

# RX ファミリ

R01AN6948JJ0200

Rev.2.00

2024.01.23

## TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

### Important Notice:

On November 21, 2023, Microsoft announced that they have decided to contribute Azure RTOS to Open Source

under the stewardship of the Eclipse foundation and Azure RTOS becomes Eclipse ThreadX.

For detailed information, please refer to the announcement titled at Microsoft Contributes Azure RTOS to Open Source.

The support strategy scheme for Eclipse ThreadX will be determined and communicated at a later date.

Microsoft will discontinue the Azure RTOS and Azure RTOS Middleware under the existing agreement LICENSED-HARDWARE.txt.

It's important to note that updates for Azure RTOS on these hardware will no longer be provided.

### 要旨

Trusted Secure IP (TSIP) ドライバは SSL/TLS (本書では以後 TLS と表記) 通信用の API をサポートしています。本書では、NetX Duo を含む Azure RTOS への TSIP ドライバの組み込み例とその動作確認方法を紹介します。

本書には、Azure RTOS をベースとしたサンプルプロジェクトを添付しています。このサンプルプロジェクトは NetX Duo を含む Azure RTOS に TSIP ドライバを組み込んでおり、Microsoft Azure を介した MQTT 通信をテストすることができます。

本書のサンプルプロジェクトは Microsoft Azure IoT Hub と IoT Hub Device Provisioning Service (DPS) を介して接続します。DPS の接続認証に TLS を使用します。

### 動作確認デバイス

本書に付属するサンプルプログラムは以下のデバイスで動作確認しています。

- RX65N: R5F565NEHDF
- RX72N: R5F572NNHDFB

(※)本ドキュメントは CK-RX65N (Ethernet) を例として記載しております。

### 動作環境

本書に付属するサンプルプログラムは以下の環境で動作確認しています。

IDE	<a href="#">e2 studio 2023-10</a>
ツールチェーン	<a href="#">CCRX コンパイラ v3.05.00</a>
ターゲットボード	CK-RX65N (Ethernet) CK-RX65N+RYZ014A (Cellular) Renesas Starter Kit+ for RX65N-2MB RX72N Envision Kit
デバッガ	E2 Lite エミュレータ (CK-RX65N、RX72N Envision Kit は内蔵)
Azure RTOS	v6.2.1_rel-rx-1.3.0
ドライバパッケージ (RDP)	<a href="#">RX Driver Package V1.41</a>
TSIP ドライバ	<a href="#">バージョン 1.19 (バイナリ版)</a>
Security Key Management Tool	<a href="#">Security Key Management Tool V.1.04</a>
Tera Term	バージョン 4.106
OpenSSL	3.1.3 Light
Gpg4win (Kleopatra)	4.1.0
シェルスクリプト (bash) 実行環境	Cygwin version 3.4.6

(※) 本書には対応するターゲットボードごとにサンプルプロジェクトを用意しております。お使いのターゲットボードに合わせたサンプルプロジェクトをインポートしてご使用ください。また、CK-RX65N はボードに実装されている LAN を使用する Ethernet 版と、セルラーモジュール RYZ014A をボードに接続してモバイルネットワークを使用する Cellular 版があります。それぞれ使用するサンプルプロジェクトが異なるのでご注意ください。

### 関連ドキュメント

TSIP を用いた TLS 通信の詳細については以下のアプリケーションノートを参照してください。

- ・RX ファミリ TSIP(Trusted Secure IP)モジュール Firmware Integration Technology ([R20AN0548](#))

また、FreeRTOS を使用した TSIP ドライバの TLS 実装例として、以下のアプリケーションノートも参考にすることができます。

- ・RX ファミリ TSIP ドライバを用いた TLS 実装方法([R01AN5880](#))

Azure の操作については以下のドキュメントを参考にしてください。

- ・RX65N グループ RX65N Cloud kit で Azure RTOS を用いて センサデータを可視化する方法 ([R01AN6011](#))

目次

1.	概要	5
1.1	TSIP を用いた TLS 通信のメリット	5
1.2	TSIP を用いた TLS フロー	5
1.3	TSIP ドライバでサポートしている Cipher Suite	5
1.4	TSIP ドライバの TLS 向け API 一覧	6
1.5	用語の定義	7
2.	サンプルプロジェクトの準備	8
2.1	ワークスペースの作成	9
2.2	プロジェクトのダウンロード	10
2.3	プロジェクトのインポート	10
2.4	鍵と証明書の準備	12
2.4.1	OpenSSL のインストール	12
2.4.2	ルート CA 証明書の入手	13
2.4.3	RSA の鍵ペアとクライアント証明書の生成	20
2.4.4	ルート CA 証明書の署名生成と証明書ファイル形式の変換	22
2.4.5	鍵のラップとプロジェクトへの登録	24
2.4.5.1	UFPK と W-UFPK の作成	25
2.4.5.2	鍵データのラップ	34
3.	Microsoft Azure ポータルでの操作	42
3.1	Azure IoT Hub との接続準備 (Azure portal)	42
3.1.1	IoT Hub の作成	42
3.1.2	IoT Hub Device Provisioning Service (DPS) の作成	42
3.1.3	IoT Hub と DPS を用いてデバイスをプロビジョニング	42
3.2	Microsoft Azure との通信設定	48
3.2.1	Azure IoT の設定	48
3.2.2	IP アドレスの設定	51
3.2.3	クライアント証明書の形式の選択	52
4.	プロジェクトのビルドおよび実行	53
4.1	プロジェクトのビルド前の確認	53
4.1.1	Renesas Starter Kit+ for RX65N-2MB の設定	53
4.1.2	ビルド時のコード生成の設定	53
4.2	プロジェクトのビルドと実行	55
4.3	Microsoft Azure との接続確認	56
4.3.1	登録状態の確認	56
4.3.2	デバイスの確認	57
5.	付録	58
5.1	Security Key Management Tool の詳細情報	58
5.2	TSIP ドライバを用いた TLS 通信の性能	58
6.	改訂履歴	60

注 :

- Git®は Software Freedom Conservancy, Inc.のトレードマークです。  
(<https://www.git-scm.com/about/trademark>)
- GitHub®は GitHub, Inc.のトレードマークです。( <https://github.com/logos> )
- OpenSSL™は OpenSSL Software Foundation のトレードマークです。  
(<https://www.openssl.org/policies/trademark.html>)
- Microsoft Azure は, Microsoft Corporation のトレードマークです ( <https://www.microsoft.com/en-us/legal/intellectualproperty/trademarks> ) 。

1. 概要

1.1 TSIP を用いた TLS 通信のメリット

TSIP ドライバでは TLS 向けの API をサポートしています。本 API を利用することにより以下 2 点のメリットがあります。

- TLS プロトコル処理中で平文の鍵情報を扱わないため、デバイス内に格納されたお客様の鍵情報の漏洩リスクを減らすことができます。
- ハードウェアでアクセラレートすることにより、暗号処理を高速化できます。

1.2 TSIP を用いた TLS フロー

以下に本サンプルプロジェクトにおける TLS のフローを示します。  
このフローはデバイス認証に X.509 自己署名証明書（鍵交換方式が RSA の場合）とした場合の例となります。

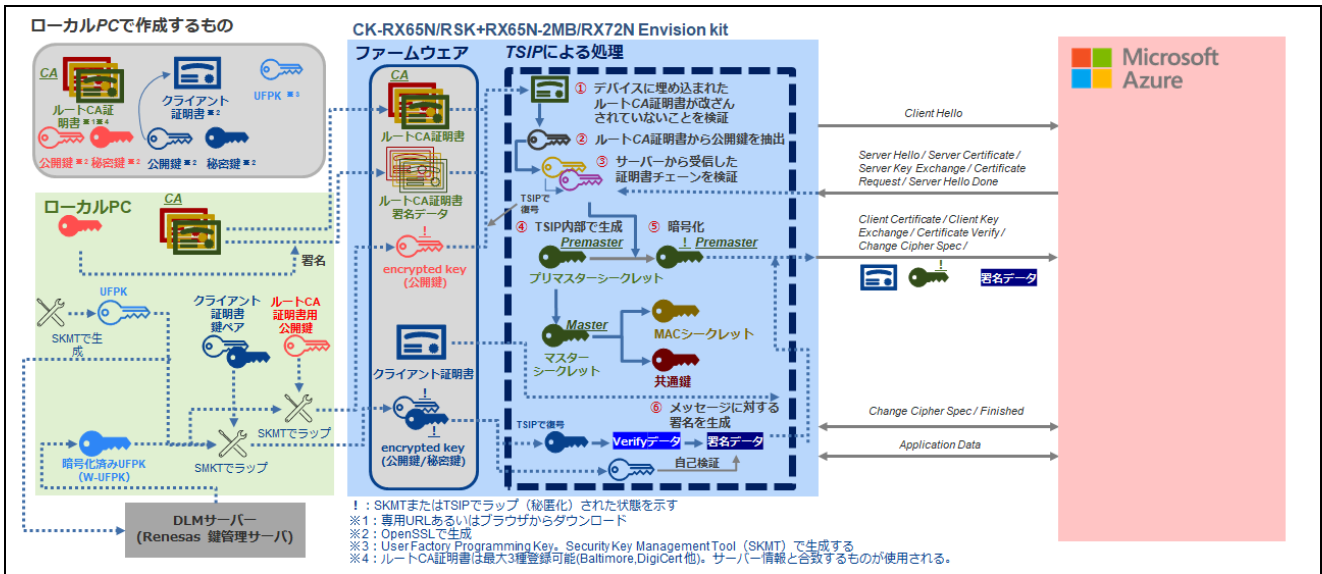


図 1-1 TSIP を用いた TLS フロー

1.3 TSIP ドライバでサポートしている Cipher Suite

TSIP ドライバは TLS1.2 に準拠した以下の Cipher Suite をサポートしています。

- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256 注
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 注
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 注

注: 本書のサンプルプロジェクトでは、サーバ および クライアント認証 (ECDSA)、暗号処理(AES-GCM) はサポートされておりません。

1.4 TSIP ドライバの TLS 向け API 一覧

表 1-1 に TLS 通信で使用する TSIP ドライバの API を示します。各 API の詳細は、アプリケーションノート「RX ファミリ TSIP(Trusted Secure IP)モジュール Firmware Integration Technology (バイナリ版) (R20AN0548)」を参照してください。

表 1-1 TLS 通信で使用する API 関数

使用場所	API 関数
証明書のインストール	R_TSIP_GenerateTlsRsaPublicKeyIndex R_TSIP_Close R_TSIP_Open R_TSIP_TlsRootCertificateVerification
乱数生成	R_TSIP_GenerateRandomNumber
クライアント認証 (鍵データ処理)	R_TSIP_GenerateRsa2048PrivateKeyIndex R_TSIP_GenerateRsa2048PublicKeyIndex
ハンドシェイクメッセージの HASH 演算	R_TSIP_Sha256Init/Update/Final
Certificate	R_TSIP_TlsCertificateVerificationExtension R_TSIP_TlsCertificateVerification
Server Key Exchange Client Key Exchange (鍵交換方式が ECDHE の場合)	R_TSIP_TlsServersEphemeralEcdhPublicKeyRetrieves R_TSIP_GenerateTlsP256EccKeyIndex R_TSIP_TlsGeneratePreMasterSecretWithEccP256Key
Client Key Exchange (鍵交換方式が RSA の場合)	R_TSIP_TlsGeneratePreMasterSecret R_TSIP_TlsEncryptPreMasterSecretWithRsa2048PublicKey
Certificate Verify	R_TSIP_RsassaPkcs1024/2048SignatureGenerate R_TSIP_RsassaPkcs1024/2048SignatureVerification R_TSIP_EcdsaP192/224/256/384SignatureGenerate R_TSIP_EcdsaP192/224/256/384SignatureVerification
Finished	R_TSIP_TlsGenerateMasterSecret R_TSIP_TlsGenerateVerifyData R_TSIP_TlsGenerateSessionKey R_TSIP_Sha1HmacGenerateInit/Update/Final R_TSIP_Sha1HmacVerifyInit/Update/Final R_TSIP_Sha256HmacGenerateInit/Update/Final R_TSIP_Sha256HmacVerifyInit/Update/Final R_TSIP_Aes128CbcEncryptInit/Update/Final R_TSIP_Aes128CbcDecryptInit/Update/Final R_TSIP_Aes256CbcEncryptInit/Update/Final R_TSIP_Aes256CbcDecryptInit/Update/Final R_TSIP_Aes128GcmEncryptInit/Update/Final R_TSIP_Aes128GcmDecryptInit/Update/Final
Application Data	R_TSIP_TlsGenerateSessionKey R_TSIP_Sha1HmacGenerateInit/Update/Final R_TSIP_Sha1HmacVerifyInit/Update/Final R_TSIP_Sha256HmacGenerateInit/Update/Final R_TSIP_Sha256HmacVerifyInit/Update/Final R_TSIP_Aes128CbcEncryptInit/Update/Final R_TSIP_Aes128CbcDecryptInit/Update/Final R_TSIP_Aes256CbcEncryptInit/Update/Final R_TSIP_Aes256CbcDecryptInit/Update/Final R_TSIP_Aes128GcmEncryptInit/Update/Final R_TSIP_Aes128GcmDecryptInit/Update/Final

## 1.5 用語の定義

本書で使用される用語の定義を以下に示します。

表 1-2 用語

用語	内容
鍵注入	工場でデバイスに Wrapped Key を注入すること。
ユーザ鍵 (User key)	ユーザが使用する平文状態の暗号鍵。デバイス上では使用しない。RSA、ECC の場合は公開鍵、秘密鍵がそれぞれユーザ鍵となる。
Encrypted key	ユーザ鍵に UFPK による暗号化および MAC 値の付加をして生成される鍵情報。同一のユーザ鍵に対する Encrypted Key は 各デバイスで共通の値となる。
Wrapped Key	Encrypted Key を鍵の注入により TSIP で使用できる形式に変換したデータ。Wrapped Key は HUK でラッピングされているため、同一の Encrypted Key に対する Wrapped Key でもデバイスごとに固有の値となる。
UFPK (User Factory Programming Key)	鍵注入においてユーザ鍵から Encrypted Key を生成するために使用する、ユーザが設定する鍵束。 デバイス上では使用しない。
W-UFPK (Wrapped UFPK)	UFPK を DLM サーバ上の HRK によりラッピングすることで生成される鍵情報。TSIP 内部にて HRK で UFPK に復号されて使用される。
Hardware Root Key (HRK)	TSIP 内部とルネサス内セキュアルームのみに存在する共通の暗号鍵
Hardware Unique Key (HUK)	TSIP 内部で導出する、鍵の保護のために使用するデバイス固有の暗号鍵
ラップ	本書では encrypted key を生成する過程で UFPK を使用して暗号化・MAC 付与を行うことをラップと呼ぶ。 TSIP ドライバは平文のユーザ鍵を入力として受け入れないため、暗号鍵はラップした状態で入力する必要がある。
DLM(Device Lifecycle Management)サーバ ( <a href="https://dlm.renesas.com/">https://dlm.renesas.com/</a> )	Renesas 鍵管理サーバ。UFPK のラッピングに使用する。

### 2. サンプルプロジェクトの準備

本サンプルプロジェクトは、CK-RX65N クラウドキットを用いて Microsoft Azure と TLS 接続し、MQTT 通信を行うデモです。Microsoft Azure IoT Hub へは IoT Hub Device Provisioning Service (DPS) を介して接続を行います。

本章ではデモに使用するサンプルプロジェクトの作成方法をガイドします。

以下に CK-RX65N で本サンプルプロジェクトを実行する際の接続情報を示します。デバッグとシリアル通信のため、CK-RX65N と PC を 2 本の USB ケーブルで接続してください。また、インターネットに接続するために CK-RX65N とルータを Ethernet ケーブルで接続してください。

(※)Renesas Starter Kit+ for RX65N-2MB、RX72N Envision Kit の場合も Ethernet で Azure と接続します。なお、本書での説明の CK-RX65N は Ethernet 版となります。Cellular 版をお使いの場合は、付属の RY2014A ボードを CK-RX65N の PMOD1 端子に接続し、モバイルネットワークで接続します

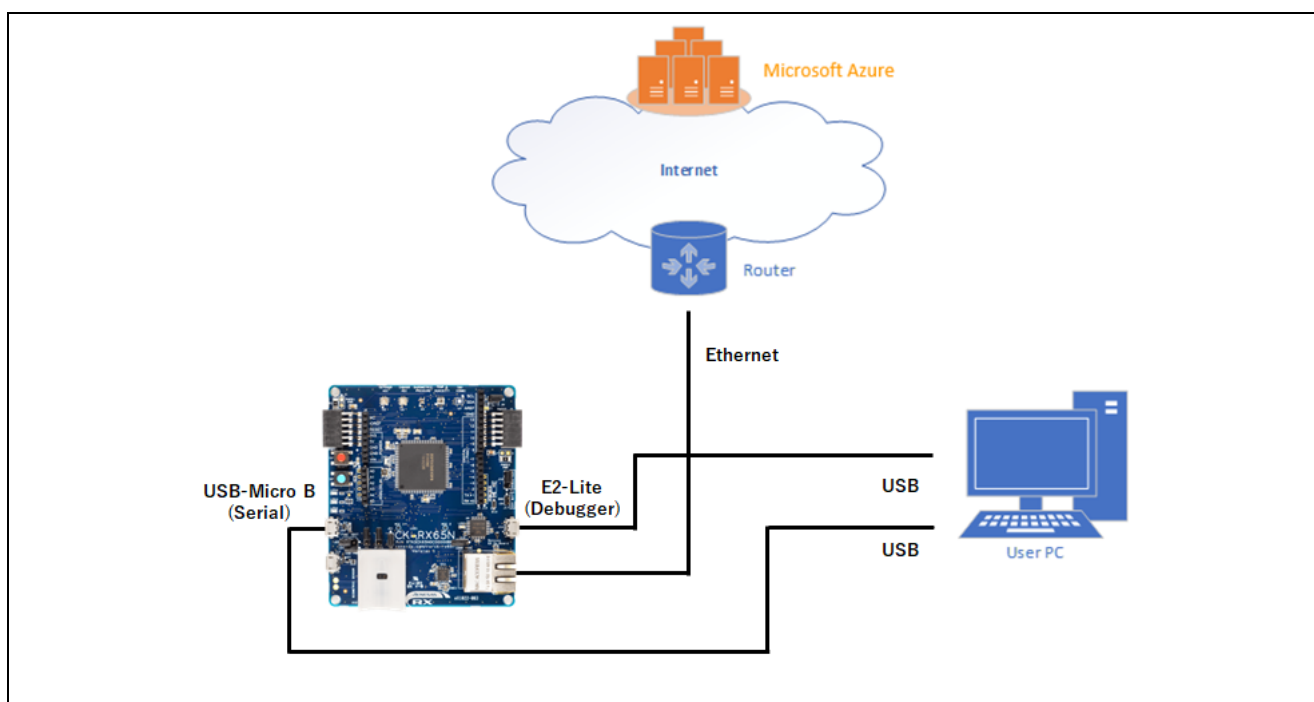


図 2-1 サンプルプロジェクトの接続関係

サンプルプロジェクトは Azure RTOS のプロジェクトをベースにしています。Azure RTOS は IoT 機器向けに必要なソースコードをまとめた IoT Libraries を提供しており、この中で暗号ライブラリとして NetX Secure ライブラリを使用しています。

次にサンプルプロジェクトのソフトウェア構造を示します。



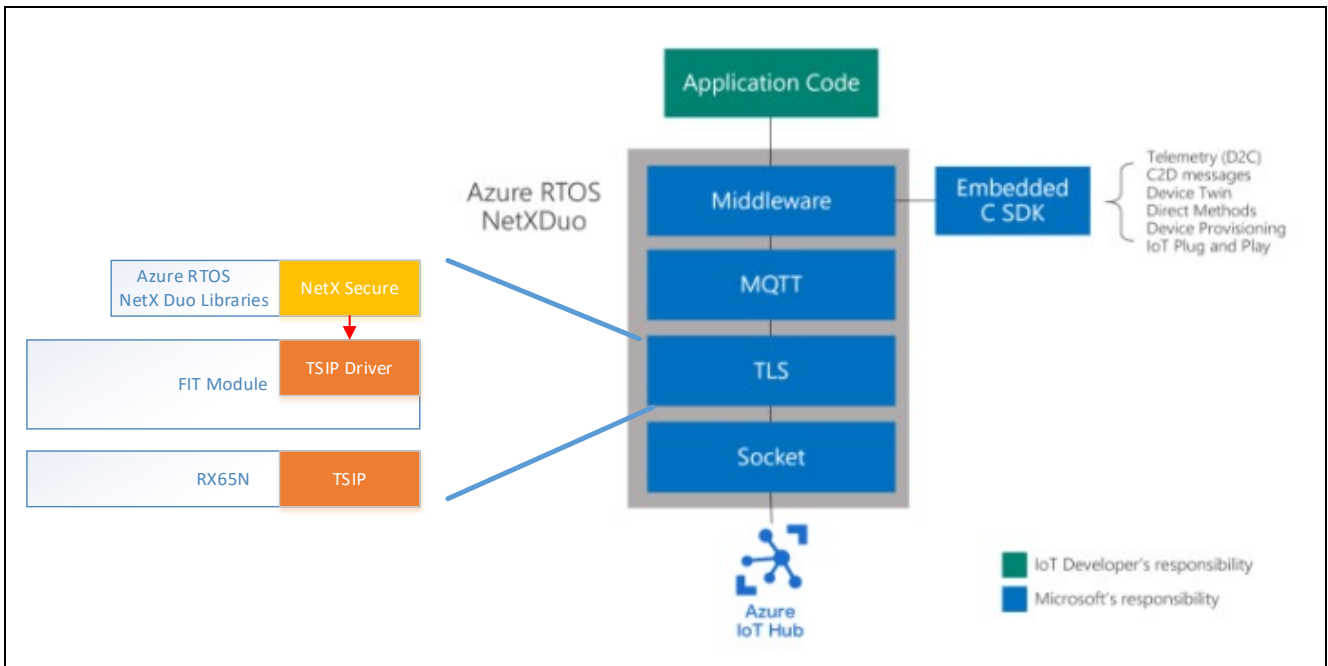


図 2-2 サンプルプロジェクトのソフトウェア構造

サンプルプロジェクトは、以下のリポジトリにある RX 向けの Azure RTOS プロジェクトの v6.2.1 をベースに、TSIP 連携するための変更を加えています。変更点はサンプルプロジェクトと以下のベースプロジェクトを比較した差分をご確認ください。

[https://github.com/renesas/azure-rtos/releases/tag/v6.2.1\\_rel-rx-1.3.0](https://github.com/renesas/azure-rtos/releases/tag/v6.2.1_rel-rx-1.3.0)

