Application Note

CONFIDENTIAL

Sample Dynamic TA

Application Note: Software

R-Car Series, 3rd Generation

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.3.0.3 Aug. 2021

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Trademark

- Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.
- Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
- Windows and Windows Media are registered trademarks of Microsoft Corporation in the United States and other countries.
- Other company names and product names mentioned herein are registered trademarks or trademarks of their respective owners.
- Registered trademark and trademark symbols (® and ™) are omitted in this document

How to Use This Manual

- **[Readers]**
  This manual is intended for engineers who develop products which use the R-Car H3/M3/M3N/E3/D3 processor.

- **[Purpose]**
  This manual is intended to give users an understanding of the functions of the R-Car H3/M3/M3N/E3/D3 processor device driver and to serve as a reference for developing hardware and software for systems that use this driver.

- **[How to Read This Manual]**
  It is assumed that the readers of this manual have general knowledge in the fields of electrical
  — engineering, logic circuits, microcontrollers, and Linux.
  → Read this manual in the order of the CONTENTS.
  — To understand the functions of a multimedia processor for R-Car H3/M3/M3N/E3/D3
  → See the R-Car H3/M3/M3N/E3/D3 User's Manual.
  — To know the electrical specifications of the multimedia processor for R-Car H3/M3/M3N/E3/D3
  → See the R-Car H3/M3/M3N/E3/D3 Data Sheet.

- **[Conventions]**
  The following symbols are used in this manual.
  Data significance: Higher digits on the left and lower digits on the right
  Note: Footnote for item marked with Note in the text
  Caution: Information requiring particular attention
  Remark: Supplementary information
  Numeric representation: Binary ... ××××, 0b××××, or ××××B
  Decimal ... ××××
  Hexadecimal ... 0x×××× or ××××H
  Data type: Double word … 64 bits
  Word … 32 bits
  Half word ... 16 bits
  Byte ... 8 bits

Table of Contents

1. Overview

1.1 Overview

This manual describes the storage operation and AES encryption or decryption using the Dynamic TA with R-Car H3/M3/M3N/E3/D3/H3e/M3e/M3Ne/E3e/D3e.
The Sample Dynamic TA to encrypt or decrypt with the secret key stored in the storage.
The secret key is created by a random number.

1.2 References

1.2.1 Standard

The following table shows the document related to module.

Table 1.1 Standards

Number	Issue	Title	Edition
1	Global Platform	TEE Client API Specification	Rev.1.1
2	Global Platform	TEE Internal Core API Specification	Rev.1.1

1.2.2 Related Document

The following table shows the document related to this module.

Table 1.2 Related documents

Number	Issue	Title	Edition
1	Renesas Electronics	Security BSP User's Manual	#0

#0：This manual refers to the latest edition.

1.3 Restrictions

Nothing.

1.4 Terminology

The following table shows the terminology related to this module.

Table 1.3 Terminology

Terms	Explanation
AES	AES is an abbreviation for Advanced Encryption Standard.
ECB	ECB is an abbreviation for Electronic Codebook.
CBC	CBC is an abbreviation for Cipher Block Chaining.
Secret Key	Common key to be used in the AES encryption or decryption.
Secure World	It is one of the security states that defined Armv8-A architecture. When in this state, the CPU can access both the Secure and Non-secure space.
Normal World	It is one of the security states that defined Armv8-A architecture. When in this state, the CPU can access only Non-secure space.
Client Application (CA)	An application running outside of the Trusted Execution Environment making use of the TEE Client API to access facilities provided by Trusted Applications inside the Trusted Execution Environment.
Trusted Application (TA)	An application running inside the Trusted Execution Environment that provides security related functionality to Client Applications outside of the TEE or to other Trusted Applications inside the Trusted Execution Environment.

2. Operating Environment

2.1 Hardware Environment

The following table lists the hardware needed to use this function.

Table 2.1 Hardware environment (R-Car H3/M3/M3N/E3/D3)

Name	Note
Evaluation Board	R-Car H3 SiP System Evaluation Board (Salvator-X/Salvator-XS) Renesas Electronics R-Car M3 SiP System Evaluation Board (Salvator-X/Salvator-XS) Renesas Electronics R-Car M3N SiP System Evaluation Board (Salvator-XS) Renesas Electronics R-Car E3 SiP System Evaluation Board (Ebisu/Ebisu-4D) Renesas Electronics R-Car D3 SiP System Evaluation Board (Draak) Renesas Electronics
Host PC 1	It is used as debugging environment. Terminal software is executed.
Host PC 2 (Linux)	TFTP server software It is used when HyperFlash is written by U-Boot or Image is downloaded.
	NFS server software It is used when File system is mounted by NFS.

