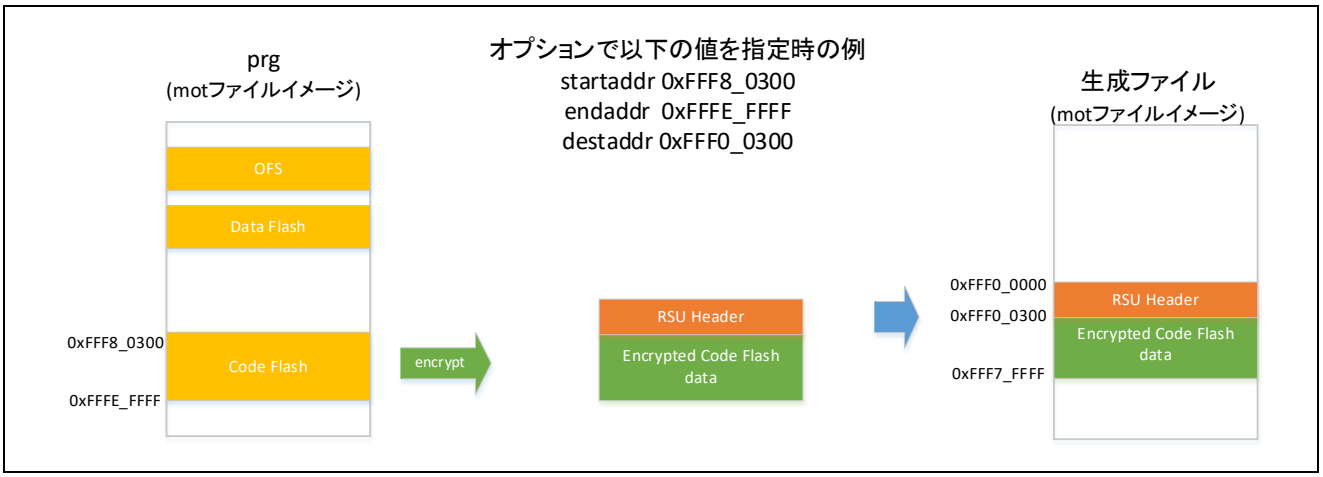

図 4.5　**mode** オプション **update** かつ **filetype** オプション **mot** 指定時のファイル生成イメージ

4.6.2 ver オプション

verオプションは、出力するファイルに付加するRSUヘッダのバージョンを指定します。

表 4-76 RSU ヘッダ V1

offset	Component	Contents name	Length	Note
0x0000	Header	Magic Code	7	“Renesas”
0x0007		Image Flags	1	imgflg オプションで指定された値
0x0008	Signature	Firmware Verification Type	32	ASCII “mac-aes128-cmac-with-tsip”
0x0028		Signature size	4	使用していません(0x00)。
0x002C		Signature	256	使用していません(0x00)。
0x012C	Option	Dataflash Flag	4	使用していません(0x00)。
0x0130		Dataflash Start Address	4	使用していません(0x00)。
0x0134		Dataflash End Address	4	使用していません(0x00)。
0x0138		Image Size	4	使用していません(0x00)。
0x013C		Reserved	124	All 0x00
0x01B8		Format Type *1	4	出力するファイルフォーマット ASCII で”rsu”(0x72, 0x73, 0x75, 0x00) もしくは “mot”(0x6D, 0x6F, 0x74, 0x00)
0x01BC		IV *1	16	ユーザプログラム暗号化時に使用した IV
0x01CC		SessionKey0 *1	16	ユーザプログラム暗号化時に使用した鍵を enckey で暗号化した鍵
0x01DC		SessionKey1 *1	16	ユーザプログラム暗号化時に使用した鍵を enckey で暗号化した鍵
0x01EC		暗号化されたユーザプログラム+MAC のワードサイズ *1	4	暗号化されたユーザプログラム+ユーザプログラム暗号化時に生成された MAC の合計ワードサイズ
0x01F0		MAC *1	16	ユーザプログラム暗号化時に生成された MAC
0x0200	Descriptor	Sequence Number	4	常に 1
0x0204		Start Address	4	ユーザプログラム領域の開始アドレス
0x0208		End Address	4	ユーザプログラム領域の終了アドレス
0x020C		Execution Address	4	ユーザプログラム実行開始アドレス格納アドレス (固定値 0xFFFEFFFC)
0x0210		Hardware ID	4	使用していません(0x00)。
0x0214		Reserved(0x00)	236	-
0x0300	Application Binary		N	暗号化されたユーザプログラム
0x0300 +N	Dataflash Binary		M	対応していません。

【*1】: RX Firmware Update FIT V.1.x で定義される `*.rsu` ファイルフォーマットから TSIP 用に変更したパラメータです。

表 4-77 RSU ヘッダ V2

offset	Component	Contents name	Length	Note
0x0000	Header	Magic Code	7	“RELFWV2”
0x0007		Reserved	1	使用していません(0x00)。
0x0008	Signature	Firmware Verification Type	32	ASCII “mac-aes128-cmac-with-tsip”
0x0028		Signature size	4	使用していません(0x00)。
0x002C		Signature	64	使用していません(0x00)。
0x006C	Option	Reserved(0x00)	332	-
0x01B8		Format Type *1	4	出力するファイルフォーマット ASCII で”rsu”(0x72, 0x73, 0x75, 0x00) もしくは “mot”(0x6D, 0x6F, 0x74, 0x00)
0x01BC		IV *1	16	ユーザプログラム暗号化時に使用した IV
0x01CC		SessionKey0 *1	16	ユーザプログラム暗号化時に使用した鍵を prog user key で暗号化した鍵
0x01DC		SessionKey1 *1	16	ユーザプログラム暗号化時に使用した鍵を prog user key で暗号化した鍵
0x01EC		暗号化されたユーザプログラム+MAC のワードサイズ *1	4	暗号化されたユーザプログラム+ユーザプログラム暗号化時に生成された MAC の合計ワードサイズ
0x01F0		MAC *1	16	プログラム暗号化時に生成された MAC
0x0200	Descriptor	プログラムデータ件数	4	ユーザプログラムのデータ数(最大 31 件 本バージョンでは 1 つのみ)
0x0204		開始アドレス[0]	4	ユーザプログラム領域の開始アドレス 1 件目
0x0208		データサイズ[0]	4	ユーザプログラム領域のデータサイズ 1 件目
0x020C		開始アドレス[1]	4	ユーザプログラム領域の開始アドレス 2 件目
0x0210		データサイズ[1]	4	ユーザプログラム領域のデータサイズ 2 件目
・ ・ ・		・ ・ ・	・ ・ ・	ユーザプログラム領域の開始アドレス 3-29 件目 ユーザプログラム領域のデータサイズ 3-29 件目
0x02F4		開始アドレス[30]	4	ユーザプログラム領域の開始アドレス 30 件目
0x02F8		データサイズ[30]	4	ユーザプログラム領域のデータサイズ 30 件目
0x02FC		Reserved(0x00)	4	-
0x0300	Application Binary		N	暗号化されたユーザプログラム
0x0300 +N	Dataflash Binary		M	対応していません。

【*1】: RX Firmware Update FIT V.2.x で定義される `*.rsu` ファイルフォーマットから TSIP 用に変更したパラメータです。

4.6.3 imgflg オプション

imgflgオプションは、出力するファイルに付加するRSUヘッダのImage Flagsフィールドに書き込む値を指定します。このオプションでは、ASCII値または16進数のいずれかを使用できます。

表 4-78 **imgflg** オプション

ASCII	値	説明
blank	0xFF	イメージは書かれていません。
testing	0xFE	イメージをアップデート中　検証実施中
Installing	0xFC	初期イメージのインストール中
valid	0xF8	アプリケーションが有効
invalid	0xF0	アプリケーションが無効
end_of_life	0xE0	アプリケーションが End Of Life

4.6.4 filetype オプション

filetypeオプションでは出力ファイル形式をASCII文字で指定します。
指定した**filetype**とファイル名の拡張子が一致しない場合、エラーになります。

表 4-79 filetype option

ASCII	拡張子	説明
mot	`*.mot`	モトローラヘキサフォーマットのファイルを出力します。
rsu	`*.rsu`	暗号化したユーザプログラムに RSU ヘッダを付加したバイナリデータを出力します。 mode オプションを update 指定時のみ指定可能です。

4.7 gencert コマンド オプション

gencertコマンドでは、以下のオプションが使用可能です。データ入力は16進データまたはファイルで指定します。ファイルを指定する場合、ファイルパスの先頭に"file="をつけてください。

表 4-80 **gencert** オプション(1)

オプション	パラメータ	説明
mode	ASCII	“**signature**” : ECDSA署名を使用したコード証明書と鍵証明書を生成します。 “**CRC**” : CRCを使用した簡易的な検証を行うためのコード証明書のみを作成します。 詳細は4.7.1 **mode** オプションを参照してください。
loadaddr	Hex data	OEM Bootloaderの開始アドレスを指定します。
cfsize	Hex data	使用するデバイスのコードフラッシュのサイズを指定します。 **oembl_size**省略時は、**loadaddr**から**cfsize**の範囲内にあるデータを署名対象にします。 署名対象領域については4.7.2 OEM_BLの署名もしくはCRC演算対象領域をご参照ください。
oembl_size	Hex data	OEM_BLのサイズを指定します。サイズは16バイトアラインサイズを指定してください。 署名対象領域については、4.7.2 OEM_BLの署名もしくはCRC演算対象領域をご参照ください。
ver	Decimal data	本オプションは**mode**が”**signature**”の場合のみ必要です。 コード証明書に記載するOEM_BLのバージョンを指定します。 1～4,294,967,295が指定可能です。 実際に使用できるバージョンは、MCU/MPUの仕様に依存しています。MCU/MPUのHardware User's Manualもしくはアプリケーションノートをご参照ください。
oembl	File Path	FSBLが検証対象とするOEM_BLのモトローラヘキサファイルを指定します。

表 4-81 **gencert** オプション(2)

オプション	パラメータ	説明
oembl_private	Hex data / File Path	本オプションは**mode**が"**signature**"の場合に必要です。 OEM_BL検証秘密鍵を指定します。 HEXデータ指定の場合、32バイトHEXデータを入力します。 ファイル入力の場合、OEM_BL検証秘密鍵のバイナリ(*.key)、もしくはテキストファイル(*.txt)、もしくは公開鍵データが含まれるPEMファイル(*.pem)の指定が可能です。 本オプションを省略した場合、ツール内部でOEM_BLのsecp256r1の鍵ペアを生成します。【注】
oembl_public	Hex data / File Path	本オプションは**mode**が"**signature**"の場合に必要です。 OEM_BL検証公開鍵を指定します。 HEXデータ指定の場合、64バイトHEXデータを入力します。 ファイル入力の場合、Qx、Qyの鍵データのバイナリ(*.key)、もしくはテキストファイル(*.txt)の指定が可能です。 **oembl_private**でPEMファイル入力時は省略可能です。
oemroot_private	Hex data / File Path	本オプションは**mode**が"**signature**"の場合に必要です。 OEM_BL検証ルート秘密鍵を指定します。 HEXデータ指定の場合、32バイトHEXデータを入力します。 ファイル入力の場合、OEM_BL検証ルート秘密鍵のバイナリ(*.key)、もしくはテキストファイル(*.txt)、もしくは公開鍵データが含まれるPEMファイル(*.pem)の指定が可能です。
oemroot_public	Hex data / File Path	本オプションは**mode**が"**signature**"の場合に必要です。 OEM_BL検証ルート公開鍵を指定します。 HEXデータ指定の場合、64バイトHEXデータを入力します。 ファイル入力の場合、Qx、Qyの鍵データのバイナリ(*.key)、もしくはテキストファイル(*.txt)の指定が可能です。 **oemroot_private**でPEMファイル入力時は省略可能です。
output_codecert	File Path	コード証明書の出力ファイル名を指定します。
output_keycert	File Path	本オプションは**mode**が"**signature**"の場合に必要です。 鍵証明書の出力ファイル名を指定します。
keyfileoutput	File Path	**mode**が"**signature**"の場合かつ、**oembl_private**オプション省略時に、ツール内で生成、使用したsecp256r1の鍵ペアのファイルを出力します。出力ファイル名を指定してください。

注: ツール内で生成するランダム値は十分な乱数性を保証するものではありません。

mode CRC設定時の使用例：

```
> skmt.exe /gencert /mode "CRC" /loadaddr "02000000" /oembl_size "1000"
/oembl "D:¥example¥user.mot" /output_codecert "D:¥example¥ccert.bin"
```

mode signature設定時の使用例：

```
> skmt.exe /gencert /mode "signature" /loadaddr "02000000" /cfsize "200000" /ver "1"
/oembl "D:¥example¥user.mot" /oembl_private file="D:¥example¥is_ec256_priv.pem"
/oemroot_private file="D:¥example¥ms_ec256_priv.pem"
/output_codecert "D:¥example¥ccert.bin" /output_keycert " D:¥example¥kcert.bin"
```

4.7.1 mode オプション

modeオプションは、ASCII文字で出力する証明書の種類を指定します。

表 4-82 **mode** オプション

ASCII	説明
signature	OEM_BLに署名する鍵(OEM_BL検証鍵)と、OEM_BL検証鍵を署名する鍵(OEM_BL検証ルート鍵)を使用して、鍵証明書とコード証明書を生成します。 **oemroot_private**、**oemroot_public**オプションでOEM_BL検証ルート鍵ペア、**oembl_private**、**oembl_public**オプションでOEM_BL検証鍵ペアを指定してください。 証明書のChain Of Trustの構造は、FSBL機能を実装するデバイスのUser Manualをご参照ください。
CRC	OEM BootloaderのCRC値を計算し、コード証明書のみを生成します。【注】

注：”**CRC**”を指定してコード証明書を生成した場合、Signer ID にダミー値として CRC 値を出力します。FSBL 動作モードで”CRC + report measurement”選択し、OEM_BL 検証鍵を使用した measurement report を出力したい場合は、”**signature**”を選んでコード証明書を生成してください。

4.7.2 OEM_BL の署名もしくは CRC 演算対象領域

OEM_BLの署名もしくはCRC演算対象は、**oemblオプション**で指定されたmotファイルと**oembl_sizeオプション**と**cfsizeオプション**で指定したエリアから抽出したイメージ部分になります。

4.7.2.1 oembl_size オプション指定時

loadaddrオプションで指定されたアドレスから**oembl_size**分のイメージを署名もしくはCRC演算対象にします。入力されたmotファイルに**oembl_size**指定分のデータがない領域は0xFFでデータを埋めます。

例として、**loadaddr** “02000000”、**oembl_size** “4000”を指定時に以下の3例を図示します。

例1:入力されたmotファイルのイメージが0x02002FFFまでしかない

例2:入力されたmotのイメージが0x02004FFFまである

例3:入力されたmotのイメージ内0x02002000から0x02002FFFまでデータが存在しない

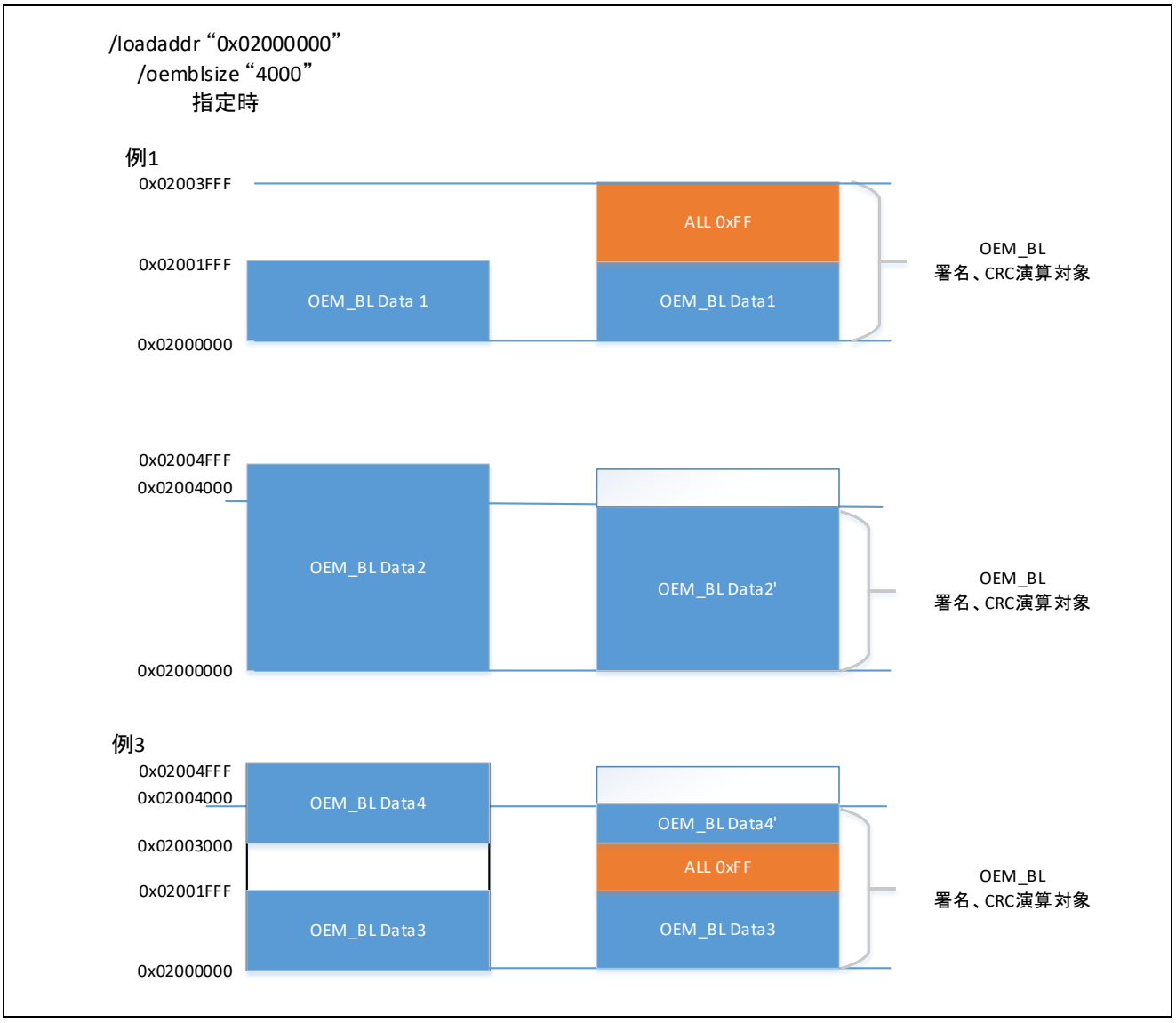

図 4.6 **oembl_size** オプション指定時の署名、CRC 演算対象

4.7.2.2　cfsize オプションのみ指定時

oembl_sizeオプションの指定を省略した場合、**oembl**オプションで指定されたmotファイル内の、**loadaddr**オプションで指定されたアドレスから**cfsize**で指定されたエリアにあるデータを署名、CRC演算対象とします。

例として、**loadaddr** “02000000”、**oembl_size** “10000”を指定時に以下の4例を図示します。

例1:入力されたmotファイルのイメージが0x02001FF8までしかない

例2:入力されたmotのイメージが0x020014FFFまである

例3:入力されたmotのイメージ内0x02010000以降にもデータが存在している

例4:入力されたmotのイメージ内0x02010000までにデータが存在しないエリアがある

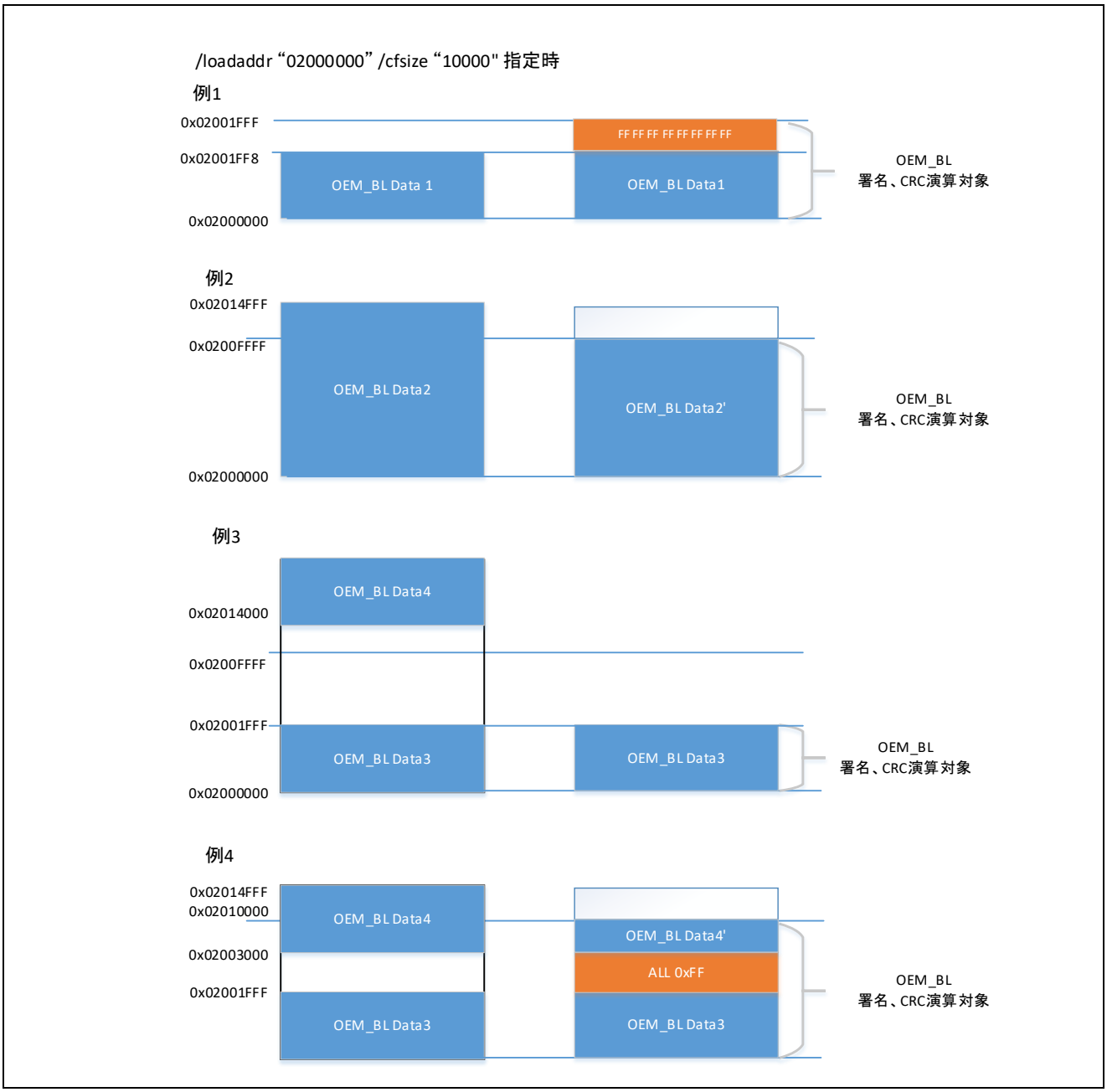

図 4.7　**cfsize** オプションのみ指定時の署名、CRC 演算対象

4.8 encdotf コマンド オプション

encdotfコマンドでは、以下のオプションが使用可能です。データ入力は16進データまたはmotファイルで指定します。ファイルを指定する場合、ファイルパスの先頭に”file=”をつけます。

表 4-83 **encdotf** オプション

オプション	パラメータ	説明
keytype	ASCII	使用するAES暗号鍵のbit長を指定します。 詳細は表 4-84を参照してください。
enckey	Hex data/ File Path	**keytype**オプションで指定したbit長の鍵データ、または鍵ファイルを指定します。 詳細は4.8.2**enckey** オプションを参照してください。
nonce	Hex data	暗号化で使用するnonceデータを指定します。データサイズは16byteです。入力されたデータの上位100bitをnonceとして使用します。 本オプションは省略可能です。本オプションを省略した場合、ツール内で生成したランダム値(*注1)をnonceとして使用します。
startaddr	Hex data	暗号化の開始アドレスを指定します。アドレスサイズは32bitです。 本オプションは省略可能です。本オプションを省略した場合、ファイル全体を暗号化します。(*注2) 16バイト境界のアドレスを指定してください。
endaddr	Hex data	暗号化の終了アドレスを指定します。アドレスサイズは32bitです。 本オプションは省略可能です。本オプションを省略した場合、ファイル全体を暗号化します。(*注2) **startaddr**オプションから**endaddr**オプションで指定したエリアのサイズが16バイト単位になるようにアドレスを指定してください。
prg	File Path	暗号化するファイルを指定します。
output	File Path	出力ファイル名を指定します。
motaddr0	なし	本オプション指定時、出力するmotファイルのアドレスの開始アドレスを0にします。 本オプション指定時は、**incplain**オプションおよび**destaddr**オプションは指定できません。
incplain	なし	**startaddr**オプションならびに**endaddr**オプション指定時に、暗号化対象以外のデータを出力ファイルにつけて、ファイル出力します。 本オプション指定時は、**motaddr0**オプションは指定できません。
destaddr	Hex data	暗号化するデータの転送先アドレスを指定します。 転送先アドレスは、暗号化時のカウンタ下位28bitに使用します。 本オプションは省略可能です。本オプションを省略した場合、暗号化範囲の先頭アドレスを転送先アドレスとします。 本オプション指定時は、**motaddr0**オプションは指定できません。