また、本章の説明では、以下のツールを使用しますので予めご用意ください。

- シェルスクリプト (bash) の実行環境 (本書では Cygwin を使用しています)
- OpenSSL
- Security Key Management Tool

## 2.1 ワークスペースの作成

e<sup>2</sup> studio を起動して新しいワークスペースを作成してください。

ワークスペースやプロジェクトファイル名はなるべく短くするようにしてください。最下層のフルパスの長さが 256byte を超えるとビルド時にエラーが発生する場合があります。

またパスに日本語が存在するとエラーとなる場合がありますので、名前は英数字で入力してください。

【例】 C:\workspace にワークスペースを作成する場合

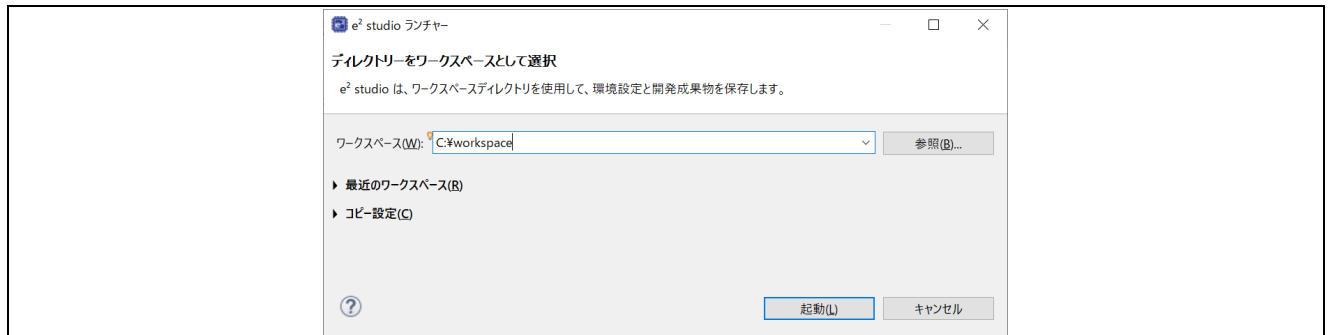


図 2-3 ワークスペース作成画面

## 2.2 プロジェクトのダウンロード

本書に同梱するサンプルプロジェクトは以下のようなフォルダ構成となっています。これらのフォルダを「2.1 ワークスペースの作成」で作成したワークスペースフォルダに配置してください。

以下は第 1 階層のフォルダまで記載しています。

```
ckrx65n_ccrx
|--key_cert_sig_generator
| |--ca
| |--ca-sign-keypair-rsa2048
| |--client-rsa2048
| |--output
|--patch
| |--tsip_integration
|--sample_azure_iot_embedded_sdk
| |--libs
| |--src
```

図 2-4 プロジェクトフォルダ構成

各フォルダにはサンプルプロジェクトのソースコードのほか、サンプルプロジェクトを構築するためのツール類も格納されています。各フォルダの概要は以下の通りです。

表 2-1 サンプルプロジェクトの内容

フォルダ名	内容
key_cert_sig_generator	暗号化に使用する鍵、証明書を生成するツールと作業フォルダを格納しています
Patch	サンプルプロジェクトを生成した場合に、プロジェクトを修正するパッチファイルを格納しています。 プロジェクトをインポートした場合はパッチ適用済みのため、今回は使用しません
sample_azure_iot_embedded_sdk	サンプルプロジェクトの本体です 次項の手順で本フォルダをインポートして使用します

## 2.3 プロジェクトのインポート

e<sup>2</sup> studio の起動後、[ファイル]メニュー ⇒ [インポート] ⇒ [一般] ⇒ [既存プロジェクトをワークスペースへ] を選択し、インポートダイアログを起ち上げてください。

[ルートディレクトリの選択]で[参照]ボタンを押下して、配置したプロジェクトフォルダの"ckrx65n-ccrx"内の"sample\_azure\_iot\_embedded\_sdk"を指定してください。

"プロジェクト"の欄に"sample\_azure\_iot\_embedded\_sdk"が表示されていることを確認して、チェックを入れて[終了]ボタンを押下してください。

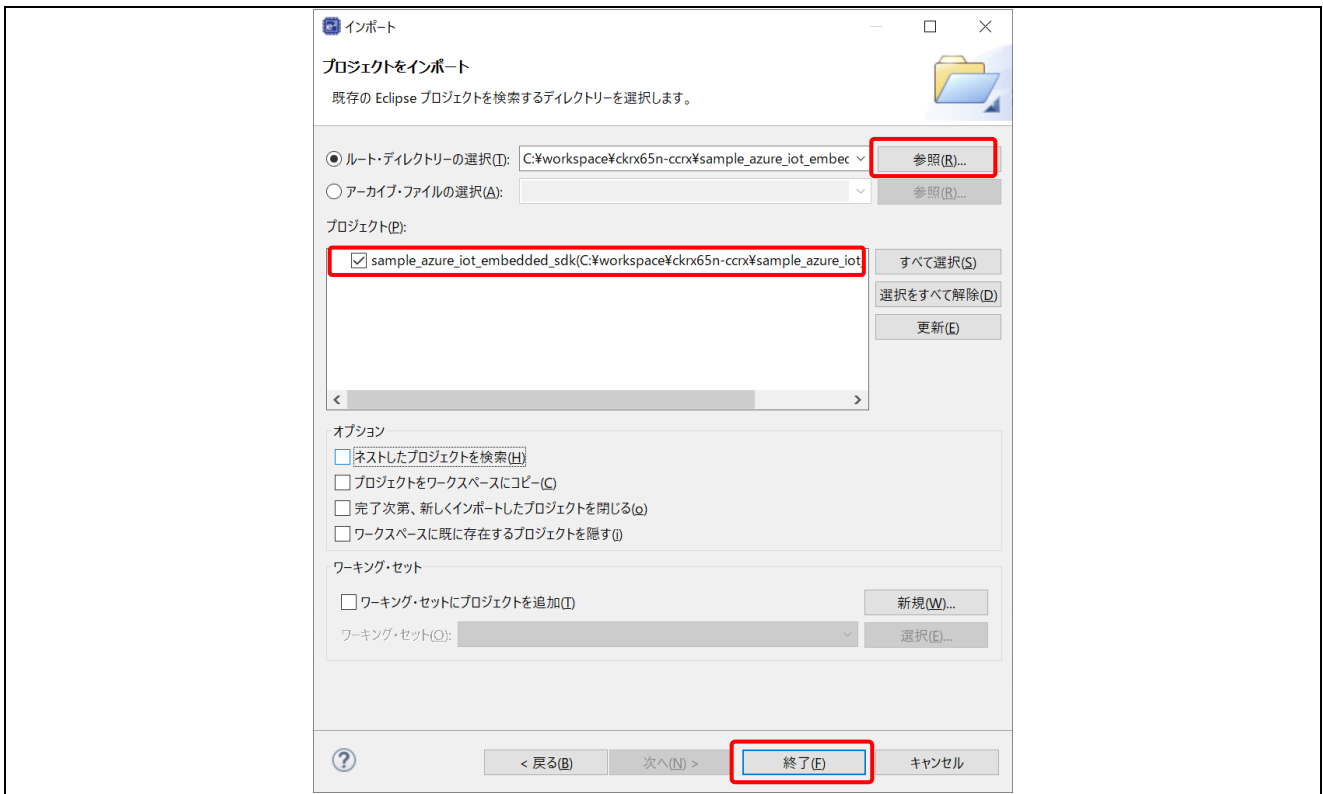


図 2-5 インポートダイアログ画面

プロジェクトが正常にインポートされると以下のようにプロジェクト・エクスプローラーにインポートした "sample\_azure\_iot\_embedded\_sdk" が追加されます。  
 プロジェクト・エクスプローラーが表示されない場合は、画面右上のパースペクティブの [C/C++] を押下してから、[ウィンドウ] ⇒ [ビューの表示] ⇒ [プロジェクト・エクスプローラー] を選択してください。

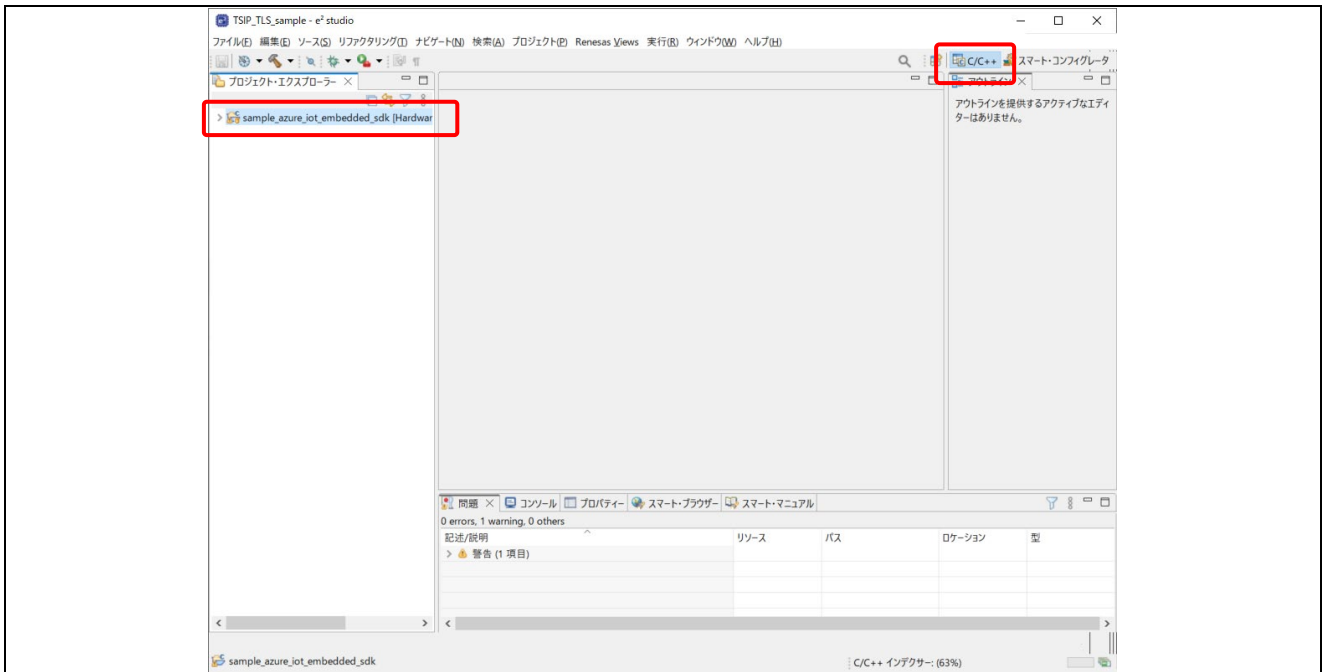


図 2-6 サンプルプロジェクトインポート後の画面

## 2.4 鍵と証明書の準備

サンプルプログラムでは表 2-2 に示す、鍵・証明書の情報を登録する必要があります。本節では、これらの鍵と証明書を手入方法と、TSIP ドライバで使用するために変換手順とプロジェクトファイルへの登録手順について説明します。

手順に従って各鍵と証明書の情報を生成し、プロジェクトへ登録を行ってください。表には必要な情報と入手方法の概要を示しています。

表 2-2 サンプルプロジェクトで使用する鍵と証明書の入手方法

鍵/証明書	入手方法	項
RSA ルート CA 証明書	専用 URL (DER 形式) または ブラウザ (PEM 形式) からダウンロードする	2.4.2
RSA 鍵ペア	OpenSSL 等のツールを使ってユーザが作成する	2.4.3
RSA クライアント証明書	OpenSSL 等のツールを使ってユーザが作成する	2.4.3
ルート CA 証明書の署名生成 / 署名検証用鍵ペア	OpenSSL 等のツールを使ってユーザが作成する	2.4.4 2.4.5

### 2.4.1 OpenSSL のインストール

一部の鍵・証明書の生成には OpenSSL を使用します。以下の手順で OpenSSL をインストールしてください。

- ① Win32/Win64 OpenSSL の[ダウンロードサイト](#)にアクセスして、使用している OS に合わせて OpenSSL のインストーラーをダウンロードしてください。  
64bit 版 Windows の場合は以下の v3.1.3 Light をダウンロードします。  
バージョンが更新されているときは最新バージョンをダウンロードしてください。

Download Win32/Win64 OpenSSL		
Download Win32/Win64 OpenSSL today using the links below!		
File	Type	Description
<a href="#">Win64 OpenSSL v3.1.3 Light</a> <a href="#">EXE</a>   <a href="#">MSI</a>	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.1.3 (Recommended for users by the creators of <a href="#">OpenSSL</a> ). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
<a href="#">Win64 OpenSSL v3.1.3</a> <a href="#">EXE</a>   <a href="#">MSI</a>	140MB Installer	Installs Win64 OpenSSL v3.1.3 (Recommended for software developers by the creators of <a href="#">OpenSSL</a> ). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
<a href="#">Win32 OpenSSL v3.1.3 Light</a> <a href="#">EXE</a>   <a href="#">MSI</a>	4MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v3.1.3 (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
<a href="#">Win32 OpenSSL v3.1.3</a> <a href="#">EXE</a>   <a href="#">MSI</a>	116MB Installer	Installs Win32 OpenSSL v3.1.3 (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
<a href="#">Win64 OpenSSL v3.1.3 Light for ARM (EXPERIMENTAL)</a> <a href="#">EXE</a>   <a href="#">MSI</a>	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.1.3 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
<a href="#">Win64 OpenSSL v3.1.3 for ARM (EXPERIMENTAL)</a> <a href="#">EXE</a>   <a href="#">MSI</a>	113MB Installer	Installs Win64 OpenSSL v3.1.3 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

図 2-7 OpenSSL のダウンロード

- ② ダウンロードした exe (または MSI) を実行し、OpenSSL のインストールを行ってください。
- ③ インストール後は Windows の[システムのプロパティ]⇒[システムの環境変数]の Path に OpenSSL のインストールフォルダを追加してください。  
64bit 版の場合 : C:\Program Files\OpenSSL-Win64\bin

### 2.4.2 ルート CA 証明書の入手

Azure クラウドでは、[Baltimore CyberTrust Root]と[DigiCert Global Root G2]の 2 種のルート CA 証明書を使用します。

本サンプルプロジェクトでは、最大 3 種類のルート CA 証明書を登録することが可能です。以下手順では上記 2 種のルート CA 証明書を登録する手順をガイドします。

RSA ルート CA 証明書を入手し、サンプルプロジェクトへ登録を行います。

入手には以下の 2 通りの方法がありますので、使用される環境に応じた手順をご使用ください。

- Microsoft Azure のサイトから入手する
- Web ブラウザ (Microsoft Edge) よりエクスポートして入手する

いずれの方法でも同じ RSA ルート CA 証明書が入手可能です。

なお、Azure サービスで使用される TLS 証明書は変更が予定されています。詳しくは下記 URL を参照ください。

<https://techcommunity.microsoft.com/t5/internet-of-things-blog/azure-iot-tls-critical-changes-are-almost-here-and-why-you/ba-p/2393169>

今回のサンプルプロジェクトでは以下 2 種のルート CA 証明書を使用します。

- Baltimore Cyber Trust Root
- DigiCert Global Root G2

#### (1) ルート CA 証明書の入手手順

##### (a) Microsoft Azure のサイトから入手する場合

以下の URL からルート CA 証明書を入手可能です。

<https://docs.microsoft.com/ja-jp/azure/security/fundamentals/tls-certificate-changes#what-is-changing>

RSA の証明書として、[Baltimore CyberTrust Root]と[DigiCert Global Root G2]をクリックしてダウンロードしてください。

変更箇所	
変更前、Azure サービスで使用されている TLS 証明書のほとんどは、次のルート CA にチェーンされています。	
CA の共通名	サムプリント (SHA1)
Baltimore CyberTrust Root <a href="#">↗</a>	d4de20d05e66fc53fe1a50882c78db2852cae474
変更後、Azure サービスによって使用される TLS 証明書は、次のいずれかのルート CA にチェーンされるようになります。	
CA の共通名	サムプリント (SHA1)
<b>DigiCert Global Root G2 <a href="#">↗</a></b>	df3c24f9bfd666761b268073fe06d1cc8d4f82a4
DigiCert Global Root CA <a href="#">↗</a>	a8985d3a65e5e5c4b2d7d66d40c6dd2fb19c5436
<b>Baltimore CyberTrust Root <a href="#">↗</a></b>	d4de20d05e66fc53fe1a50882c78db2852cae474
D-TRUST Root Class 3 CA 2 2009 <a href="#">↗</a>	58e8abb0361533fb80f79b1b6d29d3ff8d5f0f0
Microsoft RSA Root Certificate Authority 2017 <a href="#">↗</a>	73a5e64a3bff8316ff0edccc618a906e4eae4d74
Microsoft ECC Root Certificate Authority 2017 <a href="#">↗</a>	999a64c37ff47d9fab95f14769891460eec4c3c5

図 2-8 RSA ルート証明書の入手

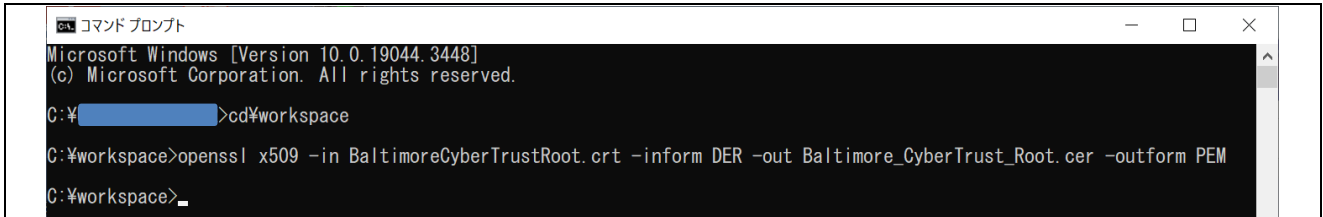
ダウンロードされたルート CA 証明書は以下のファイル名の DER 形式 (拡張子 crt) となります。

- BaltimoreCyberTrustRoot.crt
- DigiCertGlobalRootG2.crt

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

本サンプルプロジェクトではルート CA 証明書を PEM 形式で扱います。このため、以下手順でダウンロードした DER 形式のルート CA 証明書を OpenSSL のコマンドで PEM 形式 (拡張子 cer) に変換してください。

- ① ルート CA 証明書のコピー  
作業フォルダにダウンロードした「BaltimoreCyberTrustRoot.crt」と「DigiCertGlobalRootG2.crt」をコピーします。
- ② ルート CA 証明書 Baltimore Cyber Trust Root の PEM 形式への変換  
Windows のコマンドラインを起動し、以下の青文字の OpenSSL のコマンドを実行します。  
`openssl x509 -in BaltimoreCyberTrustRoot.crt -inform DER -out Baltimore_CyberTrust_Root.cer -outform PEM`



```
コマンド プロンプト
Microsoft Windows [Version 10.0.19044.3448]
(c) Microsoft Corporation. All rights reserved.
C:\> cd\workspace
C:\workspace> openssl x509 -in BaltimoreCyberTrustRoot.crt -inform DER -out Baltimore_CyberTrust_Root.cer -outform PEM
C:\workspace>
```

図 2-9 OpenSSL で PEM 形式へ変換 (Baltimore)

- ③ ルート CA 証明書 DigiCert Global Root G2 の PEM 形式への変換  
Windows のコマンドラインを起動し、以下の青文字の OpenSSL のコマンドを実行します。  
`openssl x509 -in DigiCertGlobalRootG2.crt -inform DER -out DigiCertGlobalRootG2.cer -outform PEM`



```
コマンド プロンプト
Microsoft Windows [Version 10.0.19044.3448]
(c) Microsoft Corporation. All rights reserved.
C:\> cd\workspace
C:\workspace> openssl x509 -in DigiCertGlobalRootG2.crt -inform DER -out DigiCertGlobalRootG2.cer -outform PEM
C:\workspace>
```

図 2-10 OpenSSL で PEM 形式へ変換 (DigiCert)

- ④ 変換が完了したら、以下の 2 種の PEM ファイル形式のファイルが作成されます。
  - Baltimore\_CyberTrust\_Root.cer
  - DigiCertGlobalRootG2.cer

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

(b) WEB ブラウザ (Microsoft Edge) を使用して入手する場合

以下にブラウザ (Microsoft Edge) からルート CA 証明書を PEM 形式でダウンロードする方法を示します。

① Microsoft Edge の画面右上の[...]をクリックし、メニューから[設定]をクリックします。

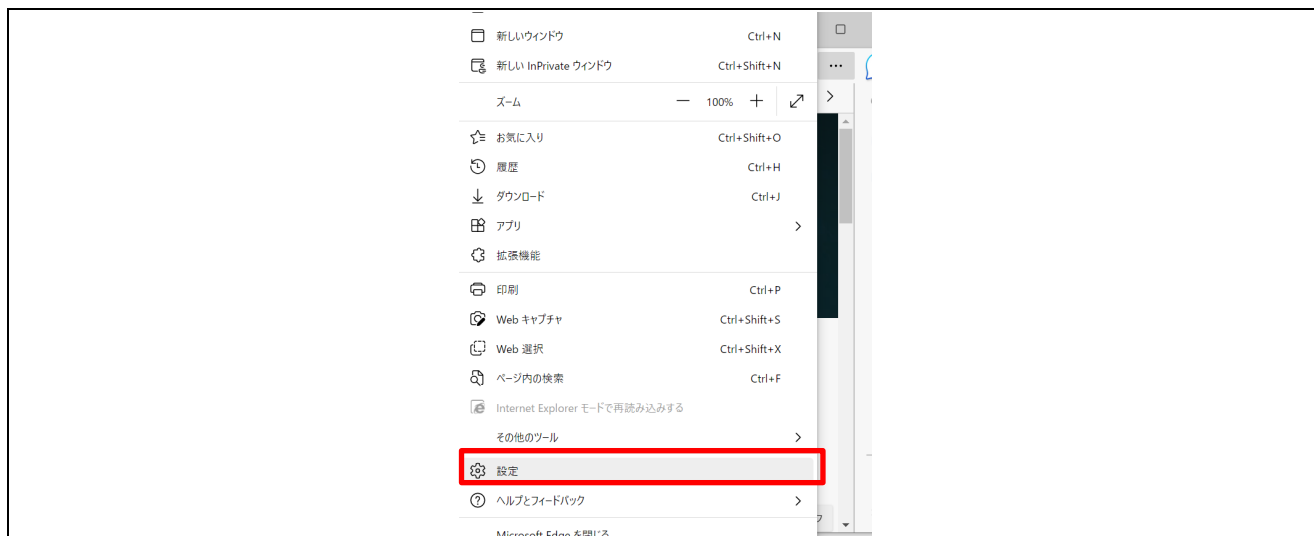


図 2-11 Microsoft Edge の設定画面

② Edge の設定メニューから[プライバシー、検索、サービス]を選択し、[セキュリティ]項目の[証明書の管理]右上のアイコンをクリックします。

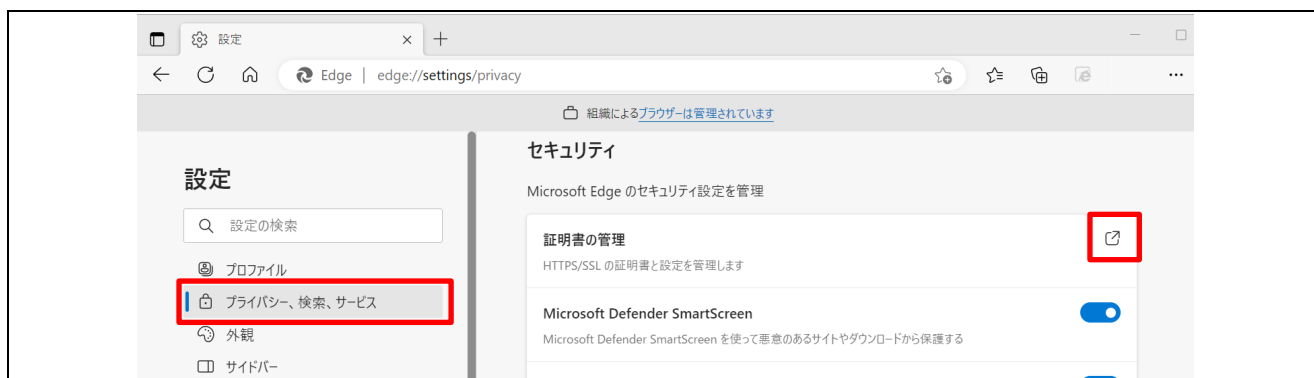


図 2-12 Edge の設定メニュー

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

- ③ 証明書画面より[信頼されたルート証明機関]のタブを選択し、証明書のリストより発行先 および 発行者が"Baltimore CyberTrust Root"のルート CA 証明書を選択して[エクスポート]ボタンをクリックします。証明書のエクスポートウィザード画面の開始画面が表示されます。

