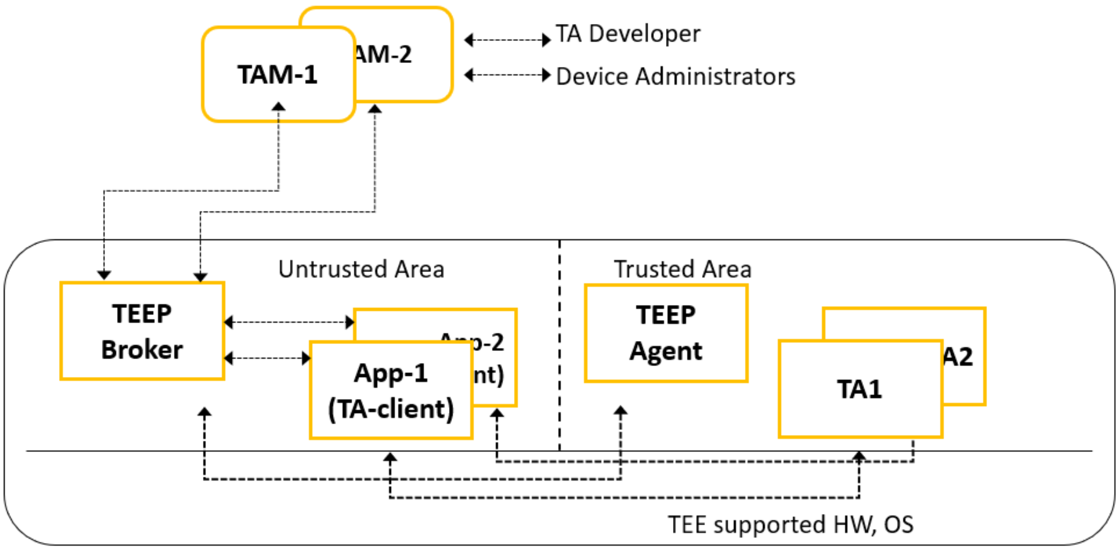
TEEP-DEVICE

The National Institute of Advanced Industrial Science and Technology

2022-02-08

1 Overview of TEEP-Device