注 1： ツール内で生成するランダム値は十分な乱数性を保証するものではありません。

注 2 : /**staraddr** と/**endaddr** オプションを省略した場合、ファイル全体を暗号化します。データがないエリアならびに、開始アドレス、終了アドレスが 16 バイトアラインを取れていない場合、0x00 でデータを補完します。

暗号化範囲指定例：

```
> skmt.exe /encdotf /keytype "AES-128" /enckey "000102030405060708090A0B0C0D0E0F"
 /nonce "00112233445566778890000000" /startaddr "80000000" /endaddr "80000FFF"
 /prg "D:¥work¥program.srec" /incplain /output "D:¥work¥dotf_program.srec"
```

暗号化範囲を指定しない例：

```
> skmt.exe /encdotf /keytype "AES-128" /enckey "000102030405060708090A0B0C0D0E0F"
  /nonce "00112233445566778890000000" /prg "D:¥work¥program.srec"
  /output "D:¥work¥dotf_program.srec"
```

4.8.1 keytype オプション

keytypeオプションは、暗号化で使用する鍵の鍵長を、ASCIIで指定します。

表 4-84 暗号化のための鍵長

ASCII	説明
AES-128	鍵長128bitのAES鍵を設定します。
AES-192	鍵長192bitのAES鍵を設定します。
AES-256	鍵長256bitのAES鍵を設定します。

4.8.2 enckey オプション

enckeyオプションは、ヘキサデータやファイルを指定します。

keytype オプションで指定された ASCII 文字に従い、以下のフォーマットのデータまたはバイナリファイル、テキストファイルを渡します。データ入力またはテキストファイル入力の場合は、1 バイトを 16 進 ASCII 文字列で表現します。

表 4-85 AES-128

Bytes	Data
0-15	AES-128bit key data

表 4-86 AES-192

Bytes	Data
0-23	AES-192bit key data

表 4-87 AES-256

Bytes	Data
0-31	AES-256bit key data

4.9 encsfp コマンド オプション

encsfpコマンドでは、以下のオプションが使用可能です。データ入力は16進データまたはバイナリファイルで指定します。ファイルを指定する場合、ファイルパスの先頭に”file=”をつけてください。

注: 本バージョンの macOS 版では、本コマンドをサポートしていません。

表 4-88 **encsfp** オプション(1)

オプション	パラメータ	説明
mcu	ASCII	Secure Factory Programmingをサポートするマップ情報を指定します。 4.9.1**mcu** オプションに記載したASCII文字で指定します。
enckey	Hex data / File Path	ユーザプログラム、パラメータ情報の暗号化で使用するAES128bit鍵データを指定します。HEXデータ指定の場合、16バイトHEXデータを入力します。 ファイル入力の場合、バイナリ(`*.key`)、もしくはテキストファイル(`*.txt`)の指定が可能です。 本オプションは省略可能です。本オプションを省略した場合、ツール内で生成したランダム値(*注)を**enckey**として使用します
nonce_prm	Hex data	パラメータ情報の暗号化で使用するnonceデータを指定します。データサイズは12byteです。 本オプションは省略可能です。本オプションを省略した場合、ツール内で生成したランダム値(*注)を**nonce_prm**として使用します
nonce_prg	Hex data	ユーザプログラムの暗号化で使用するnonceデータを指定します。データサイズは12byteです。 本オプションは省略可能です。本オプションを省略した場合、ツール内で生成したランダム値(*注)を**nonce_prg**として使用します
trn	ASCII / Hex data	DLM遷移先情報を指定します。 4.9.2 **trn** オプションに記載したASCII文字で指定します。
prg	File Path	暗号化するユーザプログラム(motファイル)を指定します。 ユーザプログラムは最大３つまで指定することができます。 詳細は4.9.3 **prg** オプションを参照してください。
al2key	Hex data / File Path	AL2_KEYを指定します。HEXデータ指定の場合、16バイトHEXデータを入力します。ファイル入力の場合、バイナリ(`*.key`)、もしくはテキストファイル(`*.txt`)の指定が可能です。 **trn**オプションで**OEM_PL0_AL2**を指定した場合は入力が必要です。 本オプションは省略可能です。**trn**オプションで**OEM_PL0_AL2**を指定時に本オプションを省略した場合、ツール内で生成したランダム値(*注)を**al2key**として使用します

表 4-89 **encsfp** オプション(2)

オプション	パラメータ	説明
ufpk	Hex data / File Path	**enckey**, **al2key**の暗号化時に使用するUFPK値、または**genufpk**コマンドで生成されるUFPKファイルを指定します。データサイズは32byteです。
wufpk	File Path	Renesas Key Wrapping serviceでラップしたUFPK値が書かれたW-UFPKファイルを指定します。
iv_enckey	Hex data	**enckey**の暗号化時に使用するIV値を指定します。データサイズは16byteです。 本オプションは省略可能です。本オプションを省略した場合、ツール内で生成したランダム値(*注)をIVとして使用します
iv_al2key	Hex data	**trn**オプションで**OEM_PL0_AL2**指定時、**al2key**の暗号化時に使用するIV値を指定します。データサイズは16byteです。 本オプションは省略可能です。本オプションを省略した場合、ツール内で生成したランダム値(*注)をIVとして使用します
output	File Path	出力ファイル名(sfpファイル)を指定します。 sfpファイルのフォーマットは 付録 Secure Factory Programming File Formatをご参照ください。
output_al2key	File Path	**trn**オプションで**OEM_PL0_AL2**を指定し**al2key**オプションを省略した場合、鍵として使用したツール内で生成したランダム値(*注)を鍵ファイル(keyファイル)として出力します。
output_enckey	File Path	**enckey**オプションを省略した場合、鍵として使用したツール内で生成したランダム値(*注)を鍵ファイル(keyファイル)として出力します。

注: ツール内で生成するランダム値は十分な乱数性を保証するものではありません。

trnオプションでOEM_PL0_AL2指定例：

```
> skmt /encsfp /mcu "RA8x1" /enckey "000102030405060708090a0b0c0d0e0f"
  /nonce_prg "111111111111222222222222" /nonce_prm "333333333333444444444444"
  /trn "OEM_PL0_AL2" /prg "D:¥work¥program.mot"
  /al2key "77777777777777778888888888888888"
  /ufpk file="ufpk.key" /wufpk "ufpk.key_enc.key"
  /iv_enckey "99999999999999999aaaaaaaaaaaaaaa"
  /iv_al2key "ddddddddddddddddeeeeeeeeeeeeeeee"
  /output " D:¥work¥ program.sfp"
```

enckeyならびにal2_keyオプション省略例：

```
> skmt /encsfp /mcu "RA8x1" /trn "OEM_PL0_AL2" /prg "D:¥work¥program.mot"
  /ufpk file="ufpk.key" /wufpk "ufpk.key_enc.key"
  /output "D:¥work¥program.sfp"
  /output_enckey "D:¥work¥enckey.key" /output_al2key "D:¥work¥al2key.key"
```

4.9.1 mcu オプション

mcuオプションは、ASCII文字でMCU情報を指定します。

表 4-90 **MCU 情報の指定**

ASCII	説明
RA8x1	RA8x1のマップ情報を指定します。

4.9.2 trn オプション

trnオプションは、ASCII文字でDLM遷移情報を指定します。

表 4-91 DLM 遷移と入力する認証鍵

ASCII	説明
OEM_PL0_AL2	DLM OEM PL0 に転送かつ AL2 鍵を書き込みます。
LCK_BOOT	DLM LCK_BOOT に転送します。

trnオプションで OEM_PL0_AL2指定例：

```
> skmt /encsfp /mcu "RA8x1" /enckey "000102030405060708090a0b0c0d0e0f"
  /nonce_prg "111111111111222222222222" /nonce_prm "333333333333444444444444"
  /trn "OEM_PL0_AL2" /prg "D:¥work¥program.mot"
  /al2key "77777777777777778888888888888888"
  /ufpk file="ufpk.key" /wufpk "ufpk.key_enc.key"
  /iv_enckey "9999999999999999aaaaaaaaaaaaaaaa"
  /iv_al2key "ddddddddddddddddeeeeeeeeeeeeeeee"
  /output " D:¥work¥ program.sfp"
```

trnオプションで LCK_BOOT指定例：

```
> skmt /encsfp /mcu "RA8x1" /enckey "101112131415161718191a1b1c1d1e1f"
  /nonce_prg "111111111111222222222222" /nonce_prm "333333333333444444444444"
  /trn "LCK_BOOT" /prg "D:¥work¥program.mot"
  /ufpk file="ufpk.key" /wufpk "ufpk.key_enc.key"
  /iv_enckey "9999999999999999aaaaaaaaaaaaaaaa" /output " D:¥work¥ program.sfp"
```

4.9.3 prg オプション

prgオプションは、Secure Factory Programming機能で暗号化するユーザプログラムを指定します。
prgオプションは、最大３つまで指定することができます。

prgオプション 3ファイル指定例：

```
> skmt /encsfp /mcu "RA8x1" /enckey "101112131415161718191a1b1c1d1e1f"
  /nonce_prg "111111111111222222222222" /nonce_prm "333333333333444444444444"
  /trn "LCK_BOOT"
  /prg "D:¥work¥program1.mot" "D:¥work¥program2.mot" "D:¥work¥program3.mot"
  /ufpk file="ufpk.key" /wufpk "ufpk.key_enc.key"
  /iv_enckey "9999999999999999aaaaaaaaaaaaaaaa" /output " D:¥work¥ program.sfp"
```

4.10 **calcresponse** コマンド オプション

表 4-92 **calcresponse** オプション

オプション	パラメータ	説明
challenge	Hex data	チャレンジ値を指定します。（通常はデバイスのユニークIDを指定） データサイズは16バイト。
key	Hex data	DLM鍵データを指定します。データサイズは16バイト。
algorithm	Name	計算アルゴリズムを指定します。以下が選択可能です。 **HMAC-SHA256** **AES-128-CMAC**

calcresponseオプション使用時の使用例：

```
> skmt.exe /calcresponse /challenge "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345"
        /key "000102030405060708090A0B0C0D0E0F" /algorithm HMAC-SHA256
```

5.　操作手順

5.1　Standalone 版

5.1.1　Windows 版

インストーラ SecurityKeyManagementTool_installer_vXXX.exe を実行し、任意のフォルダへインストールしてください。

【注意】
書き込み権限があるなるべく浅い階層で「日本語名」や「スペース」が入っていない
フォルダにインストールしてください。
深い階層やフォルダの名前が長い場合に、起動できない場合があります。

5.1.1.1　GUI 版

インストールフォルダ内のSecurityKeyManagementTool.exeが実行ファイルです。

図 5.1　Security Key Management Tool – 起動時のダイアログ

5.1.1.2　　CLI 版

インストールしたフォルダ内のCLIフォルダに格納されている以下ファイルが、CLI版の一式になります。CLI版だけで使用される場合、下記ファイルとフォルダ(**Bold文字**)一式を任意のフォルダに置いて使用してください。

- skmt.exe
- **device**
- **address**

コマンドプロンプトからskmt.exeコマンドを入力してください。

図 5.2　　コマンドプロンプトによる Security Key Management Tool - CLI 実行例

5.1.2　Linux 版

5.1.2.1　GUI 版

Linux版の解凍したフォルダ内のSecurityKeyManagementToolが実行ファイルです。
解凍時、実行権限を維持するためtar xvzfp xxxxx.tar.gzコマンドで解凍してください。

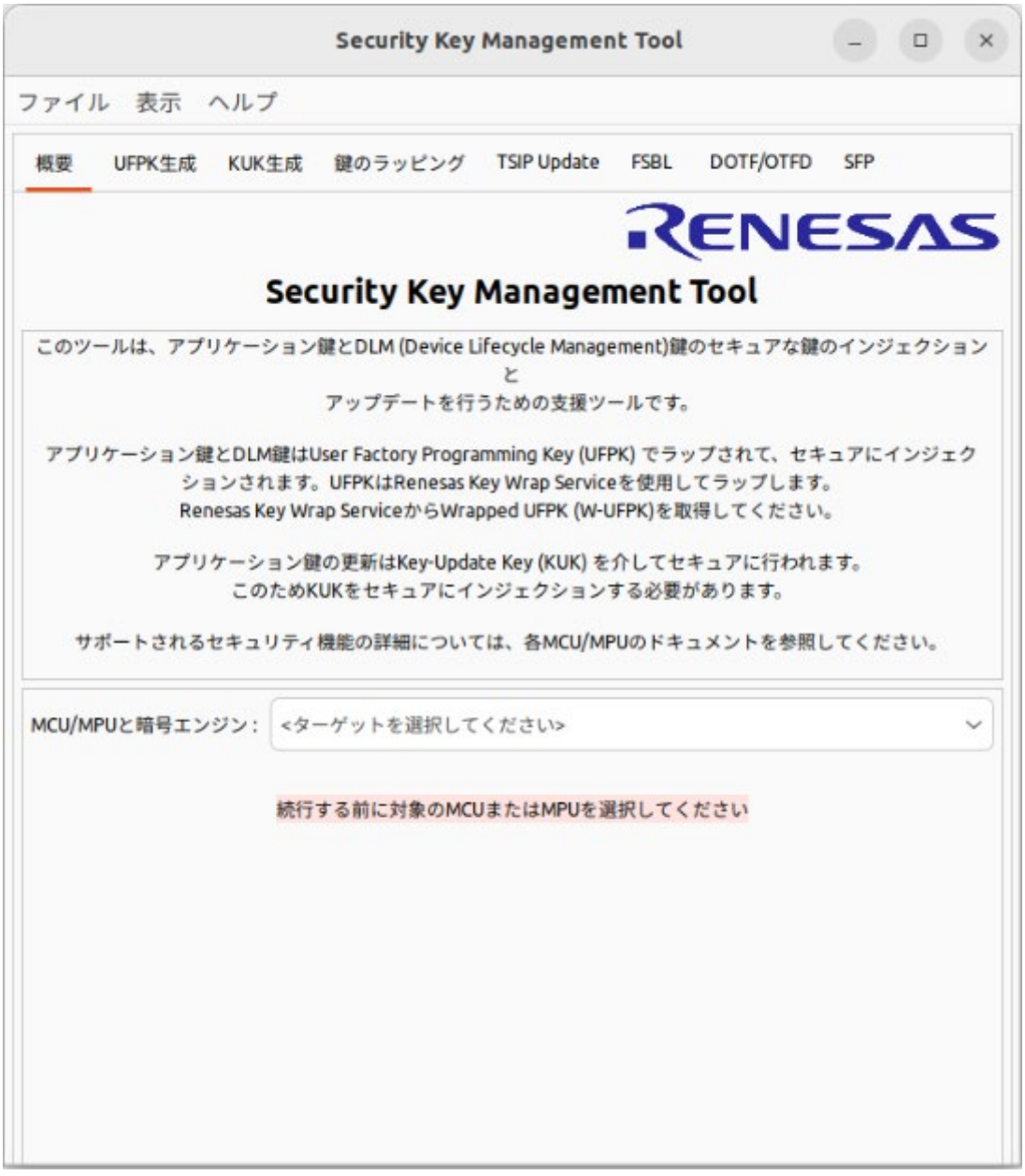

図 5.3　Security Key Management Tool – 起動時ダイアログ

5.1.2.2　　CLI 版

ダウンロードしたLinux版のパッケージを任意のフォルダに解凍してください。CLIフォルダに入っている以下ファイル一式を任意のフォルダに置いてください。

- skmt
- **device**
- **address**

ターミナルソフトからskmtコマンドを入力してください。

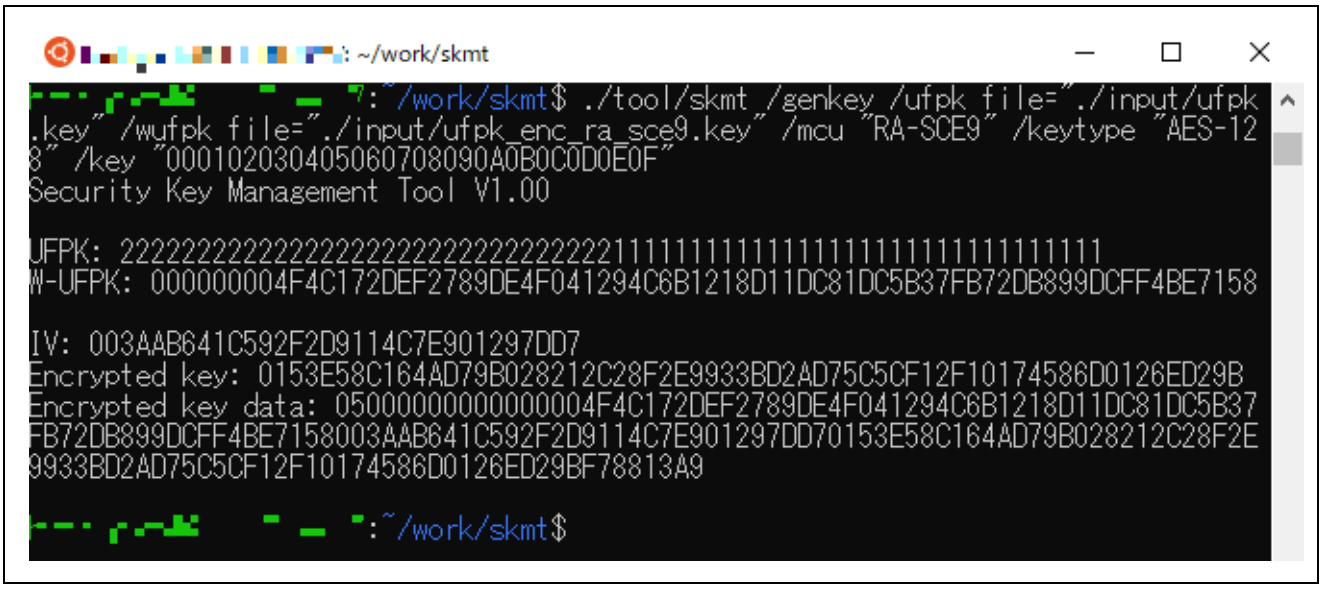

図 5.4　　Linux ターミナルソフトによる Security Key Management Tool - CLI 実行例

5.1.3　macOS 版

ダウンロードしたパッケージを任意のフォルダで解凍してください。
解凍後のファイル構成は以下のようになっています。
SecurityKeyManagementTool.app.zip　：GUI 版
skmt-cli.zip　　：　CLI 版

5.1.3.1　GUI 版

解凍したフォルダ内のSecurityKeyManagementTool.app.zipを任意のフォルダで解凍してください。
SecurityKeyManagementTool.appが実行ファイルです。

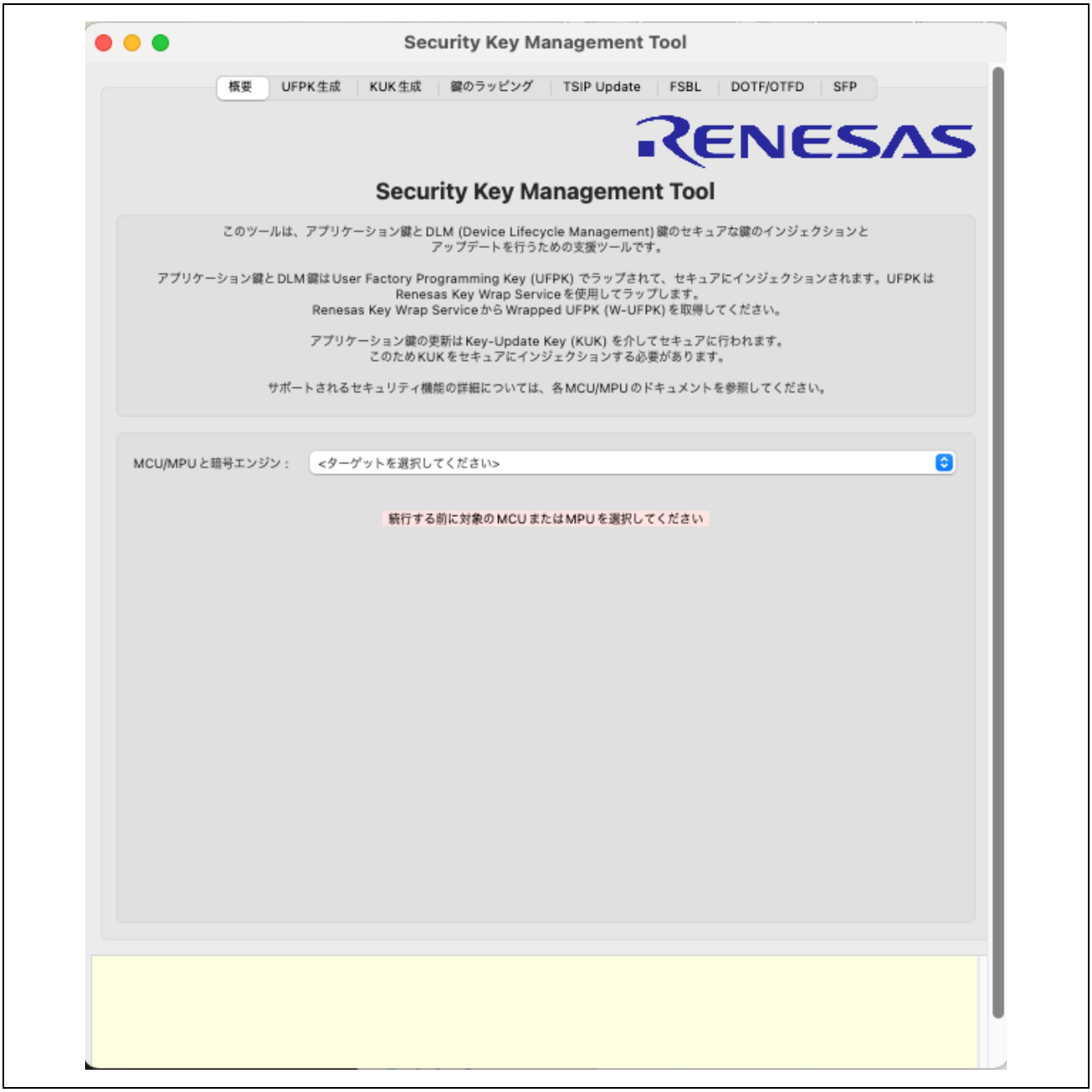

図 5.5　Security Key Management Tool – 起動時ダイアログ

5.1.3.2　CLI 版

解凍したフォルダ内の、SKMT_CLI.zipを任意のフォルダに解凍してください。フォルダに入っている以下ファイル一式を任意のフォルダに置いてください。

- skmt
- **device**
- **address**

ターミナルソフトからskmtコマンドを入力してください。

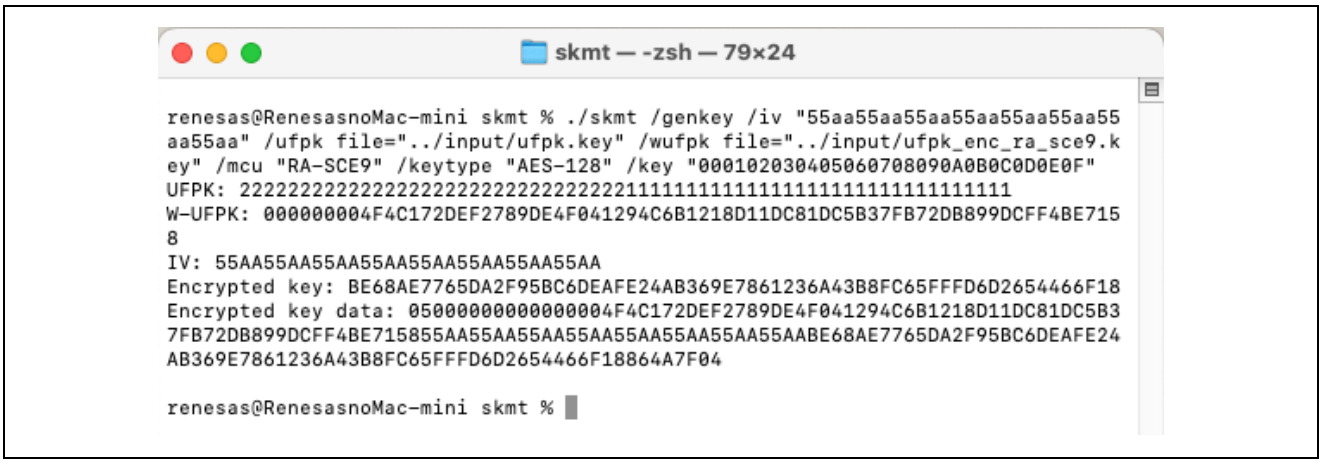

図 5.6　macOS ターミナルソフトによる Security Key Management Tool - CLI 実行例

5.2　e²studio plugin 版

Windows 版、Linux 版、macOS 版は、同じ手順でインストール、アンインストールを行います。
以下手順でインストールならびにアンインストールを行ってください。