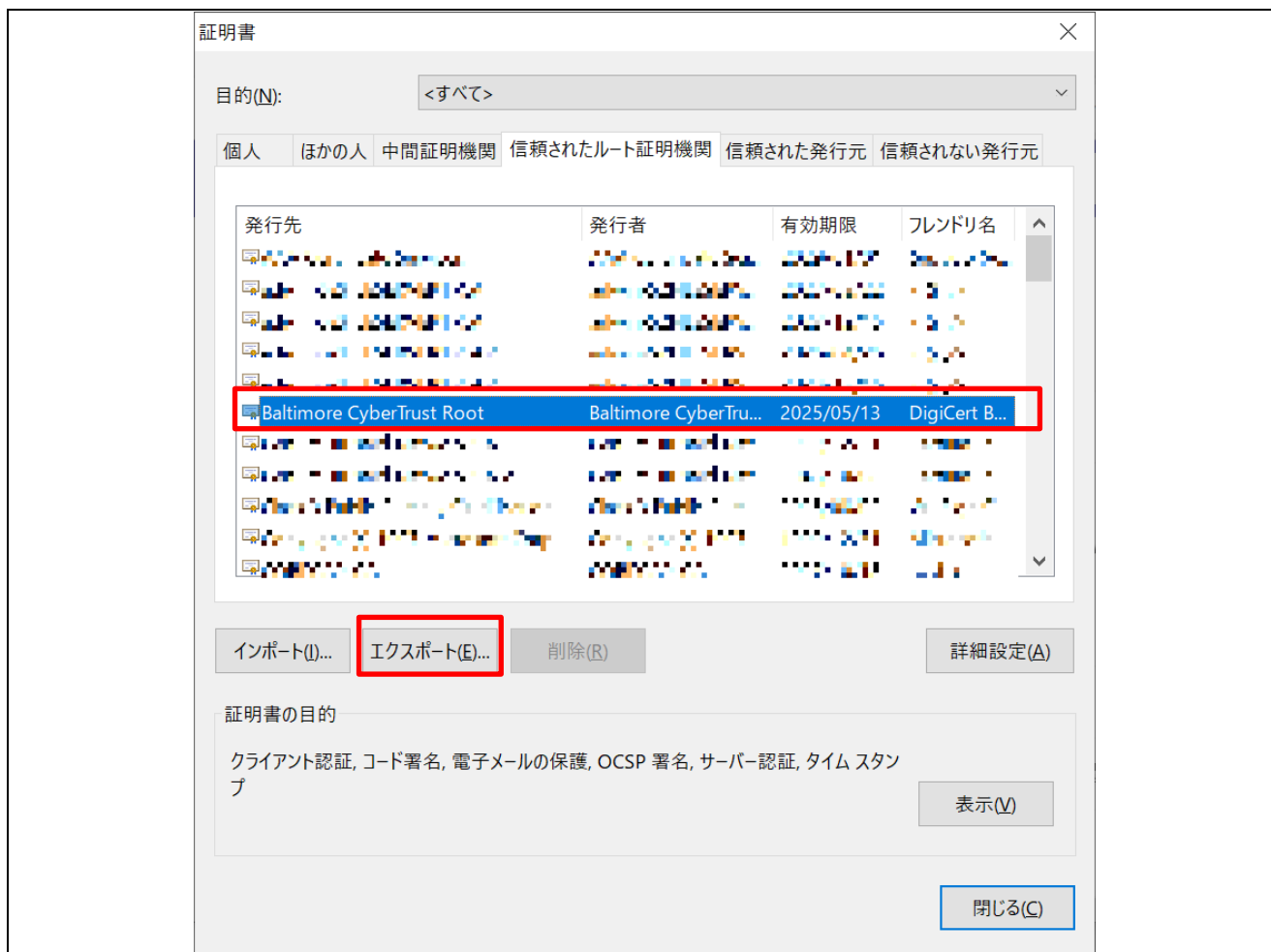


図 2-13 証明書画面 (Baltimore)



## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

- ④ 証明書のエクスポートウィザード画面の開始で[次へ]をクリックしてエクスポートファイルの形式の選択で"Base 64 encoded X.509 (.CER)(S)"を選択し、[次へ]をクリックしてください。



図 2-14 エクスポートファイルの形式

- ⑤ エクスポートするファイルの設定画面で[参照]ボタンを押し、エクスポートしたいフォルダを指定してから、ファイル名に "Baltimore\_CyberTrust\_Root.cer"と入力して[次へ]ボタンを押してください。

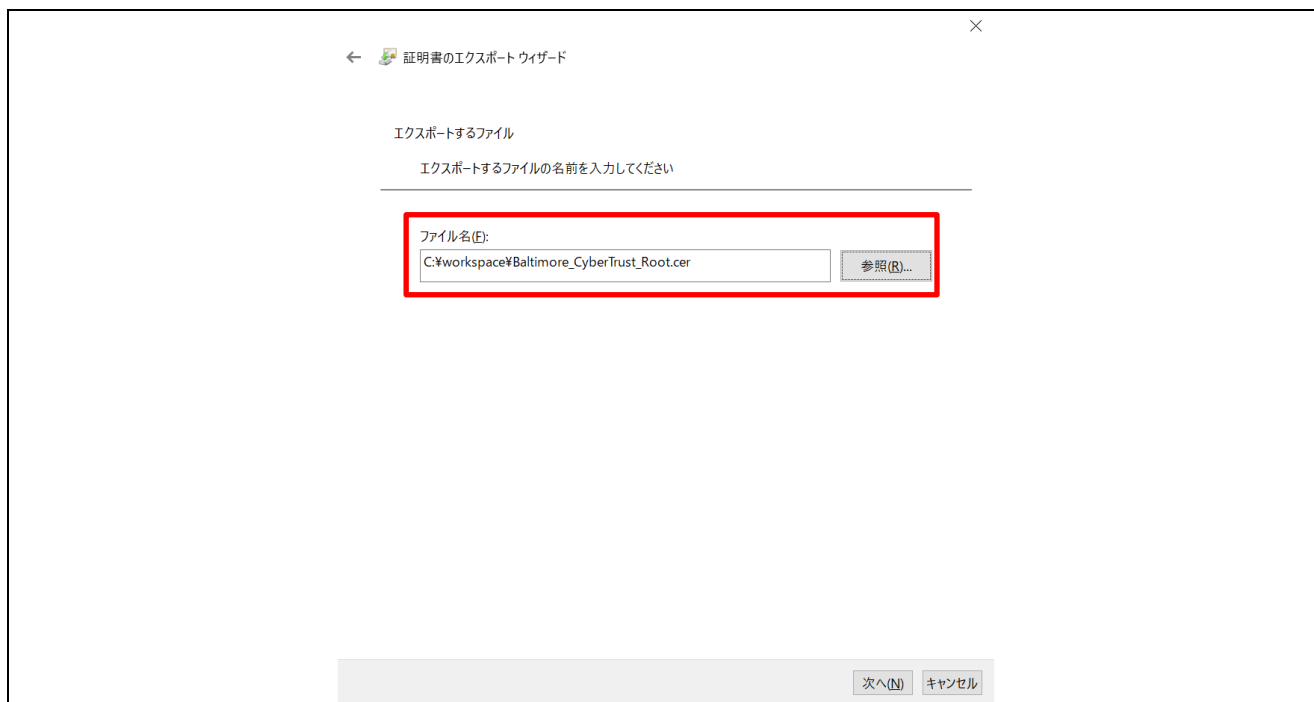


図 2-15 エクスポートファイル名の指定

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

- ⑥ 証明書のエクスポートウィザードの完了画面が表示されるので[完了]ボタンを押してください。ルート CA 証明書がエクスポートされます。

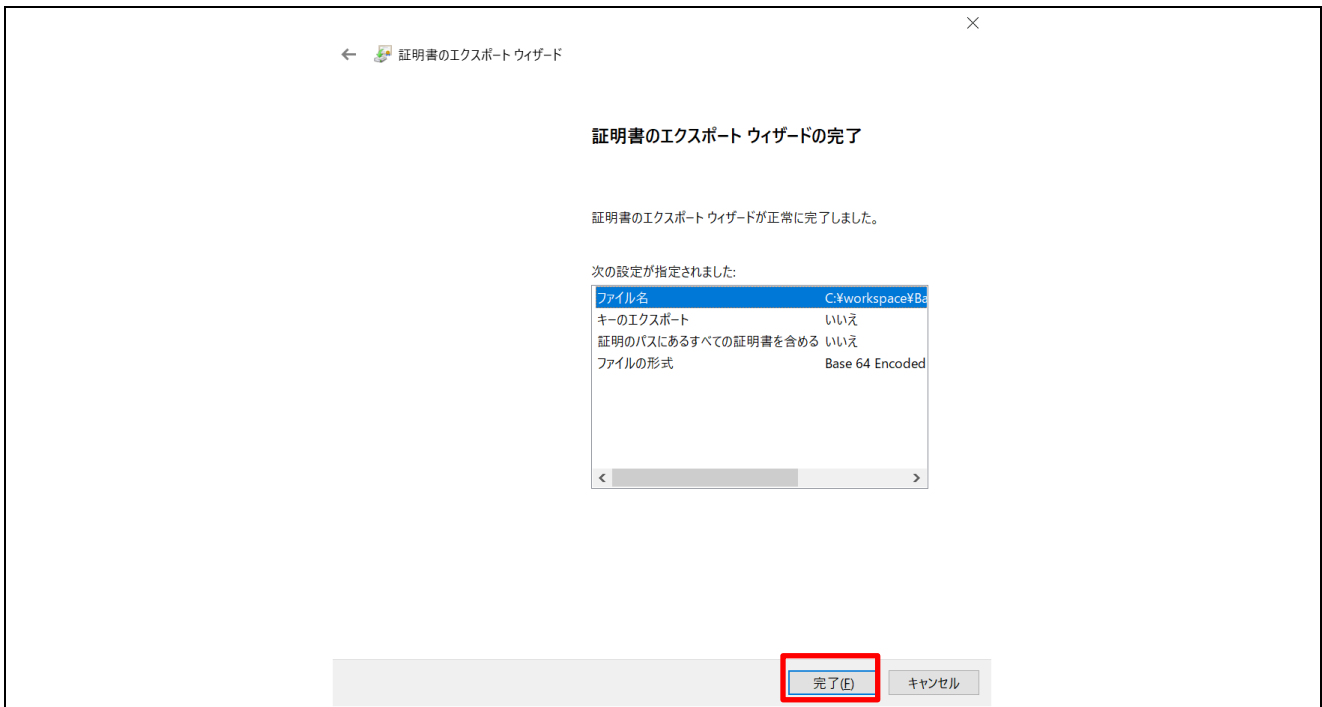


図 2-16 証明書エクスポートウィザードの完了画面

- ⑦ "DigiCert Global Root G2"のルート CA 証明書に対しても「①」～「⑥」の手順にて、エクスポートを実行してください。  
エクスポートの際にファイル名は"DigiCertGlobalRootG2.cer"と入力します。

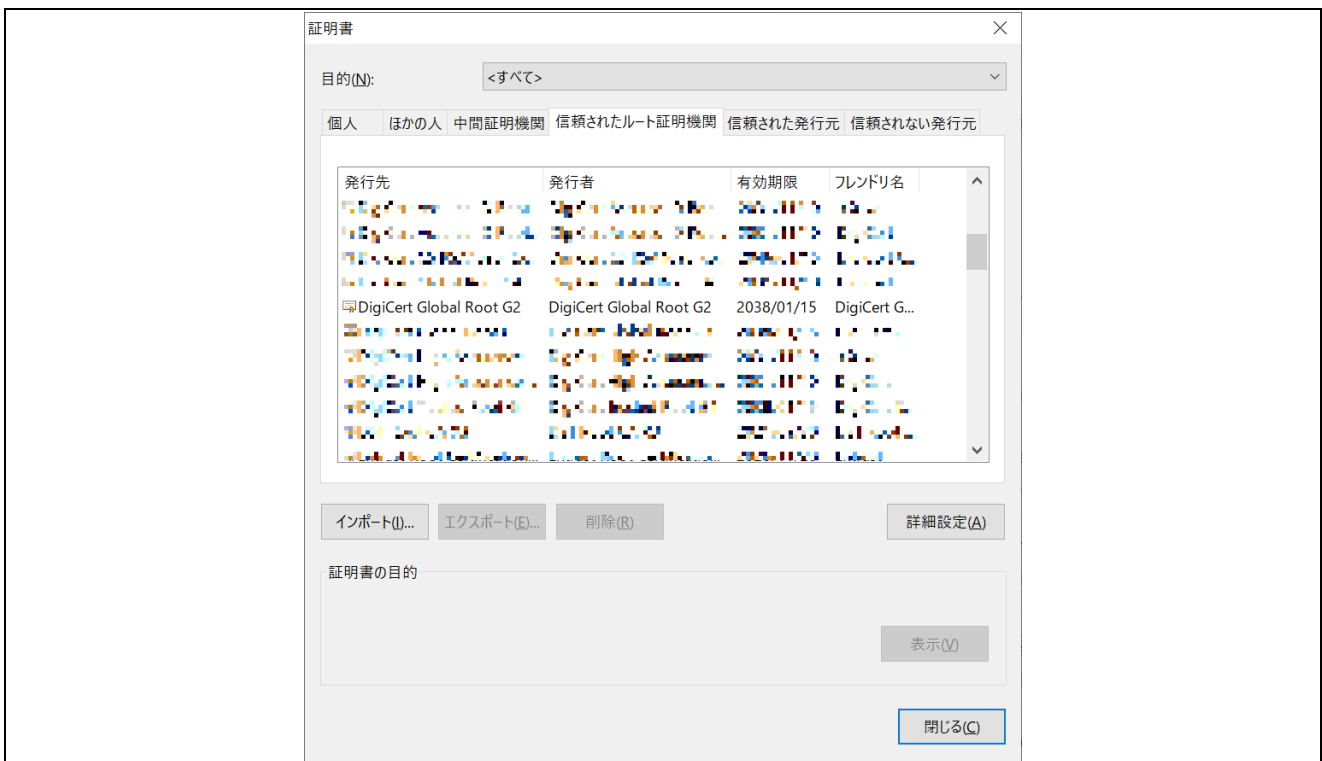


図 2-17 証明書画面 (DigiCert)

### (2) 作成したルート CA 証明書のプロジェクトフォルダへの配置

「(1)」の手順でエクスポートしたルート CA 証明書は、サンプルプロジェクトの key\_cert\_sig\_generator フォルダ下の CA フォルダに以下のように配置してください（赤字の箇所）。また、ファイル名は後述するスクリプト実行に使用するため変更しないでください。

```
key_cert_sig_generator
|-- ca
|   |-- Baltimore_CyberTrust_Root.cer
|   |-- DigiCertGlobalRootG2.cer
|-- ca-sign-keypair-rsa2048
|-- client-rsa2048
|-- output
|--1_rsa2048_convertCert.sh
|--3_showkeyValues.sh
|--convertCert.sh
```

図 2-18 ルート CA 証明書の配置

### (3) サンプルプロジェクトで使用するルート CA 証明書について

Azure IoT Hub と IoT Hub Device Provisioning Service (DPS) へ接続する際に使用するルート CA 証明書をサンプルプロジェクトに設定します。サンプルプロジェクトには、ルート CA 証明書データ登録用に以下 3 ファイルを準備しています。

- CyberTrust\_Root\_cert\_array\_1.txt
- CyberTrust\_Root\_cert\_array\_2.txt
- CyberTrust\_Root\_cert\_array\_3.txt

(2)で準備した.cer ファイルをサンプルプロジェクトにルート CA 証明書データとして使用できるように、上記ファイルに変換し、サンプルプロジェクトに登録する手順を次項より説明します。ルート CA 証明書の検証に必要な PSS 署名ファイルも次項以降で生成します。

プロジェクトには最大 3 種類のルート CA 証明書および PSS 署名ファイルが登録可能です。登録された情報を用いて自動的に有効な証明書を検証しますので、使用する証明書の設定は不要です。

※2023 年 10 月現在、Azure DPS は Baltimore CyberTrust Root、IoT Hub は DigiCert Global Root G2 による接続のみサポートしています。

また、Azure DPS は DigiCert Global Root G2 に移行予定です。Azure でのルート CA 証明書の対応状況は Microsoft の Azure サポート情報等で確認してください。





### (2) 作成した鍵と証明書のプロジェクトフォルダへの配置

「(1)」で作成した鍵ペア"device1-private.pem.key"とクライアント証明書" device1-certificate.pem.crt"をサンプルプロジェクトの key\_cert\_sig\_generator フォルダ下の client-rsa2048 フォルダに以下のように配置してください (赤文字の個所)。

また、ファイル名は後述するスクリプト実行に使用するため変更しないでください。

```
key_cert_sig_generator
|-- ca
|   |-- Baltimore_CyberTrust_Root.cer
|   |-- DigiCertGlobalRootG2.cer
|-- ca-sign-keypair-rsa2048
|-- client-rsa2048
|   |-- device1-certificate.pem.crt
|   |-- device1-private.pem.key
|-- output
|--1_rsa2048_convertCert.sh
|--3_showkeyValues.sh
|--convertCert.sh
```

図 2-23 鍵と証明書の配置

### 2.4.4 ルート CA 証明書の署名生成と証明書ファイル形式の変換

ルート CA 証明書とクライアント証明書を生成し、サンプルプロジェクト (ソースコード) へ登録を行います。

TLS で使用する証明書は一般的に PEM 形式で提供されますが、本サンプルの TSIP ドライバで使用するためには証明書を PEM 形式から DER 形式へと変換する必要があります。

以下の手順にて証明書の変換を行ってください。

これらの変換にはプロジェクトフォルダの key\_cert\_sig\_generator 内にスクリプトファイルを用意しています。スクリプトファイルはシェルスクリプト (bash) にて実行してください。

以下は cygwin を使用した例で記載します。

スクリプト実行の前に「2.4.2」項、「2.4.3」項の手順にて各鍵・証明書のファイルを所定のフォルダに配置しておいてください。

#### (1) RSA の証明書の変換

スクリプトを実行し、RSA のルート CA 証明書とクライアント証明書を DER 形式に変換します。

スクリプトではルート CA 証明書の署名生成と署名検証に使用する RSA-2048bit の鍵ペアを生成し、生成した鍵ペアの秘密鍵を使用して、ルート CA 証明書に対する署名を生成します。

変換後の各証明書は、サンプルプロジェクトのソースコードに登録できるように C 言語の配列形式に変換されます。

cygwin を実行し、サンプルプロジェクトの key\_cert\_sig\_generator フォルダへ移動してください。

以下のコマンドを入力し、スクリプトを実行してください

```
./1_rsa2048_convertCert.sh
```

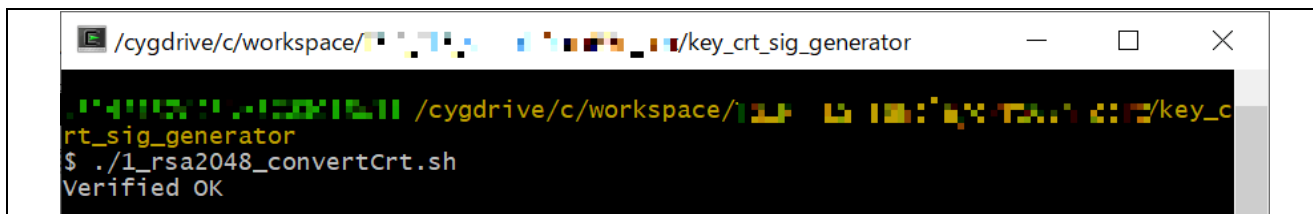


図 2-24 スクリプトの実行

### (2) 変換ファイルのプロジェクトへの登録

スクリプトの実行後、key\_crt\_sig\_generator/output フォルダに以下の 8 つのファイルが生成されます。

- Baltimore\_CyberTrust\_Root\_crt\_array.txt
- Baltimore\_CyberTrust\_Root\_sig\_array.txt
- DigiCertGlobalRootG2\_crt\_array.txt
- DigiCertGlobalRootG2\_sig\_array.txt
- client\_rsa2048\_crt\_array.txt
- Baltimore\_CyberTrust\_Root\_crt.der
- Baltimore\_CyberTrust\_Root\_sig.sig
- client\_rsa2048\_crt.der

上記の出力ファイルのうち、赤字のファイルを実験プロジェクトで使用します。これらのファイルは、生成されたバイナリデータを C 言語の uint8\_t 型配列の形式で記述したものです。ルート CA 証明書に関連する 4 ファイルのファイル名を以下のように変更してください。