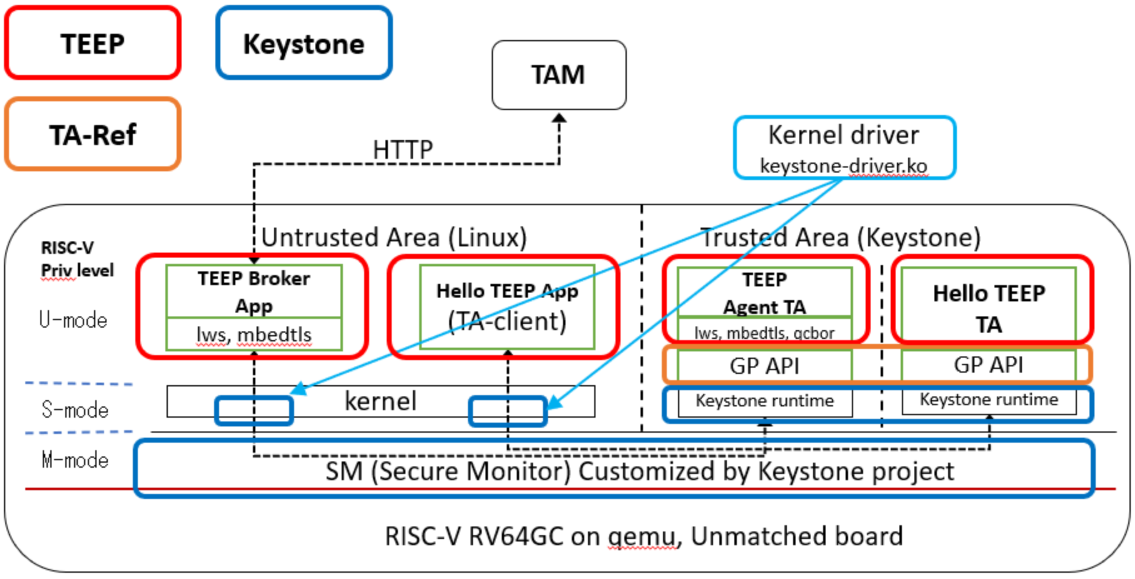


1.1 Features of TEEP-Device

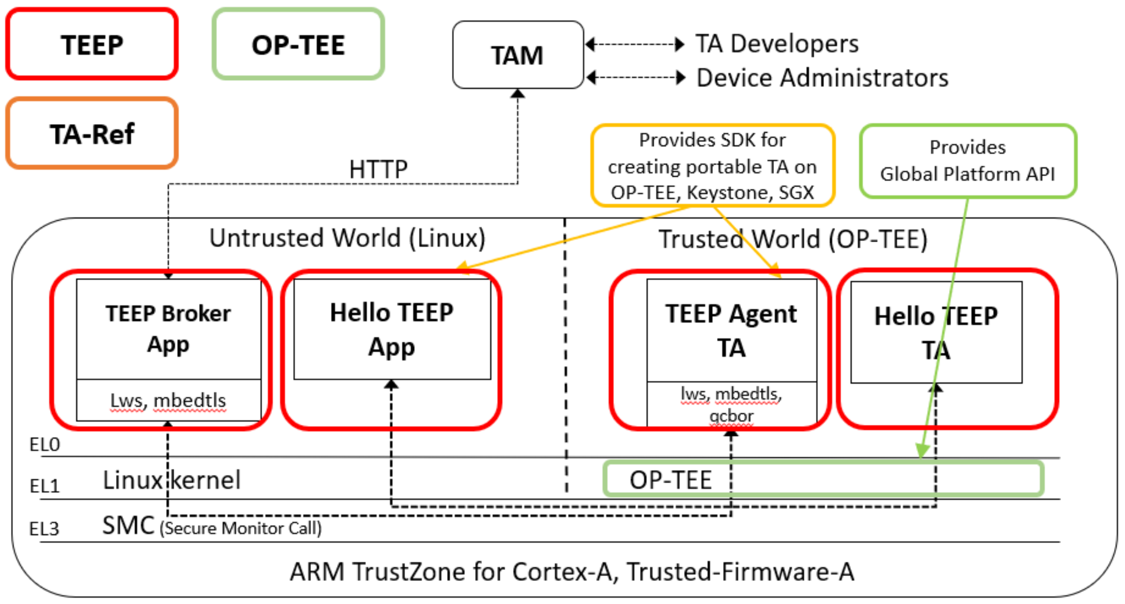
- AIST will prepare

1.2 Components of TEEP-device and TA-Ref