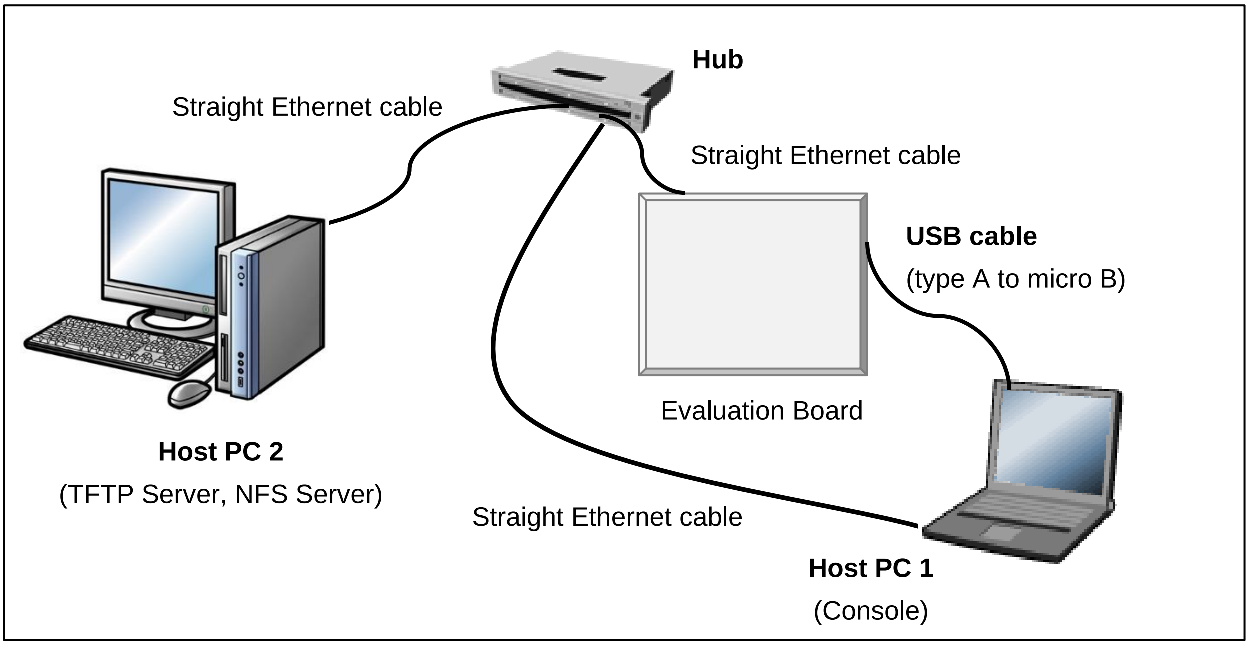


Figure 2.1 Recommended Environment

2.2 Software relationship

The following paint figure shows the description scope of Sample Dynamic TA.

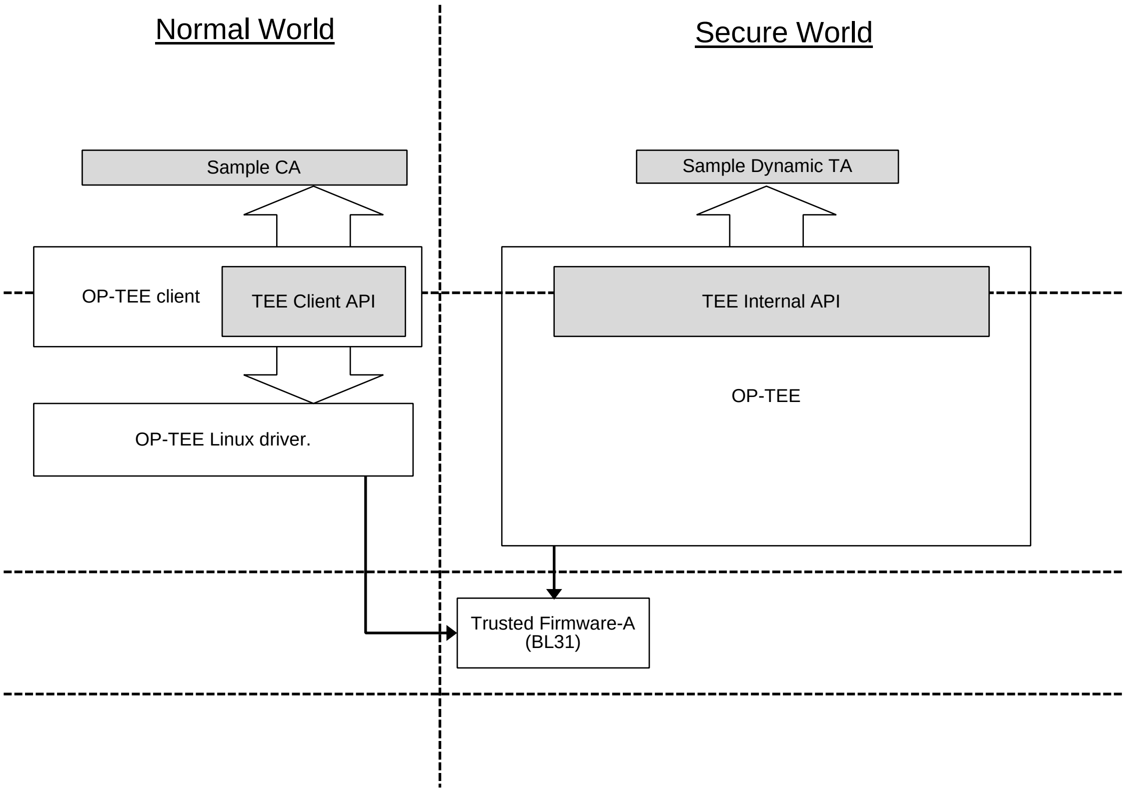


Figure 2.2 Software relationship

Table 2.2 Relationship items

Name	Note
Sample CA	Sample CA. “3.1 Function” reference for the functions that are implemented in the Sample CA.
TEE Client API	This specification defines a communications API for connecting Client Applications running in a rich operating environment with security related Trusted Applications running inside a Trusted Execution Environment (TEE). Please refer to the “TEE Client API Specification” for details.
Sample Dynamic TA	Sample Dynamic TA. “3.1 Function” reference for the functions that are implemented in the Sample Dynamic TA.
TEE Internal API	This specification defines a set of C APIs for the development of Trusted Applications (TAs) running inside a Trusted Execution Environment (TEE). Please refer to the “TEE Internal Core API Specification” for details.

3. Software

3.1 Function

This Sample Dynamic TA has the following functions.

- If the secret key does not exist in storage to create a secret key (128bit) with a random number. If the secret key is present in the storage it will use its secret key.
- Input file is encrypted or decrypted by cryptographic algorithm (ECB / CBC).
- Result of the encryption or decryption will be output to a file.

3.2 Module structure

This Sample Dynamic TA structure is shown below.

```
sampleta/                              : root directory
|-- host/
|     |-- sampleca.c                   : Source file of CA
|     `-- Makefile                     : Makefile of the CA
|-- ta/
|     |-- include/
|     |     `-- common.h               : Common definition of CA and TA
|     |-- sampleta.c                   : Source file of TA
|     |-- Makefile                     : Makefile of the CA
|     |-- sub.mk                       : Sub Makefile of the TA
|     `-- user_ta_header_defines.h     : Configurations file of the TA
`-- Makefile                           : CROSS_COMPILE of the CA and TA
```

Figure 3.1 Module structure

3.3 The processing outline of Sample Dynamic TA

The following is an outline of operation in a program.