表 2-3 ルート CA 証明書 ファイル名の変更

変更前ファイル名	変更後ファイル名
Baltimore_CyberTrust_Root_crt_array.txt	CyberTrust_Root_crt_array_1.txt
Baltimore_CyberTrust_Root_sig_array.txt	CyberTrust_Root_sig_array_1.txt
DigiCertGlobalRootG2_crt_array.txt	CyberTrust_Root_crt_array_2.txt
DigiCertGlobalRootG2_sig_array.txt	CyberTrust_Root_sig_array_2.txt

ファイル名を変更した 4 ファイルと client\_rsa2048\_crt\_array.txt ファイルを実験プロジェクトの sample\_azure\_iot\_embedded\_sdk/src/userdata\_tsip 内に上書き保存します。

また、上記の証明書の生成過程で、key\_crt\_sig\_generator/ca-sign-keypair-rsa2048 フォルダに以下のルート CA 証明書の署名検証用の鍵ペアと公開鍵ファイルが生成されます。鍵ペアファイル (rsa2048-private.pem) は、2.4.5 項の処理でも使用されます。

- rsa2048-private.pem : 鍵ペアファイル
- rsa2048-public.pem : 公開鍵ファイル

### 2.4.5 鍵のラップとプロジェクトへの登録

「2.4.3、2.4.4 項」で生成したルート証明書の署名検証用鍵とクライアント証明書用鍵情報をサンプルプロジェクト（ソースコード）へ登録を行います。

TSIP ドライバは、平文のユーザ鍵を入力として受け入れないため、鍵をラップして TSIP ドライバが受け入れられる形式に変換する必要があります。

TLS で使用する鍵も証明書と同様、一般に PEM 形式で提供されます。TSIP ドライバで使用するためには PEM 形式の鍵ファイルから鍵データを抽出します。その後、[Renesas DLM サーバ](#)（Renesas Key Wrap サービス）および Security Key Management Tool を使用して鍵をラップします。

Renesas DLM サーバでの鍵データのやり取りは、OpenPGP 暗号化が必要です。

本手順では OpenPGP 暗号化に Gpg4win (Kleopatra) を使用します。

手順の概要は以下のようになります。

- ① Security Key Management Tool を使用して平文の UFPK を作成
- ② Kleopatra を使用し、UFPK を PGP 暗号化する (Key Wrap でやり取りするため)
- ③ Renesas Key Wrap サービスを使用して、PGP 暗号化した UFPK をルネサスへ送信
- ④ ルネサスより暗号化された UFPK (送信用に PGP 暗号化されたもの) を受信する
- ⑤ Kleopatra を使用し PGP 暗号化を復号し、暗号化された UFPK (W-UFPK) を得る
- ⑥ Security Key Management Tool を使用してルート証明書の署名検証用鍵とクライアント証明書用鍵ペア情報を UFPK と W-UFPK でラップする。
- ⑦ ラップした暗号化鍵ファイルをソースコードに登録する。

次に詳細の手順を示します。



### 2.4.5.1 UFPK と W-UFPK の作成

任意の User Factory Programming Key (UFPK) を生成し、DLM サーバにアップロードして W-UFPK (Renesas Key Wrapping サービスでラップされた UFPK) を生成します。UFPK は署名検証用の公開鍵をラップするために使用する鍵です。

Security Key Management Tool を使用して、DLM サーバが受け付けるフォーマットの UFPK<sup>(※)</sup>ファイルを作成することができます。

(※)本項ではラップに使用する UFPK および暗号化された UFPK (W-UFPK) の生成手順を説明します。ただし、この手順で生成する鍵情報はサンプルのため実製品では使用することはできません。量産等に適用する場合は独自の鍵を生成する必要があるため、それらの詳細が書かれたアプリケーションノートを別途ご用意しています。

ルネサスマイコンをご採用/ご採用予定のお客様に提供させていただいておりますので、お取引のあるルネサスエレクトロニクス営業窓口にお問合せください。<https://www.renesas.com/contact/>

#### (1) Renesas Key Wrap サービスを登録する

DLM サーバにて暗号化を行う際、初回のみユーザ登録と Renesas DLM サーバとの OpenPGP 鍵交換が必要です。以下 URL にログインして初回登録を実施してください。

[Key Wrap サービス Login \(renesas.com\)](#)

詳細は Key Wrap サービスの FAQ に記載されています。

[KeyWrap サービス システム操作マニュアル.pdf \(renesas.com\)](#)

また、PGP 鍵の生成には OpenPGP を使用します。本書では Gpg4win と Kleopatra を使用した例で手順を説明します。上記 KeyWrap サービスマニュアルの「8.付録」の章に Gpg4win のインストールと使用方法が記載されていますのでインストールを行ってください。

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

### (2) OpenPGP 鍵ペアの作成とルネサス DLM サーバと PGP 公開鍵の交換

Renesas DLM サーバと公開鍵の交換するため、Kleopatra にて鍵ペアを作成します。

この手順は初回のみ実施します。

Kleopatra を起動したら、画面より[New Key Pair]ボタンを押下してください。証明書の作成画面が表示されるので、[詳細設定]ボタンを押下します。

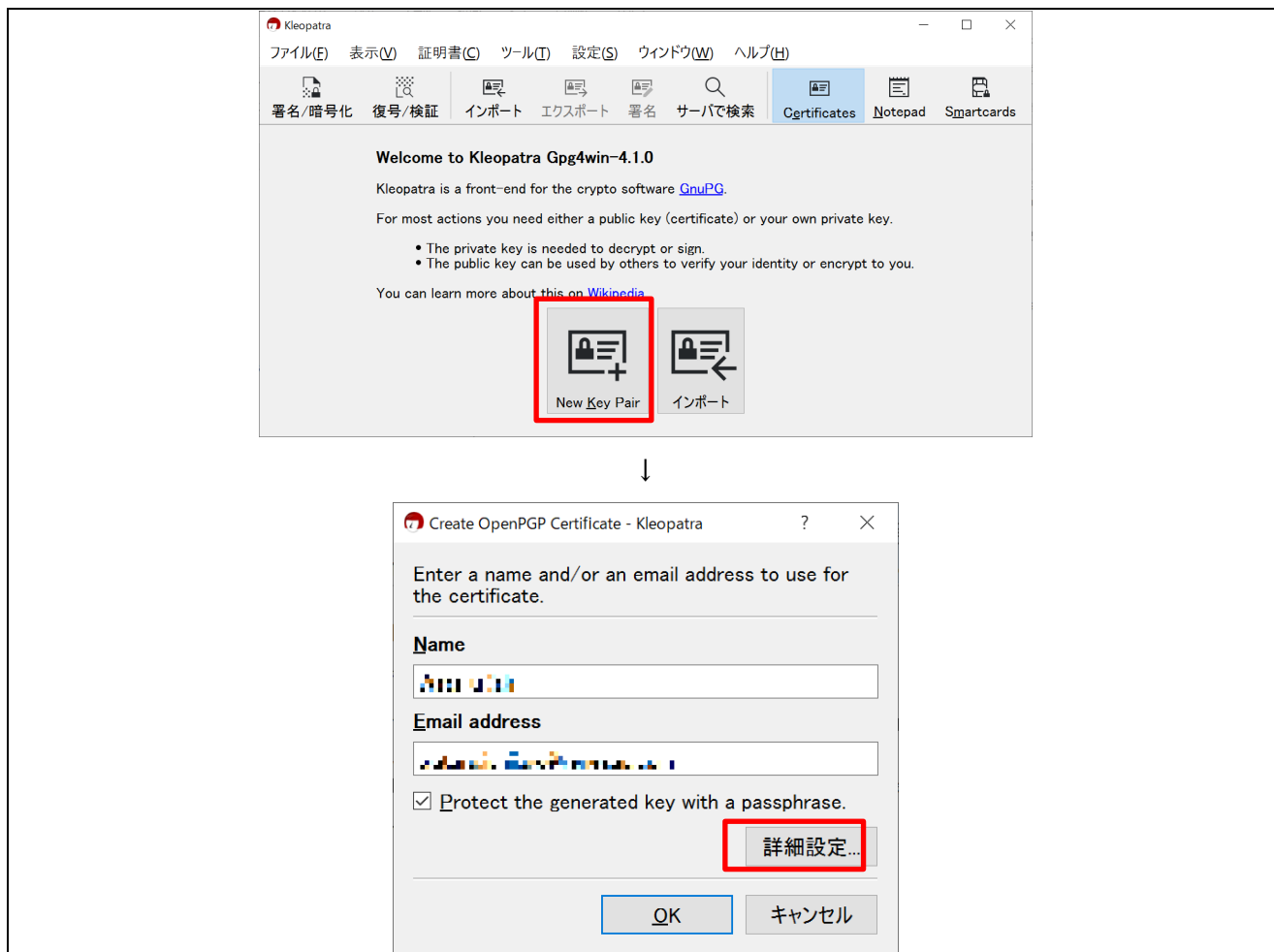


図 2-25 鍵ペアの作成画面

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

詳細設定画面で"RSA 4096 ビット"を設定し[OK]ボタンを押下します。  
Renesas DLM サーバでは RSA 鍵のみ交換可能です。

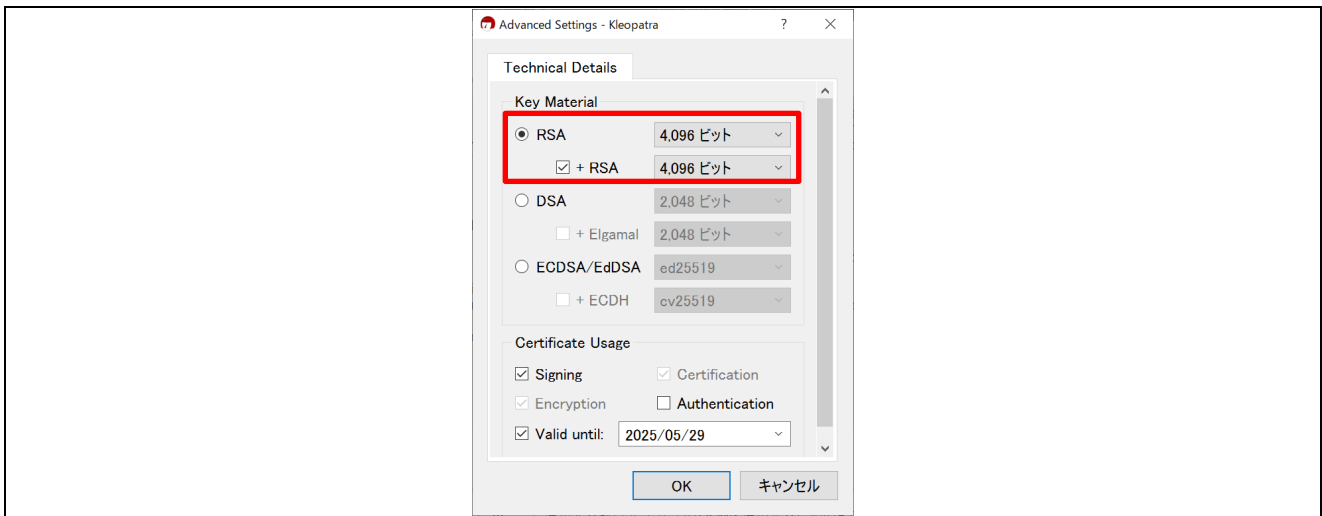


図 2-26 鍵の詳細設定画面

鍵ペアの作成画面で、"名前"と"メールアドレス"を入力し、[OK]ボタンを押してください。  
"Protect the generated key with a passphrase"にチェックを入れパスワードでセキュリティを強化することもできます。設定した場合はパスワードを忘れないようにしてください。

鍵ペアが作成出来たら、以下のように Kleopatra の画面に鍵ペアの情報が登録されます。  
登録した鍵ペアを選択し、[エクスポート]ボタンを押下して、OpenPGP 公開鍵を出力してください。ファイル名は、「～.asc」となります。

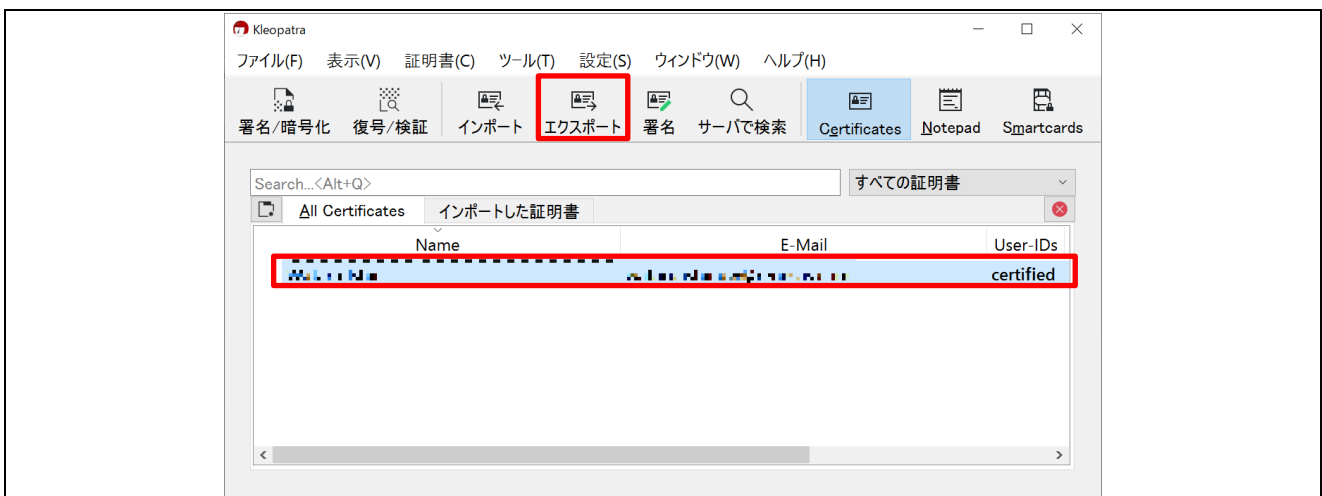


図 2-27 自分の PGP 公開鍵の出力

ルネサス Key Wrap サービスの WEB サイトより[PGP 鍵交換]をクリックし、作成した OpenPGP 公開鍵を登録します。  
正常に登録が完了すると、メールでルネサスの PGP 公開鍵が送られてくるので保存してください。

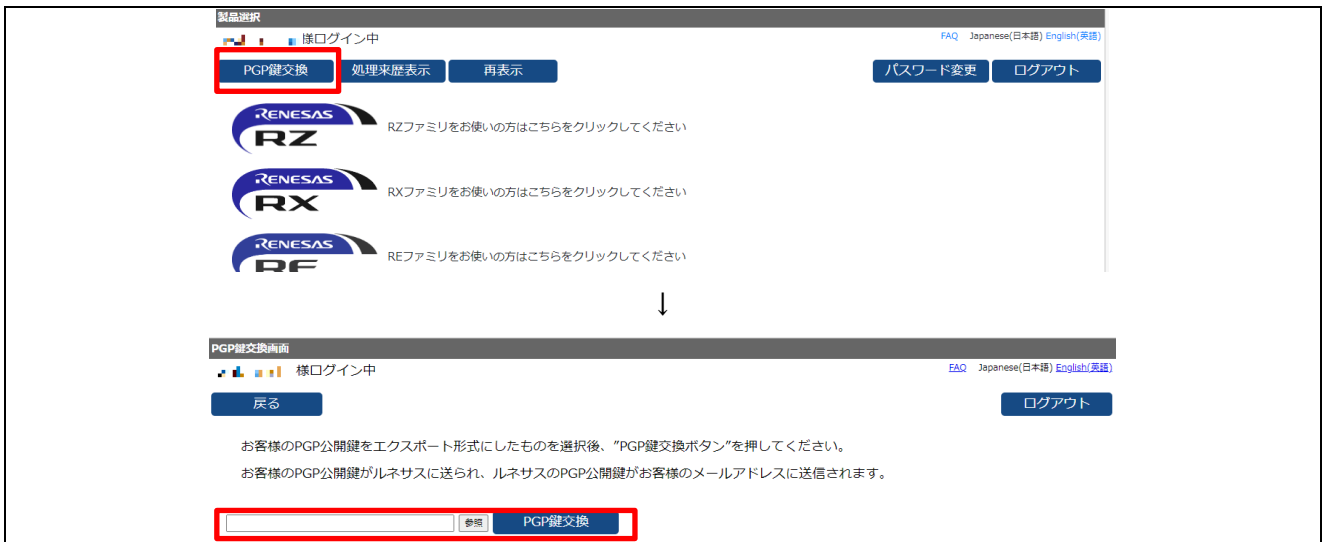


図 2-28 PGP 公開鍵の交換

(3) ルネサスの OpenPGP 公開鍵の登録

ルネサス PGP 公開鍵は、DLM サーバで PGP 暗号化された鍵を復号するのに使用します。Kleopatra にメールで受信したルネサスの PGP 公開鍵を登録します。Kleopatra のメニューより[インポート]を押下します。

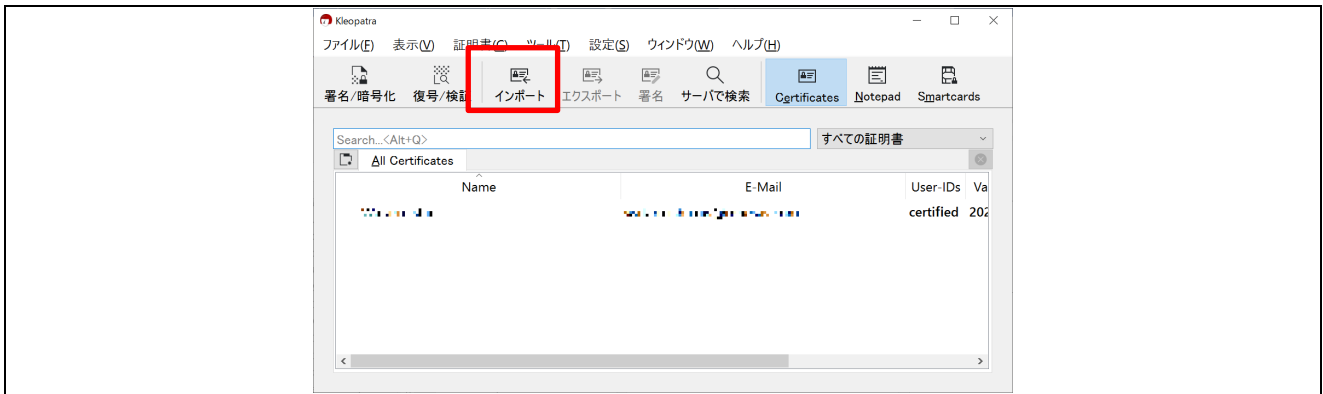


図 2-29 ルネサス PGP 公開鍵のインポート

インポートするファイル選択画面が表示されるので、ファイルの拡張子を"任意のファイル(\*)"に設定し、ルネサスより送られてきた PGP 公開鍵"keywrap-pub.key"を指定して、[開く]ボタンを押下してください。

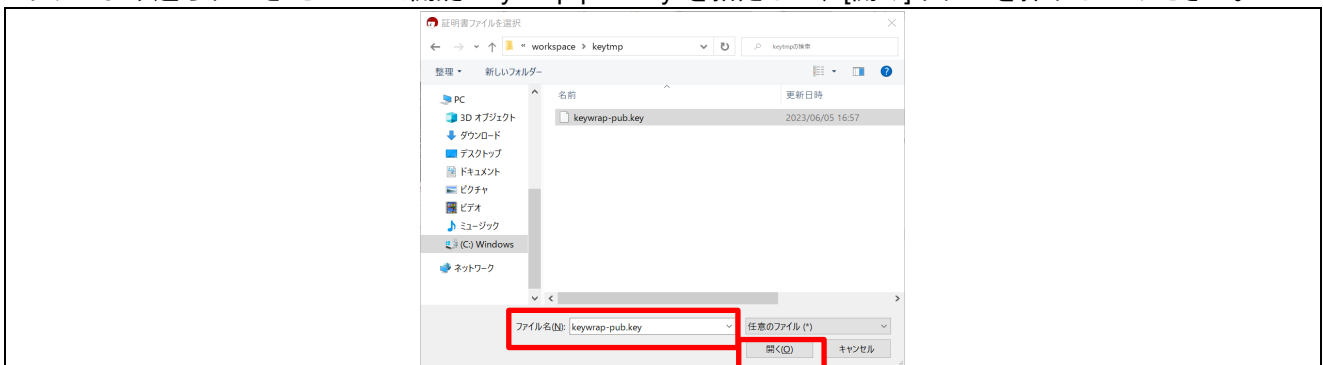


図 2-30 インポートするファイルの選択

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

インポートが完了したら[OK]ボタンを押下してインポート画面を閉じてください。  
インポートの際にファイルに署名/暗号化の確認画面が出た場合は登録済みの鍵ペアを選択してください。

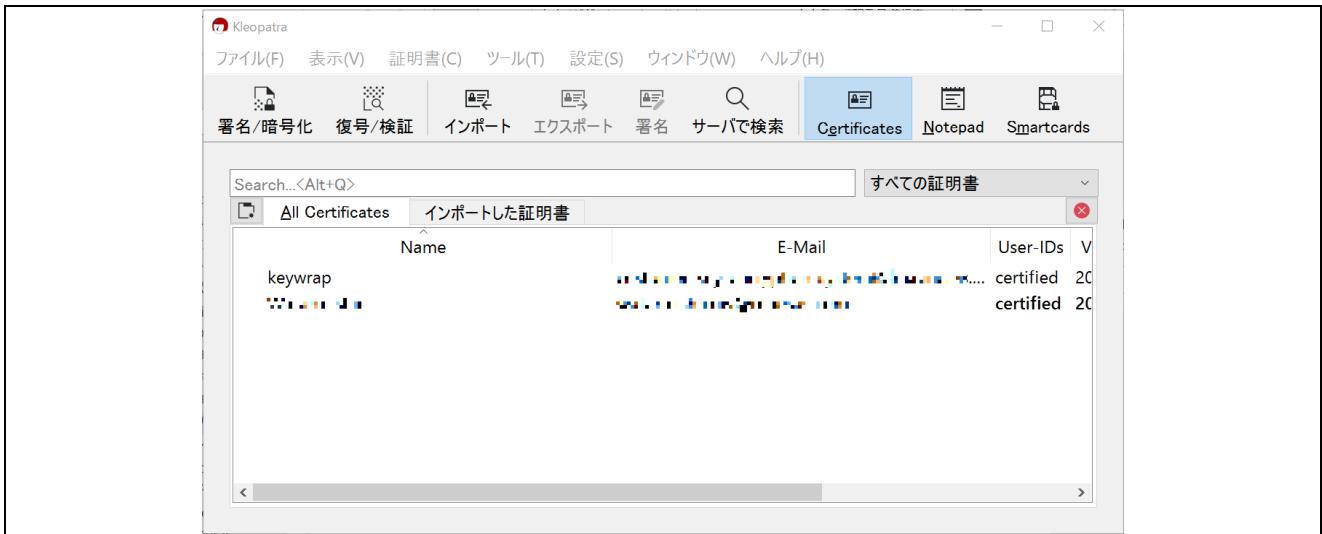


図 2-31 ルネサス PGP 公開鍵のインポート状況

### (4) UFPK (平文) ファイルの生成

Security Key Management Tool を使用して平文の UFPK を作成します。

以下 web ページの[ダウンロード] から最新版の Security Key Management Tool をダウンロードします。ダウンロード完了後、.zip ファイルを解凍し、格納されている.exe ファイルを実行してインストールしてください。