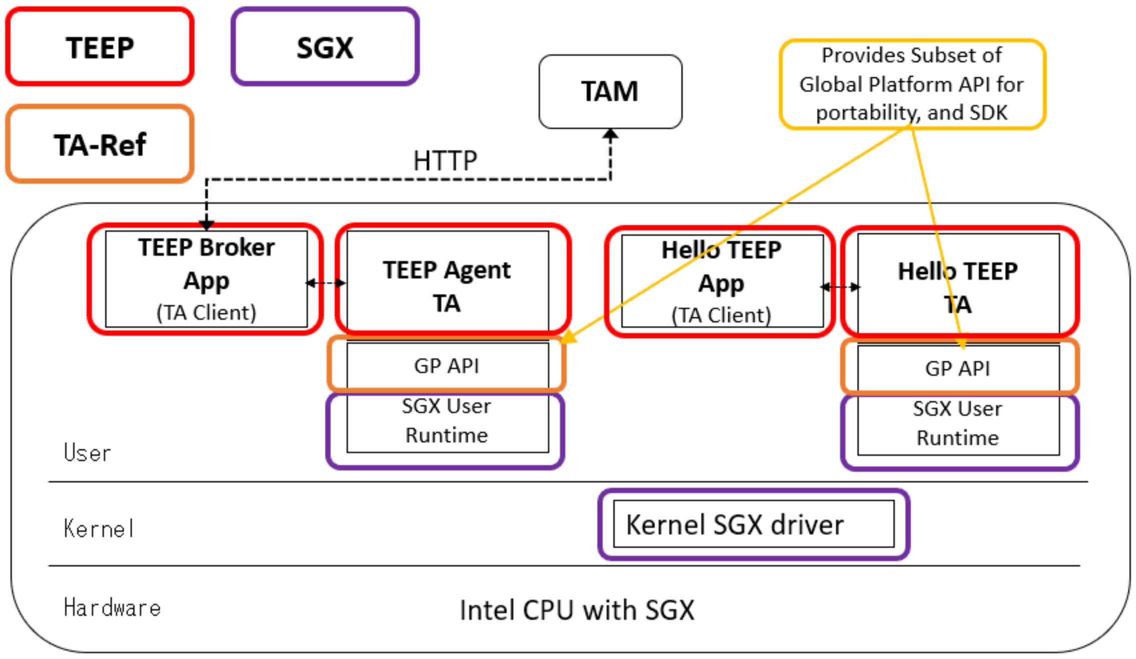
1.2.1 TEEP-device and TA-Ref Components on Keystone



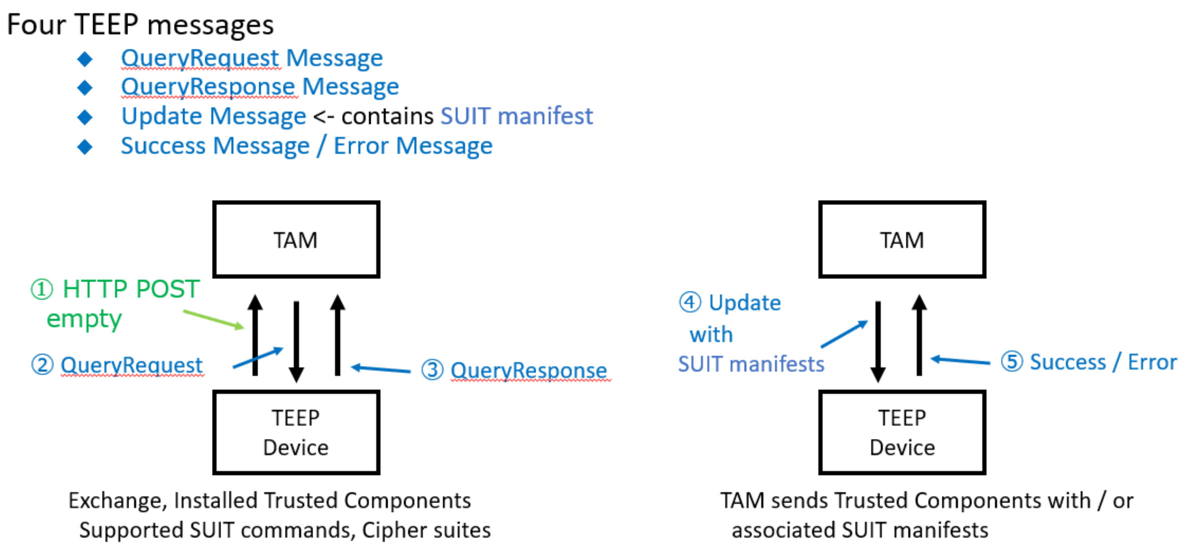
1.2.2 TEEP-device and TA-Ref Components on OP-TEE