5.2.1　インストール方法

1. Renesas Webサイトからダウンロードした SecurityKeyManagementTool_Plugin_vXXX_Windows.zip、SecurityKeyManagementTool_Plugin_vXXX_mac.zipもしくは SecurityKeyManagementTool_Plugin_vXXX_Linux.zipを、任意のフォルダに置いてください。
2. e²studioを起動します。
3. e²studioのメニュー [**ヘルプ(H)**]→[**新規ソフトウェアのインストール...**]メニューを選択し、[**インストール**]ダイアログを開きます。

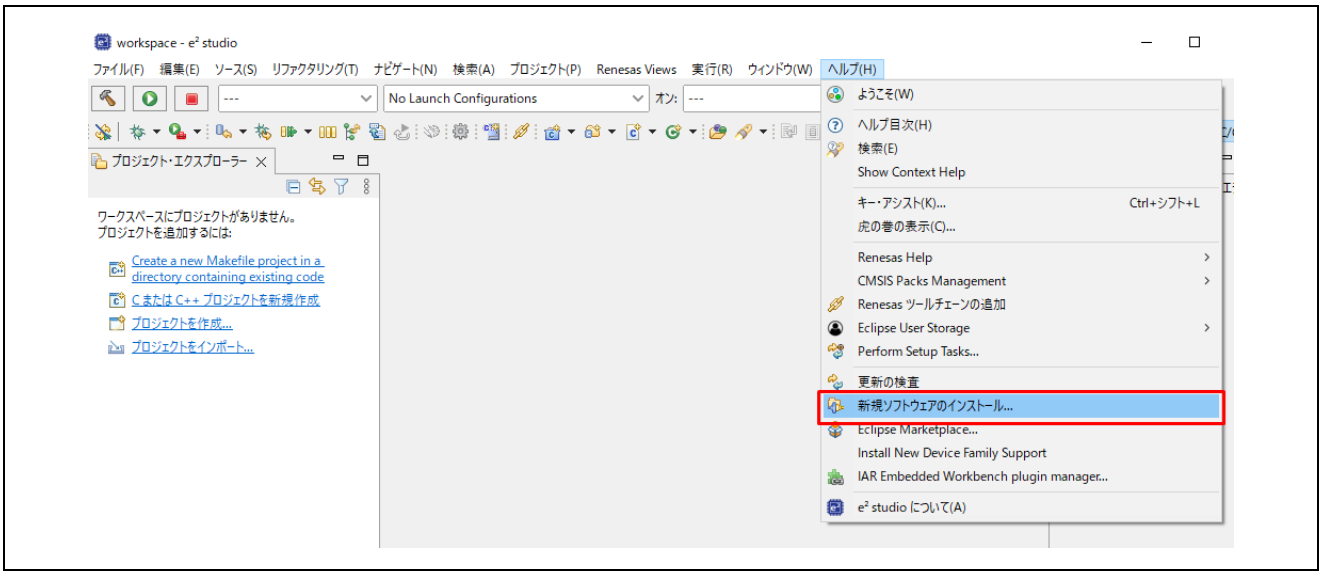

図 5.7　e²studio のメニュー [ヘルプ(H)]→[**新規ソフトウェアのインストール...**]

4. [**インストール**]ダイアログの[**追加**]ボタンを押して、[**リポジトリーを追加**]ダイアログを開きます。「必要なソフトウェアを見つけるために、インストール中に更新サイトすべてに接続」のチェックを入れたままだと、インストールに時間かがかかりますので、チェックは外してください。

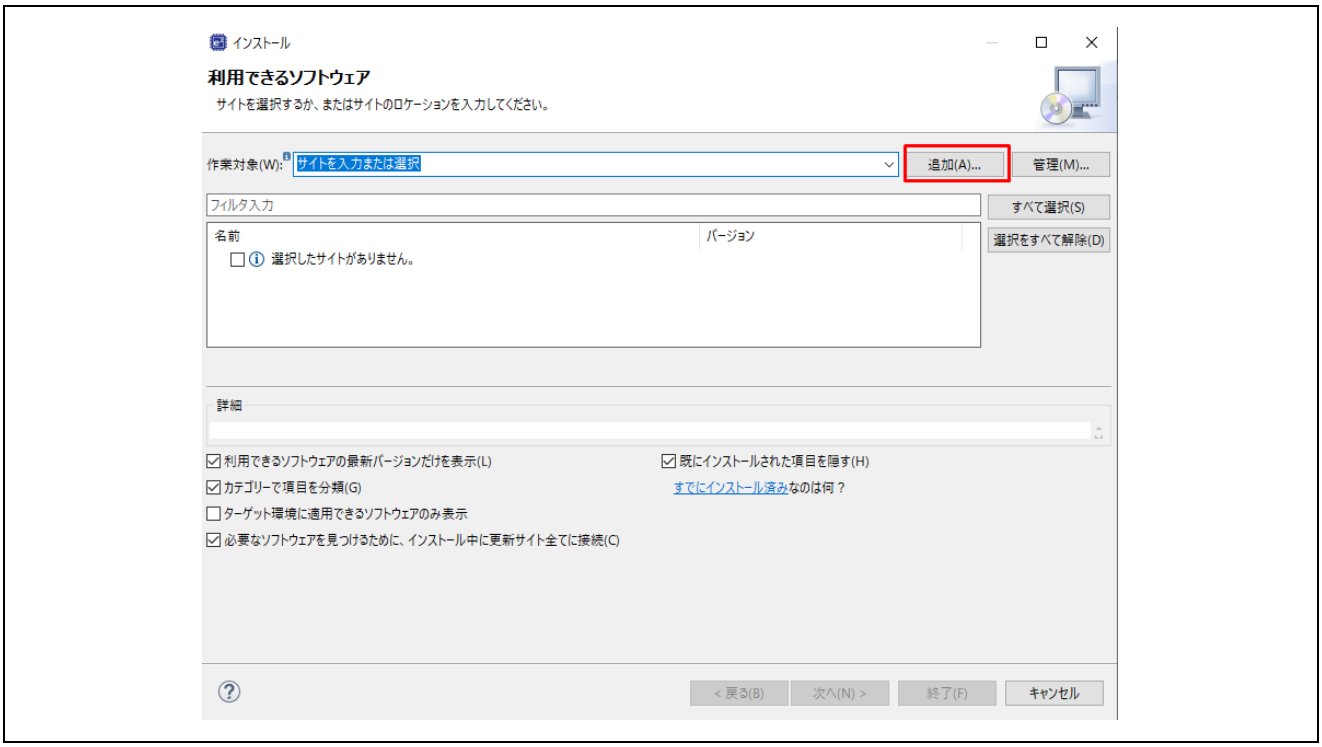

図 5.8　e²studio [**インストール**]ダイアログ

5. [**リポジトリーを追加**]ダイアログの"**アーカイブ**"ボタンを押して、1で用意したSecurityKeyManagementTool_Plugin_vXXX_Window/Linux/mac.zipを指定後、"**追加**"ボタンを押します。

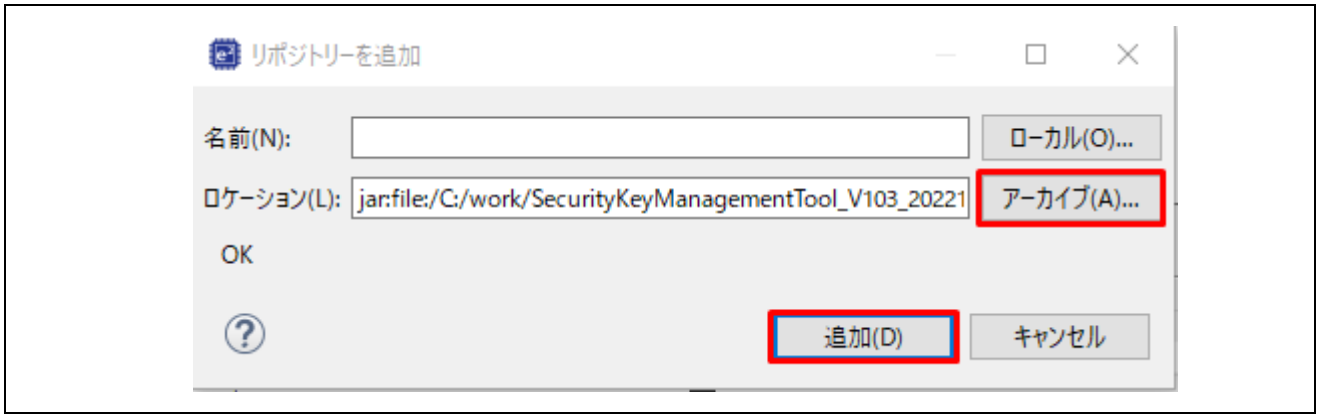

図 5.9　e²studio [**リポジトリーを追加**]ダイアログ

6. [**インストール**]ダイアログに、「Renesas Solution Toolkit」が追加されますので、チェックボックスにチェックを入れて、”**次へ**”ボタンを押します。
“必要なソフトウェアを見つけるために、インストール中に更新サイト全てに接続”のチェックをつけたままの場合、インストールに時間がかかる場合があります。チェックを外すことをお勧めします。

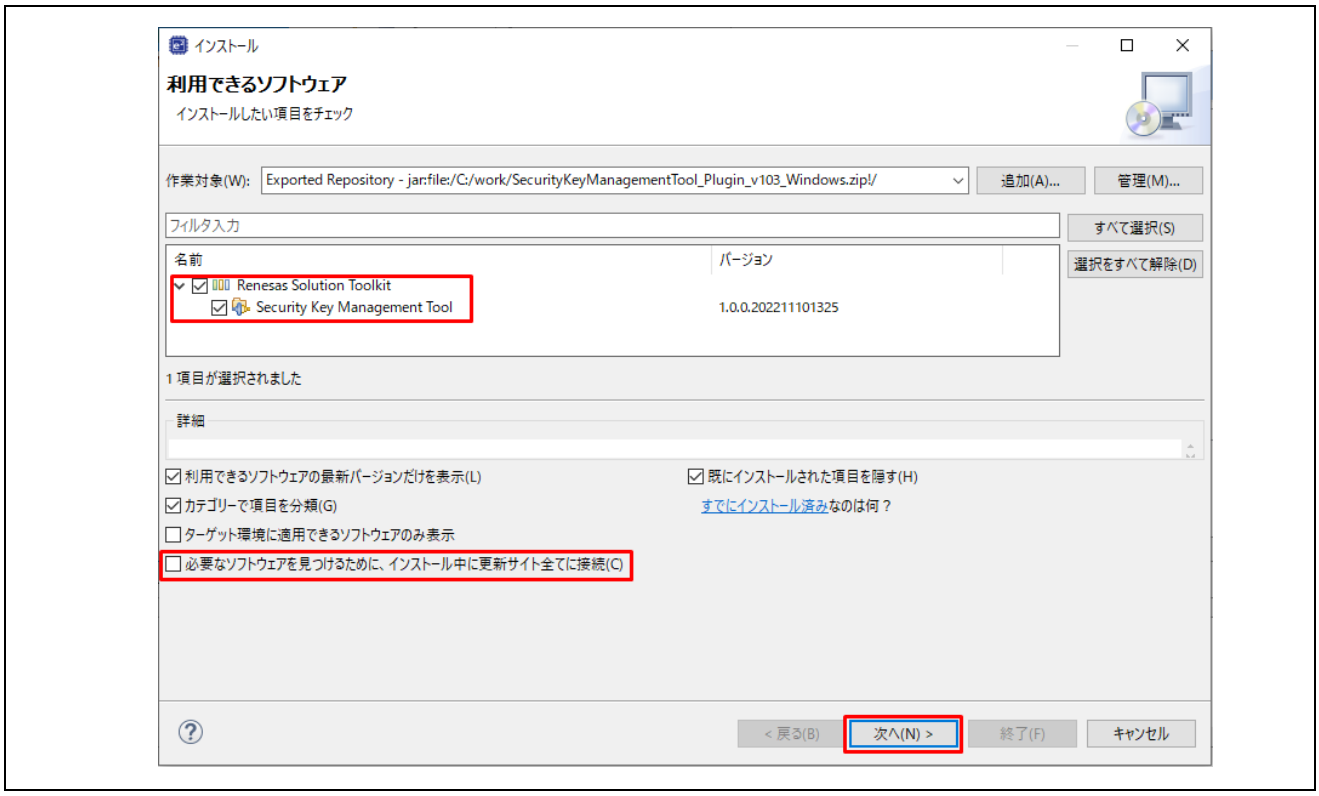

図 5.10　e²studio [**インストール**]ダイアログ - Security Key Management Tool の指定

7. [**インストール**]ダイアログが表示されるので、「Security Key Management Tool」がインストールされることを確認して、”**終了**”ボタンを押してください。

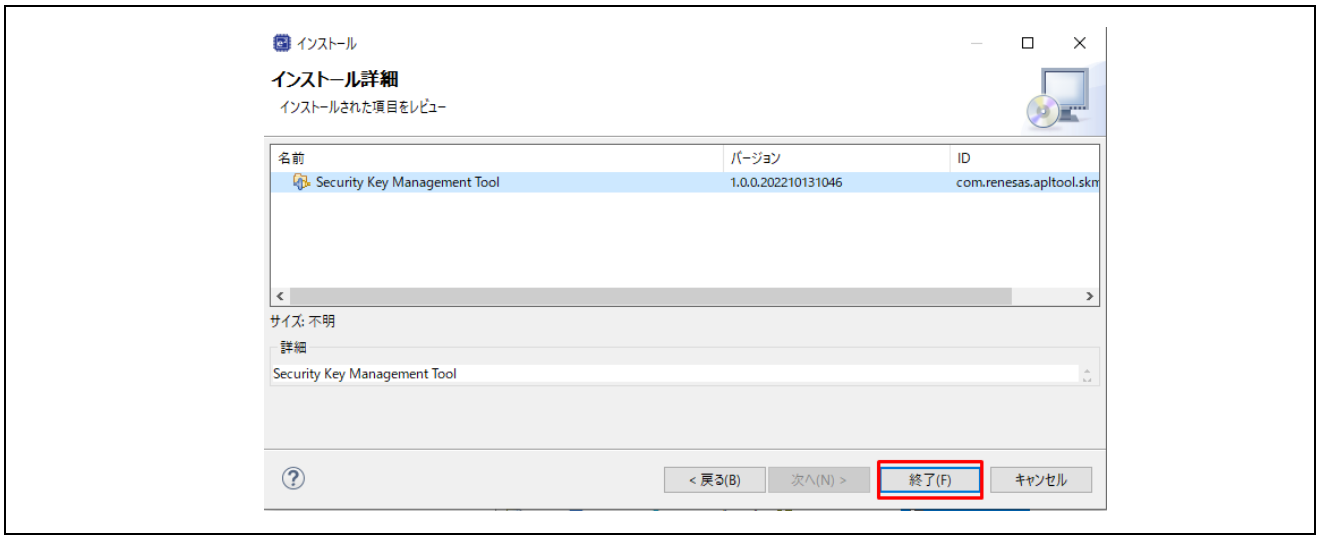

図 5.11　e²studio [**インストール**]ダイアログ - インストール詳細

8. 続いて[ライセンスをレビュー]画面が表示されます。プラグイン版をダウンロード時に提示されたライセンス条項のURLが表示されます。”使用条件の条項に同意します(A)”を選択し、”**終了**”ボタンを押してください。

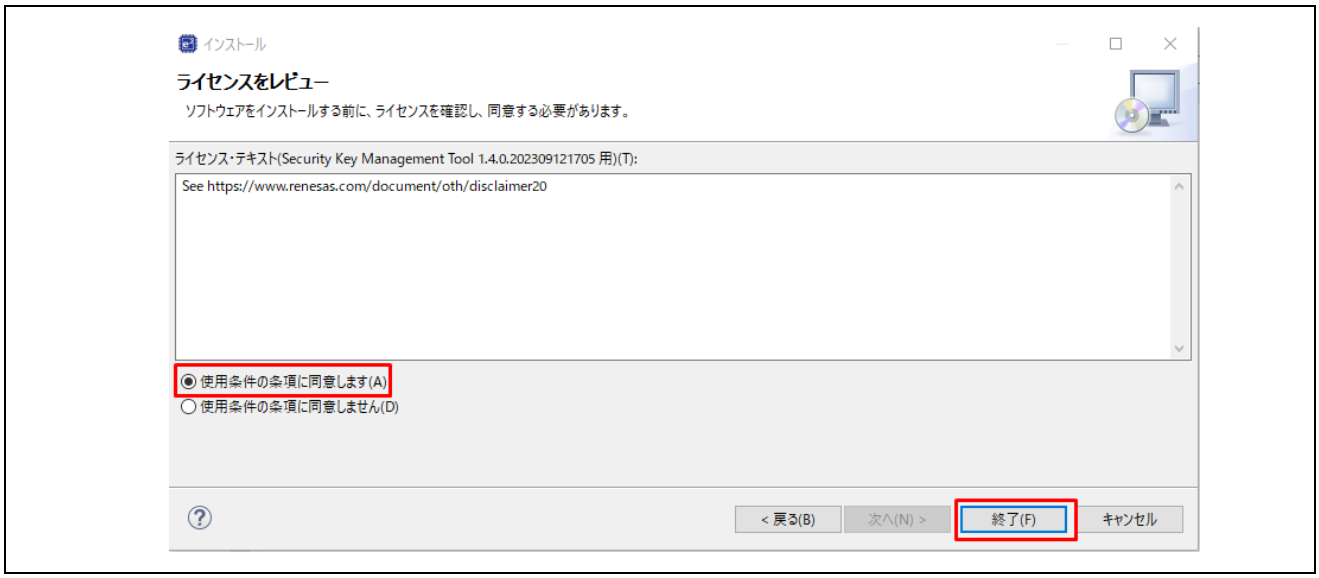

図 5.12　e²studio [**インストール**]ダイアログ – ライセンスをビュー

9. 信頼する証明書の選択ダイアログが表示された場合、表示された証明書をチェックした後、”Trust Selected”ボタンを押して、インストールを継続してください。

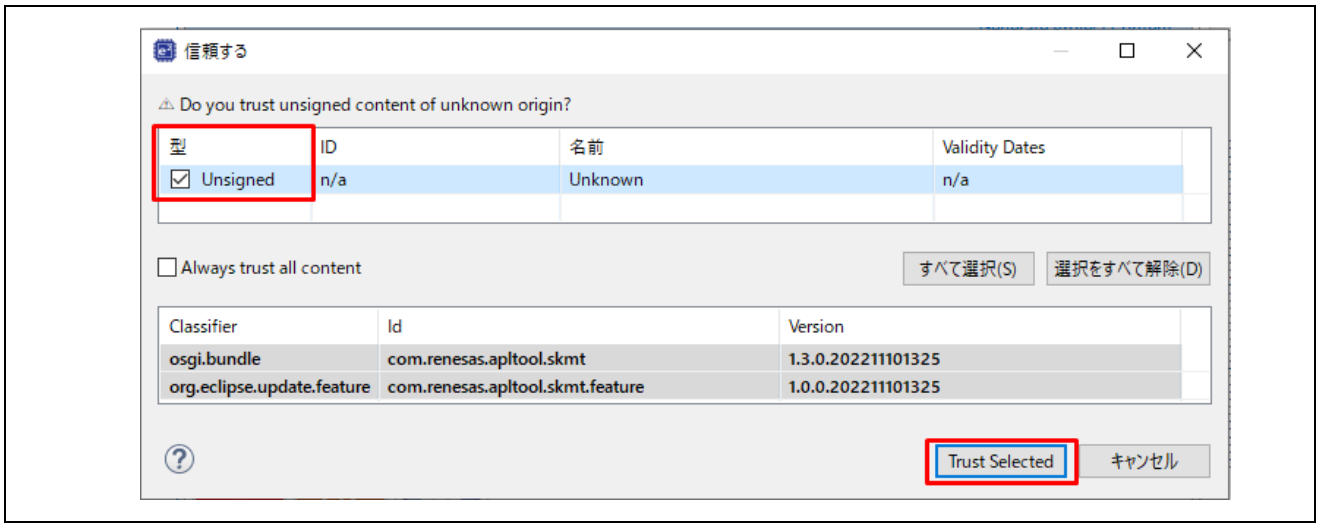

図 5.13　e²studio [**信頼する**]ダイアログ

10. e²studioの再起動を促されるので、e²studioを再起動します。

11. インストールが完了したら、プロジェクトの[**プロパティ**]ダイアログに、Security Key Management Toolが追加されます。

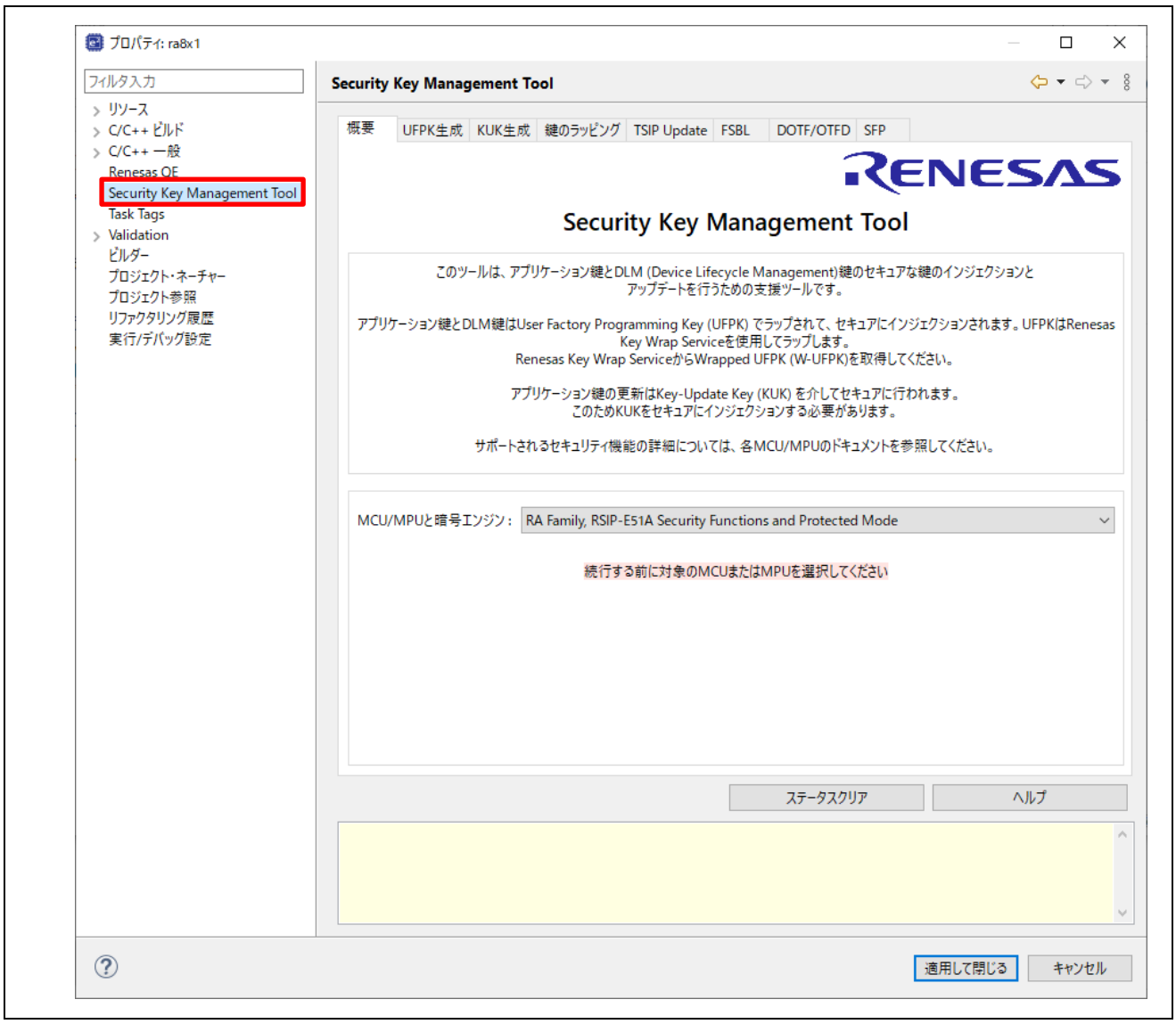

図 5.14　e²studio Security Key Management Tool プラグイン

5.2.2　アンインストール方法

1. e²studioのメニュー [**ヘルプ(H)**]→[**e²studioについて**]を選択し、[e²studioについて]ダイアログを表示します。
2. [**e²studioについて**]ダイアログの"**インストール詳細**"ボタンを押します。