[Security Key Management Tool | Renesas](#)

インストールした Security Key Management Tool を起動し、[概要]タブをクリックして、“RX Family, TSIP”を選択します。

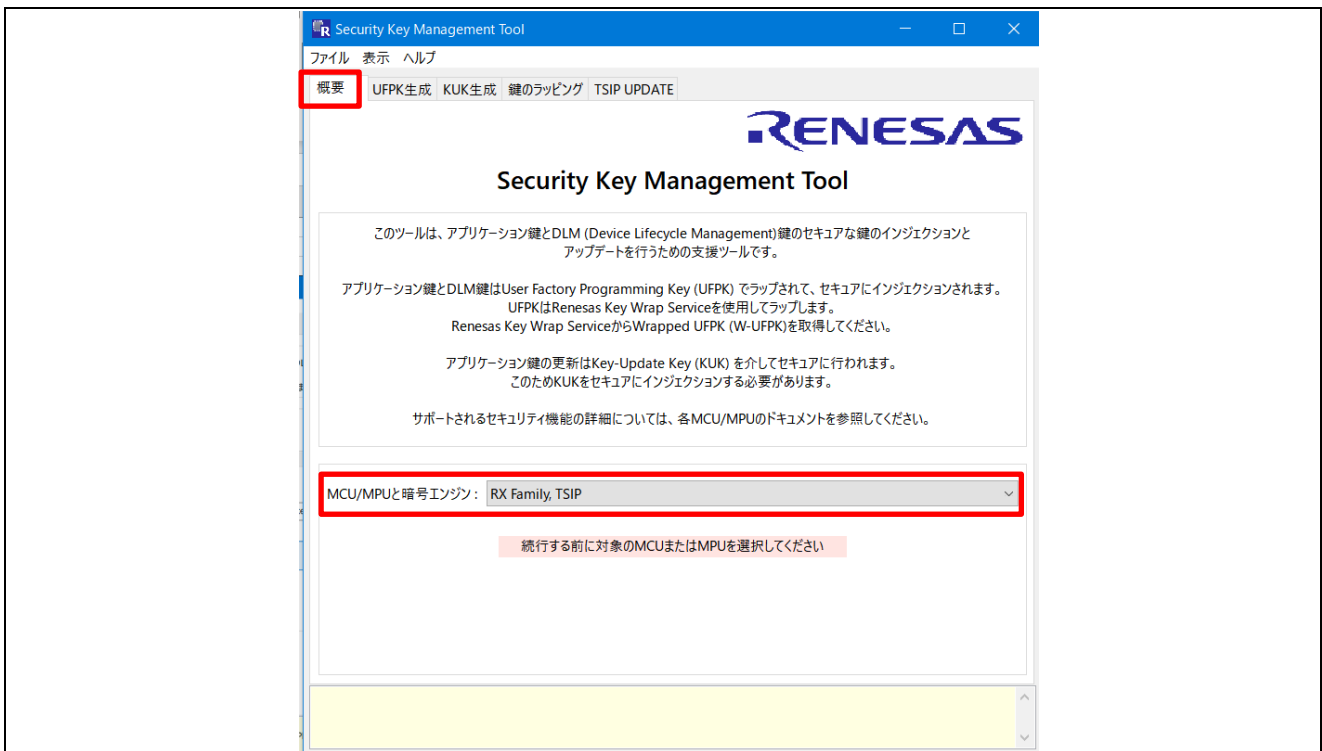


図 2-32 MCU の選択

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

Security Key Management Tool アプリの[UFPK 生成]タブをクリックして、[乱数生成機能を使用する]を選択します。 [出力ファイル(.key) : ]に出力する UFPK ファイルのパスおよびファイル名を設定します。ファイル名を" sample.key"と設定し、[UFPK ファイルを生成する]をクリックして任意のフォルダに UFPK ファイルを出力してください。

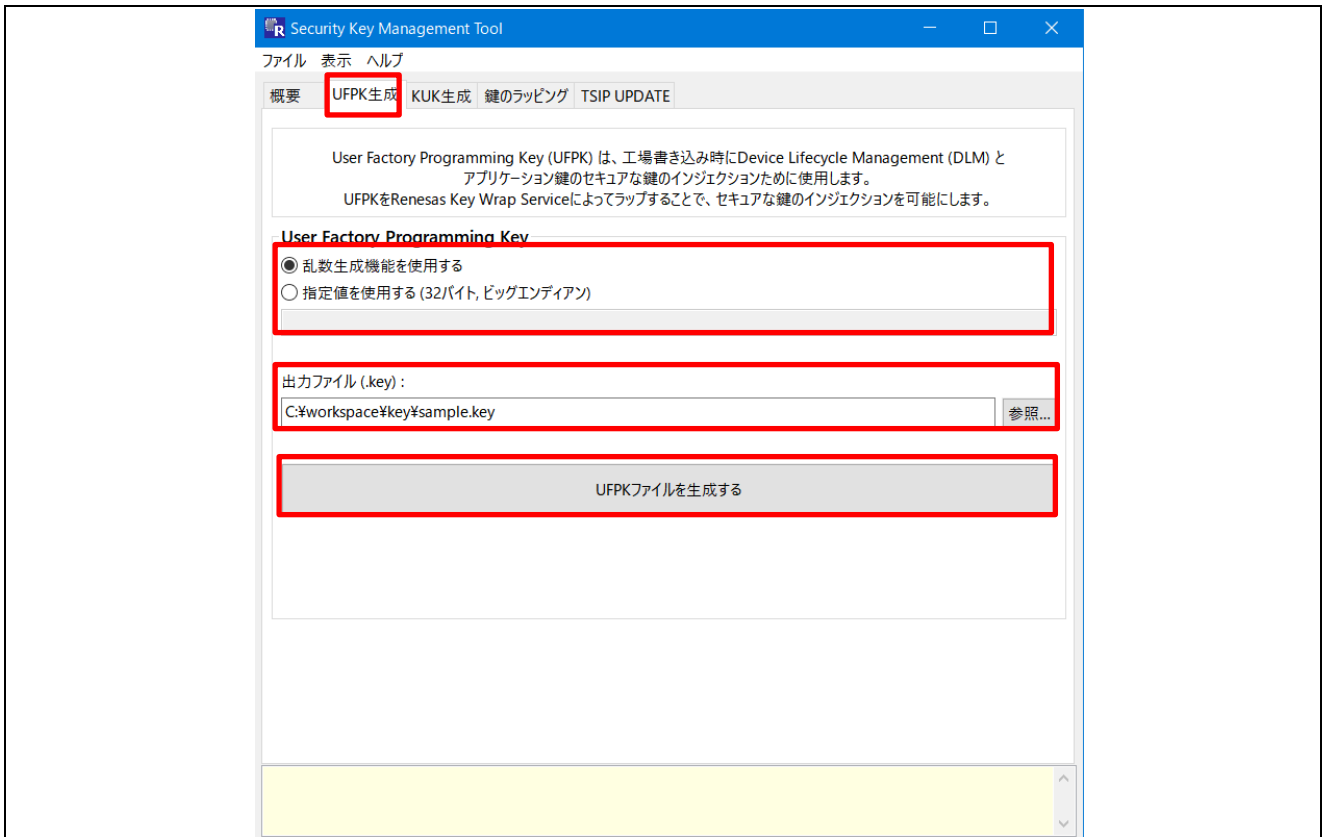


図 2-33 UFPK の生成

### (5) UFPK の PGP 暗号化

Kleopatra を使用し、「(4)」で作成した UFPK (sample.key) を作成済みの OpenPGP 鍵ペアとルネサスの PGP 公開鍵で PGP 暗号化します。

Kleopatra の画面で、[署名/暗号化]ボタンを押下して、ファイルの選択で"sample.key"を指定してください。ファイルに署名/暗号化の確認画面が出るので、署名と暗号化に作成済みの鍵ペアを指定します。

- Sign as : 自分の鍵ペアを指定
- Encrypt for me : 自分の鍵ペアを指定
- Encrypt for others : ルネサス PGP 公開鍵を指定

出力先フォルダを設定したら[sign/Encrypt]ボタンを押下してください。指定したフォルダに PGP 暗号化された UFPK "sample.key.pgp"ファイルが作成されます。

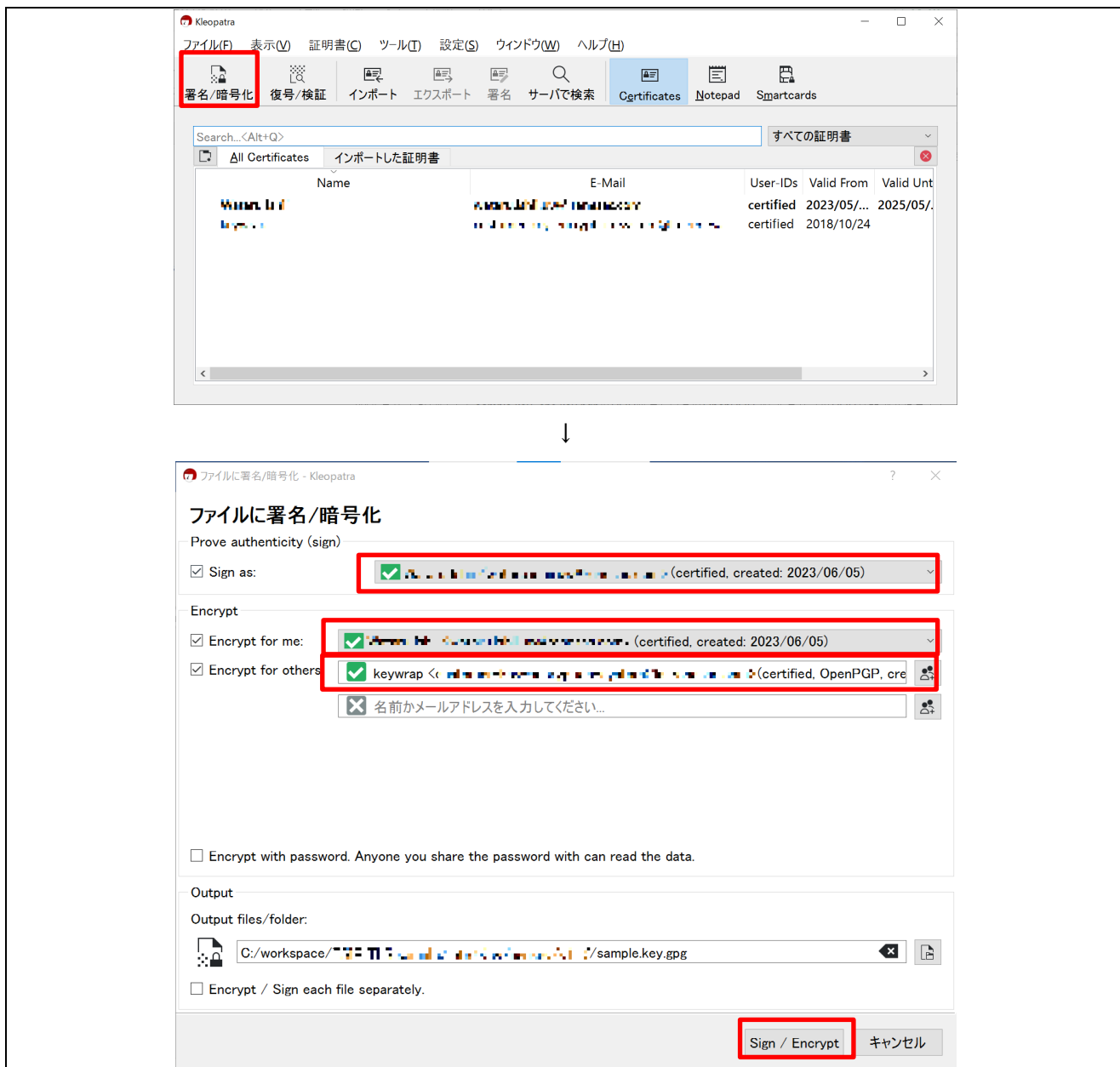


図 2-34 sample.key を PGP 暗号化

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

### (6) PGP 暗号化した UFPK を DLM サーバで暗号化

次にルネサス [Key Wrap サービスの Web サイト](#) を使用して PGP 暗号化した UFPK を DLM サーバにアップロードし暗号化を行います。

Key Wrap サービスの Web サイトより、[RENEASAS RX]⇒[RX65N/RX651 お客様データ暗号化]<sup>(※)</sup>⇒[製品用暗号化サービス]のリンクをそれぞれクリックします。

アップロード画面が表示されるので[参照]ボタンを押下して、「(4)」で作成した"sample.key.pgp"を指定し、[確定]ボタンを押下します。

(※) RX72N Envision Kit をご利用の場合は、[RX72N お客様データ暗号化]をクリックしてください。

The image shows a sequence of four screenshots from the Renesas Key Wrap service website, illustrating the steps to upload UFPK. Each screenshot is connected to the next by a downward arrow.

- Screenshot 1: 製品選択 (Product Selection)**
  - Header: 製品選択, 様ログイン中, Language options (FAQ, Japanese, English).
  - Buttons: PGP鍵交換, 処理履歴表示, 再表示, パスワード変更, ログアウト.
  - Options: RZ, **RX** (highlighted with a red box), RE, RA.
  - Footer: Renesas Synergy™
- Screenshot 2: Key Wrap サービス画面 (Key Wrap Service Page)**
  - Header: Key Wrap サービス画面, 様ログイン中, Language options.
  - Buttons: PGP鍵交換, 処理履歴表示, 再表示, 戻る, パスワード変更, ログアウト.
  - Options: RX231, **RX65N/RX651** (highlighted with a red box), RX23W.
- Screenshot 3: RX65N/RX651 処理画面 (RX65N/RX651 Processing Page)**
  - Header: RX65N/RX651 処理画面, 様ログイン中, Language options.
  - Buttons: 戻る, ログアウト.
  - Section: RX65N/RX651 顧客データ選択画面.
  - Option: **製品用暗号化サービス** (highlighted with a red box).
  - Text: お客様で作成されたKey2 (顧客鍵) をHRK-1 (RX65Nチップ内蔵鍵) で暗号化後、お客様に返信します。 TSIPドライバはV1.08以降をご使用ください。 V1.07以前のバージョンをご利用の場合は、TSIPドライバ配布元までお問い合わせ下さい。
- Screenshot 4: RX65N/RX651 製品用暗号化サービス (RX65N/RX651 Product Encryption Service Page)**
  - Header: RX65N/RX651 製品用暗号化サービス, 様ログイン中, Language options.
  - Buttons: メニューに戻る, 戻る, ログアウト.
  - Text: お客様作成のKey2 (顧客鍵) を弊社PGP公開鍵で暗号化してください。 暗号化されたファイルを参照ボタンで指定後、確定ボタンを押してください。 (例) xxxx.pgp 等
  - Buttons: 参照 (highlighted with a red box), 確定 (highlighted with a red box).

図 2-35 UFPK のアップロード



## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

アップロードが完了すると以下の画面が表示され、Renesas DLM サーバにて暗号化が実行されます。暗号化が完了すると、ルネサスよりメールにて"sample.key\_enc.key.pgp"が送られてきますので、任意のフォルダに保存してください。

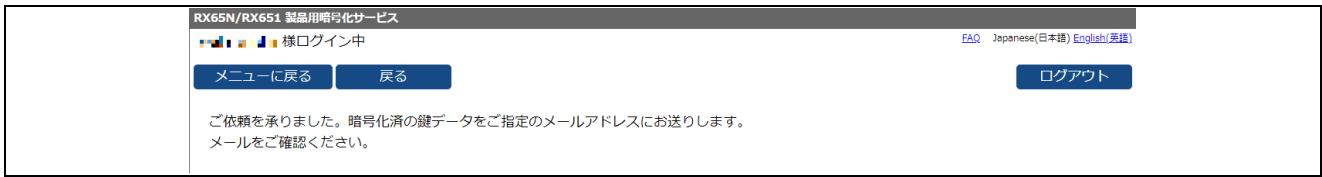


図 2-36 DLM サーバへアップロード完了

### (7) sample.key\_enc.key.pgp の OpenPGP 復号

ルネサスより送られてきた sample.key\_enc.key.pgp を自身の OpenPGP 鍵で復号して、暗号化された UFPK (W-UFPK) を作成します。

Kleopatra で[復号/検証]ボタンを押下して、ファイルの選択で" sample.key\_enc.key.pgp "を選択してください。復号が開始されます。

復号が完了したら完了画面が表示されるので[すべて保存]ボタンを押下して、復号した鍵を保存してください。同じフォルダに復号された、" sample.key\_enc.key "が出力されます。

以上で、sample.key の暗号化は完了です。

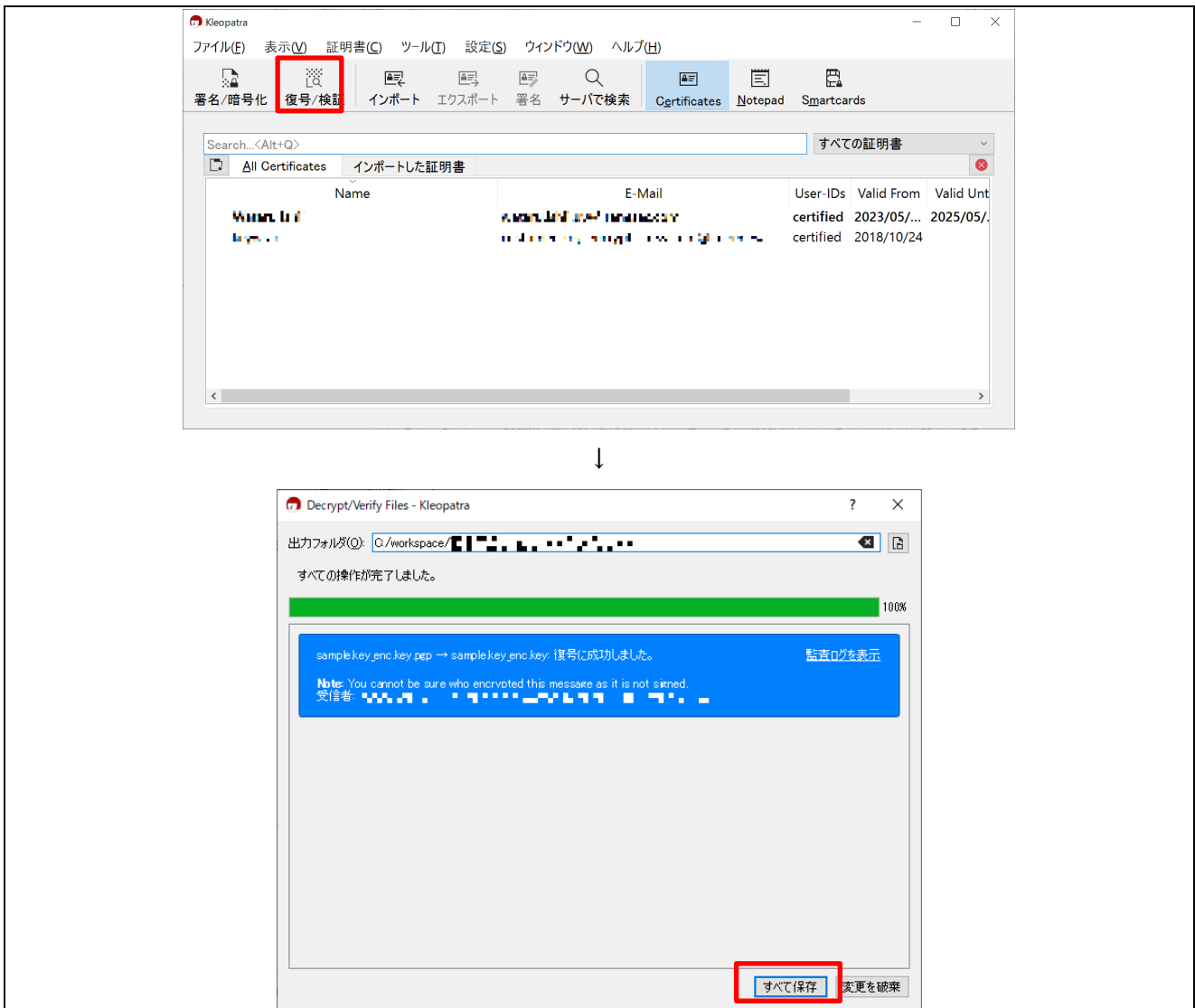


図 2-37 W-UFPK の PGP 復号

### 2.4.5.2 鍵データのラップ

ルート CA 証明書の署名検証用鍵データおよびクライアント証明書の鍵データを「2.4.5.1 UFPK と W-UFPK の作成」で作成した sample.key と sample.key\_enc.key を使用してラップし、プロジェクトファイルへ登録するデータへ変換します。

#### (1) 鍵データの抽出

「2.4.3 RSA の鍵ペアとクライアント証明書の生成」・「2.4.4 ルート CA 証明書の署名生成と証明書ファイル形式の変換」で作成した PEM 形式の鍵ファイルから、ルート CA 証明書の署名検証用鍵データとクライアント証明書の鍵データを抽出します。ここでは、以下の 2 つの鍵ファイルを使用します。

- ルート CA 署名検証用公開鍵ファイル(PEM)  
/key crt sig\_generator /ca-sign-keypair-rsa2048 /rsa2048-public.pem
- クライアント証明書用鍵ペアファイル(PEM)  
/key crt sig\_generator /client-rsa2048 /device1-private.pem.key

#### (2) 鍵データのラッピングと暗号化鍵ファイルの出力

「2.4.5.1」項、「(1)」項で作成した以下の 4 つの情報を Security Key Management Tool に入力し、暗号化鍵ファイルを生成します。

この暗号化鍵ファイルには、UFPK と W-UFPK でラップされた鍵 (encrypted key) としてプロジェクトファイルに組み込み可能なソースコードとして出力されます。

- UFPK (sample.key)
- W-UFPK (sample.key\_enc.key)
- ルート CA 証明書の署名検証用公開鍵ファイル
- クライアント証明書用鍵ペアファイル