1.2.3 TEEP-device and TA-Ref Components on SGX



2 TEEP-DEVICE Operations



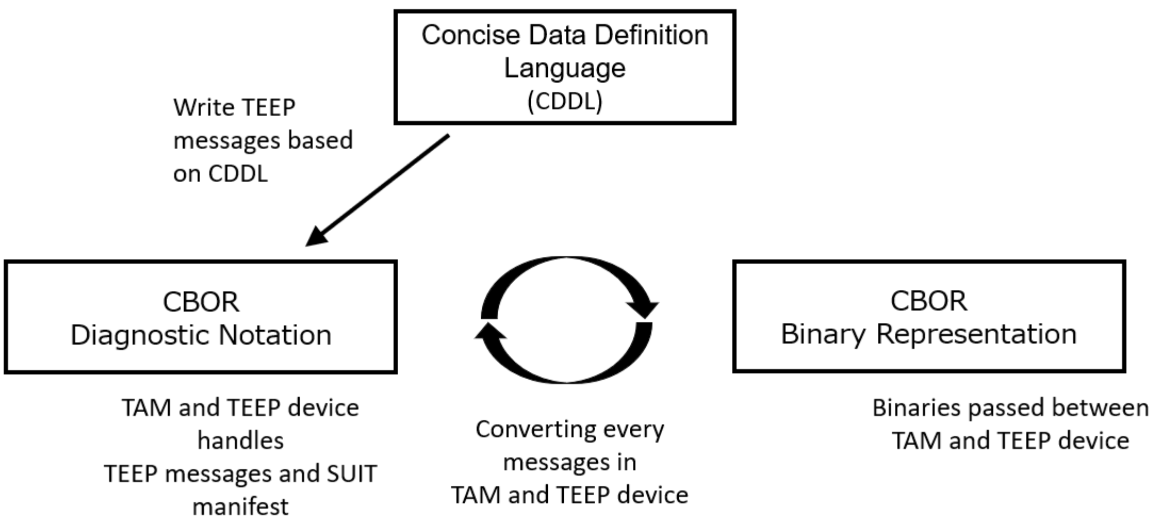
Four TEEP messages

- QueryRequest Message
- QueryResponse Message
- Update Message <- contains SUIT manifest
- Success Message / Error Message

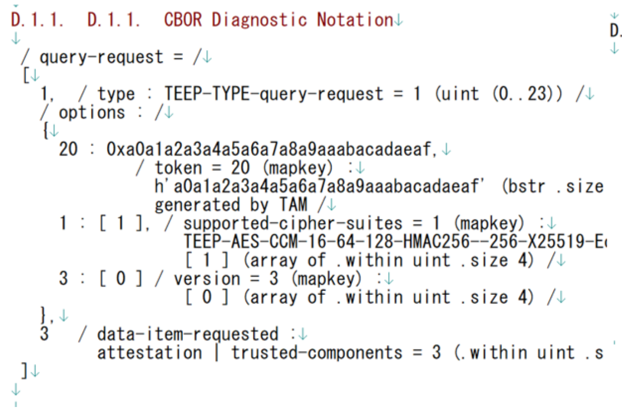
TAM
① HTTP POST empty
② QueryRequest
③ QueryResponse
TEEP Device
Exchange, Installed Trusted Components
Supported SUIT commands, Cipher suites
TAM
④ Update with SUIT manifests
⑤ Success / Error
TEEP Device
TAM sends Trusted Components with / or associated SUIT manifests

3 CBOR in TEEP-Device

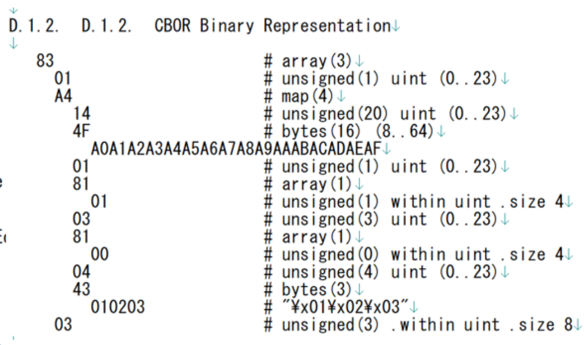
3.1 Three format representations in TEEP and SUIT