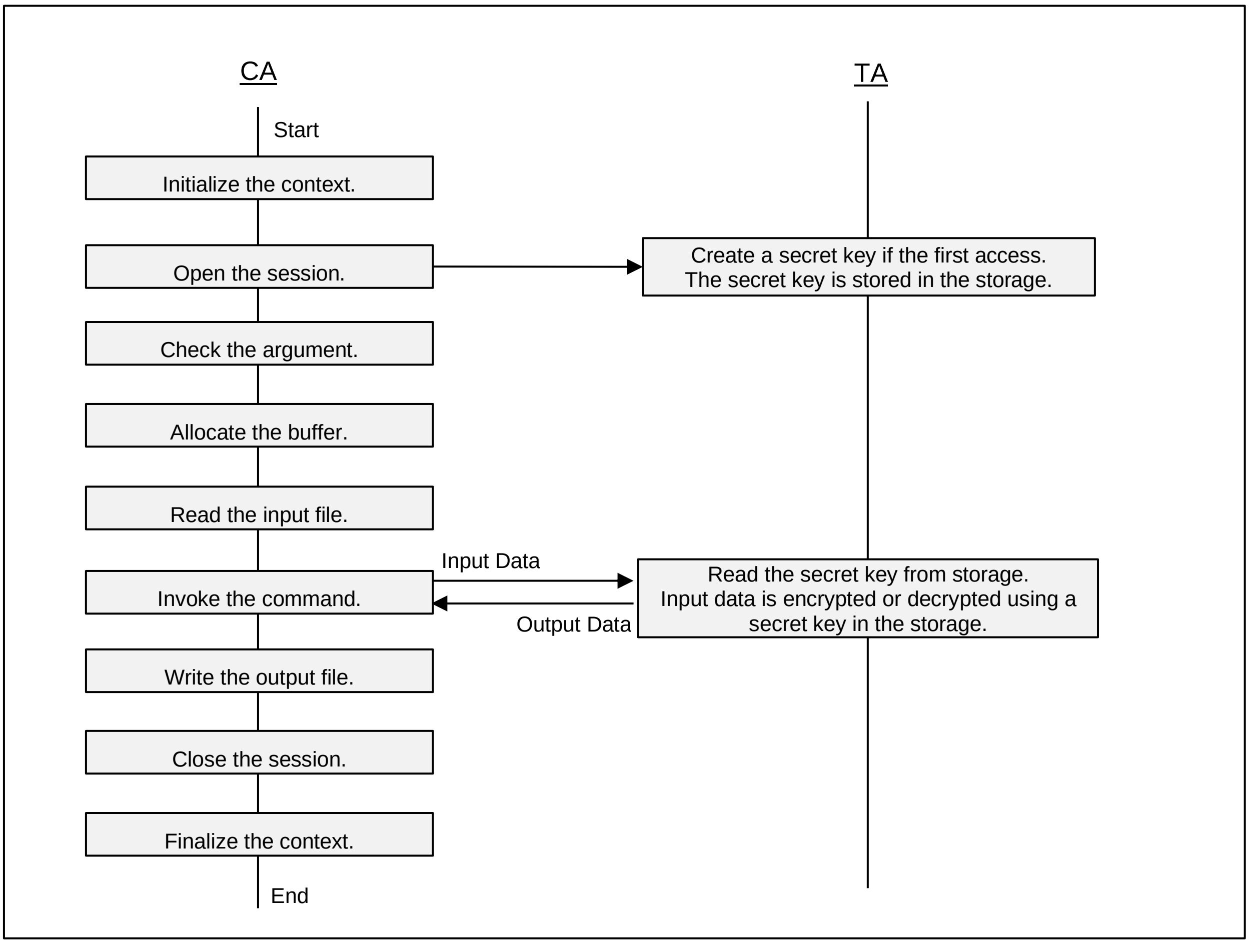


Figure 3.2 Outline of process for the Sample Dynamic TA

4. Integration

4.1 Build Sample Dynamic TA

Please acquire the sample source in the following procedures.

Step1 git clone

```
# cd $WORK
# git clone https://github.com/renesas-rcar/sampleta.git
```

Note) $WORK is the working directory.

Please refer to "4.4.3.5 How to build Dynamic TA" of the "Security BSP User's Manual" for the build procedure.
The following table shows the built output file.

Table 4.1 Output file list

Number	Application	Output path	File name
1	Sample CA	$WORK/sampleta/host/	sampleca
2	Sample Dynamic TA	$WORK/sampleta/ta/out/	2fa77580-ec0a-11e5-a47f0002a5d5c51b.ta

5. Execute the applications

This chapter shows how to perform the Sample Dynamic TA.

5.1 Execute the application

Step1 Execute application

```
# sampleca [algorithm] [mode] [input file] [output file]
```

Note) sampleca is a premise that is located in /usr/bin.

5.2 Command parameters

The following table lists the arguments that are specified by the CA.

Table 5.1 Argument

Name	Note
[algorithm]	The following parameter can be specified. ecb : AES ECB algorithm cbc : AES CBC algorithm
[mode]	The following parameter can be specified. enc : Encryption mode dec : Decryption mode
[input file]	Specify the input file. The input file is read in binary. Note) Input data must be a multiple of 16 bytes. Input data cannot be specified size larger than 262144 bytes.
[output file]	Specify the output file. The output file is written in binary.

Appendix

This chapter describes how to implement based on the sample source code.
The source is related processing colored.

Shared memory

There are two ways of how to use shared memory.

- How to copy the buffer of the CA to the shared memory.
- How to reference to the buffer pointer of the CA.