次に、暗号化鍵ファイルの生成手順を示します。

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

- ① インストールしている"Security Key Management Tool "を起動します。  
Security Key Management Tool アプリの[鍵のラッピング]タブをクリックして、[鍵の種類]タブで[RSA]にチェックを入れ、"2048bits, public"を選択してください。

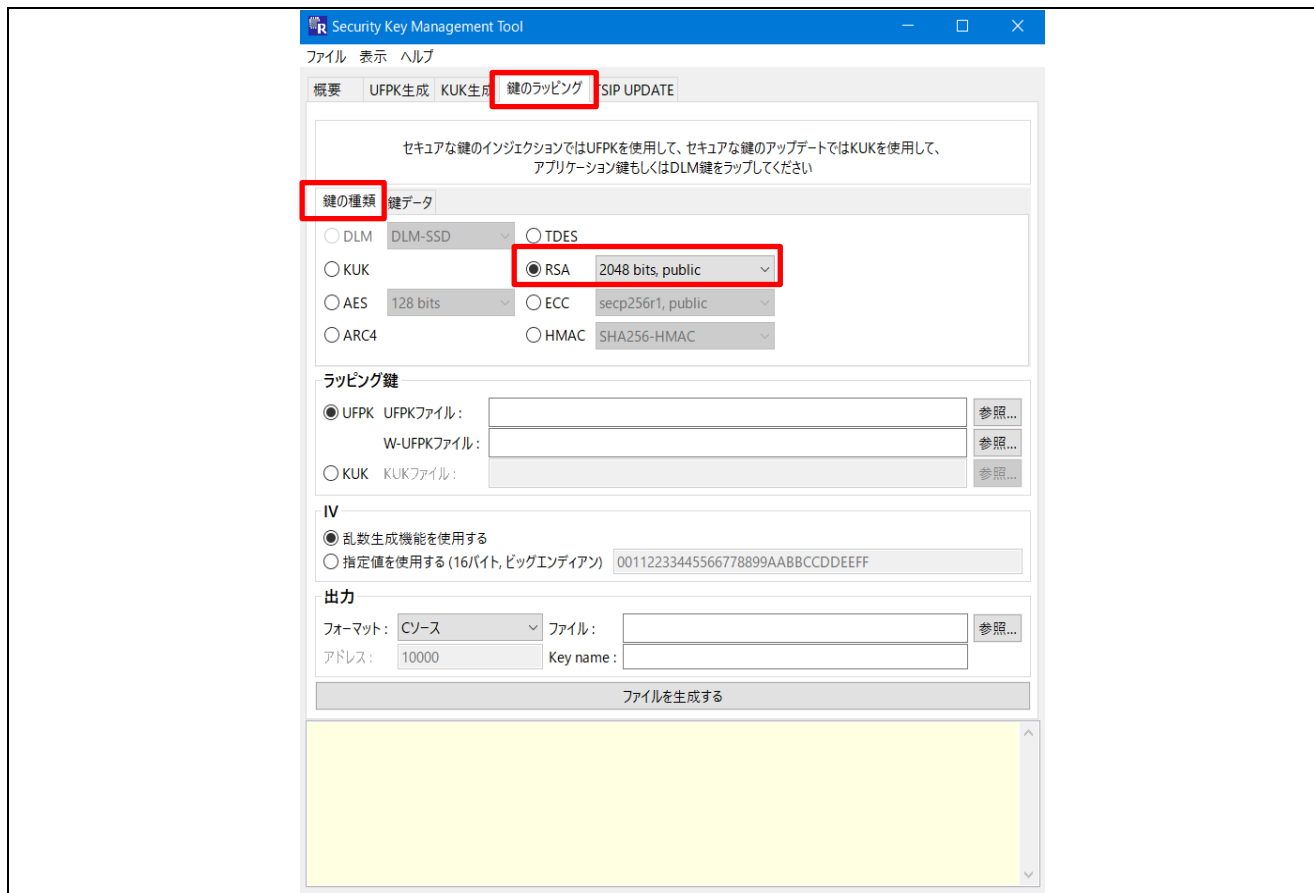


図 2-38 Security Key Management Tool を使用した公開鍵のラッピング

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

- ② UFPK と W-UFPK の登録を行います。  
[鍵データ]タブのラッピング鍵欄で[UFPK]を選択し、"UFPK ファイル"と"W-UFPK ファイル" の[参照] ボタンをそれぞれ押下して、「2.4.5.1」項で作成した UFPK (sample.key) と W-UFPK (sample.key\_enc.key) を指定します。  
IV 値は[乱数生成機能を使用する]を選択してください。ランダムな値が生成されます。

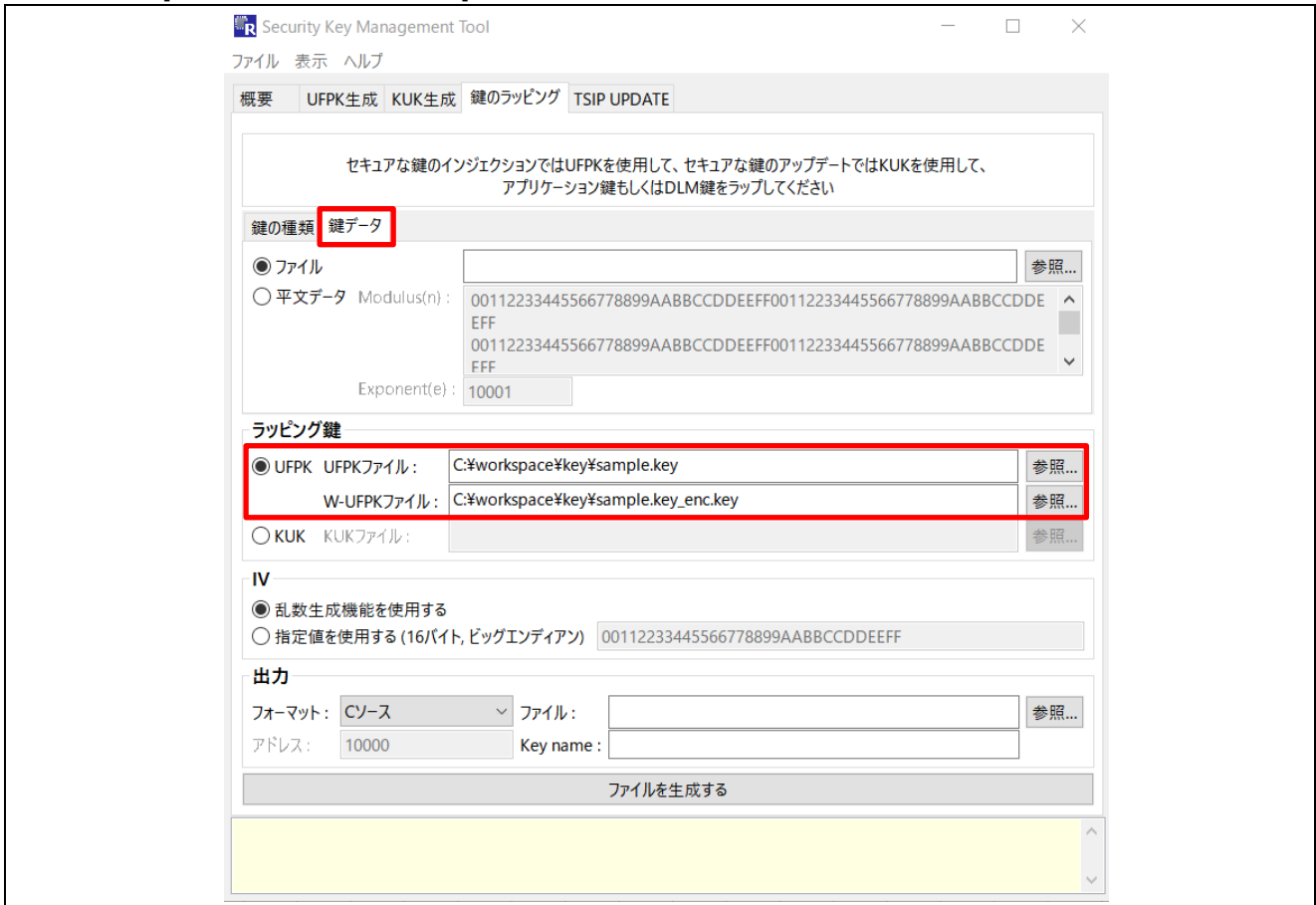


図 2-39 UFPK と W-UFPK の指定

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

- ③ ルート CA 証明書の署名検証用公開鍵データを生成します。  
[鍵データ] で [ファイル] を選択し、[参照] ボタンを押下して、[PEM 鍵データ (\*.pem)] を表示する設定とし、ルート CA 署名検証用鍵ペアファイル (PEM) の /key crt\_sig\_generator/ca-sign-keypair-rsa2048/rsa2048-public.pem を選択します。  
出力欄の "ファイル" に、ファイル名を "encrypted\_user\_rsa2048\_ne\_key" として任意のフォルダを選択、Key name も同じく "encrypted\_user\_rsa2048\_ne\_key" と入力し、[ファイルを生成する] をクリックして、ルート CA 証明書の署名検証用の公開鍵データを生成します。  
ファイル名 "encrypted\_user\_rsa2048\_ne\_key" および Key name はソースコード内で使用しているため、名称は変更しないようにしてください。

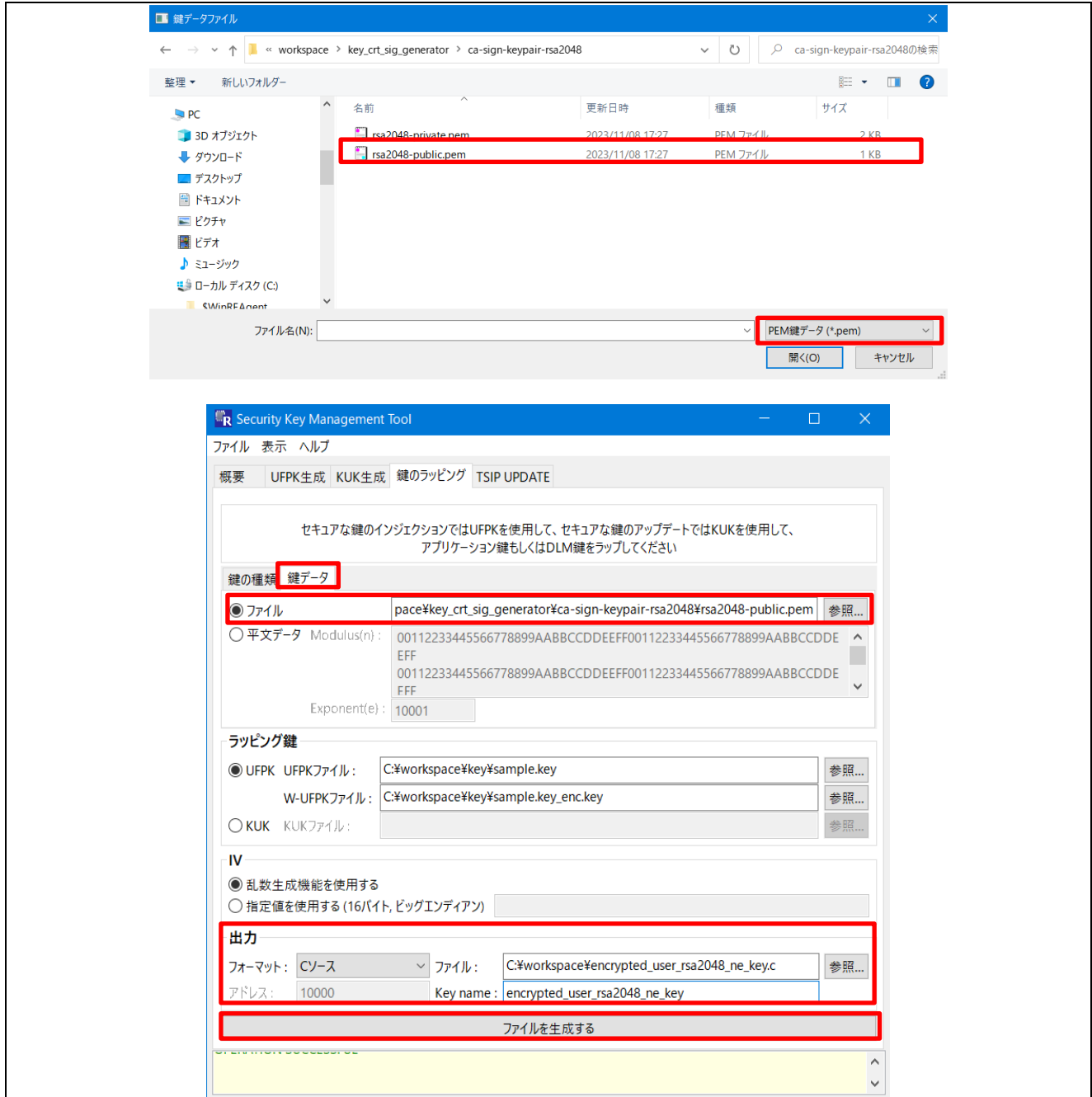


図 2-40 ルート CA 証明書の署名検証用公開鍵データの生成

ツール画面下のコンソール部分に "OPERATION SUCCESSFUL" と表示されたら生成は完了です。

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

### ④ クライアント証明書の公開鍵を生成します。

あらかじめ、/key crt\_sig\_generator /client-rsa2048/device1-private.pem.key を “device1-private.pem” にリネームしておいてください。[鍵の種類]タブに移動し、①と同様に[RSA] の"2048bits, public"が選択されていることを確認します。

再び[鍵データ]タブに移動して[ファイル]を選択し、[参照]ボタンを押下して、[PEM 鍵データ (\*.pem)]を表示する設定とし、先ほどリネームしたクライアント証明書用鍵ペアファイル(PEM) の device1-private.pem を選択します。

出力欄の”ファイル”に、ファイル名を"encrypted\_user\_rsa2048\_ne\_key2"として任意のフォルダを選択、Key name も同じく"encrypted\_user\_rsa2048\_ne\_key2"と入力し、[ファイルを生成する]をクリックして、クライアント証明書の公開鍵データを生成します。

ファイル名"encrypted\_user\_rsa2048\_ne\_key2"および Key name はソースコード内で使用しているため、名称は変更しないようにしてください。

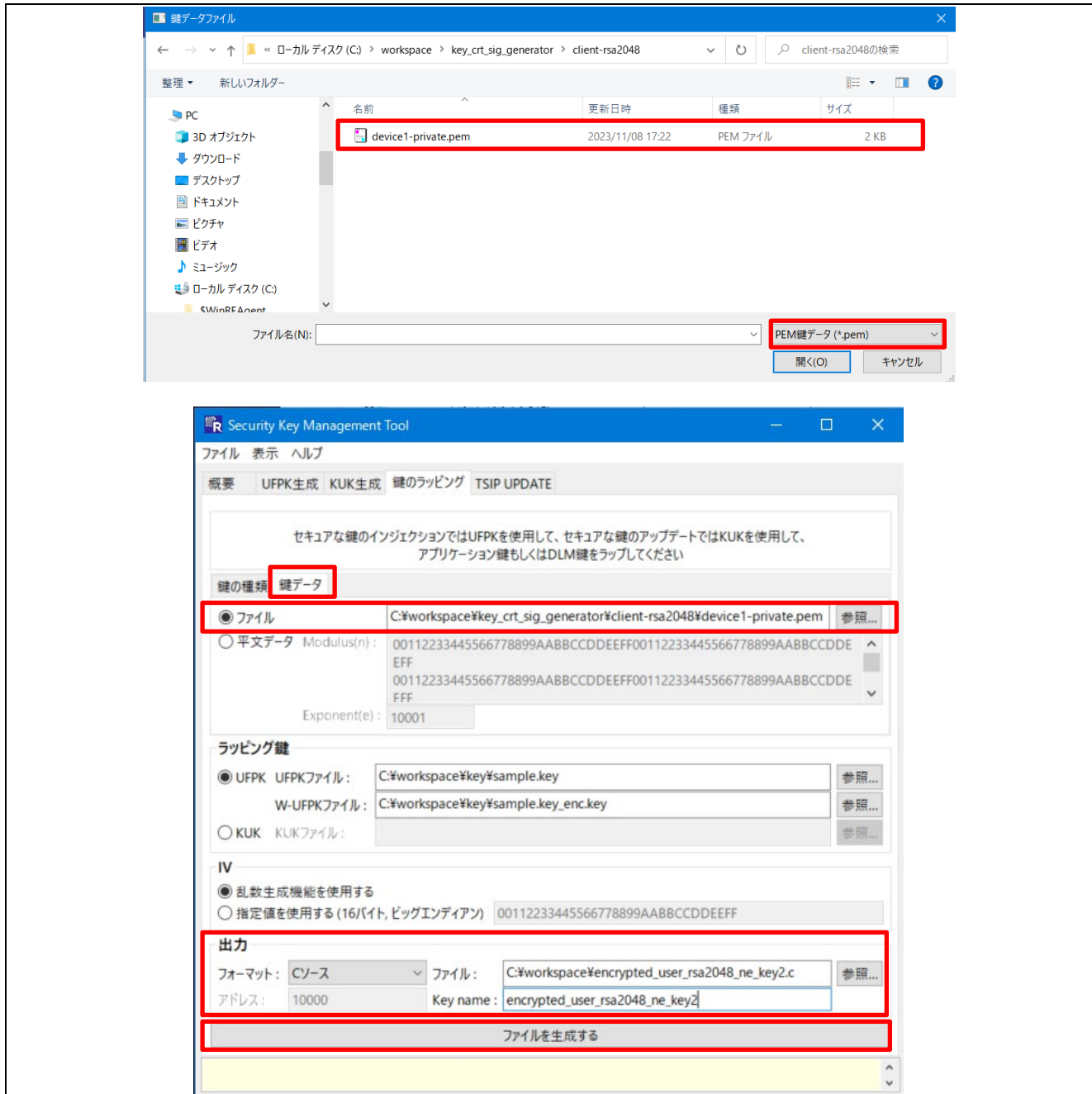


図 2-41 クライアント証明書の公開鍵の生成

ツール画面下のコンソール部分に"OPERATION SUCCESSFUL" と表示されたら生成は完了です。

- ⑤ クライアント証明書の秘密鍵を生成します。  
[鍵の種類]タブに移動し、[RSA] の"2048bits, private"を選択してください。

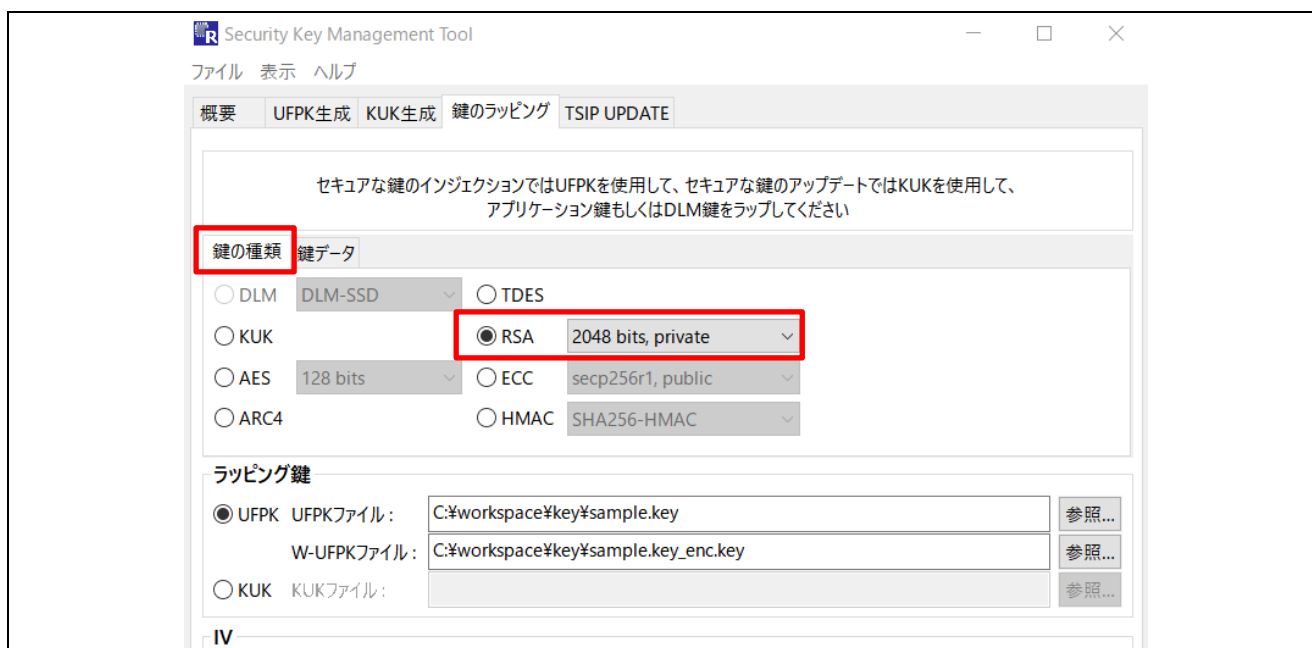


図 2-42 クライアント証明書の秘密鍵

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

再び[鍵データ]タブに移動します。[ファイル]を選択し、[参照]ボタンを押下して、[PEM 鍵データ (\*.pem)]を表示する設定とし、先ほどリネームしたクライアント証明書用鍵ペアファイル(PEM) の /key crt\_sig\_generator /client-rsa2048/device1-private.pem を選択します。

出力欄の"ファイル"に、ファイル名を"encrypted\_user\_rsa2048\_nd\_key"として任意のフォルダを選択、Key name も同じく"encrypted\_user\_rsa2048\_nd\_key"と入力し、[ファイルを生成する]をクリックして、クライアント証明書の秘密鍵データを生成します。

ファイル名"encrypted\_user\_rsa2048\_nd\_key"および Key name はソースコード内で使用しているため、名称は変更しないようにしてください。