Concise Data Definition Language (CDDL)
Write TEEP messages based on CDDL
CBOR Diagnostic Notation
TAM and TEEP device handles TEEP messages and SUIT manifest
Converting every messages in TAM and TEEP device
CBOR Binary Representation
Binaries passed between TAM and TEEP device

3.2 TEEP message format examples

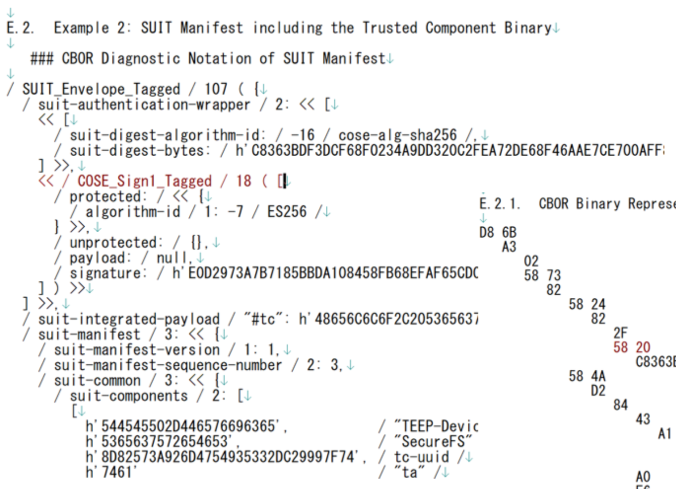


```
D.1.1.  D.1.1.  CBOR Diagnostic Notation

 / query-request = /
 [
   1,  / type : TEEP-TYPE-query-request = 1 (uint (0..23)) /
   / options : /
   {
     20 : 0xa0a1a2a3a4a5a6a7a8a9aaabacadaeaf,
             / token = 20 (mapkey) :
               h'a0a1a2a3a4a5a6a7a8a9aaabacadaeaf' (bstr .size
               generated by TAM /
     1 : [ 1 ], / supported-cipher-suites = 1 (mapkey) :
                  TEEP-AES-CCM-16-64-128-HMAC256--256-X25519-E
                  [ 1 ] (array of .within uint .size 4) /
     3 : [ 0 ] / version = 3 (mapkey) :
                  [ 0 ] (array of .within uint .size 4) /
   },
   3   / data-item-requested :
         attestation | trusted-components = 3 (.within uint .s
 ]
```

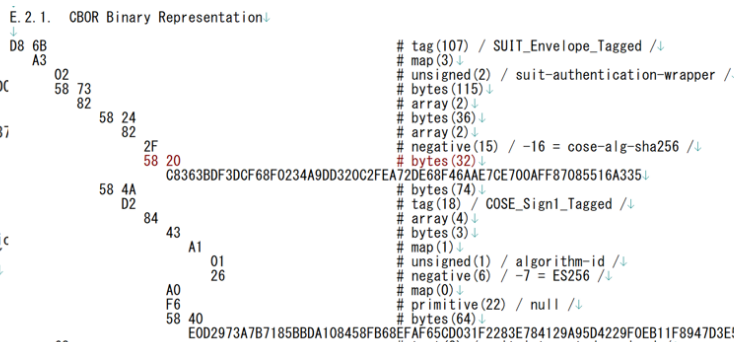


```
D.1.2.  D.1.2.  CBOR Binary Representation

  83                          # array(3)
    01                        # unsigned(1) uint (0..23)
    A4                        # map(4)
      14                      # unsigned(20) uint (0..23)
      4F                      # bytes(16) (8..64)
        A0A1A2A3A4A5A6A7A8A9AAABACADAEAF
      01                      # unsigned(1) uint (0..23)
      81                      # array(1)
        01                    # unsigned(1) within uint .size 4
      03                      # unsigned(3) uint (0..23)
      81                      # array(1)
        00                    # unsigned(0) within uint .size 4
      04                      # unsigned(4) uint (0..23)
      43                      # bytes(3)
        010203                # "¥x01¥x02¥x03"
    03                        # unsigned(3) .within uint .size 8
```