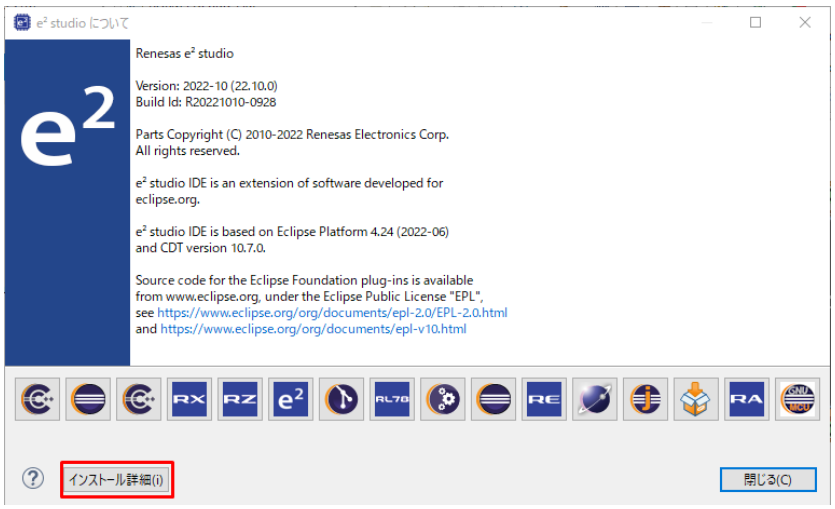

図 5.15　e²studio [**e²studio について**]ダイアログ

3. [**e²studioのインストール詳細**]ダイアログの[**インストールされたソフトウェア**]タブ→Security Key Management Toolを選択し、"**アンインストール(U)...**"ボタンを押します。

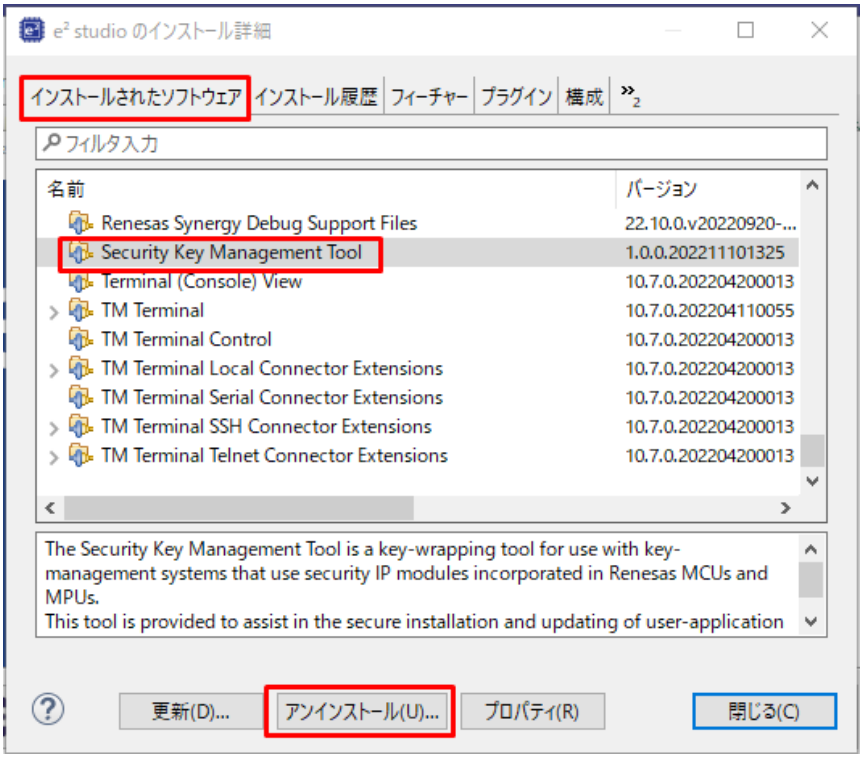

図 5.16　e²studio [**e²studio のインストール詳細**]ダイアログ

4. [**アンインストール**]ダイアログが表示されるので、Security Key Management Toolを選択して、”**終了**”ボタンを押してください。

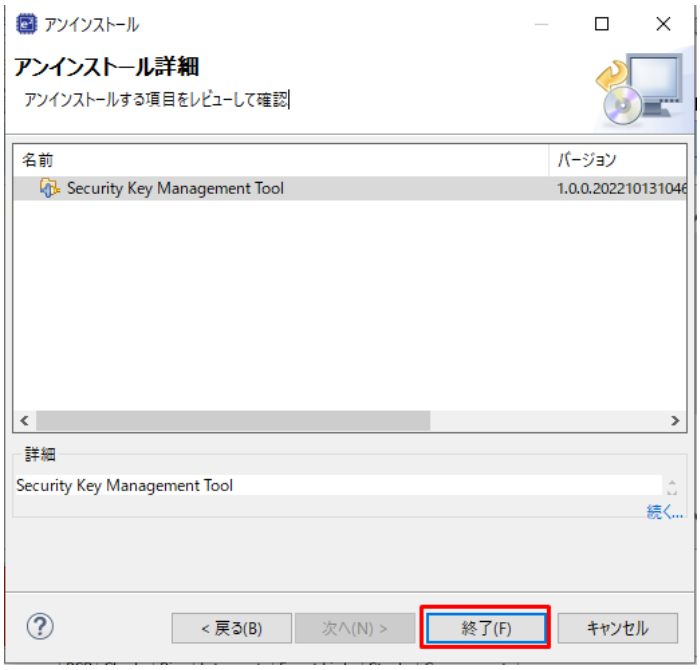

図 5.17　e²studio [**アンインストール**]ダイアログ

5. e²studioの再起動を促されるので、e²studioを再起動してください。

6. 使用例

6.1 Example 1 – RX ファミリ TSIP の AES 128 の鍵のインジェクション手順

RX ファミリの TSIP は、MCU 上で実行されるファームウェアを介したセキュアな鍵のインジェクションをサポートしています。必要なドライバは、表 1-1 MCU/MPU 関連サイトの TSIP を参照してください。

生産工程では、多くの場合、同じ鍵を持つデバイス群をプログラムする必要があります。鍵のインジェクション情報はプロビジョニングコードに埋め込むことができますが、異なる鍵を持つ複数のグループがある場合、鍵情報をプロビジョニングコードから分離することが望ましい場合があります。次の手順で Motorola Hex ファイルを作成し、プロビジョニングコードと一緒にプログラムすることで、1 つまたは複数の鍵をセキュアにインジェクションすることができます。この例では、AES128 鍵をインジェクションについて説明します。

6.1.1 GUI 版の Security Key Management Tool を使用する場合

1. MCU/MPU と暗号エンジンの選択
   [**概要**]タブで MCU/MPU と暗号エンジンを選択します。

図 6.1 [**概要**]タブ

2. UFPK を生成

[UFPK 生成] タブに UFPK 値を設定して、拡張子*.key というファイル名で、UFPK ファイルを生成します。この例では ufpk.key というファイル名を使用しています。

図 6.2 **[UFPK 生成]**タブ 指定値で UFPK を生成する場合の例

“**UFPK ファイルを生成する**”ボタンを押すと UFPK ファイルが生成されます。正常にファイルが生成された場合、以下のような実行結果が出力されます。

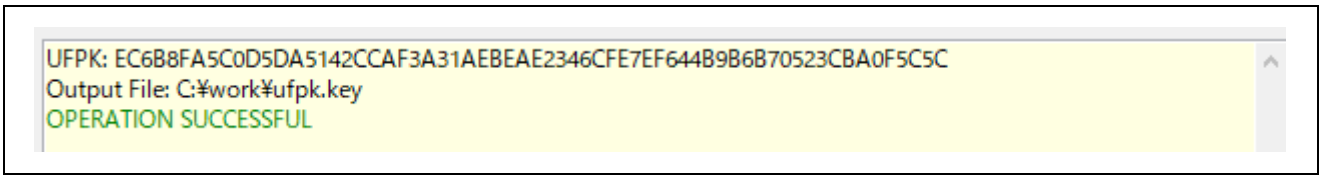

図 6.3 **[UFPK 生成]**タブ 実行結果

3. W-UFPK の取得
2 で生成した ufpk.key ファイルを Renesas Key Wrap service(https://dlm.renesas.com/keywrap)に送付して W-UFPK を取得します。
詳細な取得情報は、Renesas Key Wrap Service の FAQ もしくはアプリケーションノートをご参照ください。

4. AES128 鍵ファイルを Motorola Hex ファイルで生成
   [**鍵のラッピング**]タブで AES 128 鍵ファイルを生成します。
   [**鍵の種類**]タブで”AES128”を選択後、[**鍵データ**]タブで AES128 の鍵データを入力してください。
   “Wrapping key”には、手順 2 で生成した UFPK ファイルと手順 3 で取得した W-UFPK ファイルを設定してください。この例では簡略化のため IV は”**乱数生成機能を使用する**”を選択します。
   フォーマットに”**モトローラヘキサ**”を選択し、アドレスに FFFF0000 を入力します。
   エンディアンは”Little”を指定します。

図 6.4　[**鍵のラッピング**] - [**鍵の種類**]タブ AES128 鍵ファイル Motorola Hex 出力設定例

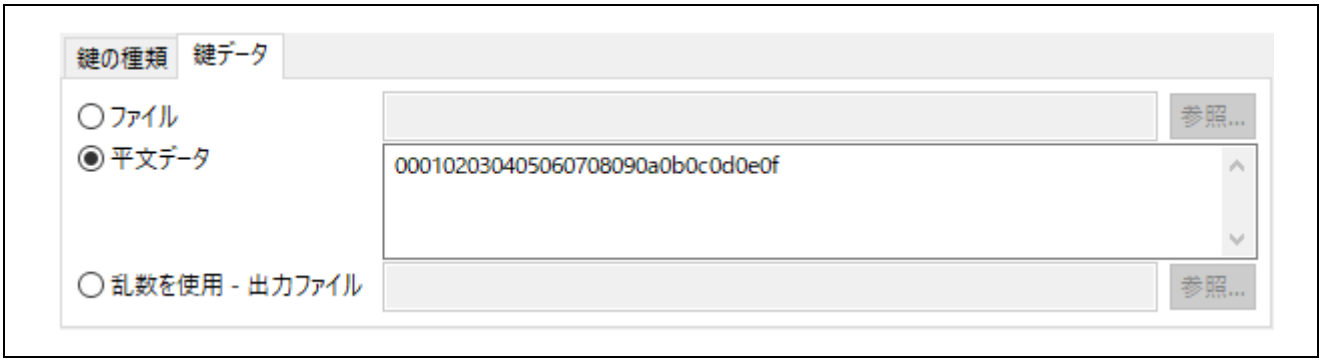

図 6.5　[**鍵のラッピング**] - [**鍵のデータ**]タブ AES128 鍵ファイル Motorola Hex 出力設定例

正常に終了すると以下のように表示されます。

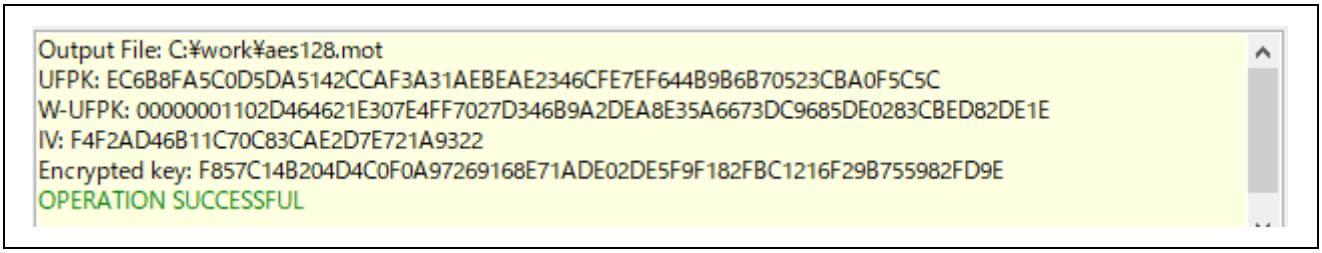

図 6.6　[**鍵のラッピング**]タブ 実行結果

6.1.2　CLI 版の Security Key Management Tool を使用する場合

1. UFKP の生成
   ターミナルソフトで以下のコマンドを実行します。

   > skmt.exe /genufpk
   /ufpk "ec6b8fa5c0d5da5142ccaf3a31aebeae2346cfe7ef644b9b6b70523cba0f5c5c"
   /output "C:¥work¥ufpk.key"

```
C:¥work¥skmt¥tool>skmt.exe /genufpk /ufpk "ec6b8fa5c0d5da5142ccaf3a31aebeae2346cfe7
ef644b9b6b70523cba0f5c5c" /output "C:¥work¥ufpk.key"
UFPK: EC6B8FA5C0D5DA5142CCAF3A31AEBEAE2346CFE7EF644B9B6B70523CBA0F5C5C
Output File: C:¥work¥ufpk.key
```

図 6.7　**genufpk** コマンド実行結果

2. W-UFPK の取得
   1 で生成した `ufpk.key` ファイルを Renesas Key Wrap service(https://dlm.renesas.com/keywrap)に送付して W-UFPK を取得します。
   詳細な取得情報は、Renesas Key Wrap Service の FAQ もしくはアプリケーションノートをご参照ください。

3. AES128 鍵ファイルを Motorola Hex ファイルで生成
   以下の **genkey** コマンドをターミナルソフトで実行します。

   > skmt.exe /genkey /ufpk file="c:¥work¥ufpk.key" /wufpk file="C:¥work¥ufpk.key_enc.key"
   /mcu "RX-TSIP" /keytype "AES-128" /key "000102030405060708090A0B0C0D0E0F"
   /filetype "mot" /address "FFFF0000" /output "C:¥work¥aes128.mot"

   UFPK ファイルは手順 1、W-UFPK ファイルは手順 2 で生成したものを使用します。

```
C:¥work¥skmt¥tool>skmt.exe /genkey /ufpk file="c:¥work¥ufpk.key" /wufpk file="C:¥wo
rk¥ufpk.key_enc.key" /mcu "RX-TSIP" /keytype "AES-128" /key "000102030405060708090A
0B0C0D0E0F" /filetype "mot" /address "FFFF0000" /output "C:¥work¥aes128.mot"
Output File: C:¥work¥aes128.mot
UFPK: EC6B8FA5C0D5DA5142CCAF3A31AEBEAE2346CFE7EF644B9B6B70523CBA0F5C5C
W-UFPK: 00000001102D464621E307E4FF7027D346B9A2DEA8E35A6673DC9685DE0283CBED82DE1E
IV: 62AB03F5F23A3E9F0D1AEAABD79DEECC
Encrypted key: 3DF63F89F22E4795F333B937D6971AAAF8A41C393636ED8B866E3FDDC091190B
```

図 6.8　**genkey** コマンド実行例

6.2 Example 2 – RA ファミリ SCE9 Protected Mode Key-Update Key のインジェクション

SCE9 搭載の RA ファミリ MCU は、プログラミングインタフェース経由でのセキュアな鍵のインジェクションをサポートしています。Renesas Flash Programmer はこの機能をサポートしています。

プログラミングインタフェースを使用して鍵をセキュアにインジェクションする利点は、MCU に特別なプロビジョニングコードが必要ないことです。鍵とアプリケーションコードは、同じプログラミングプロセスの一部としてインジェクションすることができます。この例では、KUK を 1 つインジェクションすることで、現場での鍵のアップデートを可能にします。

6.2.1 GUI 版の Security Key Management Tool を使用する場合

1. MCU/MPU と暗号エンジンの選択
   [**概要**]タブで MCU/MPU と暗号エンジンを選択します。

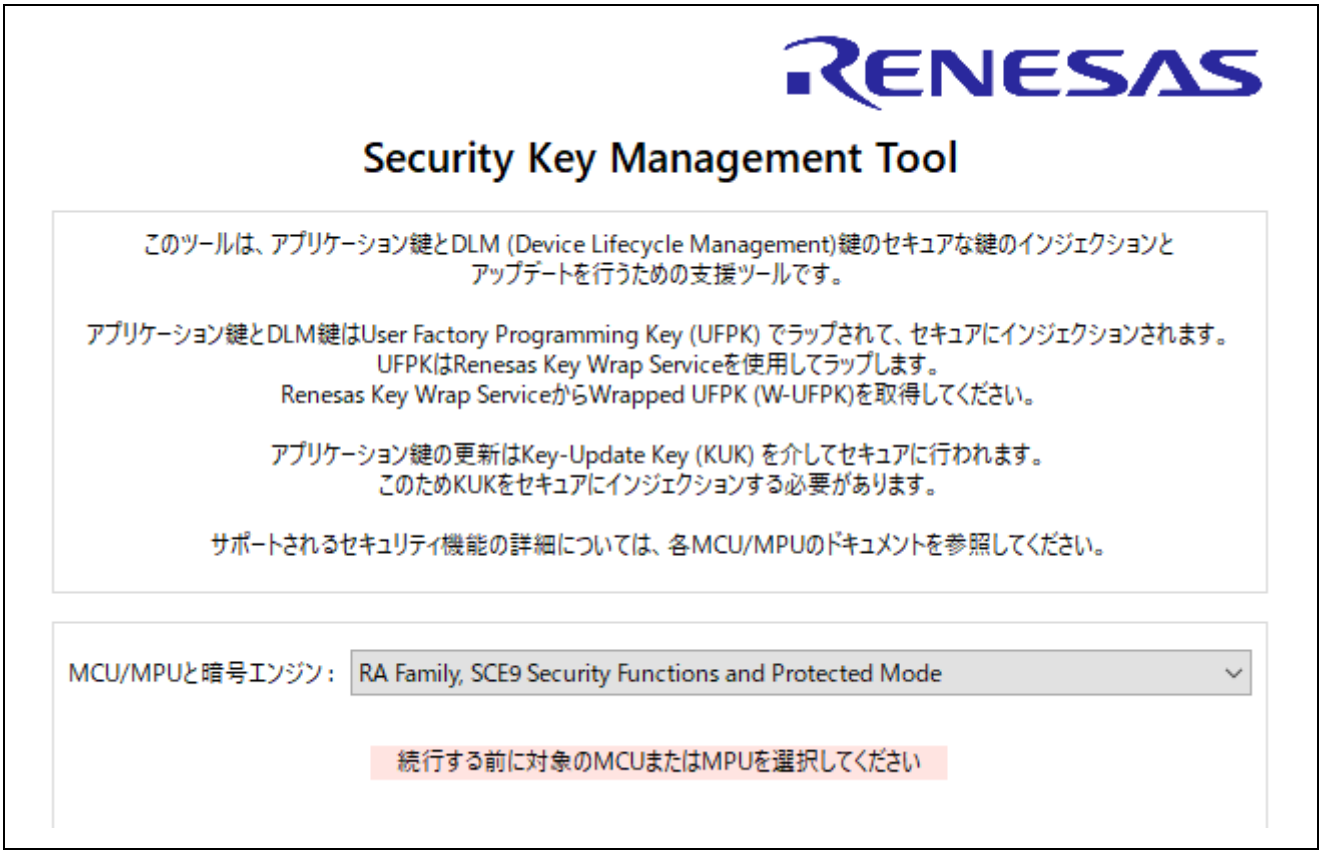

図 6.9 [**概要**]タブ

2. UFPK の生成
   [**UFPK 生成**]タブで UFPK 値をセットして、拡張子*.key というファイル名の UFPK ファイルを生成します。この例では ufpk.key というファイル名を使用します。

図 6.10 [**UFPK 生成**]タブ UFPK 生成設定例

"UFPK **ファイルを生成する**"ボタンを押すと UFPK ファイルが生成されます。
正常終了すると以下の表示がされます。

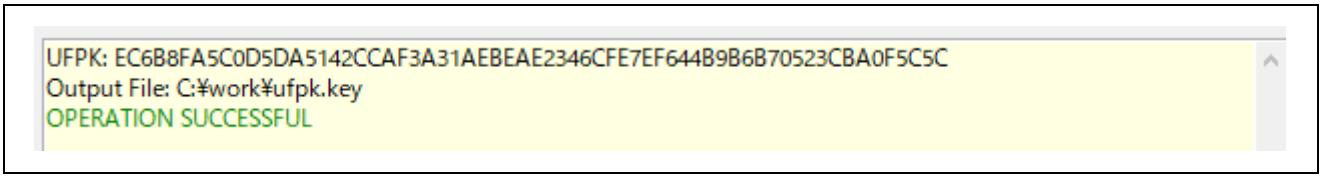

図 6.11 [**UFPK 生成**]タブ 実行結果

3. W-UFPK 取得
   2 で生成した ufpk.key ファイルを Renesas Key Wrap service(https://dlm.renesas.com/keywrap)に送付して W-UFPK を取得します。
   詳細な取得情報は、Renesas Key Wrap Service の FAQ もしくはアプリケーションノートをご参照ください。

4. KUK の生成
   [**KUK 生成**]タブでツールを使って KUK 用のランダム値を生成するか、256bit の KUK を入力するか選択します。出力ファイル名は拡張子*.key という名前で設定します。この例では kuk.key というファイル名を使用します。

図 6.12　[**KUK 生成**]タブ KUK ファイル生成設定例

"KUK **ファイルを生成する**"ボタンを押すと KUK ファイルが生成されます。正常終了すると以下のように表示されます。

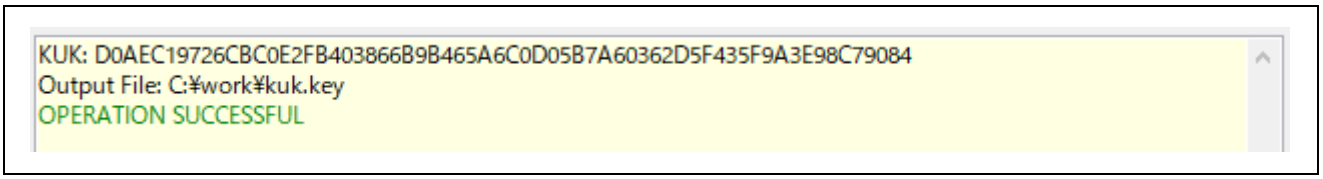

図 6.13　[**KUK 生成**]タブ 実行結果

5. RFP ファイルフォーマットの KUK ファイルの生成
   [**鍵のラッピング**] タブの [**鍵の種類**] タブで KUK を選択します。[**鍵のデータ**]タブで、前のステップで生成した KUK ファイル名(この例では kuk.key)を入力します。[**鍵のラッピング**]は "UFPK "を選択し、手順 2 で生成した UFPK ファイルと手順 3 で取得した W-UFPK ファイルを選択します。この例では簡略化のため、IV は"**乱数生成機能を使用する**"を選択します。"**出力**"パネルで、フォーマットとして「RFP」を選択し、拡張子が*.rkey のファイル名を入力します。

図 6.14 [**鍵のラッピング**] – [**鍵の種類**]タブ KUK ファイルを RFP ファイルで出力する設定例

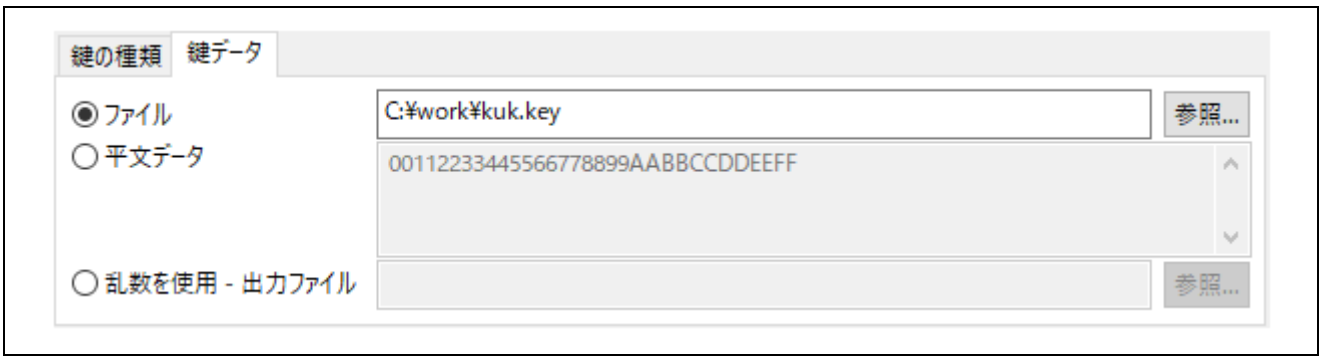

図 6.15 [**鍵のラッピング**] – [**鍵データ**]タブ KUK ファイルを RFP ファイルで出力する設定例

正常終了すると以下のように出力されます。

図 6.16 [**鍵のラッピング**]タブ 実行結果

6.2.2　CLI 版の Security Key Management Tool を使用する場合

1. UFKP の生成
   ターミナルソフトで **genufpk** コマンドを実行します。

> skmt.exe /genufpk
/ufpk "ec6b8fa5c0d5da5142ccaf3a31aebeae2346cfe7ef644b9b6b70523cba0f5c5c"
/output "C:¥work¥ufpk.key"

```
C:¥work¥skmt¥tool>skmt.exe /genufpk /ufpk "ec6b8fa5c0d5da5142ccaf3a31aebeae2346cfe7
ef644b9b6b70523cba0f5c5c" /output "C:¥work¥ufpk.key"
UFPK: EC6B8FA5C0D5DA5142CCAF3A31AEBEAE2346CFE7EF644B9B6B70523CBA0F5C5C
Output File: C:¥work¥ufpk.key
```