The following sample source code has entrusted the allocation of shared memory to TEE_Client_API.
output_buf has been copied to the shared memory.

```
uint8_t              *input_buf         = NULL;
uint8_t              *output_buf        = NULL;
struct               stat    st;
TEEC_SharedMemory       input_SharedMemory;
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/* Allocates a buffer. */
if (res == (TEEC_Result)TEEC_SUCCESS) {
    input_SharedMemory.size = st.st_size;
    input_SharedMemory.flags = TEEC_MEM_INPUT;
    res = TEEC_AllocateSharedMemory(&ctx, &input_SharedMemory);
    if (res != (TEEC_Result)TEEC_SUCCESS) {
        (void)printf("TEEC_AllocateSharedMemory error 0x%08x¥n", res);
    } else {
        input_buf = input_SharedMemory.buffer;
    }
}
if (res == (TEEC_Result)TEEC_SUCCESS) {
    output_buf = malloc((size_t)st.st_size);
    if (output_buf == NULL) {
        res = TEEC_ERROR_GENERIC;
        (void)printf("Memory allocate error¥n");
    }
}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/* Invoke command. */
if (res == (TEEC_Result)TEEC_SUCCESS) {
    (void)memset(&op, 0, sizeof(op));
    op.paramTypes = (uint32_t)TEEC_PARAM_TYPES(TEEC_VALUE_INPUT, TEEC_VALUE_INPUT,
        TEEC_MEMREF_PARTIAL_INPUT, TEEC_MEMREF_TEMP_OUTPUT);
        op.params[0].value.a          = algo;
        op.params[1].value.a          = mode;
        op.params[2].memref.offset = 0U;
        op.params[2].memref.size      = readSize;
        op.params[2].memref.parent = &input_SharedMemory;
        op.params[3].tmpref.buffer = (void*)output_buf;
        op.params[3].tmpref.size      = readSize;

    res = TEEC_InvokeCommand(&sess, (uint32_t)E_TEEC_CRYS_CMD_AES, &op, &err_origin);
    if (res != (TEEC_Result)TEEC_SUCCESS) {
        (void)printf("TEEC_InvokeCommand failed with code 0x%08x origin 0x%08x¥n", res, err_origin);
    }
}
/* Output file */
if (res == (TEEC_Result)TEEC_SUCCESS) {
    fp_out = fopen(argv[4], "wb");
    if (fp_out == NULL) {
        (void)printf("fopen error out_file=%s¥n", argv[4]);
    } else {
        writeSize = fwrite(output_buf, 1U, op.params[3].tmpref.size, fp_out);
        (void)fclose(fp_out);

        if (writeSize != op.params[3].tmpref.size) {
            (void)printf("File write error %ld¥n", writeSize);
        }
    }
}
/* Release */
TEEC_ReleaseSharedMemory(&input_SharedMemory);
myFree(output_buf);
```

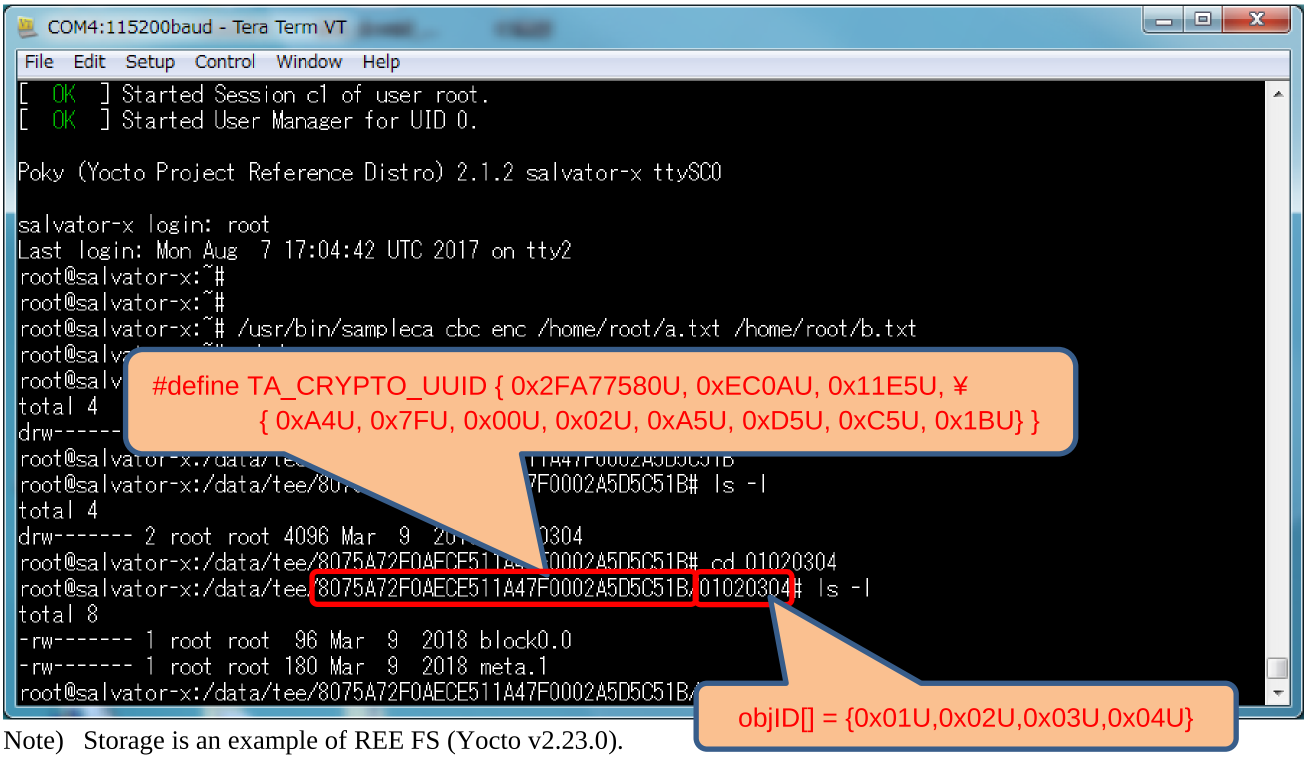
The following sample source code is then allocated on the shared memory, puts the input data to the allocated areas. Pointer of input_buf will be the address of the shared memory.

```
uint8_t             *input_buf         = NULL;
uint8_t             *output_buf        = NULL;
struct              stat    st;
TEEC_SharedMemory       input_SharedMemory;
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/* Allocates a buffer. */
if (res == (TEEC_Result)TEEC_SUCCESS) {
      input_SharedMemory.size = st.st_size;
      input_SharedMemory.flags = TEEC_MEM_INPUT;
      res = TEEC_AllocateSharedMemory(&ctx, &input_SharedMemory);
      if (res != (TEEC_Result)TEEC_SUCCESS) {
            (void)printf("TEEC_AllocateSharedMemory error 0x%08x¥n", res);
      } else {
            input_buf = input_SharedMemory.buffer;
      }
}
if (res == (TEEC_Result)TEEC_SUCCESS) {
      output_buf = malloc((size_t)st.st_size);
      if (output_buf == NULL) {
            res = TEEC_ERROR_GENERIC;
            (void)printf("Memory allocate error¥n");
      }
}
/* Input file */
if (res == (TEEC_Result)TEEC_SUCCESS) {
      fp_in = fopen(argv[3], "rb");
      if (fp_in == NULL) {
            res = TEEC_ERROR_GENERIC;
            (void)printf("fopen error in_file=%s¥n", argv[3]);
      } else {
            readSize = fread(input_buf, 1U, (size_t)st.st_size, fp_in);
            (void)fclose(fp_in);

            if (readSize != st.st_size) {
                  res = TEEC_ERROR_GENERIC;
                  (void)printf("File read error %ld¥n", readSize);
            }
      }
}
/* Invoke command. */
if (res == (TEEC_Result)TEEC_SUCCESS) {
      (void)memset(&op, 0, sizeof(op));
      op.paramTypes = (uint32_t)TEEC_PARAM_TYPES(TEEC_VALUE_INPUT, TEEC_VALUE_INPUT,
            TEEC_MEMREF_PARTIAL_INPUT, TEEC_MEMREF_TEMP_OUTPUT);
            op.params[0].value.a           = algo;
            op.params[1].value.a           = mode;
            op.params[2].memref.offset = 0U;
            op.params[2].memref.size      = readSize;
            op.params[2].memref.parent = &input_SharedMemory;
            op.params[3].tmpref.buffer = (void*)output_buf;
            op.params[3].tmpref.size      = readSize;

      res = TEEC_InvokeCommand(&sess, (uint32_t)E_TEEC_CRYS_CMD_AES, &op, &err_origin);
      if (res != (TEEC_Result)TEEC_SUCCESS) {
            (void)printf("TEEC_InvokeCommand failed with code 0x%08x origin 0x%08x¥n", res, err_origin);
      }
}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/* Release */
TEEC_ReleaseSharedMemory(&input_SharedMemory);
myFree(output_buf);
```