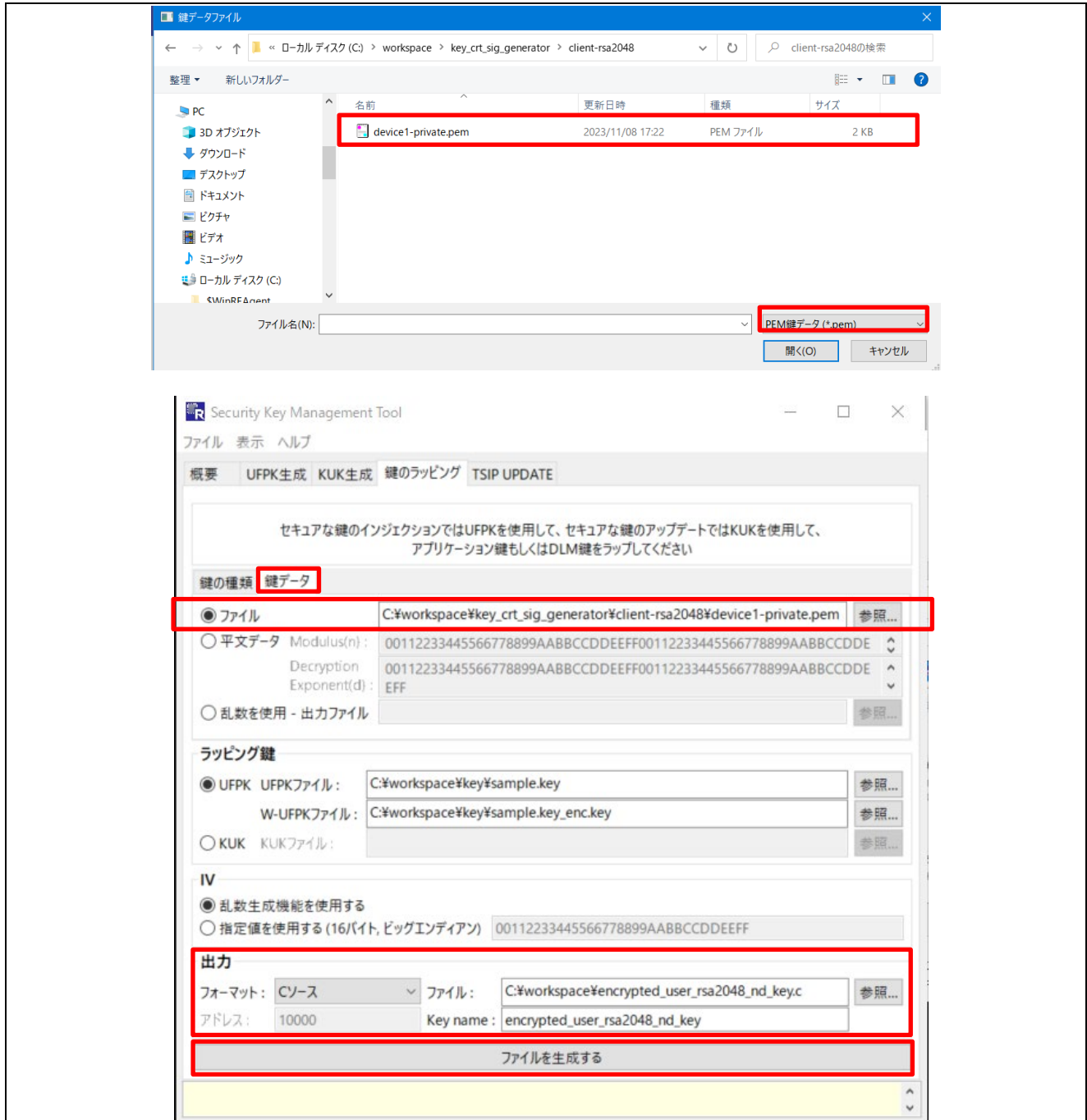


図 2-43 Security Key Management Tool を使用した秘密鍵のラッピング

ツール画面下のコンソール部分に"OPERATION SUCCESSFUL" と表示されたら生成は完了です。



⑥ 生成された以下の 6 つの暗号化鍵ファイルは、サンプルプロジェクトの `sample_azure_iot_embedded_sdk/src/userdata_tsip` 内に上書き保存してください。

- ・ `encrypted_user_rsa2048_ne_key.c`
- ・ `encrypted_user_rsa2048_ne_key.h`
- ・ `encrypted_user_rsa2048_ne_key2.c`
- ・ `encrypted_user_rsa2048_ne_key2.h`
- ・ `encrypted_user_rsa2048_nd_key.c`
- ・ `encrypted_user_rsa2048_nd_key.h`

以上で基本となるプロジェクトファイルの準備は完了となります。

3 章で Microsoft Azure ポータルの設定を行った後に Azure 関連の設定値の登録を行い、4 章でプロジェクトのビルドと実行を行います。

### 3. Microsoft Azure ポータルでの操作

本書のサンプルプロジェクトを実行するための Microsoft Azure の操作手順を説明します。  
本サンプルでは、Azure IoT Hub へ IoT Hub Device Provisioning Service (DPS) を用いて接続します。

#### 3.1 Azure IoT Hub との接続準備 (Azure portal)

##### 3.1.1 IoT Hub の作成

Microsoft Azure と接続するために必要な準備をします。  
アプリケーションノート「[RX65N Cloud kit で Azure RTOS を用いて センサデータを可視化する方法](#)」の「3.1 Azure 準備 (3.1.1、3.1.2)」に従って IoT Hub の作成を行ってください。  
本書の手順では DPS を介して IoT Hub へ接続するため、「3.1.3 IoT デバイス作成」の項でのデバイス作成は行いません。

##### 3.1.2 IoT Hub Device Provisioning Service (DPS) の作成

下記に従って IoT Hub Device Provisioning Service を作成してください

<https://learn.microsoft.com/ja-jp/azure/iot-dps/quick-setup-auto-provision#create-a-new-iot-hub-device-provisioning-service-instance>

DPS が作成できたら、下記に従って IoT Hub と Device Provisioning Service をリンクさせてください。

<https://learn.microsoft.com/ja-jp/azure/iot-dps/quick-setup-auto-provision#link-the-iot-hub-and-your-device-provisioning-service-instance>

以上で IoT Hub と DPS の作成は完了です。

##### 3.1.3 IoT Hub と DPS を用いてデバイスをプロビジョニング

次に、作成した IoT Hub と Device Provisioning Service (DPS) を用いて、デバイスをプロビジョニングする手順について説明します。

- ① Azure ポータルのホーム画面より、[すべてのサービス]⇒[モノのインターネット]カテゴリ⇒[Azure IoT Hub Device Provisioning Service]をクリックしてください。  
Azure IoT Hub Device Provisioning Service のリストが表示されるので、「3.1.2IoT Hub Device Provisioning Service (DPS) の作成」で作成した、DPS を選択します。



図 3-1 DPS のリスト表示

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

- ② 選択した DPS メニューより[登録を管理します]をクリックします。登録を管理します画面が表示されます。



図 3-2 DPS の登録の管理


- ③ 登録を管理します画面より[個々の登録]タブ⇒[個々の登録の追加]をクリックすると[登録の追加]画面が表示されます。



図 3-3 登録を管理します画面

- ④ 登録の追加画面で、DPS の設定を行います。以下の項目を入力してください。特に指定のない項目はデフォルトの設定としてください。

1. "登録とプロビジョニング"画面にて以下の項目を設定します。

- 構成証明メカニズム"X.509 クライアント証明書"を設定します(\*)
- プライマリ証明書ファイルのフォルダアイコン  をクリックしてクライアント証明書のファイルを選択します。  
クライアント証明書のファイルは"2.4.3(1)⑤"項で作成した"device1-certificate.pem.crt"となります。ファイルの拡張子は、".pem" または ".cer" である必要がありますので、クライアント証明書のファイルを任意のフォルダにコピーしてから、ファイル名を"device1-certificate.pem"にリネームしてから選択してください。  
選択すると、"プライマリ証明書ファイル"の下に、証明書に設定した共通名 (Common Name) が表示されます。  
設定が完了したら[次へ : IoT ハブ]ボタンを押下してください。

## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

(※)対称キーを選択した場合は、登録 ID へ任意の ID と、対称キーをプロジェクトに設定する必要があります。

すべてのサービス > Azure IoT Hub Device Provisioning Service > | 登録を管理します >

### 登録の追加

登録とプロビジョニング IoT ハブ デバイス設定 確認と作成

#### 構成証明

構成証明は、登録中にデバイスの ID を確認するプロセスです。デバイスは、登録の選択した構成証明メカニズムを使用して ID を証明する必要があります。

構成証明メカニズム \*

X.509 クライアント証明書

#### X.509 証明書の設定

デバイスプロビジョニングサービスは、クライアント X.509 証明書構成証明を使用して、デバイスの証明書を登録証明書に対して検証します。登録には 1 つまたは 2 つの証明書が含まれる可能性があります。アップロードされた証明書は、共通名を共有する必要があります。

プライマリ証明書ファイル

device1-certificate.pem

共通名: TSIP sample test

セカンダリ証明書ファイル

証明書ファイルを選択してください

#### 登録 ID

登録されている各デバイスには、デバイスプロビジョニングサービスで一意の登録 ID が割り当てられます。ID の形式は構成証明メカニズムによって異なります。

① 登録 ID は、選択した証明書のサブジェクトの共通名と同じになります。

#### プロビジョニングの状態

この登録をプロビジョニングと再プロビジョニングを行うデバイスから有効または無効にすることができます。

この登録を有効にする

#### 再プロビジョニング ポリシー

プロビジョニングされたデバイスは、再プロビジョニングの要求をトリガーする可能性があります。再プロビジョニング ポリシーは、デバイスを再プロビジョニングするかどうかと、デバイスの既存の状態データの処理方法を指定します。

再プロビジョニング ポリシー

デバイスの再プロビジョニングと現在の状態の移行

確認と作成 < 前へ 次へ: IoT ハブ >

図 3-4 登録の追加 : 登録とプロビジョニング

- "IoT ハブ"画面にて"ターゲット IoT ハブ"にデバイスをプロビジョニングしたい IoT Hub 名を設定します。  
設定が完了したら[次へ：デバイス設定]ボタンを押下してください。



図 3-5 登録の追加：IoT ハブ



## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

- ⑤ 正常に設定の保存が完了すると、"個々の登録"画面に新規作成した DPS が登録 ID リストに追加されます。



図 3-8 DPS の登録 ID リスト

この登録 ID は、使用するクライアント証明書に登録された共通名 (Common Name) で登録されます。(任意のデバイス名を登録した場合は登録したデバイス名になります)。

登録 ID はプロジェクトのソースコードで指定する必要がありますので、後で使用できるようにメモをしておいてください。

また、登録 ID 名をクリックすると、以下のように登録状態が表示されます。登録直後は登録状態が空の状態になっていますが、DPS が機能し IoT Hub への登録がされると登録状態が表示されます。



図 3-9 登録の詳細

Azure portal での設定は以上となります。

### 3.2 Microsoft Azure との通信設定

2章で準備したサンプルプロジェクトのソースコードに各種設定を行います。

#### 3.2.1 Azure IoT の設定

サンプルプロジェクト sample\_azure\_iot\_embedded\_sdk フォルダ内の src/sample\_config.h を開き、以下の項目を設定してください。

##### (1) パラメータ設定

以下のように暗号化等の動作パラメータを sample\_config.h に定義します。

表 3-1 設定するパラメーター一覧

パラメータ	設定値	本サンプル の設定値	備考
SEL_CIPHER_SUITE	0:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (default) 1:暗号スイート選択	0	
SEL_DEVICE_AUTH	0:対称キー 1:X.509 自己署名証明書 2:X.509CA 署名証明書 (未サポート)	1	
SEL_DPS	0:IoT Hub 接続 (DPS 接続なし) 1:DPS 接続+IoT Hub 接続	1	
SEL_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA	0 : Client Hello message で指定なし 1 : Client Hello message で指定	0	SEL_CIPHER_SUITE = 0 設定時無効
SEL_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA	0 : Client Hello message で指定なし 1 : Client Hello message で指定	0	
SEL_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA256	0 : Client Hello message で指定なし 1 : Client Hello message で指定	0	
SEL_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA256	0 : Client Hello message で指定なし 1 : Client Hello message で指定	0	
SEL_TSIP_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	0 : Client Hello message で指定なし 1 : Client Hello message で指定	0	

```

3      /* Copyright (c) Microsoft Corporation. All rights reserved. */
11
12  #ifndef SAMPLE_CONFIG_H
13  #define SAMPLE_CONFIG_H
14
15  #ifdef __cplusplus
16  extern "C" {
17  #endif
18
19
20  #include "platform.h"
21  #if (BSP_CFG_MCU_PART_ENCRYPTION_INCLUDED == true)
22  #define SEL_CIPHER_SUITE 0 // 0:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (default), 1:Select cipher suite
23  #define SEL_DEVICE_AUTH 1 // 0:Symmetric, 1:X.509 Self Signed Certificate, 2:X.509 CA Certificate
24  #define SEL_DPS 1 // 0:DPS Not Connect, 1:DPS Connect
25
26  #if (SEL_CIPHER_SUITE == 1)
27  #define SEL_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA 0 // TLS_RSA_WITH_AES_128_CBC_SHA (No.1)
28  #define SEL_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA 0 // TLS_RSA_WITH_AES_256_CBC_SHA (No.2)
29  #define SEL_TSIP_TLS_RSA_WITH_AES_128_CBC_SHA256 0 // TLS_RSA_WITH_AES_128_CBC_SHA256 (No.3)
30  #define SEL_TSIP_TLS_RSA_WITH_AES_256_CBC_SHA256 0 // TLS_RSA_WITH_AES_256_CBC_SHA256 (No.4)
31  #define SEL_TSIP_TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 0 // TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (No.6)
32  #endif /* SEL_CIPHER_SUITE */
33
34  #endif /* BSP_CFG_MCU_PART_ENCRYPTION_INCLUDED */
    
```

図 3-10 sample\_config.h パラメータ設定



## (2) Azure IoT Hub DPS 認証情報

"3.1"節で作成した Azure IoT Hub DPS の登録情報を以下のように sample\_config.h に定義します。

- ①#define ENDPOINT : 接続先 DPS のサービスエンドポイント
- ②#define ID\_SCOPE : 接続先 DPS の ID スコープ
- ③#define REGISTRATION\_ID : 接続先 DPS に登録した登録 ID
- ④#define DEVICE\_SYMMETRIC\_KEY : 接続先 DPS の主キー(※)

(※)主キーは、DPS 登録時に構成証明メカニズムに"対称キー"を選択した時のみ定義します。

上記マクロは、以下のように” ” の間に値を入力してください。

```

111
112 /* Required when DPS is used. */
113 #ifndef ENDPOINT
114 #define ENDPOINT
115 #endif /* ENDPOINT */
116
117 #ifndef ID_SCOPE
118 #define ID_SCOPE
119 #endif /* ID_SCOPE */
120
121 #ifndef REGISTRATION_ID
122 // Begin Modify for TSIP
123 /*
124 #define REGISTRATION_ID
125 */
126 #if ( SEL_DEVICE_AUTH == 0 )
127 #define REGISTRATION_ID
128 #elif ( SEL_DEVICE_AUTH == 1 )
129 #define REGISTRATION_ID "TSIP sample test"
130 #elif ( SEL_DEVICE_AUTH == 2 )
131 #define REGISTRATION_ID
132 #endif
133 // End Modify for TSIP
134 #endif /* REGISTRATION_ID */
135
136 #endif /* ENABLE_DPS_SAMPLE */
137
138 /* Optional SYMMETRIC KEY. */
139 #ifndef DEVICE_SYMMETRIC_KEY
140 #define DEVICE_SYMMETRIC_KEY
141 #endif /* DEVICE_SYMMETRIC_KEY */
    
```

図 3-11 sample\_config.h 認証情報設定

上記認証情報は Azure ポータルで確認します。Azure ホームより「3.1.2」項で作成した DPS を選択し、各画面より設定値をコピーしてソースコードに張り付けてください。

「①サービスエンドポイント」と「②ID スコープ」は DPS を選択し、メニューの[概要]をクリックすることで参照できます。



図 3-12 DPS の概要画面

"③登録 ID"は、DPS を選択後に登録した登録 ID 名を参照します。



図 3-13 DPS の登録 ID リスト

「④主キー」は構成証明メカニズムに「対称キー」を選択した時 (SEL\_DEVICE\_AUTH 0 と定義) のみ定義します。対称キーを設定した DPS の登録 ID の登録の詳細画面より主キーをコピーしてください。構成証明メカニズム"X.509 クライアント証明書"とした場合は、本設定は不要です。



図 3-14 対称キーを設定したときの登録の詳細画面

### (3) DPS を使用しないときの接続情報

DPS 接続を使用しない場合は、IoT Hub ヘドバイスの登録が必要です。

「[RX65N Cloud kit で Azure RTOS を用いて センサデータを可視化する方法](#)」の「3.1.3 IoT デバイス作成」の手順でデバイスの登録を行ってください。

また、上記ドキュメントの「3.2 ソフトウェア準備」の 8 に従って、IoT Hub の設定が必要です。サンプルプロジェクトへは以下の IoT Hub の設定を行ってください。

- HOST\_NAME
- DEVICE\_ID
- DEVICE\_SYMMETRIC\_KEY (対称キーを使用する場合)

また、「表 3-1 設定するパラメータ一覧」を参照し、DPS の使用を無しに設定してください。

```

3      3
11     /* Copyright (c) Microsoft Corporation. All rights reserved. */
12
13     #ifndef SAMPLE_CONFIG_H
14         #define SAMPLE_CONFIG_H
15
16     #ifdef __cplusplus
17         extern "C" {
18     #endif
19
20     #include "platform.h"
21
22     #if ( BSP_CFG_MCU_PART_ENCRYPTION_INCLUDED == true )
23         #define SEL_CIPHER_SUITE          0 // 0:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (default),
24         #define SEL_DEVICE_AUTH          1 // 0:Symmetric, 1:X.509 Self Signed Certificate, 2:X.509 Certificate
25         #define SEL_DPS                  0 // 0:DPS Not Connect, 1:DPS Connect
26
27     #endif
28
29     /* Required when DPS is not used. */
30     /* These values can be picked from device connection string which is of format : HostName=<host1>;DeviceId=<device1>;SymmetricKey=<key1>. */
31     #ifndef HOST_NAME
32         #define HOST_NAME                  "azure-devices.net"
33     #endif /* HOST_NAME */
34
35     #ifndef DEVICE_ID
36         // Begin Modify for TSIP
37         /*
38         #define DEVICE_ID                  ""
39         */
40     #if ( SEL_DEVICE_AUTH == 0 )
41         #define DEVICE_ID                  "TSIP_sample_test_sym"
42     #elif ( SEL_DEVICE_AUTH == 1 )
43         #define DEVICE_ID                  "TSIP_sample_test"
44     #elif ( SEL_DEVICE_AUTH == 2 )
45         #define DEVICE_ID                  ""
46     #endif
47     // End Modify for TSIP
48     #endif /* DEVICE_ID */
49
50     #endif /* SAMPLE_CONFIG_H */
51
52     #endif /* __cplusplus */
53 }
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

図 3-15 DPS を使用しない場合の設定

### 3.2.2 IP アドレスの設定

本サンプルプロジェクトは、Ethernet を使用した通信を行う場合、デフォルトの設定ではネットワークの接続に DHCP を使用します。ターゲットボードを接続するルータの DHCP 機能が無効の場合、以下の設定を行ってください。

sample\_azure\_iot\_embedded\_sdk フォルダ内の src/main.c を開き、#define SAMPLE\_DHCP\_DISABLE を追加する。

sample\_azure\_iot\_embedded\_sdk フォルダ内の src/main.c を開き、IP アドレス、デフォルトゲートウェイ、DNS サーバアドレス、サブネットマスクを入力する。

CK-RX65N+RYZ014A (Cellular)ボードをご使用の場合は、お使いの SIM カードに合わせた設定が必要です。設定方法を次項より説明します。

- ① sample\_azure\_iot\_embedded\_sdk フォルダ内の sample\_azure\_iot\_embedded\_sdk.scfg を開き、[コンポーネント]タブに移動する
- ② [RTOS Library] の [ewf] をクリックして開く
- ③ [The SIM operator APN] にご使用の SIM カードのアクセスポイントを設定する
- ④ ファイル (sample\_azure\_iot\_embedded\_sdk.scfg) を保存し、右上の[コードの生成] をクリックしてコード生成を行う

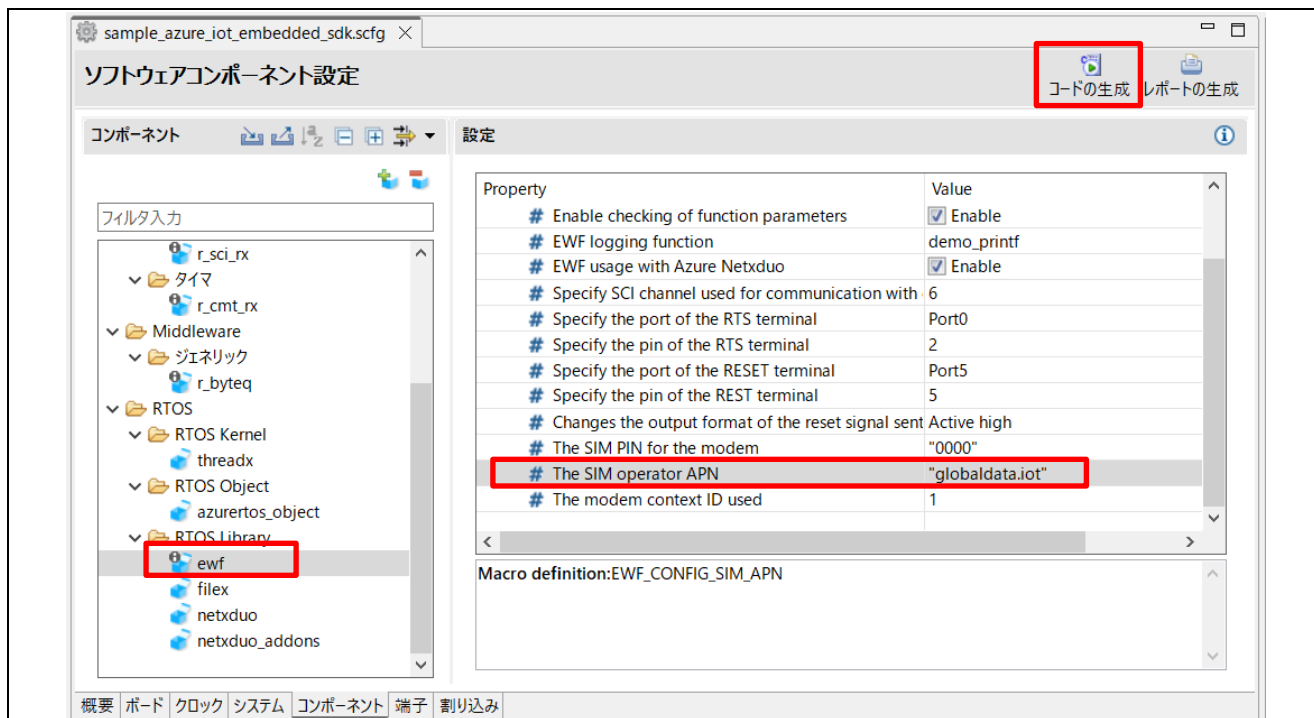


図 3-16 アクセスポイントの設定

### 3.2.3 クライアント証明書の形式の選択

デフォルトでは RSA 証明書が有効になっています。ECDSA 証明書は、本書のサンプルプロジェクトではサポートされておりません。サンプルプロジェクトでは src/userdata\_tsip/r\_trust\_certificate\_data.h に定義されています。

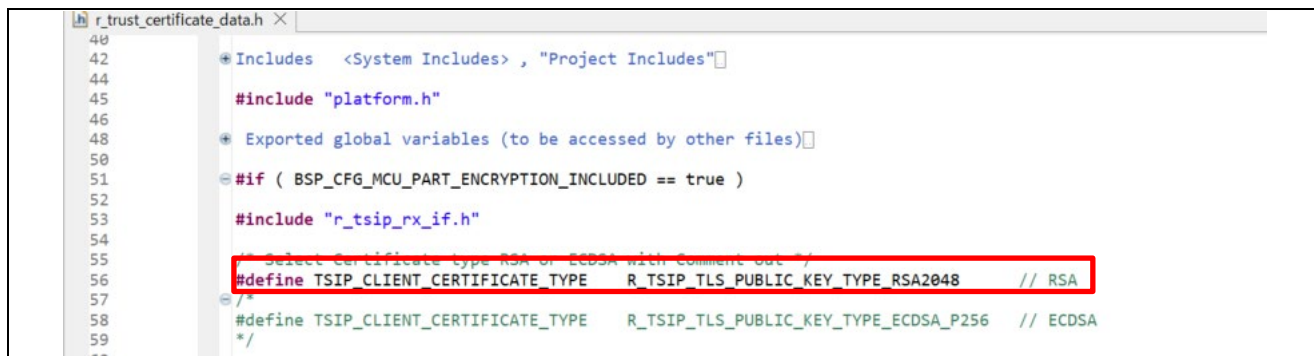


図 3-17 クライアント証明書形式選択

### 4. プロジェクトのビルドおよび実行

2章・3章で作成したプロジェクトをビルドします。  
ビルドの前に以下の4.1節の内容を確認してからビルドを実施してください。

#### 4.1 プロジェクトのビルド前の確認

##### 4.1.1 Renesas Starter Kit+ for RX65N-2MB の設定

Renesas Starter Kit+ for RX65N-2MB のプロジェクトは、TSIP 処理の判断に使用する、`sample_azure_iot_embedded_sdk/src/smc_gen/r_config/r_bsp_config.h` の以下マクロが"true"となっていることを確認して下さい。もし、"false"となっている場合は、手動で"true"に変更して下さい。