図 6.17　**genufpk** コマンド実行結果

2. W-UFPK の取得
   手順 1 で生成した `ufpk.key` ファイルを Renesas Key Wrap service(https://dlm.renesas.com/keywrap)に送って、W-UFPK ファイルを受け取ります。
   詳細は、Renesas Key Wrap Service の FAQ もしくはアプリケーションノートを参照してください。

3. KUK の生成
   ターミナルソフトで **genkuk** コマンドを実行します。

> skmt.exe /genkuk /output "C:¥work¥kuk.key"
/kuk "d0aec19726cbc0e2fb403866b9b465a6c0d05b7a60362d5f435f9a3e98c79084"

```
C:¥work¥skmt¥tool>skmt.exe /genkuk /output "C:¥work¥kuk.key" /kuk "d0aec19726cbc0e2
fb403866b9b465a6c0d05b7a60362d5f435f9a3e98c79084"
KUK: D0AEC19726CBC0E2FB403866B9B465A6C0D05B7A60362D5F435F9A3E98C79084
Output File: C:¥work¥kuk.key
```

図 6.18　**genkuk** コマンド実行結果

4. RFP ファイルフォーマットの KUK ファイルの生成
   ターミナルソフトで **genkey** コマンドを実行します。

> skmt.exe /genkey /ufpk file="c:¥work¥ufpk.key" /wufpk file="C:¥work¥ufpk.key_enc.key"
/mcu "RA-SCE9" /keytype "key-update-key" /key file="C:¥work¥kuk.key" /filetype "rfp"
/output "C:¥work¥kuk.rkey"

手順 1 で生成した UFPK ファイル、手順 2 で受け取った W-UFPK ファイルを使用します。

```
C:¥work¥skmt¥tool>skmt.exe /genkey /ufpk file="c:¥work¥ufpk.key" /wufpk file="C:¥wo
rk¥ufpk.key_enc.key" /mcu "RA-SCE9" /keytype "key-update-key" /key file="C:¥work¥ku
k.key" /filetype "rfp" /output "C:¥work¥kuk.rkey"
Output File: C:¥work¥kuk.rkey
UFPK: EC6B8FA5C0D5DA5142CCAF3A31AEBEAE2346CFE7EF644B9B6B70523CBA0F5C5C
W-UFPK: 000000004F4C172DEF2789DE4F041294C6B1218D11DC81DC5B37FB72DB899DCFF4BE7158
IV: 288CCF50DBB7188B45C77BCED8A169EF
Encrypted key: EE047D0F1C492DA03E12F87464B982D527A4AE15E0CC34BC06177B0F68658FD199CF
66293561B87381C9F8817B2ED4FA
```

図 6.19　**genkey** コマンド実行例

6.3 Example 3 – RA ファミリ SCE9 Protected Mode RSA 2048 公開鍵のアップデート

鍵は、事前にインジェクションされた KUK を使用してフィールドでアップデートすることができます。必要なドライバは、表 1-1 MCU/MPU 関連サイトの各デバイスのサポートドライバをご確認ください。

市場での鍵のアップデートは、さまざまな方法で行うことができます。1 つの方法は、ファームウェア・アップデートの一部として KUK でラップされた鍵を含めることです。これを行う簡単な方法は、C ソースファイルとしてデータを埋め込むことです。 ここでは、前章で作成されインジェクションされた KUK を使って、RSA 2048 公開鍵のアップデート例を示します。

6.3.1 GUI 版の Security Key Management Tool を使用する場合

1. MCU/MPU と暗号エンジンの選択
   [**概要**]タブで MCU/MPU と暗号エンジンを選択します。

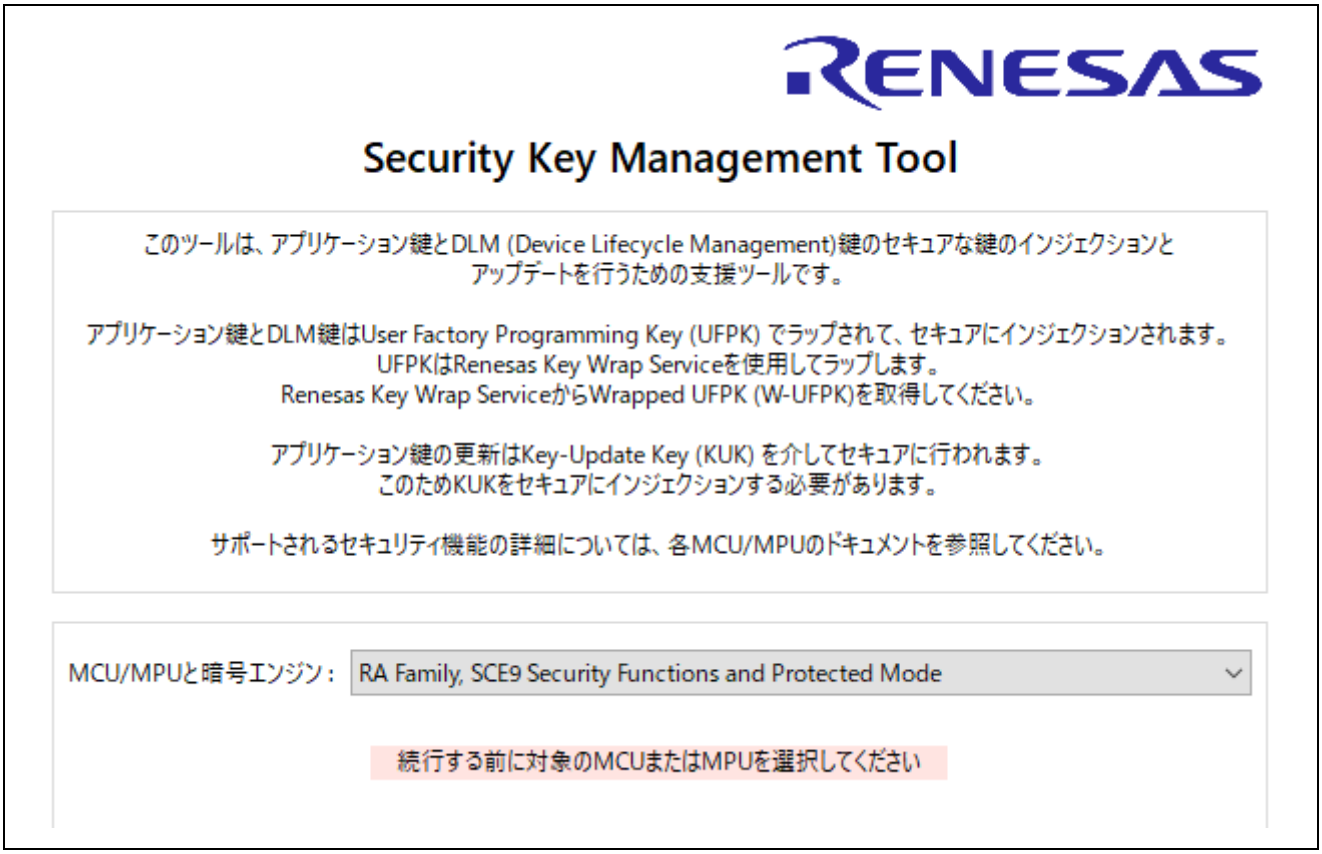

図 6.20 [**概要**]タブ

2. C ソースファイルの RSA 2048 公開鍵ファイルの生成
   [**鍵のラッピング**]タブを使用して RSA 2048 公開鍵ファイルを C ソースファイルとして生成します。[**鍵の種類**]タブで”RSA”と”2048 bits public”を選択し、[**鍵データ**] タブで RSA 2048 公開鍵データを入力します。
   “**鍵のラッピング**”に 6.2.Example 2 – RA ファミリ SCE9 Protected Mode Key-Update Key のインジェクション で作成した KUK ファイルを設定します。簡略化のため、この例では IV に”**乱数生成機能を使用する**”を選択します。”**出力**”のフォーマットに”**C ソース**”を選択し、拡張子*.c のファイル名を指定します。

図 6.21 **[鍵のラッピング] – [鍵の種類]**タブ RSA 2048 公開鍵 C ソースファイル生成例

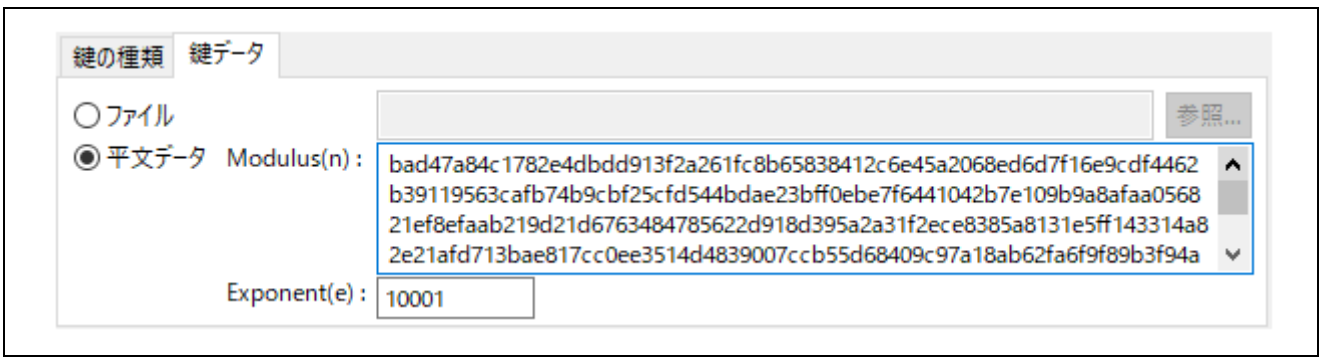

図 6.22 **[鍵のラッピング] – [鍵データ]**タブ RSA 2048 公開鍵 C ソースファイル生成例

正常終了すると以下のように出力されます。

図 6.23　[**鍵のラッピング**]タブ　実行結果

6.3.2　CLI 版の Security Key Management Tool を使用する場合

1. C ソースファイルの RSA 2048 公開鍵ファイルの生成
   ターミナルソフトで **genkey** コマンドを実行します。

> skmt.exe /genkey /kuk file="C:¥work¥kuk.key" /mcu "RA-SCE9" /keytype "RSA-2048-public"
/key "bad47a84c1782e4dbdd913f2a261fc8b65838412c6e45a2068ed6d7f16e9cdf4
462b39119563cafb74b9cbf25cfd544bdae23bff0ebe7f6441042b7e109b9a8a
faa056821ef8efaab219d21d6763484785622d918d395a2a31f2ece8385a8131
e5ff143314a82e21afd713bae817cc0ee3514d4839007ccb55d68409c97a18ab
62fa6f9f89b3f94a2777c47d6136775a56a9a0127f682470bef831fbec4bcd7b
5095a7823fd70745d37d1bf72b63c4b1b4a3d0581e74bf9ade93cc4614861755
3931a79d92e9e488ef47223ee6f6c061884b13c9065b591139de13c1ea292749
1ed00fb793cd68f463f5f64baa53916b46c818ab99706557a1c2d50d232577d1
00010001"
/filetype "csource" /output "C:¥work¥rsa2048public.c"
/keyname "rsa2048public"

KUK ファイルは 6.2.Example 2 – RA ファミリ SCE9 Protected Mode Key-Update Key のインジェクションで生成したファイルを使用します。

```
C:¥work¥skmt¥tool>skmt.exe /genkey /kuk file="C:¥work¥kuk.key" /mcu "RA-SCE9" /keyt
ype "RSA-2048-public" /key "bad47a84c1782e4dbdd913f2a261fc8b65838412c6e45a2068ed6d7
f16e9cdf4462b39119563cafb74b9cbf25cfd544bdae23bff0ebe7f6441042b7e109b9a8afaa056821e
f8efaab219d21d6763484785622d918d395a2a31f2ece8385a8131e5ff143314a82e21afd713bae817c
c0ee3514d4839007ccb55d68409c97a18ab62fa6f9f89b3f94a2777c47d6136775a56a9a0127f682470
bef831fbec4bcd7b5095a7823fd70745d37d1bf72b63c4b1b4a3d0581e74bf9ade93cc4614861755393
1a79d92e9e488ef47223ee6f6c061884b13c9065b591139de13c1ea2927491ed00fb793cd68f463f5f6
4baa53916b46c818ab99706557a1c2d50d232577d100010001" /filetype "csource" /output "C:
¥work¥rsa2048public.c" /keyname "rsa2048public"
Output File: C:¥work¥rsa2048public.h
Output File: C:¥work¥rsa2048public.c
KUK: D0AEC19726CBC0E2FB403866B9B465A6C0D05B7A60362D5F435F9A3E98C79084
IV: DEB4BB141E1BDFA11EB1B3E7195E21F1
Encrypted key: EA0150EC3B9F421AD6C4F54D64625DC8B45FAF44D1339DB1A562E673806B3953E152
67D49CD9AE8CAEB2AC12683DE831C58D538C4F1BF71994A62AC14E36E99E39364F7C0AB6B4A61661486
0FC05702233A37700D3F027F6BCE51B070B98061FB30F42ADCE6F79178F46E60B1EB401C9AEA4052048
AF94272ADB90691279D74425BA9258D540167150F2E55894B4915F2C77A3AF6CAFADF06035260C6CA79
08DAAB2E927551677047AEAF27DBA32B28E916B56397748A733B6DF1661A99399EE016CD2E10114A73C
93B57D9C8298C1E1A72E6A203C958B23ECFFB94C8E5E5606E7E1FA6273461533ACD61B632E2024F2EF3
FBC14B76DD791C0999BD48218B2CC4B87B0718CC18D4EFAF864212F7E5E04A8DBC030F31BE8243E7B96
9A4E0C5ADC
```

図 6.24　**genkey** コマンド実行例

6.4 Example 4 – RX ファミリ TSIP Secure Update 時の使用方法

TSIP を使用したセキュアなファームウェアアップデートソリューションで、ユーザプログラムを暗号化して、ターゲットデバイスに送信し、デバイス内でユーザプログラムを復号して更新することが可能です。必要なドライバならびにサンプルを使用した活用方法は、表 1-1 MCU/MPU 関連サイトの RX TSIP の資料をご確認ください。

6.4.1 GUI 版の Security Key Management Tool を使用する場合

1. MCU/MPU と暗号エンジンの選択
   [**概要**]タブで MCU/MPU と暗号エンジンを選択します。

図 6.25 [**概要**]タブ

2. ファームウェアイメージファイルの生成
   [TSIP Update]タブを使用して RSU ヘッダを付けたファームウェアイメージを生成します。
   “出力イメージ”で**”Secure Update”**を選択し、**”ファームウェアイメージ”**に暗号化する
   ファームウェアの mot ファイルを設定します。
   [暗号化アドレス範囲]タブに暗号化するアドレス範囲を設定します。
   [Image Encryption Key]タブ“**Session Key Encryption Key**”に、Secure Boot プログラムに設定している暗号化する鍵を暗号化する鍵を設定します。簡略化のため、この例では**”Image Encryption Session Key**”に”**乱数生成機能を使用する**”を選択します。
   [IV]タブでは、簡略化のため、この例では”**乱数生成機能を使用する**”を選択します。
   [RSU ヘッダ]タブでは**”イメージフラグ”**に TESTING、**”RSU ヘッダ Ver”**に 1 を設定します。
   ”**出力**”のフォーマットに”**バイナリ**”を選択し、拡張子`*.rsu`のファイル名を指定します。

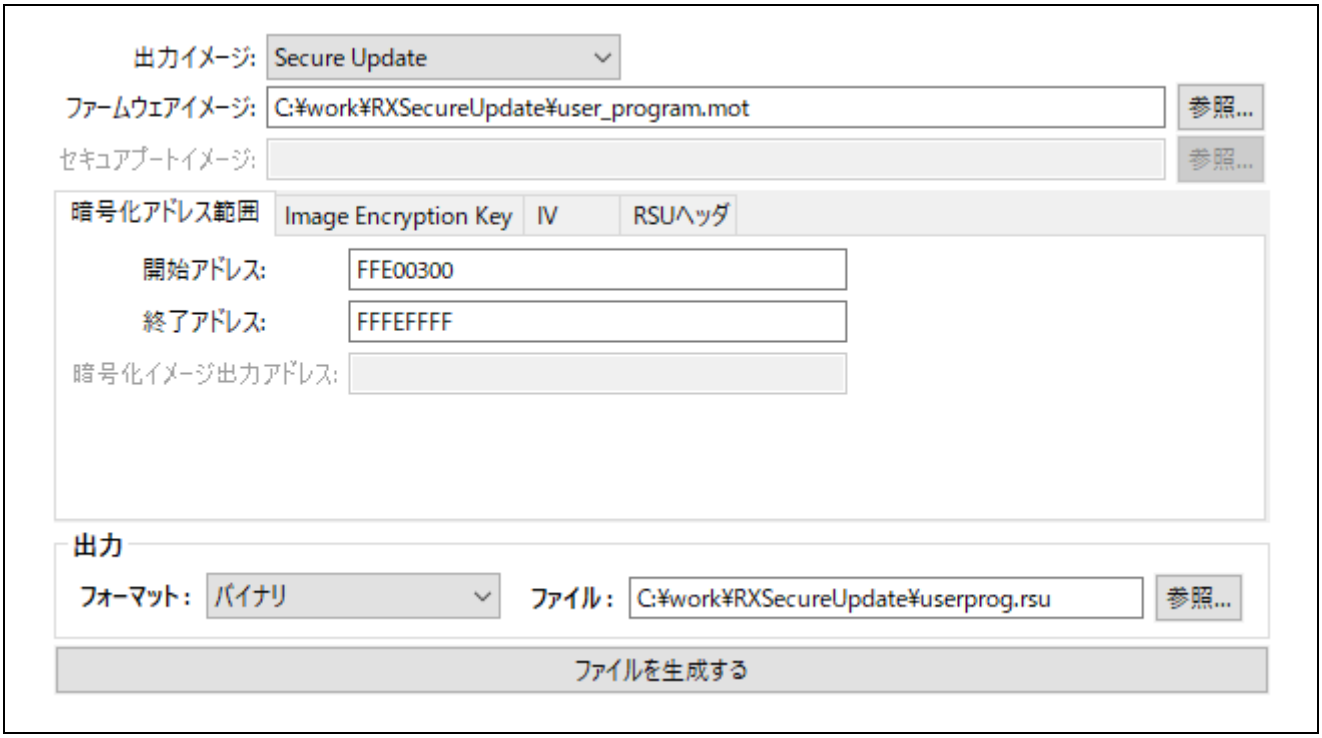

図 6.26　**[TSIP Update] – [暗号化アドレス範囲]**タブ 他の設定例

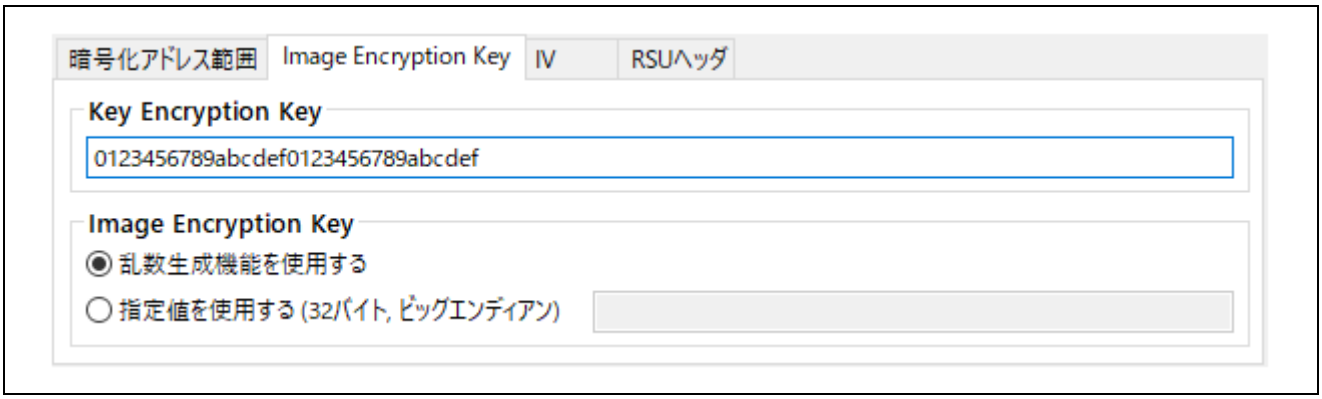

図 6.27　**[TSIP Update] – [Image Encryption Key]**タブ 設定例

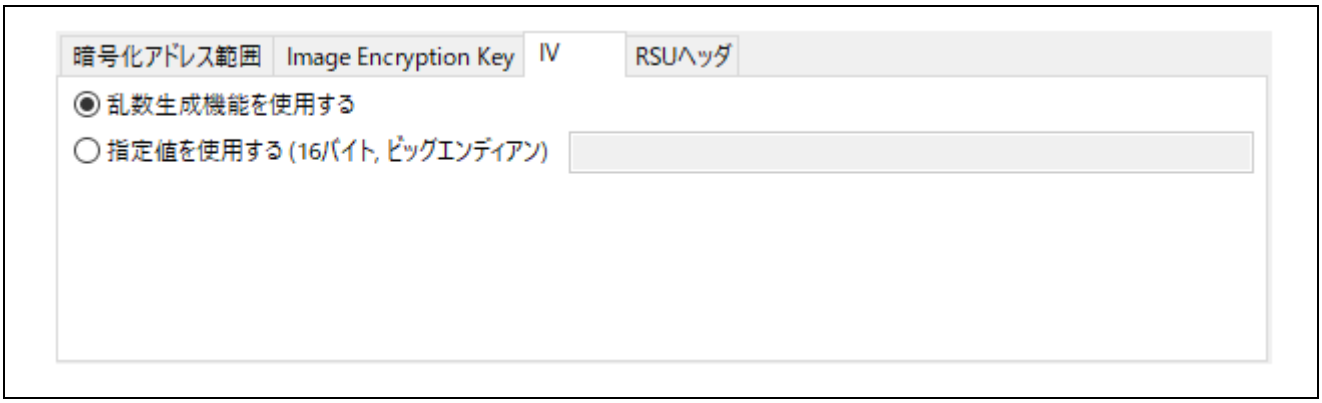

図 6.28 **[TSIP Update] – [IV]タブ 設定例**

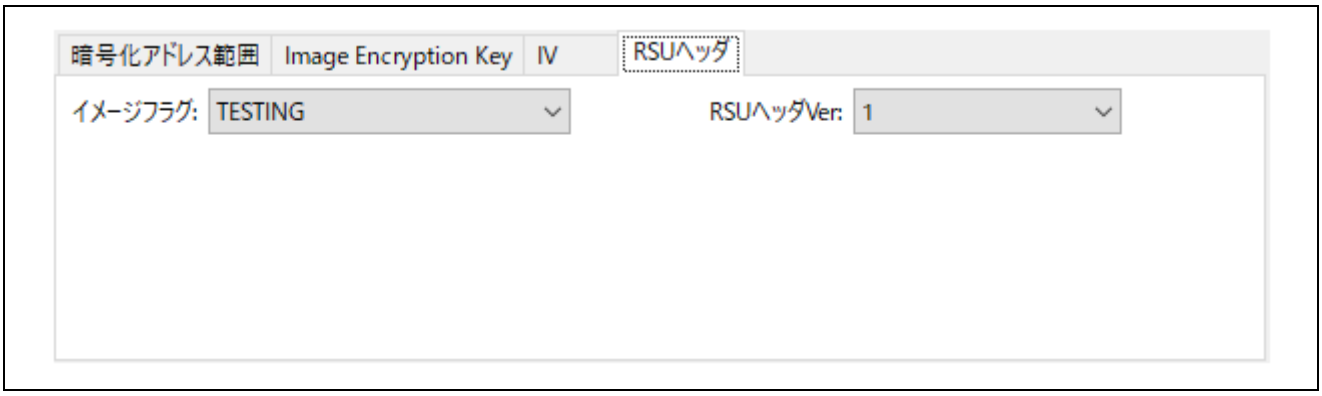

図 6.29 **[TSIP Update] – [RSU ヘッダ]タブ 設定例**

正常終了すると以下のように出力されます。

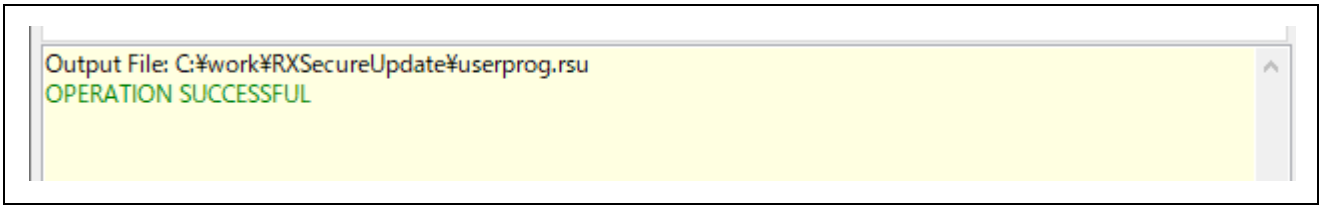

図 6.30 **[TSIP Update]タブ 実行結果**

6.4.2　CLI 版の Security Key Management Tool を使用する場合

1. ターミナルソフトで **enctsip** コマンドを実行します。

> skmt.exe /enctsip /mode "update" /ver "1" /prg "C:¥work¥RXSecureUpdate¥user_program.mot"
/enckey "0123456789abcdef0123456789abcdef"
/startaddr "FFE00300" /endaddr "FFFEFFFF" /imgflg "testing" /filetype "bin"
/output "C:¥work¥RXSecureUpdate¥userprog.rsu"