3.3 SUIT message format examples



```
E.2.  Example 2: SUIT Manifest including the Trusted Component Binary

   ### CBOR Diagnostic Notation of SUIT Manifest

/ SUIT_Envelope_Tagged / 107 ( {
  / suit-authentication-wrapper / 2: << [
    << [
      / suit-digest-algorithm-id: / -16 / cose-alg-sha256 /,
      / suit-digest-bytes: / h'C8363BDF3DCF68F0234A9DD320C2FEA72DE68F46AAE7CE700AFF
    ] >>,
    << / COSE_Sign1_Tagged / 18 ( [
      / protected: / << {
        / algorithm-id / 1: -7 / ES256 /
      } >>,
      / unprotected: / {},
      / payload: / null,
      / signature: / h'E0D2973A7B7185BBDA108458FB68EFAF65CDC
    ] ) >>
  ] >>,
  / suit-integrated-payload / "#tc": h'48656C6C6F2C20536563
  / suit-manifest / 3: << {
    / suit-manifest-version / 1: 1,
    / suit-manifest-sequence-number / 2: 3,
    / suit-common / 3: << {
      / suit-components / 2: [
        [
          h'544545502D446576696365',          / "TEEP-Devic
          h'5365637572654653',                / "SecureFS"
          h'8D82573A926D4754935332DC29997F74', / tc-uuid /
          h'7461'                             / "ta" /
```



```
E.2.1.  CBOR Binary Representation

D8 6B                                                 # tag(107) / SUIT_Envelope_Tagged /
   A3                                                 # map(3)
      02                                              # unsigned(2) / suit-authentication-wrapper /
      58 73                                           # bytes(115)
         82                                           # array(2)
            58 24                                     # bytes(36)
               82                                     # array(2)
                  2F                                  # negative(15) / -16 = cose-alg-sha256 /
                  58 20                               # bytes(32)
                     C8363BDF3DCF68F0234A9DD320C2FEA72DE68F46AAE7CE700AFF87085516A335
            58 4A                                     # bytes(74)
               D2                                     # tag(18) / COSE_Sign1_Tagged /
                  84                                  # array(4)
                     43                               # bytes(3)
                        A1                            # map(1)
                           01                         # unsigned(1) / algorithm-id /
                           26                         # negative(6) / -7 = ES256 /
                     A0                               # map(0)
                     F6                               # primitive(22) / null /
                     58 40                            # bytes(64)
                        E0D2973A7B7185BBDA108458FB68EFAF65CD031F2283E784129A95D4229F0EB11F8947D3E
```

4 TEEP-Device with docker

4.1 Preparation for Docker

For building teep-device with docker, it is required to install docker on Ubuntu.

For the first time users of docker, please have a look on `https://docs.docker.com/engine/`

The following installation steps is for Ubuntu 20.04

4.1.1 Installing Docker

```
$ sudo apt update

# Next, install a few prerequisite packages which let apt use packages over HTTPS:
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common

# Then add the GPG key for the official Docker repository to your system:
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

# Add the Docker repository to APT sources:
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"

# This will also update our package database with the Docker packages from the newly added repo.
# Make sure you are about to install from the Docker repo instead of the default Ubuntu repo:
$ apt-cache policy docker-ce

#Finally, install Docker
$ sudo apt install docker-ce
```

4.1.2 Executing Docker without sudo

By default, the docker command can only be run the root user or by a user in the docker group, which is automatically created during Docker's installation process. If you attempt to run the docker command without prefixing it with sudo or without being in the docker group, you'll get an output like this:

```
docker: Cannot connect to the Docker daemon. Is the docker daemon running on this host?.
```

To avoid typing sudo whenever we run the docker command, add your username to the docker group.

```
$ sudo groupadd docker
$ sudo gpasswd -a $USER docker
# Logout and then log-in again to apply the changes to the group
```

After you logout and login, you can probably run the docker command without `sudo`

```
$ docker run hello-world
```