Manipulating the storage

The following sample source code is the method of operation of the storage.
TEE_OpenPersistentObject failure will generate and store the secret key.

```
/******************************************************************************/
/* Global Variables                                                           */
/******************************************************************************/
static uint8_t        objID[]     = {0x01U,0x02U,0x03U,0x04U};
static uint32_t       objID_len = 4U;

/* Secret key */
static TEE_ObjectHandle secretKey;

/* IV */
static uint8_t      iv[16];
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TEE_Result TA_CreateEntryPoint(void)
{
    TEE_Result              res;
    uint8_t                 keyData[16];
    TEE_ObjectHandle        object;

    /* Opens a handle on an existing persistent object. */
    res = TEE_OpenPersistentObject((uint32_t)TEE_STORAGE_PRIVATE, objID, objID_len,
        (uint32_t)TEE_DATA_FLAG_ACCESS_READ, &object);
    if (res != (TEE_Result)TEE_SUCCESS) {
        IMSG("###################################First Access###################################¥n");

        /* Random generation of key data. */
        TEE_GenerateRandom((void*)keyData, sizeof(keyData));
        DBUFDUMP(keyData, (int32_t)sizeof(keyData));

        /* Random generation of IV. */
        TEE_GenerateRandom((void*)iv, sizeof(iv));
        DBUFDUMP(iv, (int32_t)sizeof(iv));

        /* Create persistent object. */
        res = TEE_CreatePersistentObject((uint32_t)TEE_STORAGE_PRIVATE, objID, objID_len,
                                (uint32_t)TEE_DATA_FLAG_ACCESS_WRITE,
                                NULL, NULL, 0U, &object);
        if (res != (TEE_Result)TEE_SUCCESS) {
            EMSG("TEE_CreatePersistentObject¥n");
        } else {
            /* Write the key data. */
            res = TEE_WriteObjectData(object, (void*)keyData, sizeof(keyData));
            if (res != (TEE_Result)TEE_SUCCESS) {
                EMSG("TEE_WriteObjectData¥n");
            }
            /* Write the IV. */
            res = TEE_WriteObjectData(object, (void*)iv, sizeof(iv));
            if (res != (TEE_Result)TEE_SUCCESS) {
                EMSG("TEE_WriteObjectData¥n");
            }
            TEE_CloseObject(object);
        }
    } else {
        TEE_CloseObject(object);
    }

    return res;
}
```