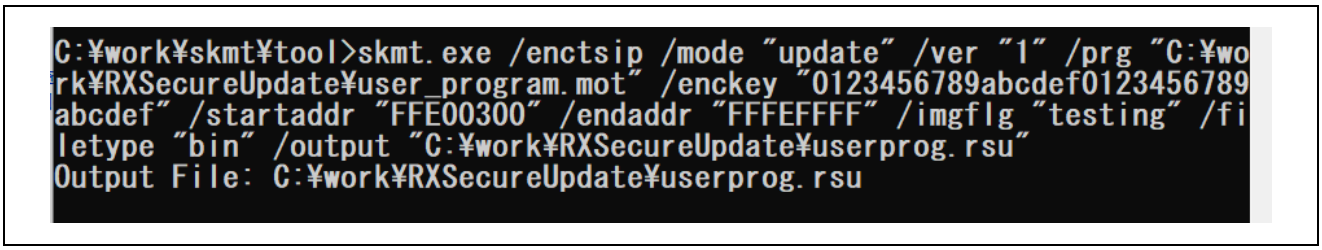

図 6.31　**enctsip** コマンド実行例

6.5 Example 5 – RA ファミリ FSBL 鍵証明書 / コード証明書の生成時の使用方法

First Stage Bootloader(FSBL)機能を搭載したルネサスデバイスで使用可能な鍵証明書とコード証明書の生成を行います。鍵証明書とコード証明書は OEM_BL を Renesas Flash Programmer を使用してデバイスに書き込む際に、鍵証明書とコード証明書の正当性検証で使用します。

必要なツールやサンプルを使用した活用方法は、表 1-1 MCU/MPU 関連サイトの資料をご確認ください。

6.5.1 GUI 版の Security Key Management Tool を使用する場合

1. MCU/MPU と暗号エンジンの選択
   [**概要**]タブで MCU/MPU と暗号エンジンを選択します。

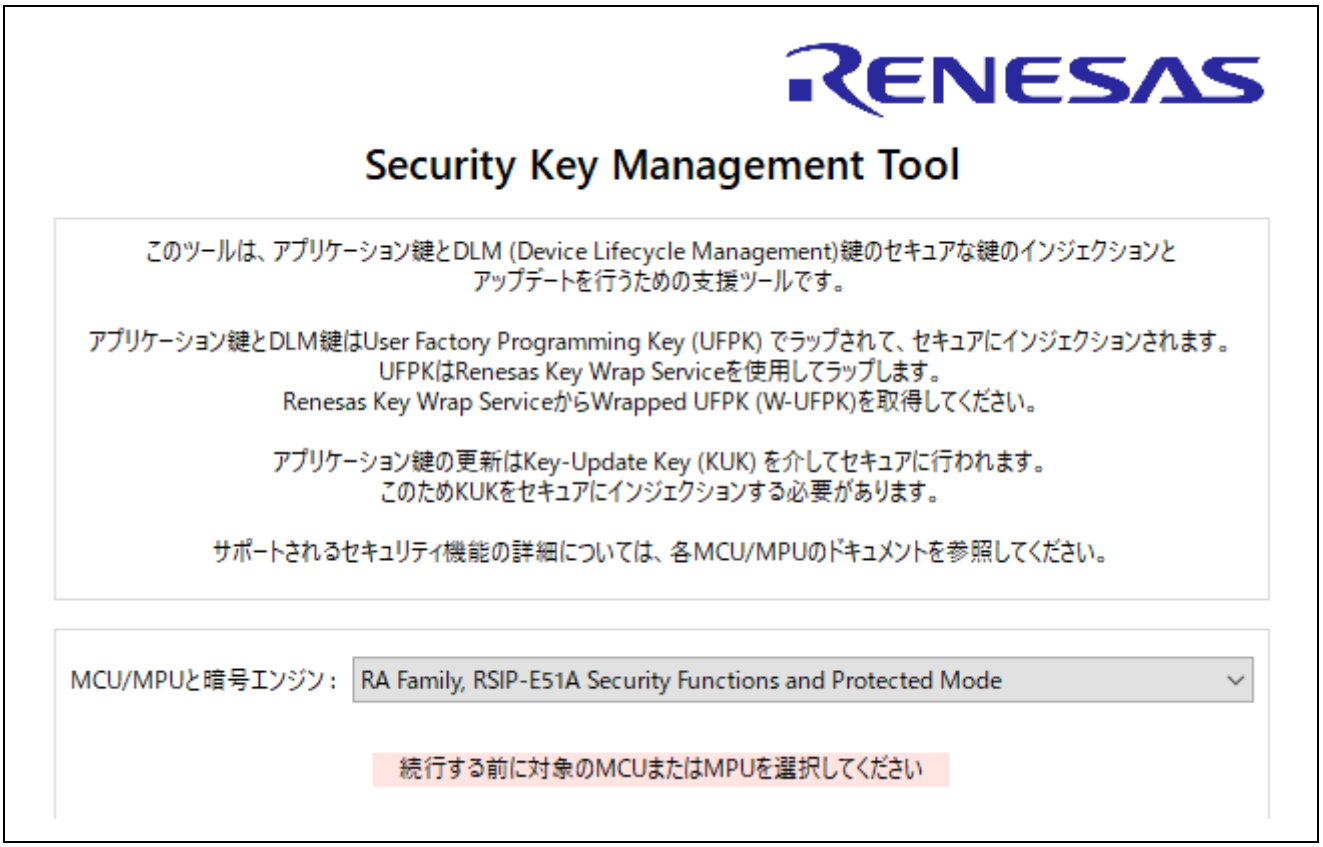

図 6.32 [**概要**]タブ

2. 鍵証明書とコード証明書の生成

[**FSBL**]タブを使用して鍵証明書とコード証明書を生成します。
この例ではプログラム検証方法で“**Signature**”の場合の手順を説明します。

“**OEM Bootloader イメージ**”で署名対象となる OEM_BL の mot ファイルを指定します。
“**プログラム検証方法**”で”**Signature**”を選択します。
“**イメージバージョン**”に **1** を設定します。

[**証明書**]タブの設定を行います。
“**コードフラッシュ開始アドレス(hex)**”に OEM Bootloader の開始アドレスを設定します。この例では 02000000 を設定します。
“**デバイスコードフラッシュ**”で使用するデバイスのコードフラッシュのサイズを選択します。この例では“**2MB**”を選択します。
“**OEM Bootloader サイズ**”で“**自動計算**”を選択します。
“**鍵証明書**”で鍵証明書のファイル名を指定します。
“**コード証明書**”でコード証明書のファイル名を指定します。

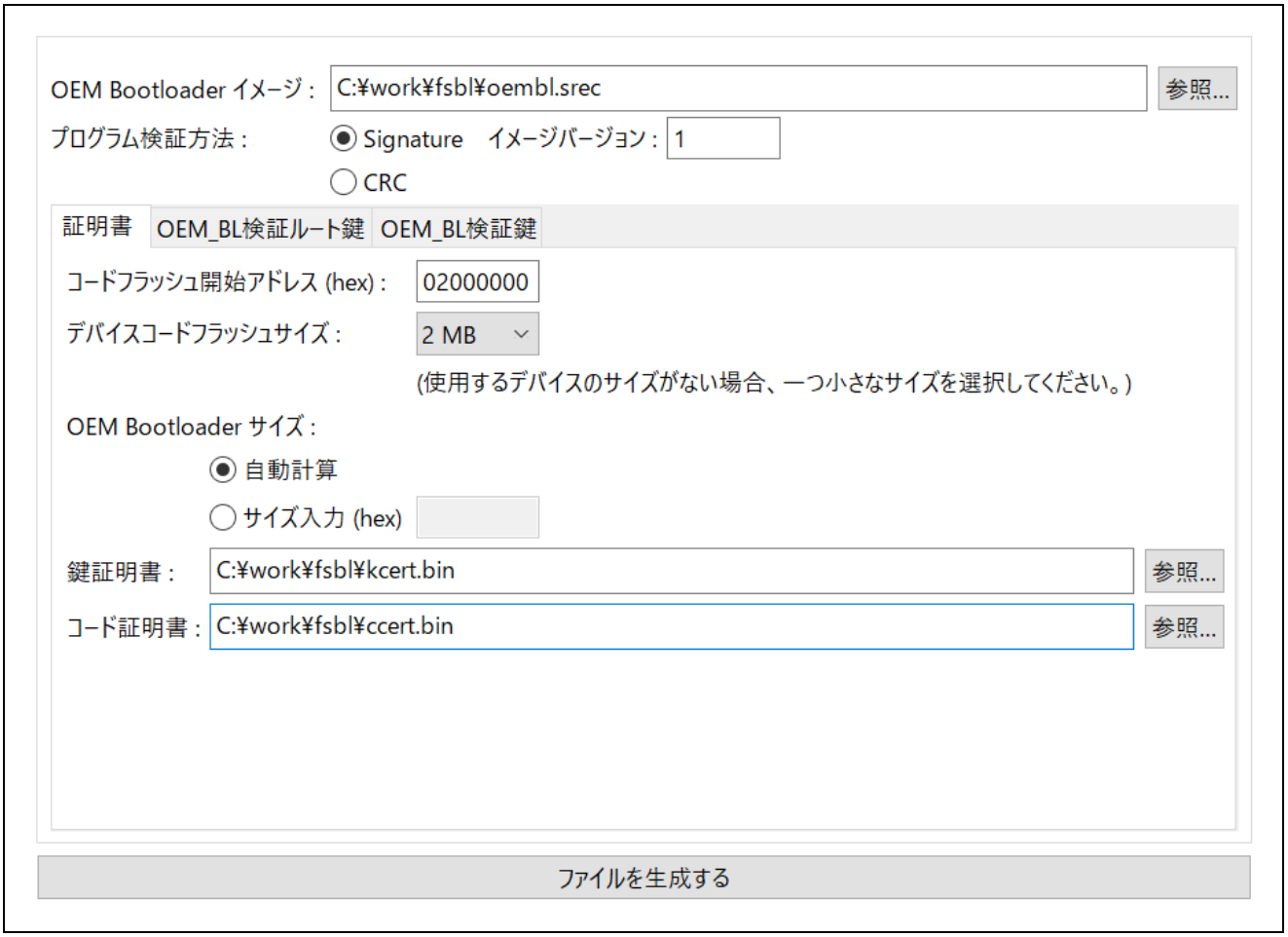

図 6.33　[**FSBL**] – [**証明書**]タブ 他の設定例

[**OEM_BL 検証ルート鍵**]タブの設定を行います。
“**OEM_BL 検証ルート秘密鍵**”で“**ファイル**”を選択し OEM_BL 検証ルート鍵の公開鍵を含んだ PEM ファイルを指定します。“**OEM_BL 検証ルート秘密鍵**”で PEM ファイルを指定したため、“**OEM_BL 検証ルート公開鍵**”の指定は不要です。

図 6.34　[**FSBL**] – [**OEM_BL 検証ルート鍵**]タブ 設定例

[**OEM_BL 検証鍵**]タブの設定を行います。
“**OEM_BL 検証秘密鍵**”で“**ファイル**”を選択し OEM_BL 検証鍵の公開鍵を含んだ PEM ファイルを指定します。“**OEM_BL 検証秘密鍵**”で PEM ファイルを指定したため、“**OEM_BL 検証公開鍵**”の指定は不要です。

図 6.35　[**FSBL**] – [**OEM_BL 検証鍵**]タブ 設定例

“**ファイルを生成する**”ボタンを押すと鍵証明書とコード証明書が生成されます。正常にファイルが生成された場合、以下のような実行結果が出力されます。

図 6.36 [**FSBL**]タブ 実行結果

6.5.2 CLI 版の Security Key Management Tool を使用する場合

1. ターミナルソフトで **gencert** コマンドを実行します。

> skmt.exe /gencert /mode "signature" /loadaddr "02000000" /cfsize "200000" /ver "1"
/oembl "userprog.mot" /oembl_private file="is_ec256_priv.pem"
/oemroot_private file="ms_ec256_priv.pem"
/output_codecert "ccert.bin" /output_keycert "kcert.bin"

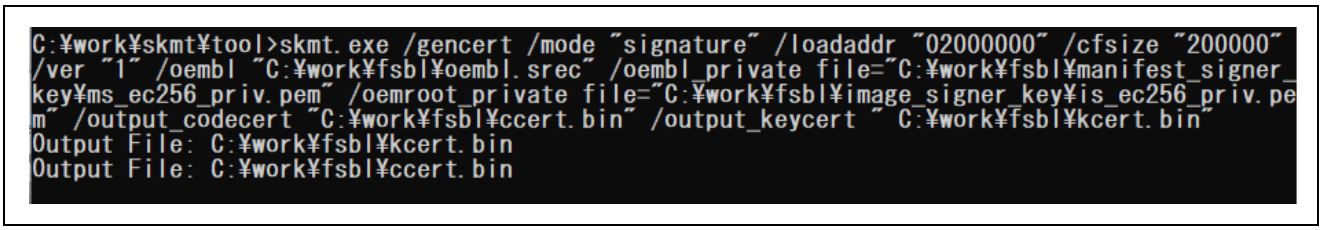

図 6.37 **gencert** コマンド実行例

6.6 Example 6 – RA ファミリ DOTF 使用時のプログラム暗号化方法

DOTF機能を搭載したルネサスデバイスで使用可能なユーザアプリケーションの暗号化を行います。
必要なドライバやサンプルを使用した活用方法は、表 1-1 MCU/MPU 関連サイトの資料をご確認ください。

6.6.1 GUI 版の Security Key Management Tool を使用する場合

1. MCU/MPU と暗号エンジンの選択
   [**概要**]タブで MCU/MPU と暗号エンジンを選択します。

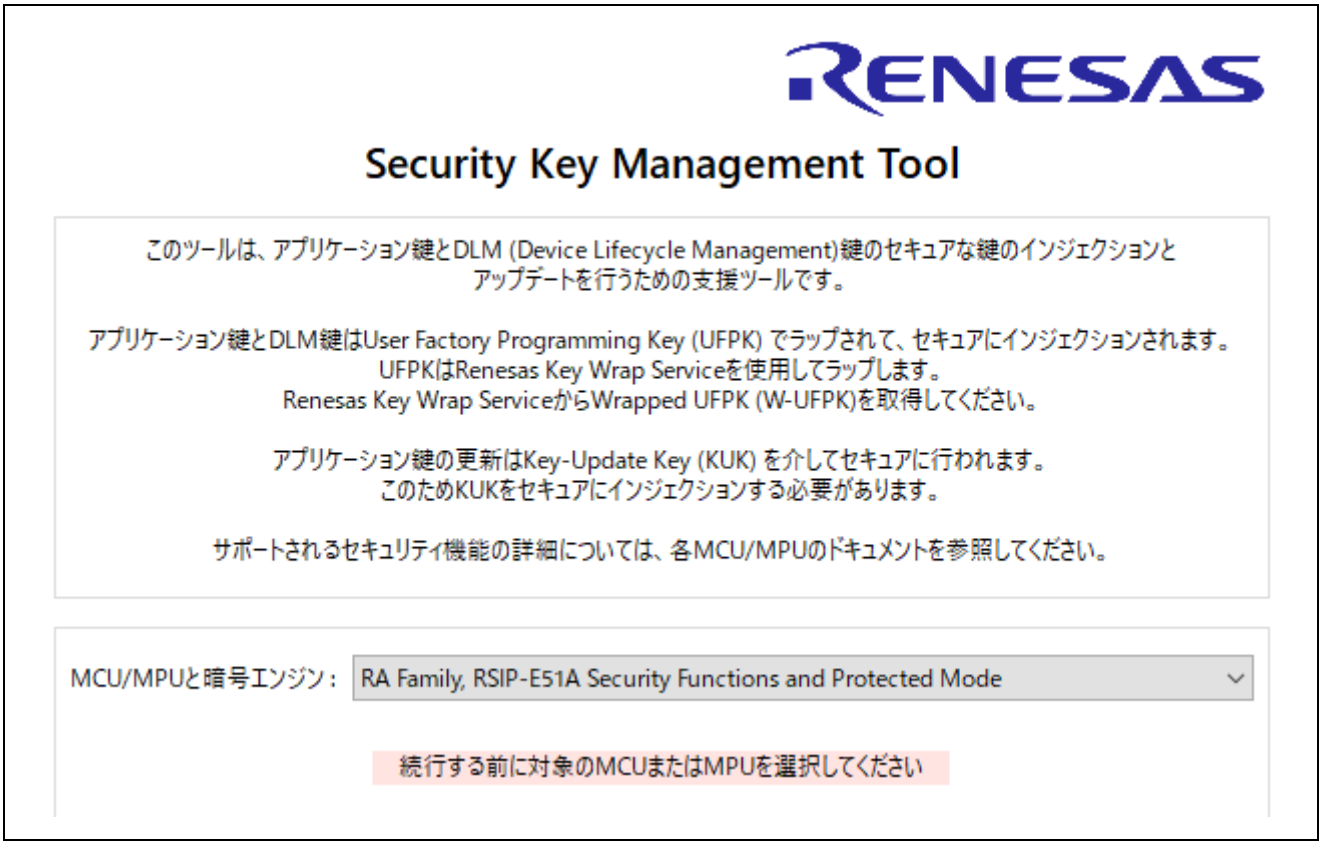

図 6.38 [**概要**]タブ

2. ファームウェアイメージの設定
   [**DOTF/OTFD**]タブを使用して、ファームウェアイメージの暗号化を行います。
   “**平文イメージ**”で暗号化するファームウェアの mot ファイルを設定します。
   “**暗号化するイメージのアドレス範囲**”に平文イメージで入力された暗号化範囲を指定します。
   “**転送先アドレス**”は暗号化するアドレスと、配置されるアドレスが異なる場合に設定をします。
   この例では、暗号化範囲は 90000000 から 90000FFF、”**転送先アドレス**”は”**平文イメージと同じ**”を設定します。

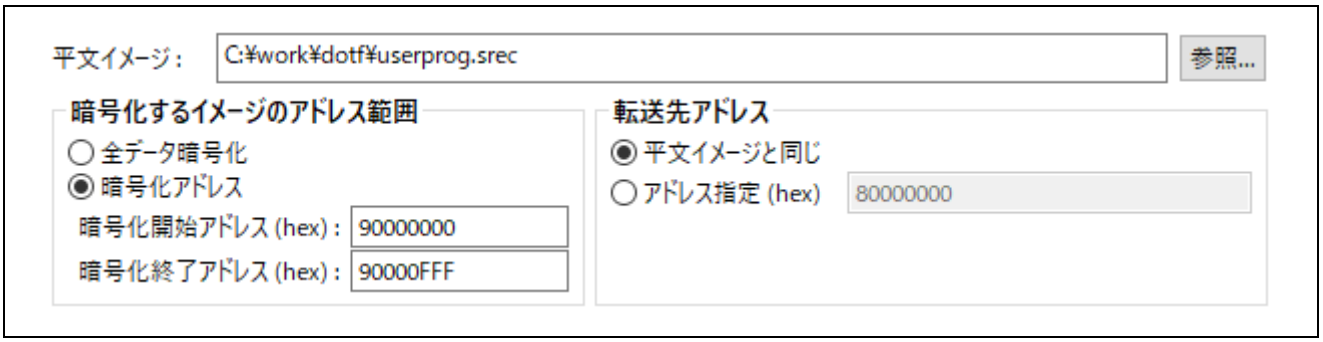

図 6.39　[**DOTF/OTFD**]**タブ** – Plaintext Image、暗号化範囲、実行アドレスの設定例

3. 暗号化鍵と IV の設定
   “**イメージ暗号化鍵**”に暗号化で使用する鍵を設定します。この例では AES128 の鍵を使用します。
   “**IV**”に平文イメージの暗号化で使用する nonce を設定します。この例では簡略化のため、IV は”**乱数生成機能を使用する**”を選択します。

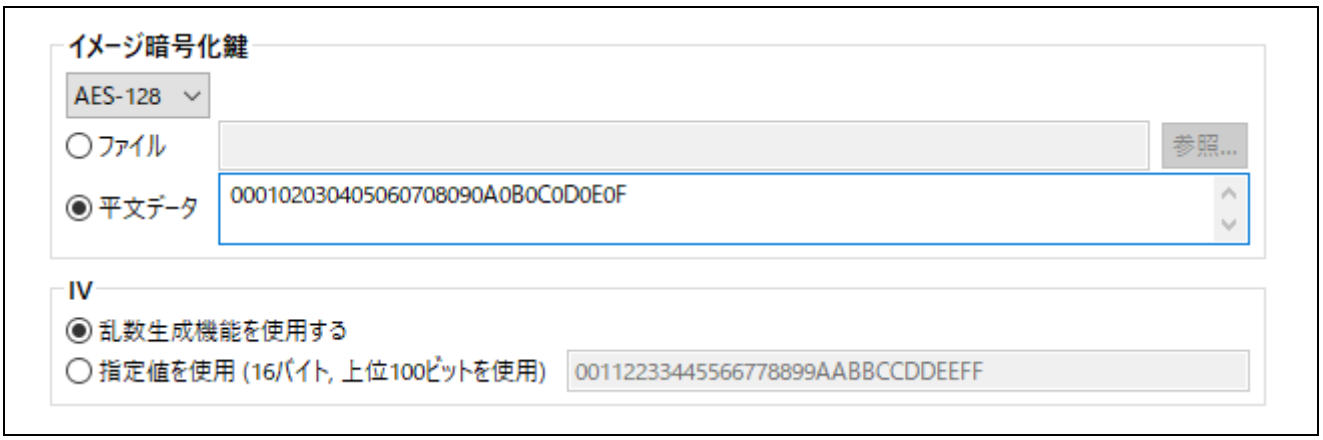

図 6.40　[**DOTF/OTFD**]**タブ** – Image Encryption Key、IV 設定例

4. 出力ファイルの設定
   “**暗号化イメージ**”に出力するファイル名を指定します。
   “**出力イメージのアドレスとコンテンツ**”で、出力するモトローラヘキサのアドレスと、暗号化しなかった平文データを含めるかを指定します。
   “**入力ファイルのアドレスを使用**”にチェックを入れ、”**暗号化したデータ以外の領域に、平文データを含める**”にチェックを入れます。

図 6.41 [**DOTF/OTFD**]タブ – 暗号化イメージ 出力イメージのアドレスとコンテンツ 設定例

正常終了すると以下のように出力されます。

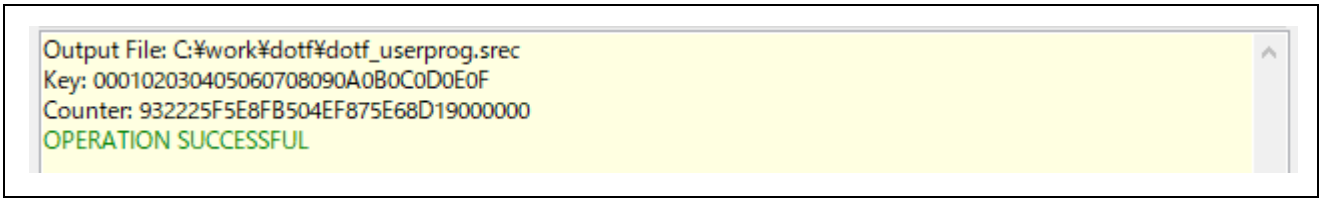

図 6.42 [**DOTF/OTFD**]タブ 実行結果

6.6.2 CLI 版の Security Key Management Tool を使用する場合

1. ターミナルソフトで **encdotf** コマンドを実行します。
   > skmt.exe /encdotf /keytype "AES-128" /enckey "000102030405060708090A0B0C0D0E0F" /startaddr "90000000" /endaddr "90000FFF" /incplain /prg "C:\work\dotf\userprog.srec" /output "C:\work\dotf\dotf_userprog.srec"

```
C:¥work¥skmt¥tool>skmt.exe /encdotf /keytype "AES-128" /enckey "000102030405060708090A0B0
C0D0E0F" /startaddr "90000000" /endaddr "90000FFF" /incplain /prg "C:¥work¥dotf¥userprog.
srec" /output "C:¥work¥dotf¥dotf_userprog.srec"
Output File: C:¥work¥dotf¥dotf_userprog.srec
Key: 000102030405060708090A0B0C0D0E0F
Counter: 81E4EE532AA836BBBB247754E9000000
```