4.1.3 Create a docker network tamproto

A docker network named tamproto is required when we run teep device with ta-ref for all targets. The local network is required to connect with tamproto service running locally.

```
$ docker network create tamproto_default
```

4.2 Pre-built Docker Image details

The following are the docker images that has pre-built and tested binaries of teep-device with ta-ref. Since this images are already prepared and built already, you can start using it directly without building the teep-device again. Make sure you have account on docker-hub. If not please create one on `dockerhub.com`

Target	docker image
Keystone	trasioteam/teep-dev:keystone
OP-TEE	trasioteam/teep-dev:optee
Intel SGX	trasioteam/teep-dev:sgx
Tamproto	trasioteam/teep-dev:tamproto
Doxygen	trasioteam/teep-dev:doxygen

4.3 Prepartion for building teep-device on docker

4.3.1 Docker images details for building

If we need to build the teep-device, docker images with all necessary packages for building teep-device for all three targets are already available. The details are mentioned below.

Target	docker image
Keystone	trasioteam/taref-dev:keystone
OP-TEE	trasioteam/taref-dev:optee
Intel SGX	trasioteam/taref-dev:sgx
Doxygen	trasioteam/taref-dev:doxygen

5 Clone and Building teep-device without docker

Clone the teep-device source code and build it for Keystone, OPTEE and SGX. To build please refer to ta-ref.pdf->preparation section

- `https://192.168.100.↩`
  `100/rinkai/ta-ref/-/blob/teep-device-tb-slim/docs/ta-ref.pdf`

5.1 Install Doxygen-1.9.2

This PDF was generated using Doxygen version 1.9.2. To install `doxygen-1.9.2` following procedure is necessary.

5.1.1 Install Required Packages

Install following packages on Ubuntu 18.04

```
sudo apt install doxygen-latex graphviz texlive-full texlive-latex-base latex-cjk-all
```

Above packages required to generate PDF using doxygen.

5.1.2 Build and Install

```
git clone https://github.com/doxygen/doxygen.git
cd doxygen
mkdir build
cd build
cmake -G "Unix Makefiles" ..
make
sudo make install
```

5.2 Tamproto Setup

To test teep-device, have to run TAM server on the PC.

Prerequisites

```
sudo apt install rustc npm
sudo pip3 install --upgrade git+https://github.com/ARMmbed/suit-manifest-generator.git@v0.0.2
```

Build and Install

```
git clone https://github.com/ko-isobe/tamproto.git
cd tamproto
git checkout cef99c07b669a49c2748b0c0ff0412ec1628b686 -b 2020-12-18
npm install
```

Make sure your PC is configures with IP address for network connectivity with TEEP device for further testing.

5.3 Keystone

Build `teep-device` with Keystone. Make sure Keystone and its supporting sources have been build already.

5.3.1 Clone and Build

Prepare the environment setup

```
export TEE=keystone
export KEYSTONE_DIR=<path to keystone dir>
export PATH=$PATH:$KEYSTONE_DIR/riscv/bin
export KEYEDGE_DIR=<path tokeyedge dir>
export KEEDGER8R_DIR=<path to keedger8r dir>
```

Clone and Build

```
git clone https://192.168.100.100/rinkai/teep-device.git
cd teep-device
git submodule sync --recursive
git submodule update --init --recursive
make
```

5.3.2 Check teep-device by running hello-app and teep-broker-app

To check teep-device on Unleased, we need to run TAM server and networking with Unleased dev board

5.3.3 Run Tamproto (TAM Server)

First start the TAM server on PC. Make sure IP address configured on PC and Unleased development board.

```
cd tamproto
npm app.js
JWKBaseKeyObject {
  keystore: JWKStore {},
  length: 4096,
  kty: 'RSA',
  kid: 'sWpWma0lDp_RfHKdtkGSVTYQaMIVQaKhESVmzjaW9jc',
  use: ",
  alg: " }
192.168.0.5
Express HTTP  server listening on port 8888
Express HTTPS server listening on port 8443
```