```
#define BSP_CFG_MCU_PART_ENCRYPTION_INCLUDED
```

上記設定はコード生成をすると"false"に戻るためご注意ください。

##### 4.1.2 ビルド時のコード生成の設定

本サンプルプロジェクトでは、上記の通りボードによってスマートコンフィグレータの生成ファイルに変更を加えている箇所があるため、コードの生成を行った際に追加している設定が失われる場合があります。このため、プロジェクトをビルドまたはクリーンする際、すべてのソースファイルが再生成されることを回避するため、次の(1)～(3)の処理を実施してください。

- (1) e<sup>2</sup> studio のプロジェクト・エクスプローラーで、対象プロジェクトを右クリックし、コンテキストメニューから[プロパティ]を選択してください。

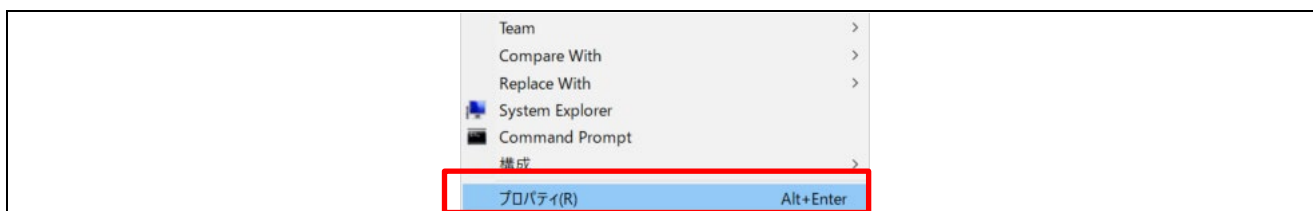


図 4-1 プロジェクトのコンテキストメニュー

- (2) 「プロパティ」ダイアログのメニューの[ビルダー]を選択し、[SC Code Generation Builder]を選択後、[編集]をクリックしてください。ビルダーの構成画面が表示されます。

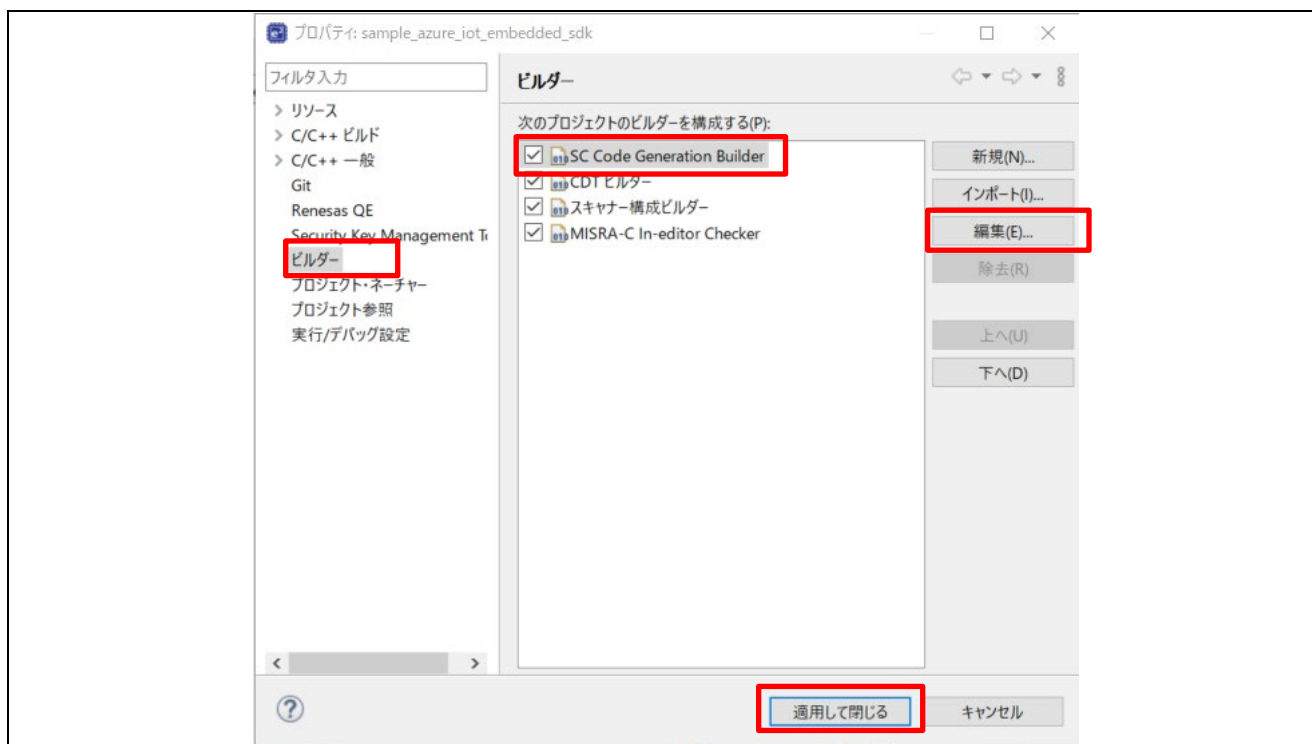


図 4-2 プロジェクトのプロパティダイアログ

- (3) ビルダーの構成画面ですべてのチェックボックスのチェックを外して、[OK]をクリックしてください。その後、プロパティダイアログの[適用して閉じる]をクリックしてください。本設定を実施することで、予期せぬコード生成を抑止することができます。(※)

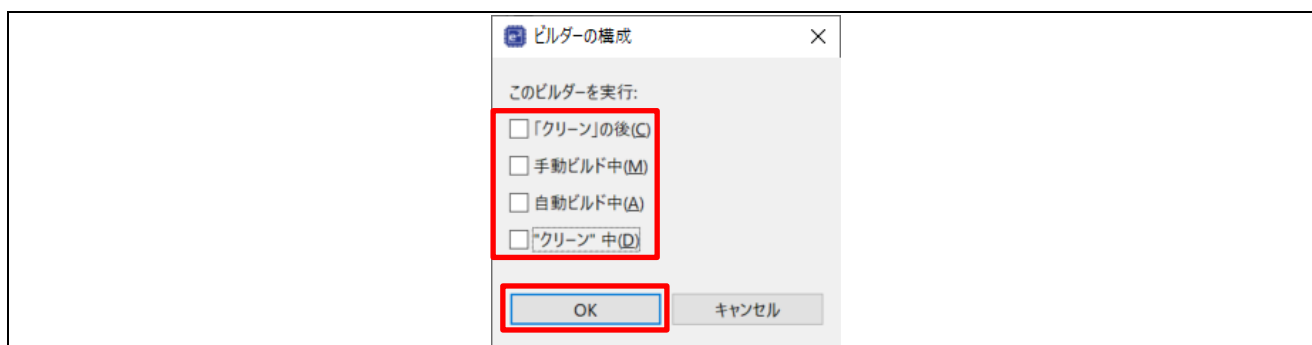


図 4-3 ビルダーの構成画面

(※) 実際にコード生成を行いたい場合は、スマートコンフィグレータのコード生成ボタンをクリックして実施してください。

### 4.2 プロジェクトのビルドと実行

[プロジェクト(P)]→[すべてビルド(A)] をクリックしてプロジェクトをビルドします。この際、Warning が出現しますが動作に問題はありません。

ビルド完了後、図 2-1 を参考にターゲットボードを PC とルータに接続し、[実行(R)]→[デバッグ(G)]→[2 Renesas GDB Hardware Debugging]をクリックしてデバッグを開始してください。

また、CK-RX65N の J12 を USB ケーブルで PC と接続することで Tera Term 等のターミナルソフトで動作状態をモニターすることができます。

J12 を PC と接続すると、Windows のデバイスマネージャーのポートに登録されるので、登録された COM ポート番号をターミナルソフトに設定してターゲットボードと接続してください。

シリアルポートの通信設定は以下に設定してください。

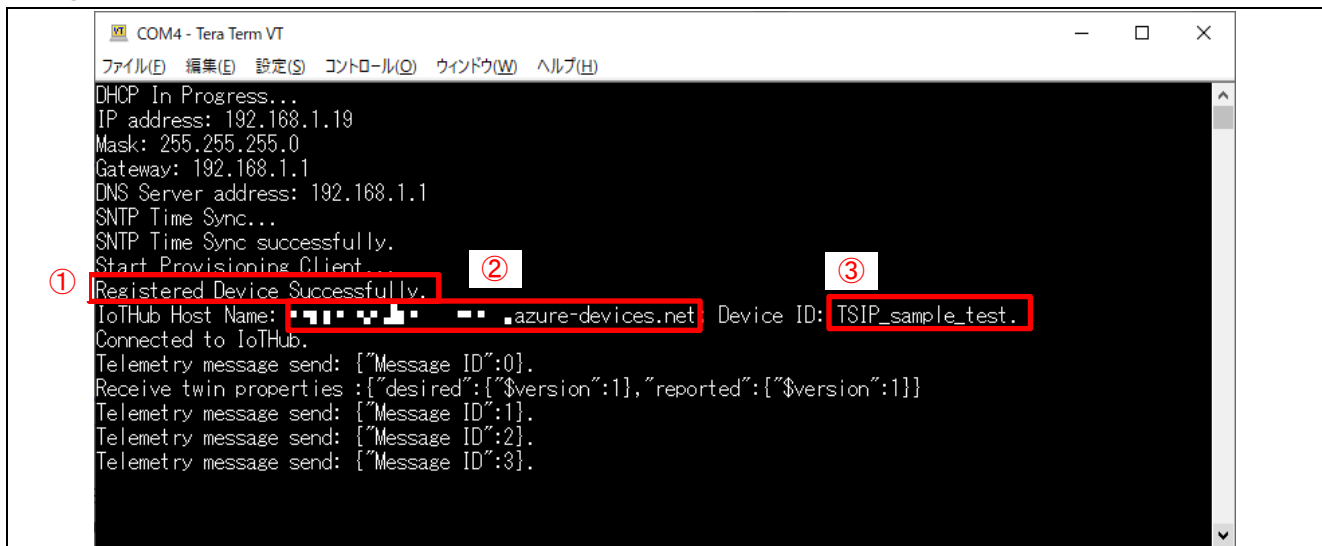
- ボーレート : 115200bps
- データビット : 8
- ストップビット : 1
- パリティ : なし

プロジェクトを実行すると、DPS 接続により IoT Hub にデバイス登録 (プロビジョニング) を行います。DPS 接続で IoT Hub にデバイス登録を行った後、IoT Hub に接続し、メッセージを MQTT で出力するプログラムが実行されます。

サンプルプログラムは以下の画面例のように動作状態をターミナルへシリアル出力します。

ターミナルの表示を確認し、以下の状態を確認してください。

- ①DPS への接続完了
- ②DPS から IoT Hub へ接続完了
- ③登録デバイス ID へ接続完了



```
COM4 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
DHCP In Progress...
IP address: 192.168.1.19
Mask: 255.255.255.0
Gateway: 192.168.1.1
DNS Server address: 192.168.1.1
SNTP Time Sync...
SNTP Time Sync successfully.
Start Provisioning Client...
① Registered Device Successfully. ②
IoTHub Host Name: [redacted] azure-devices.net Device ID: TSIP_sample_test. ③
Connected to IoTHub.
Telemetry message send: [{"Message ID":0}.
Receive twin properties : [{"desired":{"$version":1},"reported":{"$version":1}}
Telemetry message send: [{"Message ID":1}.
Telemetry message send: [{"Message ID":2}.
Telemetry message send: [{"Message ID":3}.
```

図 4-4 ターミナル出力画面例

### 4.3 Microsoft Azure との接続確認

Microsoft Azure との接続確認は、Azure IoT Explorer (preview)を使用することで、アップロードしたデータが Azure クラウドに送信されていることを確認することが可能です。「[RX65N Cloud kit で Azure RTOS を用いて センサデータを可視化する方法](#)」の「3.5 Azure IoT Explorer による通信確認」を参照ください。Azure IoT Explorer (preview) にて[接続先のIoT Hub]⇒[登録 ID]をクリック後、メニューより[Telemetry]をクリック後、[Start]を押下して下さい(※)。

"Receive Events..."以下に受信データが表示されていれば正常にメッセージが送受信されています。

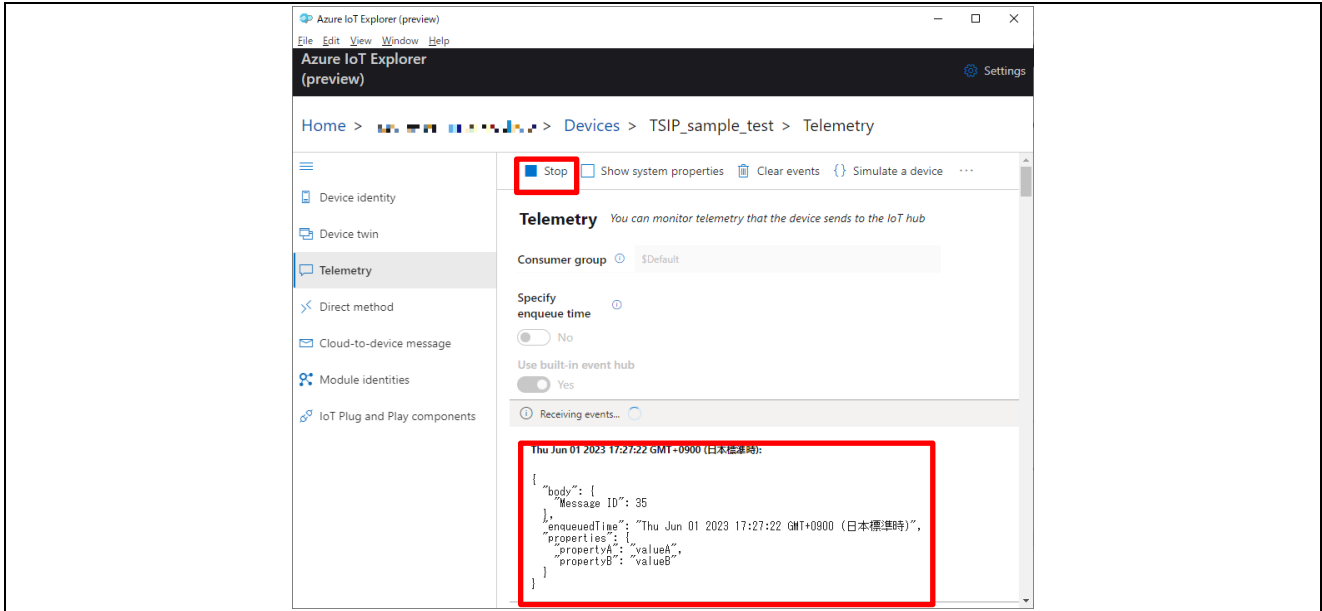


図 4-5 Azure IoT Explorer での Telemetry メッセージの受信確認

(※) [Start]を押下して受信開始すると、画面例のように[Stop]の表示となります。

#### 4.3.1 登録状態の確認

DPS 接続による IoT Hub へのデバイス登録が正常に実施されたかの確認は以下で行うことができます。Azure ポータルで、接続先の DPS を選択し、設定メニューより[登録を管理します]⇒[個々の設定]タブをクリックし、登録 ID のリストを表示してください。リストより接続先の登録 ID をクリックするとデバイス登録の詳細画面が表示されます。



図 4-6 登録 ID リストの表示



## RX ファミリ TSIP ドライバを用いた TLS 実装例 (Azure RTOS)

デバイスの登録の詳細画面で、登録状態が表示されます。DPS が割り当てた IoT Hub とデバイス ID が確認できます。



図 4-7 登録の詳細画面

### 4.3.2 デバイスの確認

「4.3.1」で DPS に登録した IoT Hub にデバイスが追加されていることを確認します。

Azure ポータルから DPS で登録された IoT Hub 名を選択し、デバイス管理メニューの[デバイス]をクリックしてください。デバイスの一覧が表示され、接続されたデバイスが新規登録されます<sup>(※)</sup>。



図 4-8 IoT Hub デバイスリスト

(※)一度もデバイスと接続していないときは、リストに表示されません。

## 5. 付録

### 5.1 Security Key Management Tool の詳細情報

Security Key Management Tool の詳細はこちらをご参照ください。

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

### 5.2 TSIP ドライバを用いた TLS 通信の性能

表 5-1 に Renesas Starter Kit+ for RX65N-2MB で TLS 通信をした際の TLS 接続確立に伴うハンドシェイク時間および接続確立後の Application Data の通信速度の例を参考までに示します。

アップロード時間計測時 (4KB)、ダウンロード時間計測時 (1MB) 分のデータの転送時間を MCU 内蔵のタイマで測定し、5 回分の転送時間の平均値から算出しています。この例では、TSIP ドライバを利用することにより、TLS 接続確立に伴うハンドシェイク時間が 2.73 s から 0.34 s、アップロード時間が 4.46 Mbps から 24.82 Mbps、ダウンロード時間が 5.34 Mbps から 27.03 Mbps に向上しています。

表 5-1 TSIP ドライバを用いた TLS 通信速度の例

Cipher Suite	Block Cipher	NetX Duo <sup>注1</sup>	NetX Duo w/ TSIP <sup>注2</sup>
TLS_RSA_WITH_AES_128_CBC_SHA	128bit AES-CBC	Connection:2.73s Up: 4.46 Mbps Down: 5.34 Mbps	Connection:0.34s Up: 24.82 Mbps Down: 27.03 Mbps

【注】 システムクロック (ICLK): 120MHz  
TSIP 動作クロック (PCLKB): 60MHz

【注】 1. NetX Duo: ソフトウェア処理  
2. NetX Duo w/ TSIP: TSIP ドライバの TLS 向け API を使用

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

<https://www.renesas.com/jp/ja/>

お問合せ先

<https://www.renesas.com/jp/ja/support/contact.html>

## 6. 改訂履歴

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2023.08.31	—	初版発行
1.10	2023.12.14	1,30	RX72N(Envision kit)対応の説明を追加 (本文説明は CK-RX65N を基本とする)
		1,9,12	IDE/Azure RTOS/RDP/OpenSSL のバージョンを更新
		5	TSIP を用いた TLS フロー図を修正
		13-19	ルート CA 証明書 DigiCert Global Root G2 の入手に関する説明を追加
		19	使用するルート CA 証明書の設定手順追加
		23	DigiCert のファイルをリストに追加
		23,32	スクリプトで生成した、ラップに使用する鍵データの説明を追加
2.00	2024.01.23	1	Important Notice の文を追加
		1,2,8	ターゲットボード追加による説明を追加
		2	CK-RX65N+RYZ014A (Cellular)、Renesas Starter Kit+ for RX65N-2MB、CK-RX65N Cellular 対応の説明を追加(本文説明は CK-RX65N を基本とする)
		2,9	IDE/Azure RTOS/RDP のバージョンを更新
		2,5,7,9, 24,25,29, 30,34-41	鍵生成ツールを Security Key Management Tool に変更
		7,24,25,29, 30,32-34	Security Key Management Tool に合わせ用語を編集
		10	鍵生成ツール変更に伴い配布方法が変わるため、プロジェクトフォルダ構成から tool フォルダを削除
		19	保持している証明書を全て検証する方式に変更したため、使用するルート CA 証明書の選択/設定を削除
		19	デフォルトでサンプルプロジェクトに準備しているルート CA 証明書登録用ファイルについて説明を追加
		22	ca フォルダに DigiCertGlobalRootG2.cer を追加
		23	DER 形式のルート CA 証明書のファイル名を変更する手順を追加
		23,41,52	設定ファイルの格納フォルダを userdata_tsip としたことでパスを編集
		29,30, 34-40	Security Key Management Tool の使い方を追加
		51,52	CK-RX65N+RYZ014A (Cellular)ボードを使用する場合の AP 設定の手順を追加
		53	nx_secure_port.h に記載していた NetXDuo Addons の設定をパッチファイルに含めるようにしたため削除
53	Renesas Starter Kit+ for RX65N-2MB ボードを使用する場合のマクロ設定の確認の手順を追加		
58	Security Key Management Tool の詳細な使い方が記載されているドキュメントを案内する文言に変更		

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しないでください。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、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、 $V_{IL}(\text{Max.})$  から  $V_{IH}(\text{Min.})$  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、 $V_{IL}(\text{Max.})$  から  $V_{IH}(\text{Min.})$  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くと、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、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

## ご注意書き

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

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

高品質水準： 輸送機器（自動車、電車、船舶等）、交通管制（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たまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア/ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証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なります。
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（豊洲フォレスト）

[www.renesas.com](http://www.renesas.com)

## 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。

## 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/](http://www.renesas.com/contact/)