図 6.43 **encdotf** コマンド実行例

6.7　Example 7 – RA ファミリ Secure Factory Programming 機能使用時のプログラム暗号化方法

Secure Factory Programming 機能で使用するための Secure Factory Programming ファイル(`*.sfp`)を出力します。

注：Secure Factory Programming ファイルには以下表の機能を含めることができません。これらの機能は、別な方法でデバイスに入力する必要があります。

表 6-1 **Secure Production Programming に追加が必要なファイル**

項目	必要な場合	入力方法
ユーザ鍵の注入	アプリケーションがセキュアに注入された鍵を必要とする場合	GUI の [**鍵のラッピング**]タブ（3.6 [鍵のラッピング]タブ）または **genkey** CLI コマンド（4.5 **genkeyコマンド** オプション）を使用して、必要な`*.rkey`ファイルをすべて生成する。鍵の注入に使用するUFPKは、Secure Factory Programmingに使用するUFPKと同じである必要はありません。
鍵証明書	署名認証付きFSBLを使用する場合	GUIの[**FSBL**]タブ（3.8 [**FSBL**]タブ）または**gencert** CLIコマンド（4.7 **gencertコマンド** オプション）を使用して、必要なバイナリ鍵証明書ファイルを生成する。
コード証明書	署名認証付きもしくはCRC検証によるFSBLを使用する場合	GUIの[**FSBL**]タブ（3.8 [**FSBL**]タブ）または**gencert** CLIコマンド（4.7 **gencertコマンド** オプション）を使用して、必要なバイナリコード証明書ファイルを生成する。

6.7.1 GUI 版の Security Key Management Tool を使用する場合

1. MCU/MPU と暗号エンジンの選択
   [**概要**]タブで MCU/MPU と暗号エンジンを選択します。

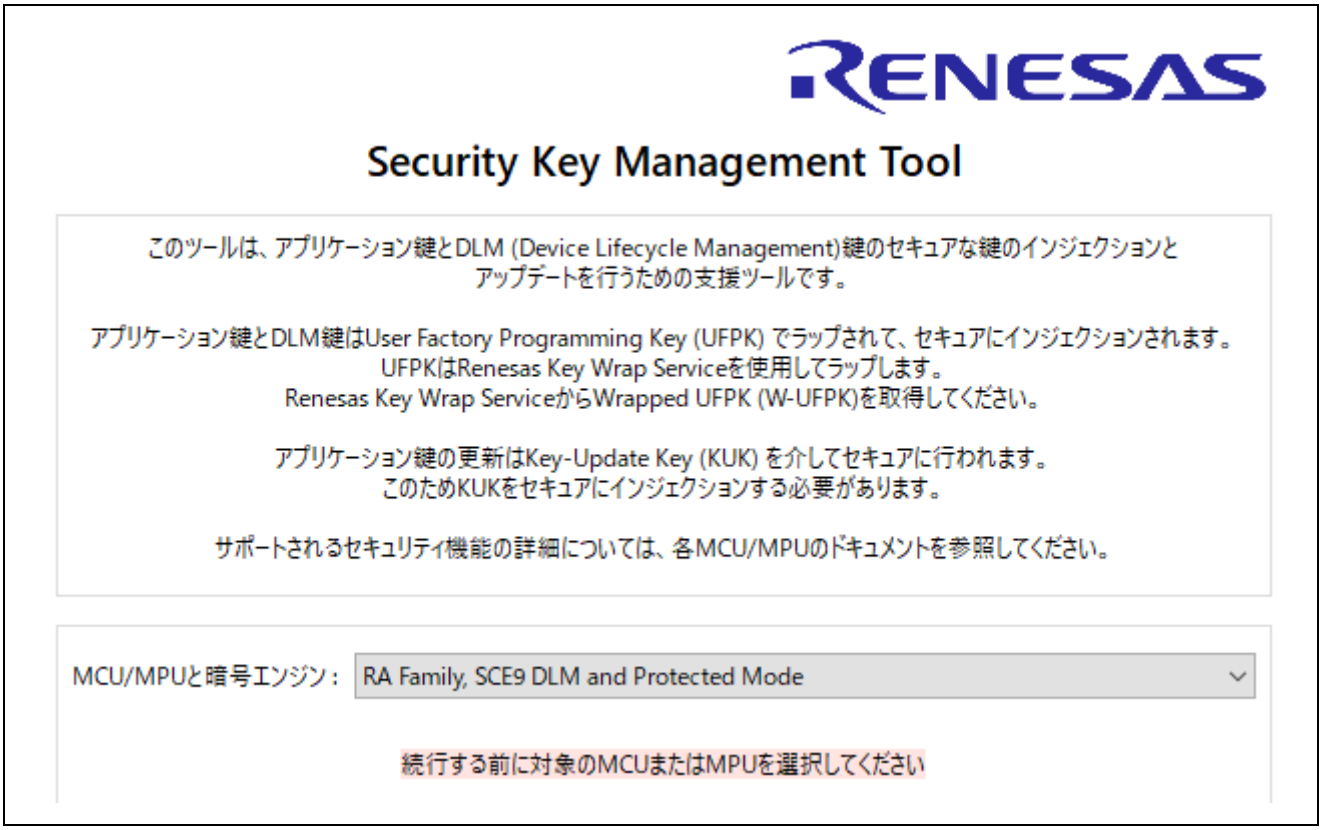

図 6.44 [**概要**]タブ

2. ファームウェアイメージファイルの生成
   [**SFP**]タブを使用して Secure Factory Programming ファイル(`*.sfp`)を生成します。
   [**ファームウェアイメージ**]タブ“**平文イメージ**”で暗号化するユーザプログラムを指定します。ファイルを選択して追加ボタンで追加します。

   “**最終 DLM/AL ステート**”でファームウェアイメージ書き込み後の遷移先 DLM/AL ステートを選択します。この例では“**OEM PL0 with AL2_KEY**”を選択します。
   “**セキュアプログラミングファイル**”で出力ファイル名を指定します。

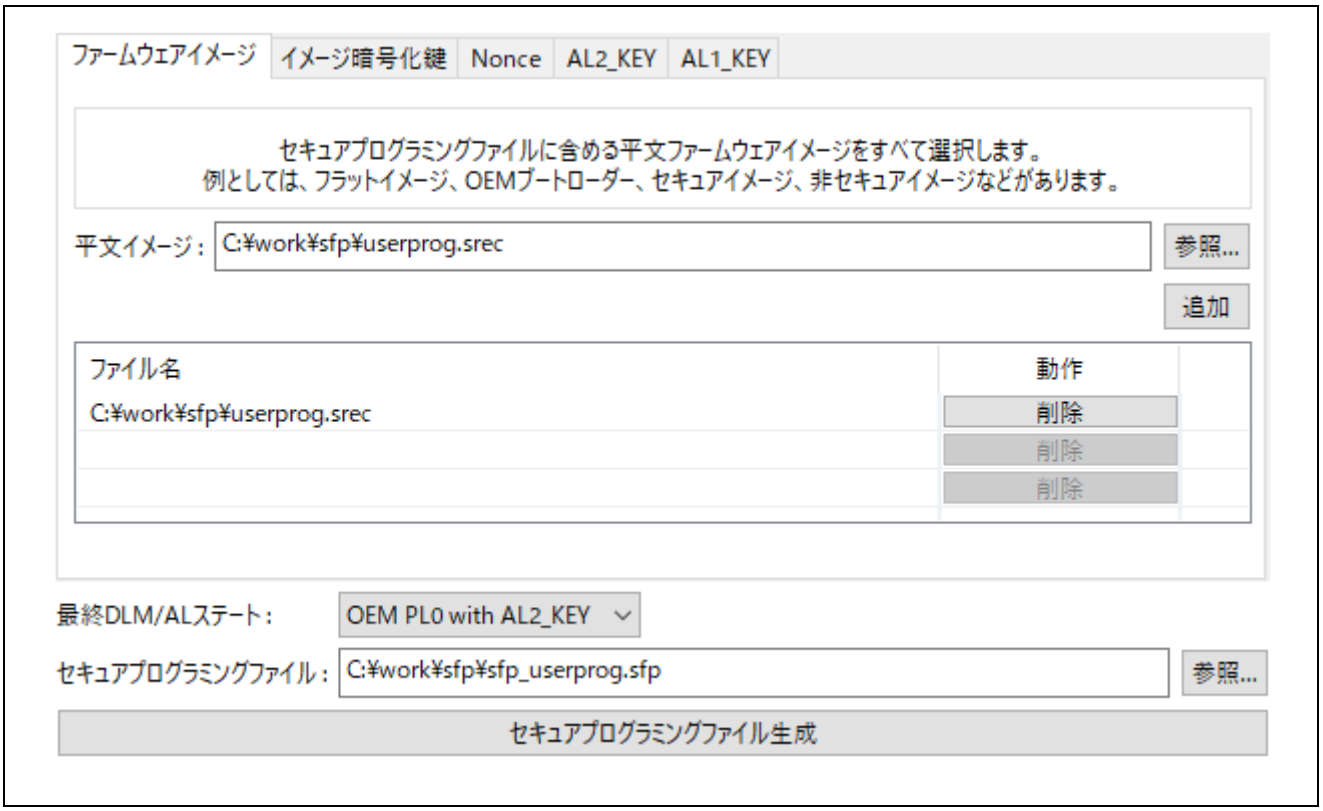

図 6.45　[**SFP**] – [**ファームウェアイメージ**]タブ 他の設定例

3. [**イメージ暗号化鍵**]タブの設定を行います。
“**鍵データ(AES-128)**”でユーザプログラムおよびパラメータ情報の暗号化に使用する AES128bit 鍵データを指定します。
“**IV**”は簡略化のため、この例では”**乱数生成機能を使用する**”を選択します。
“**UFPK ファイル**”に UFPK ファイル、“**W-UFPK ファイル**”に W-UFPK ファイルを指定します。
UFPK ファイルおよび W-UFPK ファイルの生成方法は 6.1Example 1 – RX ファミリ TSIP の AES 128 の鍵のインジェクション手順の手順 1~3 を参照してください。

図 6.46 [**SFP**] – [**イメージ暗号化鍵**]タブ 設定例

4. [**Nonce**]タブの設定を行います。
“**IV (プログラミングパラメータ)**”および“**IV (ファームウェアイメージ)**”は簡略化のため、この例では“**乱数生成機能を使用する**”を選択します。

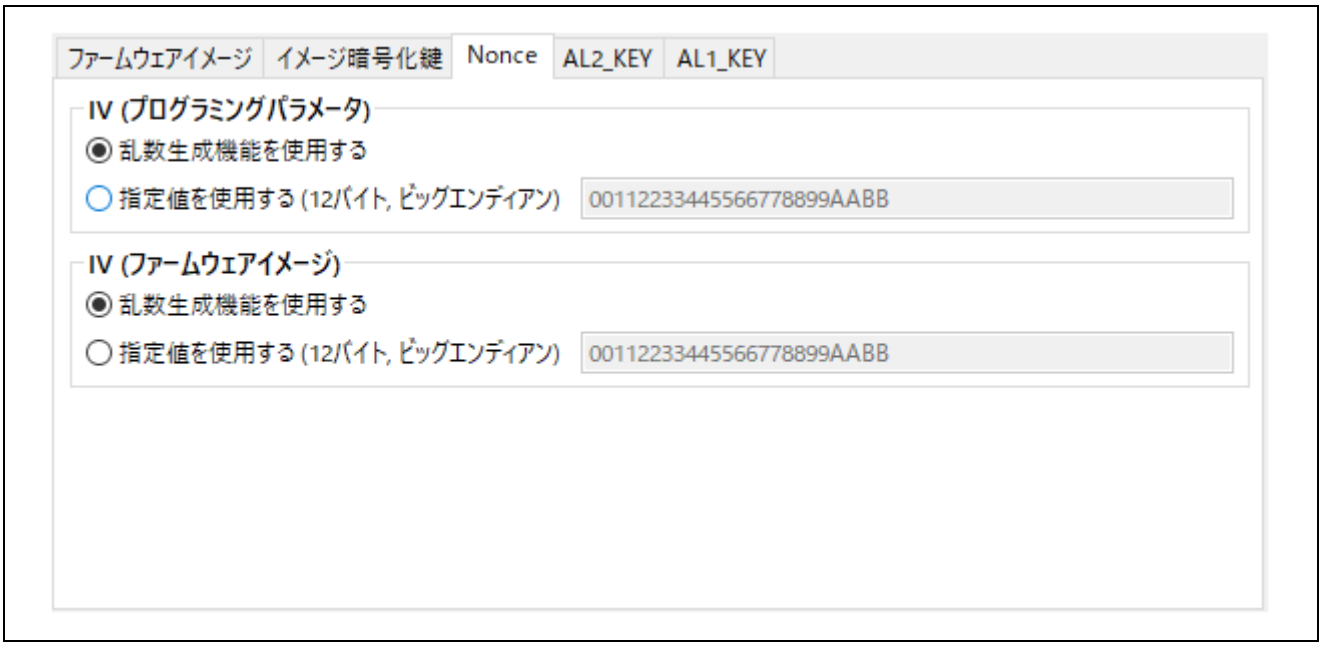

図 6.47　[**SFP**] – [**Nonce**]タブ 設定例

5. [**AL2_KEY**]タブの設定を行います。
“**鍵データ**”で AL2_KEY として使用する AES128bit 鍵データを指定します。
“**IV**”は簡略化のため、この例では”**乱数生成機能を使用する**”を選択します。

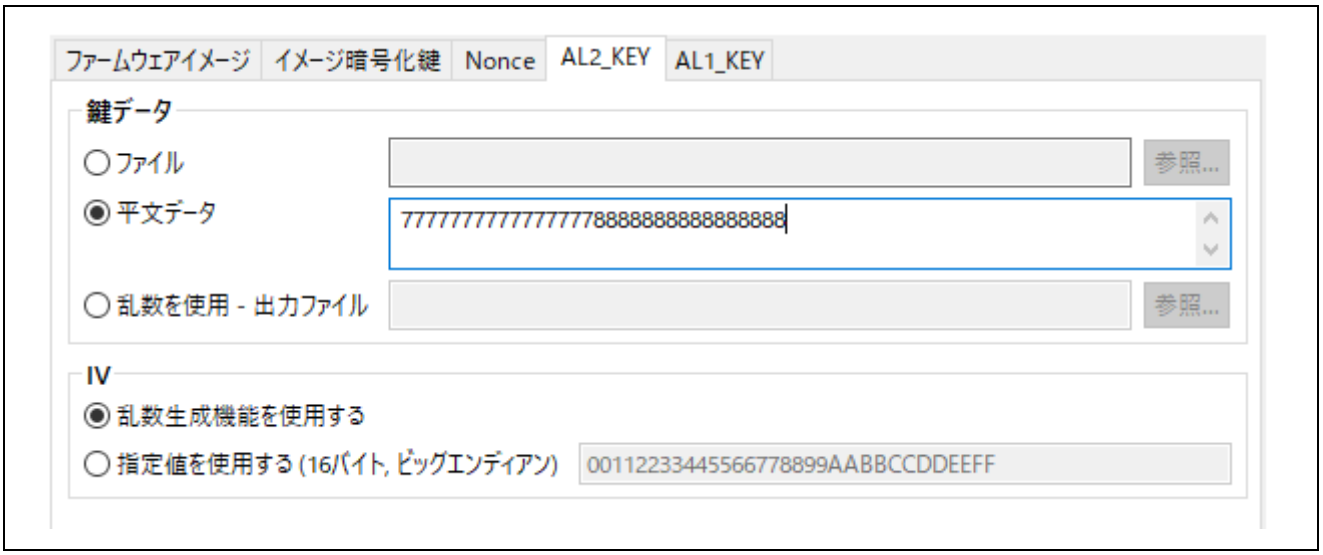

図 6.48　[**SFP**] – [**AL2_KEY**]タブ 設定例

“**セキュアプログラミングファイル生成**”ボタンを押すと Secure Factory Programming ファイルが生成されます。正常にファイルが生成された場合、以下のような実行結果が出力されます。

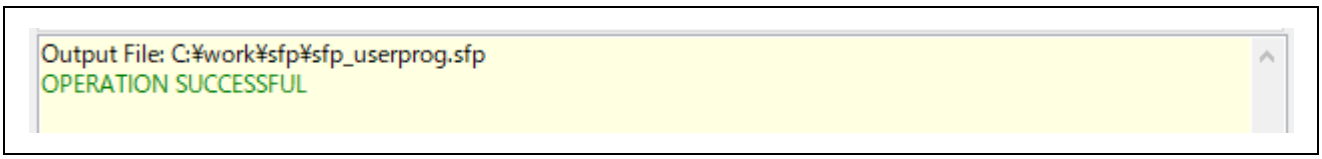

図 6.49　[**SFP**]タブ 実行結果

6.7.2　CLI 版の Security Key Management Tool を使用する場合

1. ターミナルソフトで **encsfp** コマンドを実行します。

>skmt /encsfp /mcu "RA8x1" /enckey "000102030405060708090a0b0c0d0e0f"
/trn "OEM_PL0_AL2" /prg "C:¥work¥sfp¥userprog.srec"
/al2key "77777777777777778888888888888888"
/ufpk file="C:¥work¥sfp¥ufpk.key" /wufpk "C:¥work¥sfp¥ufpk.key_enc.key"
/output "C:¥work¥sfp¥sfp_userprog.sfp"

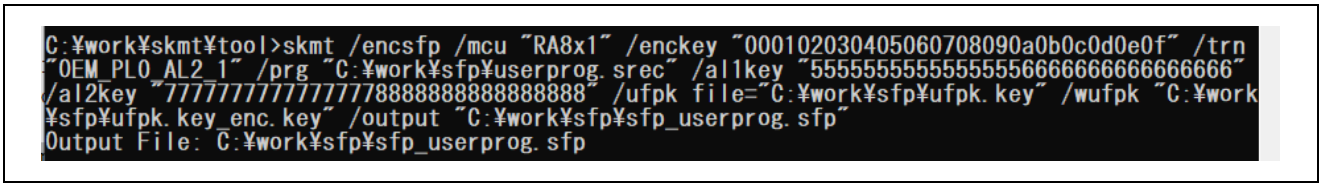

図 6.50　**encsfp** コマンド実行例

7.　注意事項

7.1　Windows 環境使用時のディスプレイ設定

Windows 環境で、スタンドアロン版もしくは e²studio プラグイン版の GUI で、ディスプレイ設定を推奨の設定以外の設定にした場合に、ダイアログ全体が表示できない場合があります。

ただし、以下の方法で表示が改善される場合があります。

1. 実行ファイルを右クリックします。
   スタンドアロン版の場合、SecurityKeyManagementTool.exe
   e²studio プラグイン版の場合、e2studio.exe
   * e2studio.exe は、e²studio インストールしたフォルダ内 eclipse フォルダにあります。
2. プロパティ→互換性タブ
   「高 DPI 設定の変更」ボタンを押してください。
3. 「高 DPI スケール設定の上書き」
   「高い DPI スケールの動作を上書きします。」にチェックを入れて
   拡大縮小の実行元：「システム」　を選んでください。

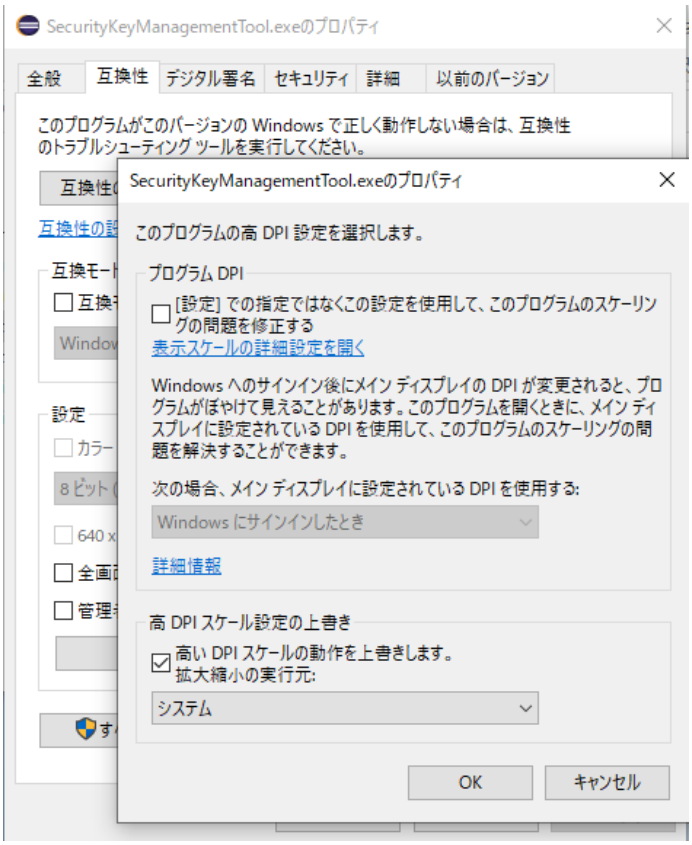

図 7.1　SecurityKeyManagementTool.exe のプロパティ　「高 DPI スケールの上書き」設定

7.2 Linux 版 e²studio plugin 使用時の注意事項

Linux 環境で、e²studio プラグイン版を使用される場合、インストール後にコマンドライン実行ファイルに実行権限を設定する必要があります。

e²studio にプラグイン版の Security Key Management Tool をインストール後、e²studio インストールフォルダの以下のファイルに実行権を付与してください。

/eclipse/plugins/com.renesas.apltool.skmt_X.X.X.XXXXXXXXXXX/cli/skmt

7.3 macOS 版 e²studio plugin 使用時の注意事項

macOS 環境で、e²studio プラグイン版を使用される場合、インストール後にコマンドライン実行ファイルに実行権限を設定する必要があります。

e²studio にプラグイン版の Security Key Management Tool をインストール後、e²studio インストールフォルダの以下のファイルに実行権を付与してください。

/eclipse/plugins/com.renesas.apltool.skmt_X.X.X.XXXXXXXXXXX/cli/skmt

7.4 macOS 版の制限事項

macOS 版はコマンドライン版、スタンドアロン版、e²studio プラグイン版の以下の機能に制限があります。

表 7-1 コマンドライン版制限事項

コマンド	オプション	制限事項
genkey	以下特定の/keytype オプション指定時に、/key オプションを省略して鍵生成を行う。 ”brainpoolP256r1-private” ”brainpoolP384r1-private” ”brainpoolP512r1-private”	使用できません。 “Error: For BRAINPOOLPxxxR1-PRIVATE, key option cannot be omitted when using MAC OS”というエラーが返ります。
	以下特定の/keytype オプション指定時に、/key オプションで PEM ファイル入力を行う。 ”brainpoolP256r1-public”、 ”brainpoolP256r1-private”、 ”brainpoolP384r1-public”、 ”brainpoolP384r1-private”、 ”brainpoolP512r1-public”、 ”brainpoolP512r1-private”、	使用できません。 “Error: For BRAINPOOLPxxxR1-PRIVATE/PUBLIC, PEM file cannot be specified when using MAC OS.”というエラーが返ります。
encsfp	-	encsfp コマンドは使用できません。 “Error: For MAC OS, encsfp command is not supported.”というエラーが返ります。

表 7-2 スタンドアロン版、e2studio plugin 版制限事項

タブ	操作	制限事項
鍵のラッピング	[KeyType]タブで、以下の鍵種を選択時、[KeyData]タブで”Random”を選択して鍵生成を行う。 ”brainpoolP256r1-private” ”brainpoolP384r1-private” ”brainpoolP512r1-private”	鍵生成機能を使用できません。 Generate file ボタンを押すと“Error: For BRAINPOOLPxxxR1-PRIVATE, key option cannot be omitted when using MAC OS”というエラーが返ります。
	[KeyType]タブで、以下の鍵種を選択時、[KeyData]タブで”File”を選択して入力に PEM ファイルを使用する。 ”brainpoolP256r1-public”、 ”brainpoolP256r1-private”、 ”brainpoolP384r1-public”、 ”brainpoolP384r1-private”、 ”brainpoolP512r1-public”、 ”brainpoolP512r1-private”、	PEM ファイルを扱えません。 Generate file ボタンを押すと“Error: For BRAINPOOLPxxxR1-PRIVATE/PUBLIC, PEM file cannot be specified when using MAC OS.”というエラーが返ります。
SFP	Generate file ボタンが有効になりません。	Secure Factory Programming 機能用のファイルを生成できません。