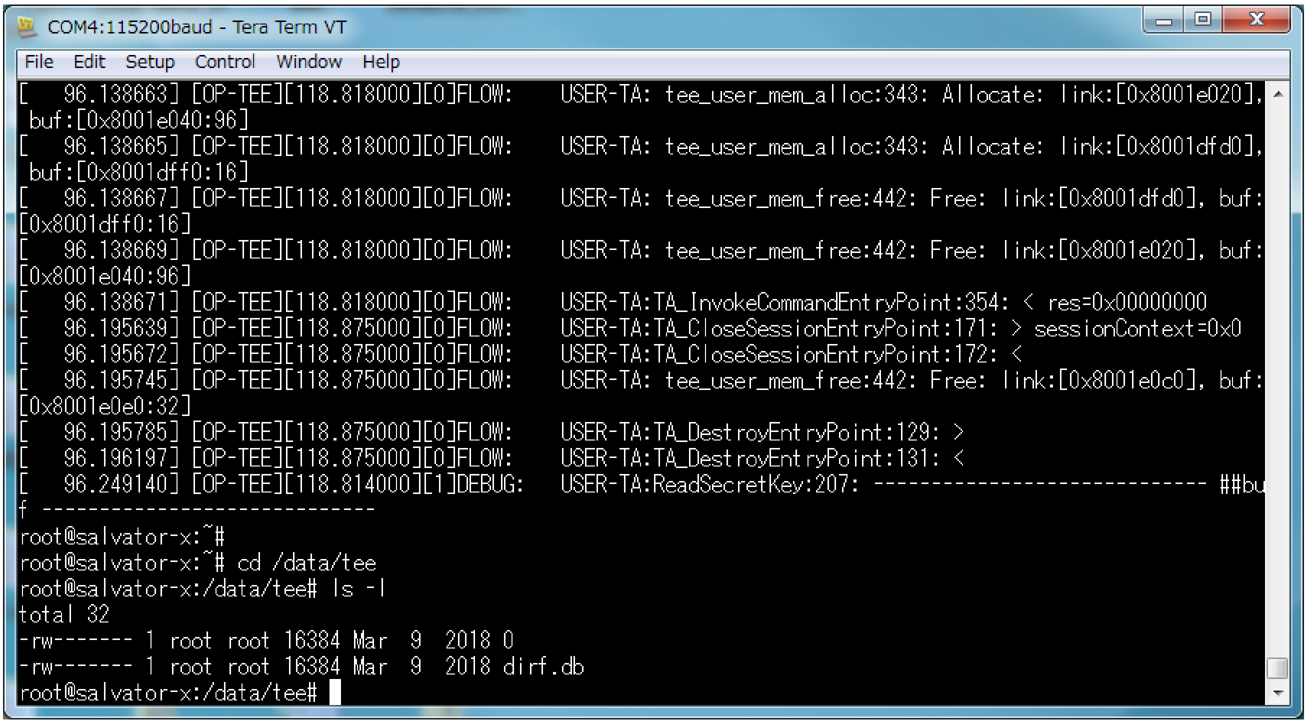
COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
[  OK  ] Started Session c1 of user root.
[  OK  ] Started User Manager for UID 0.

Poky (Yocto Project Reference Distro) 2.1.2 salvator-x ttySC0

salvator-x login: root
Last login: Mon Aug  7 17:04:42 UTC 2017 on tty2
root@salvator-x:~#
root@salvator-x:~#
root@salvator-x:~# /usr/bin/sampleca cbc enc /home/root/a.txt /home/root/b.txt
root@salva
root@salv
total 4
drw------
root@salvator-x:/data/te                         11A47F0002A5D5C51B
root@salvator-x:/data/tee/80                     7F0002A5D5C51B# ls -l
total 4
drw------- 2 root root 4096 Mar  9  20              0304
root@salvator-x:/data/tee/8075A72F0AECE511A        0002A5D5C51B# cd 01020304
root@salvator-x:/data/tee/8075A72F0AECE511A47F0002A5D5C51B/01020304# ls -l
total 8
-rw------- 1 root root  96 Mar  9  2018 block0.0
-rw------- 1 root root 180 Mar  9  2018 meta.1
root@salvator-x:/data/tee/8075A72F0AECE511A47F0002A5D5C51B/
```

#define TA_CRYPTO_UUID { 0x2FA77580U, 0xEC0AU, 0x11E5U, ¥
{ 0xA4U, 0x7FU, 0x00U, 0x02U, 0xA5U, 0xD5U, 0xC5U, 0x1BU} }

objID[] = {0x01U,0x02U,0x03U,0x04U}

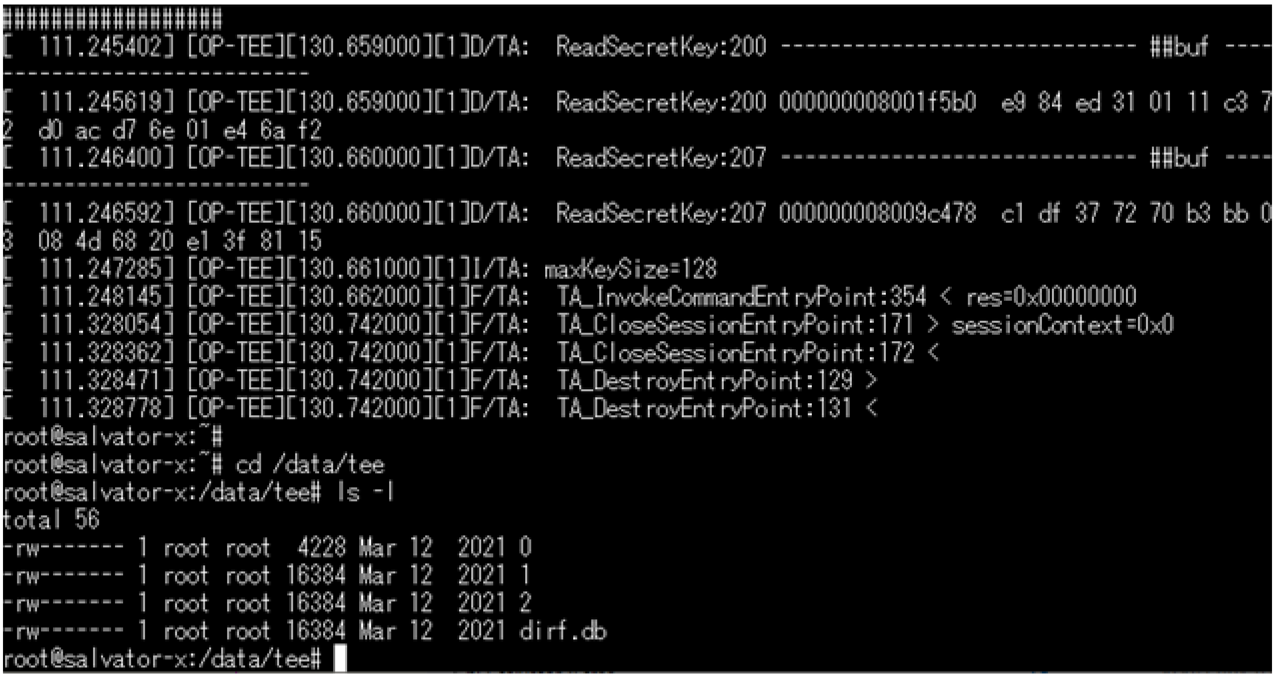
Note) Storage is an example of REE FS (Yocto v2.23.0).
The secret key is created only the first time.



COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
[   96.138663] [OP-TEE][118.818000][0]FLOW:    USER-TA: tee_user_mem_alloc:343: Allocate: link:[0x8001e020],
 buf:[0x8001e040:96]
[   96.138665] [OP-TEE][118.818000][0]FLOW:    USER-TA: tee_user_mem_alloc:343: Allocate: link:[0x8001dfd0],
 buf:[0x8001dff0:16]
[   96.138667] [OP-TEE][118.818000][0]FLOW:    USER-TA: tee_user_mem_free:442: Free: link:[0x8001dfd0], buf:
[0x8001dff0:16]
[   96.138669] [OP-TEE][118.818000][0]FLOW:    USER-TA: tee_user_mem_free:442: Free: link:[0x8001e020], buf:
[0x8001e040:96]
[   96.138671] [OP-TEE][118.818000][0]FLOW:    USER-TA:TA_InvokeCommandEntryPoint:354: < res=0x00000000
[   96.195639] [OP-TEE][118.875000][0]FLOW:    USER-TA:TA_CloseSessionEntryPoint:171: > sessionContext=0x0
[   96.195672] [OP-TEE][118.875000][0]FLOW:    USER-TA:TA_CloseSessionEntryPoint:172: <
[   96.195745] [OP-TEE][118.875000][0]FLOW:    USER-TA: tee_user_mem_free:442: Free: link:[0x8001e0c0], buf:
[0x8001e0e0:32]
[   96.195785] [OP-TEE][118.875000][0]FLOW:    USER-TA:TA_DestroyEntryPoint:129: >
[   96.196197] [OP-TEE][118.875000][0]FLOW:    USER-TA:TA_DestroyEntryPoint:131: <
[   96.249140] [OP-TEE][118.814000][1]DEBUG:   USER-TA:ReadSecretKey:207: ---------------------------- ##bu
f ----------------------------
root@salvator-x:~#
root@salvator-x:~# cd /data/tee
root@salvator-x:/data/tee# ls -l
total 32
-rw------- 1 root root 16384 Mar  9  2018 0
-rw------- 1 root root 16384 Mar  9  2018 dirf.db
root@salvator-x:/data/tee#
```