8. 付録

A. .NET

本ソフトウェアは.NET Foundationのオープンソースソフトウェア.NETを使用しています。
.NET のライセンスは以下をご参照ください。
.NET License :
https://github.com/dotnet/runtime/blob/main/LICENSE.TXT

.NET Foundation :
https://dotnetfoundation.org/

B. Inno Setup

本ソフトウェアのwindows版のインストーラはInno Setupを使用して作成しています。
Inno Setup のライセンスは以下をご参照ください。

Inno Setup License :
https://jrsoftware.org/files/is/license.txt

Inno Setup :
https://jrsoftware.org/isinfo.php

C. Renesas File Key Format

1) テキストフォーマット

表 8-1 Renesas File Key Format テキストフォーマット

Item	Specification
文字列	ASCII (Unicode や multi-byte character は使用できません)
空白文字	TAB(0x09) / Space(0x20)
1 行の最大バイト長	80 bytes
改行コード	CRLF(0x0D,0x0A) / CR(0x0D) / LF(0x0A)
Base64	Chars = Alpha / Digit / "+" / "/" Pad = "=" Line length = 64 chars (最終行は除く)

2) ファイルのデータ構造

ファイルのデータは以下の 3 要素から構成されます。

```
<Header>
<Base64Lines>
<Footer>
```

<Header> と <Footer> 固定文字列です。

Header = "-----BEGIN RENESAS KEY-----"

Footer = "-----END RENESAS KEY-----"

<Base64Lines> は 4)で示される鍵データのバイナリデータを Base64 変換した文字列です。フッター後の空欄、改行は無視します。

3) ファイルの拡張子

".rkey"

4) 鍵データ

鍵データは以下の構造とサイズになります。データは big-endian 並びです。

表 8-2 Renesas File Key Format 鍵データ構造

Name	Type	Size	Description
Magic code	Char[4]	4 Bytes	"REK1"
Suite Version	Integer	4 Bytes	データフォーマットバージョン 1
Reserved	Byte[7]	7 Bytes	0
Key Type	Byte	1 Bytes	[DLM 鍵] 0 [ユーザ鍵] Keytype 値 (Keytype 値は 4.5.2**keytype** オプションを参照)
Encrypted Key Size	Integer	4 Bytes	"Encrypted Key" (= N bytes)のサイズ
W-UFPK	Byte[36]	36 Bytes	Renesas DLM サーバーから送付される W-UFPK ファイルの値 最初の4バイトはshared key number 残りの 32 バイトが WUFPK 値
Initial Vector	Byte[16]	16 Bytes	ユーザ鍵をラップするときに使用した初期化ベクタ
Encrypted Key	Byte[N]	N Bytes	UFPK でラップしたユーザ鍵と MAC 値
Data CRC	Byte[4]	4 Bytes	データの CRC-32 演算値 初期値 = 0xFFFFFFFF Magic number = 0x04C11DB7

5) Encrypted Key 計算式

ユーザ鍵をUFPKもしくはKUKでラップするときの暗号化とMAC生成は以下の式で行います。

・ RAファミリ、RXファミリ、Synergy PlatformのEncrypted Key計算式

```
uint32_t i = 0;
uint32_t n            = User key byte size;
uint8_t IV[16];                                    // Initial Vector(128bit)
uint8_t CBCKey[16]     = UFPK[0:15] or KUK[0:15];   // CBCKEY in either UFPK or KUK
uint8_t CBCMACKey[16] = UFPK[16:31] or KUK[16:31]; // CBCMACKEY in either UFPK or KUK
uint8_t User_Key[n];                               // Plain text User key
uint8_t MAC[16] = {0};
uint32_t encrypted_key[n];                         // Encrypted Key

for (i = 0; i < n; i += 16)
{
   MAC[0:15]    =
      AES128-Enc(CBCMACkey[0: 15], xor_16byte(User_Key[i: i+15], MAC[0: 15]));
   encrypted_key[i: i+15] =
      AES128-Enc(CBCkey[0: 15],    xor_16byte(User_Key [i: i+15], IV[0: 15]));
   IV[0: 15]      = encrypted_key [i: i+15];
}
encrypted_key[i: +15] = AES128-Enc(CBCkey[0: 15], xor_16byte(MAC[0: 15], IV[0: 15]));
```

* : AES128-Enc()は、AES ECBモード 鍵長128bitの暗号化　を示します。
　第一引数：暗号化時に使用する鍵 第二引数：暗号化する平文データ

・ RZ/T2M2, RZ/T2ME, RZ/T2L, RZ/N2L, RZ-TSIPのEncrypted Key計算式

```
uint32_t n            = User key byte size;
uint8_t IV[16];                                    // Initial Vector(128bit)
uint8_t CBCKey[16]     = UFPK[0:15] or KUK[0:15];   // CBCKEY in either UFPK or KUK
uint8_t CMACKey[16]    = UFPK[16:31] or KUK[16:31]; // CMACKEY in either UFPK or KUK
uint8_t User_Key[n];                               // Plain text User key
uint8_t MAC[16] = {0};
uint32_t encrypted_key[n];                         // Encrypted Key

MAC[0:15]           = AES128-CMAC(CMACkey[0: 15], IV[0: 15], User_Key[0: n-1]); *1
encrypted_key[0: n-1] = AES128-CBC(CBCkey[0: 15],   IV[0: 15], User_Key[0: n-1]); *2
encrypted_key[n: n+15] = MAC[0:15];
```

*1 : AES128-CMAC()は、鍵長128bitのAES CMAC生成演算を示します。
　第一引数：暗号化鍵、第二引数：Initial Vector 第三引数：平文データ
*2 : AES128-CBC()は、鍵長128bitのAES CBC暗号化演算を示します。
　第一引数：暗号化鍵、第二引数：Initial Vector 第三引数：平文データ

D Secure Factory Programming File Format

1) ファイルフォーマット

固有情報を格納する『Headerフィールド』、TLVフィールドのサイズを格納する『TLV Lengthフィールド』、暗号化データを格納する『TLV(Type-Length-Value)フィールド』で構成されるバイナリファイルです。図 8.1Secure Factory Programmingファイルフォーマットにフォーマットイメージを示します。データはBig-Endianの並びでデータを配置します。

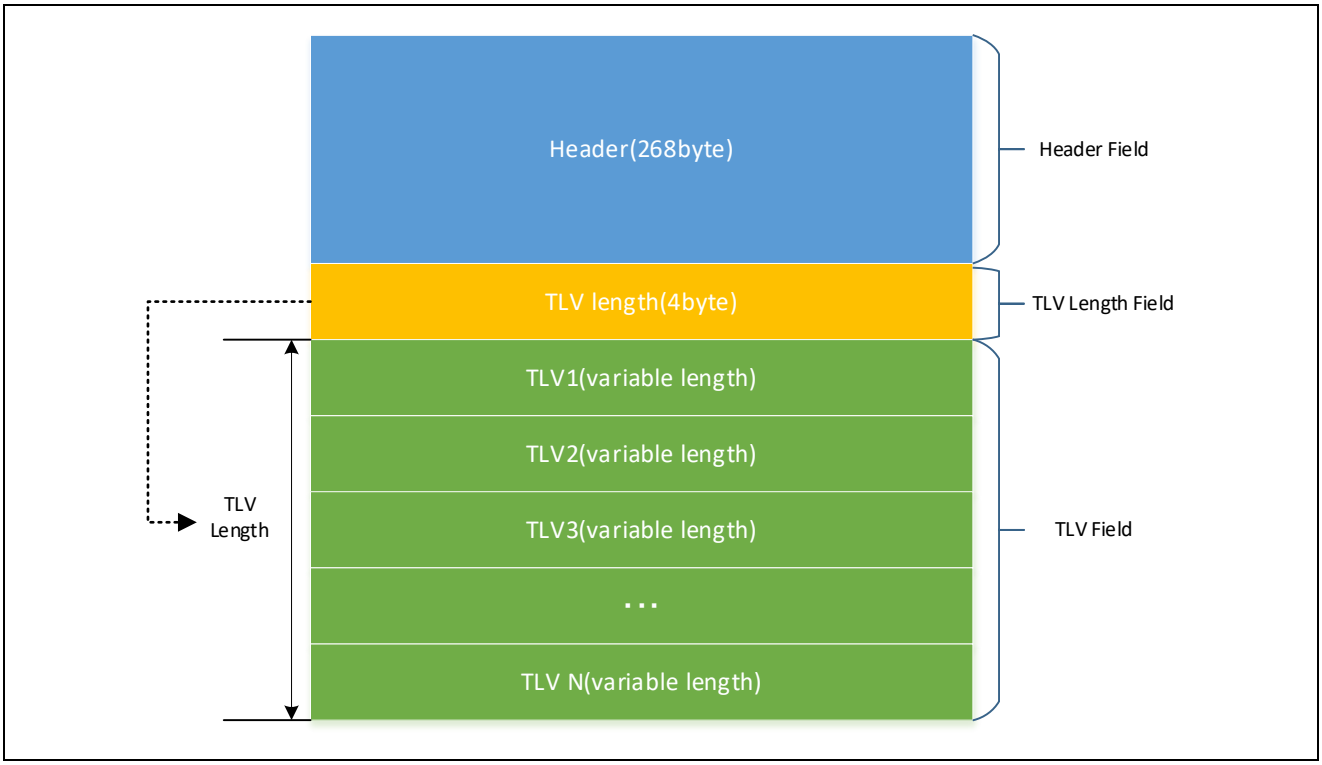

図 8.1　Secure Factory Programming ファイルフォーマット

a) Headerフィールド

Header フィールドは、固有情報を格納する 268byte のフィールドです。

表 8-3 Secure Factory Programming File Format Header フィールド

Field	Size [byte]	Description
Magic	4	固有のマジックナンバーで”sfpr”の ASCII コードを符号無し 4byte 整数で表した”0x73667072”を設定する
Version	4	フォーマットバージョン 上位 2byte がメジャーバージョン、下位 2byte がマイナーバージョンを表す V.1.00 では”0x00010000”を設定する。現在は V.1.00
W-UFPK	36	Renesas DLM サーバーから送付される W-UFPK ファイルの値 最初の 4 バイトは shared key number 残りの 32 バイトが WUFPK 値
Initialization Vector used for encrypting ENKY	16	ユーザプログラムならびにパラメータの暗号化に使用した鍵をラップするときに使用した初期化ベクタ
ENKY	32	パラメータ値ならびにユーザプログラムを暗号化するときに使用した鍵を UFPK で暗号化したデータ
Nonce used for encrypting parameters	12	パラメータを暗号化するときに使用した Nonce データ
Encrypted Parameter	16	パラメータ値を Encryption key で暗号化したデータ
MAC for Encrypted Parameter	16	パラメータ値を Encryption key で暗号化したときに生成される MAC 値
Reserved	3	0xFF
Encrypted AL2 Key Enable	1	Initialization Vector for used encrypted AL2 Key フィールドと Encrypted AL2 Key with MAC フィールドの有効無効フラグ 0：無効 それ以外：有効
Initial Vector used for encrypting AL2 Key	16	AL2 Key 暗号化時に使用した Initialization Vector 値
Encrypted AL2 Key	32	AL2 Key を UFPK で暗号化したデータ＋MAC 値
Reserved	52	0xFF
Nonce used for encrypting user data	12	ユーザプログラムを暗号化するとき使用した Nonce 値
MAC for encrypted user data	16	ユーザプログラムを暗号化するときに生成された MAC 値

b) TLV Lengthフィールド

TLV LengthはTLVフィールドのサイズを記述するフィールドです。

表 8-4 Secure Factory Programming File Format TLV Length フィールド

Field	Size [byte]	Description
TLV Length	4	TLVフィールドの合計バイトサイズ

c) TLV フィールド

TLVは、以下のいずれかの値をType-Length-Value形式で設定するフィールドです。

- Encrypted User Program(暗号化されたプログラム)

TLVのフォーマットを図 8.2TLV formatに示す。

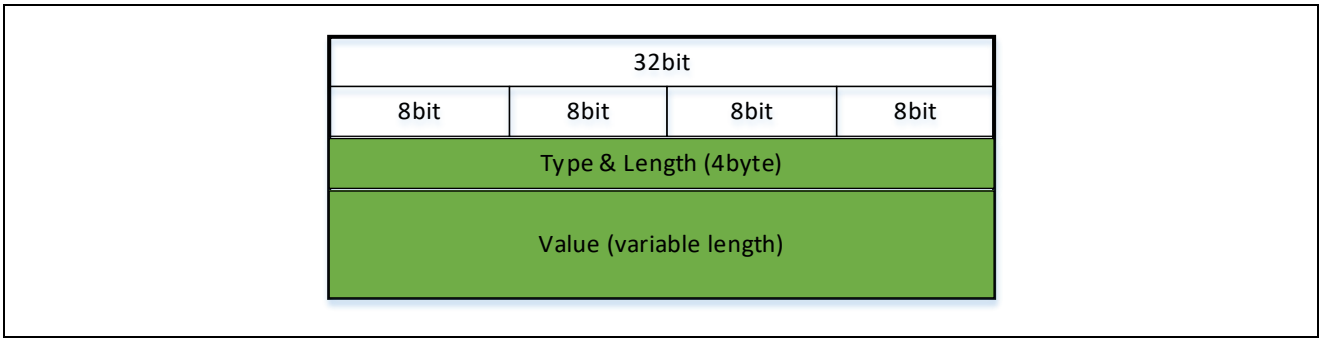

図 8.2 TLV format

表 8-5 Secure Factory Programming TLV Format

Field	Size [byte]	Description
Type & Length	4	Valueの種別及びword(32bit)サイズ
Value	可変長	タイプ毎のデータ サイズはLengthで指定された値の4倍となる (Lengthはwordサイズ指定のため)

A) Type&Length フィールドの詳細

Type&Lengthフィールドは32ビットのビットフィールドで定義します。

表 8-6 Secure Factory Programming TLV Format Type&Length フィールド

Field	Bit Field	Size(bit)	Bit position	Description
Type	Class	4	31:28	Valueの分類 bit 31 28 0 0 0 0 : Encrypted User Data ※他は予約
	<Class unique>	4	27:24	Class毎のフィールド(詳細は別途記載)
Length	word size	24	23:0	Valueのwordサイズ(1word = 4byte) [値]0～16,777,215

B) Class 毎の Unique フィールド

Class毎のClass uniqueフィールドの値について説明します。

表 8-7 Secure Factory Programming TLV Format Class フィールドが『Encrypted User Data』の場合

Bit Field	Size (bit)	Bit position	Description
Use type	4	27:24	利用タイプ [value] bit 27 24 0 0 0 0 : Encrypted User Data ※他は予約

2) ファイルの拡張子

".sfp"

E 鍵証明書 / コード証明書

1) 鍵証明書ファイルフォーマット

以下データ構造のバイナリファイルです。
詳細はFSBL機能をサポートしているデバイスのUser Manualをご参照ください。

表 8-8 鍵証明書 データフォーマット

Field		Size [byte]	Description
Header	Magic	4	0x6B657963
	Manifest Version	4	0x00010000
	Flags	4	Reserved (0x00000000)
	Reserved	20	Reserved (ALL 0)
TLV Length		4	鍵証明書の TLV フィールドのデータバイト長 0x000000AC
TLV ECC PUBKEY	Type&Length	4	0x00088010
	Value	64	/gencert /oemroot_public オプションで指定した OEM_BL 検証ルート公開鍵
TLV KEYHASH	Type&Length	4	0x10144008
	Value	32	/gencert /oembl_public オプションで指定した OEM BL 検証公開鍵の SHA2-256 HASH 値
TLV EXPECTED_SIG	Type&Length	4	0x20088410
	Value	64	署名値

2) mode “signature”で生成されるコード証明書ファイルフォーマット

以下データ構造のバイナリファイルです。
詳細はFSBL機能をサポートしているデバイスのUser Manualをご参照ください。

表 8-9 mode “signature”の場合のコード証明書 データフォーマット

Field		Size [byte]	Description
Header	Magic	4	0x636F6463
	Manifest Version	4	0x00010000
	Flags	4	0x00000000
	Load Addr	4	/gencert /loadaddr オプションで指定した OEM BL 開始アドレス
	Dest Addr	4	/gencert /loadaddr オプションで指定した OEM BL 開始アドレス
	Image size	4	/gencert /oembl_size オプションで指定した OEM BL サイズ /oembl_size 省略時は、/cfsize オプションと/oembl で入力された OEM BL ファイルから計算した OEM BL サイズ
	Image version	4	/gencert /ver オプションで指定したバージョン情報
	Build number	4	Reserved (ALL 0)
TLV Length		4	コード証明書の TLV フィールドのデータバイト長 0x000000AC
TLV ECC PUBKEY	Type&Length	4	0x00088010
	Value	64	OEM_BL 検証ルート公開鍵
TLV EXPECTED_CRC	Type&Length	4	0x40000001
	Value	4	OEM BL の CRC32 値
TLV SIGNER_ID	Type&Length	4	0x10144008
	Value	32	OEM BL 検証公開鍵の SHA2-256 HASH 値
TLV EXPECTED_SIG	Type&Length	4	0x25088410
	Value	64	署名値

3) mode “crc”で生成されるコード証明書ファイルフォーマット

以下データ構造のバイナリファイルです。
詳細はFSBL機能をサポートしているデバイスのUser Manualをご参照ください。

表 8-10 mode “crc”の場合のコード証明書データフォーマット

Field		Size [byte]	Description
Header	Magic	4	0x636F6463
	Manifest Version	4	0x00010000
	Flags	4	0x00000000
	Load Addr	4	/gencert /loadaddr オプションで指定した OEM BL 開始アドレス
	Dest Addr	4	/gencert /loadaddr オプションで指定した OEM BL 開始アドレス
	Image size	4	/gencert /oembl_size オプションで指定した OEM BL サイズ /oembl_size 省略時は、/cfsize オプションと/oembl で入力された OEM BL ファイルから計算した OEM BL サイズ
	Image version	4	0
	Build number	4	Reserved (ALL 0)
TLV Length		4	コード証明書の TLV フィールドのデータバイト長 0x000000AC
TLV EXPECTED_CRC	Type&Length	4	0x40000001
	Value	4	OEM BL の CRC32 値
TLV SIGNER_ID	Type&Length	4	0x10144008
	Value	32	OEM BL の CRC32 値で FILL します。

4) CRC32の計算式

コード証明書のTLV EXPECTED_CRCの計算値は以下になります。

- CRC32 (x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1)

多項式 ：0xEDB88320(Bit revesed)

シフト方向 ：右シフト

入力ビット反転：なし

出力ビット反転：あり

初期値 ：0xFFFFFFFF

F TSIP Update

1) ユーザプログラムの暗号化式

ユーザプログラムの暗号化とMAC生成は以下の式で行います。

```
uint32_t i = 0;
uint32_t n            = User program byte size;
uint8_t IV[16]        = IV[0:15];                              // Initial Vector(128bit)
uint8_t IEkey[16]     = Image Encryption Key[0:15];
uint8_t IEMACKey[16] = Image Encryption Key[16:31];
uint8_t KEKey[16]     = Key Encryption Key[0:15];
uint8_t User_Program[n];
uint8_t MAC[16] = {0};
uint32_t encrypted_user_program[n];                           // Encrypted user program

for (i = 0; i < (n-16); i += 16)
{
   MAC[0:15]    =
      AES128-Enc(IEMACkey[0: 15], xor_16byte(User_Program[i: i+15], MAC[0: 15]));
   encrypted_user_program[i: i+15] =
      AES128-Enc(IEkey[0: 15],    xor_16byte(User_Program[i: i+15], IV[0: 15]));
   IV[0: 15]    = encrypted_user_program[i: i+15];
}
encrypted_user_program[i: i+15]
                  = AES128-Enc(IEkey[0: 15], xor_16byte(MAC[0: 15], IV[0: 15]));
SessionKey0[0:15] = AES128-Enc(KEkey[0: 15], IEKey[0: 15]);
SessionKey1[0:15] = AES128-Enc(KEkey[0: 15], IEMACKey[0: 15]);
```

* : AES128-Enc()は、AES ECBモード 鍵長128bitの暗号化 を示します。
　第一引数：暗号化時に使用した鍵　第二引数：平文データ

改訂記録	Security Key Management Tool ユーザーズマニュアル

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Dec.28.21	ー	初版発行
1.01	Mar.31.22	ー	3 GUI 機能説明追記 5 GUI 説明追記 6 GUI 操作方法追記
1.02	Jun.30.22	ー	4.5.1 mcu オプション RA-SCE5_B 追加 4.5.3.2 ファイル入力 pem ファイル 追加 5.2.1 Linux 版 GUI 使用時の使用方法 変更
1.03	Dec.29.22	ー	5.2 e^{2}studio plugin 版 説明追記 付録 Renesas Key File Format 追記
1.04	Sep.29.23	ー	3.7 [TSIP UPDATE]タブ 追加 4.5.1 mcu オプション RE-TSIPLite 削除 4.6 enctsip コマンド オプション 追加 4.7 calcresponse コマンド オプション 追加
1.05	Dec.22.23	ー	1.5 セキュリティ機能　追加 3.8 [FSBL]タブ 追加 3.9 [DOTF]タブ 追加 3.10 [SFP]タブ 追加 4.7 gencert コマンド オプション 追加 4.8 encdotf コマンド オプション 追加 4.9 encsfp コマンド オプション 追加 6.5 FSBL 鍵証明書/コード証明書生成方法 追加 6.6 Decryption On-The-Fly の使用方法 追加 6.7 Secure Factory Programming の使用方法 追加
1.06	Mar.29.24	P.26 P.50 P.52 P.88 P.100 P.104 P.107	2.2.2 e2studio 動作確認バージョン 2024-01　変更 3.7.3 暗号化する範囲の上限は 8MB 追記 3.7.6 RSU ヘッダ Ver 2 を指定可能 変更 4.5.3.2 pem ファイルサポート非対称鍵に RSA-2048-public-TLS 追加 4.6 ver オプション 2 追加 省略時は 2 に変更 endaddr オプション 暗号化領域のサイズの上限は 8MB 追記 4.6.2 ver オプション RSU ヘッダ V1 Sequence Number は常に 1 修正 Execution Address 固定値 0xFFFEFFFC　追記 RSU ヘッダ V2 追加 4.7 ver オプション 1～4,294,967,295 で指定可能

Rev.	発行日	改訂内容	
		ページ	ポイント
1.07	Aug.30.24	P.5	用語 Encrypted KUK 追加
		P.11	1.2 表 1-1 RZ/T2M, RZ/T2ME, RZ/T2L, RZ/N2L 追加
		P.25	2.1.1 表 2-1 RX RSIP-E11A RZ/T2M, RZ/T2ME, RZ/T2L, RZ/N2L 追加
		P.26	2.1.2 表 2-2 RX RSIP-E11A RZ/T2M, RZ/T2ME, RZ/T2L, RZ/N2L 追加
		P.34	2.2.2 対応 OS Ubuntu 22.04 LTS, macOS14 追加 e2studio 動作確認バージョン 2024-07 更新
		P.40	3.6.2 表 3-6 macOS 時の注意事項 追記
		P.47	3.6.5 表 3-8 追加
		P.49	3.7 TSIP UPDATE → TSIP Update タブ名変更
		P.63	3.9 DOTF → DOTF/OTFD タブ名変更
		P.69	3.10 表 3-12 “OEM PL0 with AL2_KEY and AL1_KEY” 削除 注意事項追記
		P.74	3.10.5 注意事項 追記
		P.78	4.5 genkey オプション fileadd オプション、bswap オプション追加
		P.81	4.5.1 表 4-8 RX RSIP-E11A RZ/T2M, RZ/T2ME, RZ/T2L, RZ/N2L 追加
		P.88	4.5.3.1 表 4-44 Qx → Qy 修正
		P.90	4.5.3.2 表 4-54 macOS 時の注意事項 追記
		P.92	4.5.3.3 表 4-55 macOS 時の注意事項 追記
		P.101	4.5.5 fileadd オプション 追加 4.5.6 bswap オプション 追加
		P.119 ～121	4.9 al1_key オプション削除
		P.122	4.9.2 表 4-91 OEM_PL0_AL2_1 削除
		P.127	5.1.1.2 CLI 版のファイル構成変更 DLL ファイル削除
		P.130 ~131	5.1.3 macOS 版 追加
		P.176	7.3 macOS 版 e2studio plugin 使用時の注意事項 追記
		P.176 ～177	7.4 macOS 版 注意事項 追記
		P.178 ～179 P.187	Appendix Renesas File Key Format ユーザ鍵 暗号化式追記 TSIP Update ユーザプログラム暗号化式追加

Security Key Management Tool ユーザーズマニュアル

発行年月　2024年8月30日　Rev.1.07

発行　ルネサス エレクトロニクス株式会社
〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

Security Key Management Tool

ルネサス エレクトロニクス株式会社

R20UT5349JJ0107