Note) Storage is an example of REE FS (Yocto v3.6.0).
The secret key is created only the first time.



```
###################
[  111.245402] [OP-TEE][130.659000][1]D/TA:  ReadSecretKey:200 ----------------------------- ##buf ----
-------------------------
[  111.245619] [OP-TEE][130.659000][1]D/TA:  ReadSecretKey:200 000000008001f5b0  e9 84 ed 31 01 11 c3 7
2  d0 ac d7 6e 01 e4 6a f2
[  111.246400] [OP-TEE][130.660000][1]D/TA:  ReadSecretKey:207 ----------------------------- ##buf ----
-------------------------
[  111.246592] [OP-TEE][130.660000][1]D/TA:  ReadSecretKey:207 000000008009c478  c1 df 37 72 70 b3 bb 0
3  08 4d 68 20 e1 3f 81 15
[  111.247285] [OP-TEE][130.661000][1]I/TA: maxKeySize=128
[  111.248145] [OP-TEE][130.662000][1]F/TA:  TA_InvokeCommandEntryPoint:354 < res=0x00000000
[  111.328054] [OP-TEE][130.742000][1]F/TA:  TA_CloseSessionEntryPoint:171 > sessionContext=0x0
[  111.328362] [OP-TEE][130.742000][1]F/TA:  TA_CloseSessionEntryPoint:172 <
[  111.328471] [OP-TEE][130.742000][1]F/TA:  TA_DestroyEntryPoint:129 >
[  111.328778] [OP-TEE][130.742000][1]F/TA:  TA_DestroyEntryPoint:131 <
root@salvator-x:~#
root@salvator-x:~# cd /data/tee
root@salvator-x:/data/tee# ls -l
total 56
-rw------- 1 root root  4228 Mar 12  2021 0
-rw------- 1 root root 16384 Mar 12  2021 1
-rw------- 1 root root 16384 Mar 12  2021 2
-rw------- 1 root root 16384 Mar 12  2021 dirf.db
root@salvator-x:/data/tee#
```

Note) Storage is an example of REE FS (Yocto v5.1.0).
The secret key is created only the first time.

TEE_OpenPersistentObject will succeed next time and thereafter.
Success will create a secret key based on the data read from the storage.

```
/*****************************************************************************/
/* Global Variables                                                          */
/*****************************************************************************/
static uint8_t      objID[]    = {0x01U,0x02U,0x03U,0x04U};
static uint32_t     objID_len = 4U;

/* Secret key */
static TEE_ObjectHandle secretKey;

/* IV */
static uint8_t      iv[16];
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
static TEE_Result ReadSecretKey(void)
{
    TEE_Result          res;
    TEE_Attribute       attrs[1];
    uint8_t             keyData[16];
    uint32_t            readSize;
    TEE_ObjectHandle    object;

    /* Opens a handle on an existing persistent object. */
    res = TEE_OpenPersistentObject((uint32_t)TEE_STORAGE_PRIVATE, objID, objID_len,
        (uint32_t)TEE_DATA_FLAG_ACCESS_READ, &object);
    if (res == (TEE_Result)TEE_SUCCESS) {
        IMSG("###################################Second Access###################################¥n");

        /* Read the key data. */
        res = TEE_ReadObjectData(object, (void*)keyData, sizeof(keyData), &readSize);
        if (res != (TEE_Result)TEE_SUCCESS) {
            EMSG("TEE_ReadObjectData¥n");
        }
        DBUFDUMP(keyData, (int32_t)readSize);

        /* Read the IV. */
        res = TEE_ReadObjectData(object, (void*)iv, sizeof(iv), &readSize);
        if (res != (TEE_Result)TEE_SUCCESS) {
            EMSG("TEE_ReadObjectData¥n");
        }
        DBUFDUMP(iv, (int32_t)readSize);

        /* Create key handle */
        if (res == (TEE_Result)TEE_SUCCESS) {
            TEE_InitRefAttribute(&attrs[0], TEE_ATTR_SECRET_VALUE, (void*)keyData, sizeof(keyData));
            res = TEE_AllocateTransientObject(TEE_TYPE_AES, KEY_SIZE, &secretKey);
            if (res != (TEE_Result)TEE_SUCCESS) {
                EMSG("Error TEE_AllocateTransientObject¥n");
            }
        }
        if (res == (TEE_Result)TEE_SUCCESS) {
            res = TEE_PopulateTransientObject(secretKey, attrs, 1U);
            if (res != (TEE_Result)TEE_SUCCESS) {
                EMSG("Error TEE_PopulateTransientObject¥n");
            }
        }
        TEE_CloseObject(object);
    }

    return res;
}
```

Cryptographic

The following sample source code is an excerpt of encrypt or decrypt.
Please refer to the source for details because the following is an excerpt.

```
/* Secret key */
static TEE_ObjectHandle secretKey;
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TEE_Result TA_InvokeCommandEntryPoint(void *sessionContext, uint32_t commandID,
            uint32_t paramTypes, TEE_PARAM_ARGV params)
{
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    /* Reads the secret key */
    res = ReadSecretKey();
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    /* To encrypt or decrypt the input data with the secret key */
    if (res == (TEE_Result)TEE_SUCCESS) {
        TEE_GetObjectInfo1(secretKey, &info);
        IMSG("maxKeySize=%d¥n", info.maxKeySize);

        res = TEE_AllocateOperation(&op, Algo, Mode, info.maxKeySize);
        if (res != (TEE_Result)TEE_SUCCESS) {
            EMSG("Error TEE_AllocateOperation maxKeySize=%d¥n", info.maxKeySize);
        }
    }
    if (res == (TEE_Result)TEE_SUCCESS) {
        res = TEE_SetOperationKey(op, secretKey);
        if (res != (TEE_Result)TEE_SUCCESS) {
            EMSG("Error TEE_SetOperationKey¥n");
        }
    }
    if (res == (TEE_Result)TEE_SUCCESS) {
        TEE_CipherInit(op, iv, sizeof(iv));
        pos = 0U;
        Remainder = inbuf_size;
        /*
         * TEE_CipherUpdate since the update process is called in TEE_CipherDoFinal is not required.
         * The following comment out is an example when carry out TEE_CipherUpdate.
         * Big data can not send at one time to Secure World from Normal World.
         * If this is the case, you will need to implement separately in TEE_CipherUpdate and TEE_CipherDoFinal.
         */
        //while ((Remainder % BLOCK_SIZE) == 0U) {
        //    res = TEE_CipherUpdate(op, &inbuf[pos], BLOCK_SIZE, &outbuf[pos], &outbuf_size);
        //    if (res != (TEE_Result)TEE_SUCCESS) {
        //        EMSG("Error TEE_CipherUpdate¥n");
        //        break;
        //    }
        //    pos += BLOCK_SIZE;
        //    Remainder -= BLOCK_SIZE;
        //
        //    /* Exit to leave the last of the block */
        //    if (Remainder == BLOCK_SIZE) {
        //        break;
        //    }
        //}
        /* Last block */
        if (res == (TEE_Result)TEE_SUCCESS) {
            res = TEE_CipherDoFinal(op, &inbuf[pos], Remainder, &outbuf[pos], &outbuf_size);
            if (res != (TEE_Result)TEE_SUCCESS) {
                EMSG("Error TEE_CipherDoFinal¥n");
            }
        }
    }
    /* Free OperationHandle */
    if (op != TEE_HANDLE_NULL) {
        TEE_FreeOperation(op);
    }
    OUTMSG("res=0x%08x¥n", res);
    return res;
}
```

Revision History	Sample Dynamic TA Application Note

Rev.	Date	Description	
		Page	Summary
1.0.0	Apr. 15, 2016	—	First Edition issued.
1.0.1	Aug. 24, 2016	1	Version of a reference Security BSP User's Manual has been updated.
1.0.2	Dec. 22, 2016	9	Add a description of input data.
1.0.3	Feb. 15, 2017	1	Version of a reference Security BSP User's Manual has been updated.
1.0.4	Jul. 12, 2017	—	Fixed the format of the document (trademark, etc.)
1.0.5	Mar. 14, 2018	—	Fixed the format of the document (trademark, etc.)
		3	Add M3N and E3 board.
		11	Fixed the code in the appendix.
		14	Add REE FS output image of Yocto v2.23.0 and Yocto v3.6.0.
2.0.0	Nov. 5, 2018	—	Fixed the format of the document(Address List.)
		3	Add Ebisu-4D board.
3.0.2	Apr. 6, 2021	2	Changed ARM notation to Arm.
		3	Add D3 board.
		5	Changed the software name from ARM Trusted Firmware to Trusted Firmware-A.
		15	Add REE FS output image of Yocto v5.1.0.
3.0.3	Aug. 16, 2021	—	Fixed the format of the document (Notice, Address List.)
		—	Add information of Gen3e.

CONFIDENTIAL

Sample Dynamic TA
Application Note: Software

Publication Date: Rev.1.0.0 Apr. 15, 2016
Rev.3.0.3 Aug. 16, 2021

Published by: Renesas Electronics Corporation

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics Corporation
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

Renesas Electronics America Inc. Milpitas Campus
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics America Inc. San Jose Campus
6024 Silver Creek Valley Road, San Jose, CA 95138, USA
Tel: +1-408-284-8200, Fax: +1-408-284-2775

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 101-T01, Floor 1, Building 7, Yard No. 7, 8th Street, Shangdi, Haidian District, Beijing 100085, China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai 200333, China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, #06-02 Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit No 3A-1 Level 3A Tower 8 UOA Business Park, No 1 Jalan Pengaturcara U1/51A, Seksyen U1, 40150 Shah Alam, Selangor, Malaysia
Tel: +60-3-5022-1288, Fax: +60-3-5022-1290

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.0

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

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）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/

Colophon 3.1

Sample Dynamic TA
Application Note: Software

Renesas Electronics Corporation