Once TAM server is up, you see above messages

5.3.4 Copy the hello-app and teep-broker-app binaries to Unleased

5.3.4.1 Manual Copy

- Conneect to Unleased over serial console then assign IP address `ifconfig eth0 192.168.0.6`
- Copy the binaries from build PC over SSH (user:root, password: sifive)

Here `192.168.0.6` is IP Address of Unleased board

```
scp platform/keystone/build/hello-ta/hello-ta root@192.168.0.6:/root/teep-device
scp platform/keystone/build/hello-app/hello-app root@192.168.0.6:/root/teep-device
scp platform/keystone/build/teep-agent-ta/teep-agent-ta root@192.168.0.6:/root/teep-device
scp platform/keystone/build/teep-broker-app/teep-broker-app root@192.168.0.6:/root/teep-device
scp $KEYSTONE_DIR/sdk/rts/eyrie/eyrie-rt root@192.168.0.6:/root/teep-device
scp platform/keystone/build/libteep/ree/mbedtls/library/lib* root@192.168.0.6:/usr/lib/
scp platform/keystone/build/libteep/ree/libwebsockets/lib/lib* root@192.168.0.6:/usr/lib/
```

5.3.4.2 Write to SD card

Please follow below below steps to write the teep-device binaries to SD-card

- Insert SD card to your PC for Unleashed
- Edit `platform/keystone/script/sktinst.sh`
  - Check SD-card device name detected on yor PC and fix `prefix=?`
  - `export prefix=/dev/mmcblk0`
- execute `script/sktinst.sh` as follows
  - `cd platform/keystone; script/sktinst.sh`
- Move the sd to unleashed board and boot it

5.3.5 Check hello-app and teep-broker-app on Unleased

There are two methods to connect to Unleased.

- Serial Port using minicom (/dev/ttyUSB0)
- Over SSH: `ssh root@192.168.0.6`; password is `sifive`

Setup envrionment in Unleased (create /root/env.sh file and add following lines)

```
export PATH=$PATH:/root/teep-device
export TAM_HOST=tamproto_tam_api_1
export TAM_PORT=8888
insmod keystone-driver.ko
```

5.3.5.1 Run hello-app

```
$ source env.sh
[ 2380.618514] keystone_driver: loading out-of-tree module taints kernel.
[ 2380.625305] keystone_enclave: keystone enclave v0.2
$ cd teep-device/
$ ./hello-app hello-ta eyrie-rt
hello TA
$
```

5.3.5.2 Run teep-broker-app

Use the TAM server IP address (i.e 192.168.0.5)

```
./teep-broker-app --tamurl http://192.168.0.5:8888/api/tam_cbor
```

Upon execution, you see following log

```
teep-bro[ 2932.269897] ------------[ cut here ]------------
[ 2932.274191] WARNING: CPU: 4 PID: 164 at /home/arun/projects/ks-0.3/keystone/riscv-linux/mm/page_alloc.c:3926
       __alloc_pages_nodemask+0x150/a
[ 2932.287053] Modules linked in: keystone_driver(O)
[ 2932.291716] CPU: 4 PID: 164 Comm: teep-broker-app Tainted: G        W  O     4.15.0-00060-g65e929792fb9-dirty
       #4
[ 2932.301867] Call Trace:
[ 2932.304314] [<0000000036e46dc0>] walk_stackframe+0x0/0xa2
[ 2932.309686] [<00000000893dfe1c>] show_stack+0x26/0x34
[ 2932.314725] [<00000000c57ed7ce>] dump_stack+0x5e/0x7c
[ 2932.319759] [<00000000a68ce031>] __warn+0xca/0xe0
[ 2932.324445] [<00000000bec1f8a6>] warn_slowpath_null+0x2c/0x3e
[ 2932.330176] [<00000000e8c56bf2>] __alloc_pages_nodemask+0x14c/0x8da
[ 2932.336426] [<00000000ec1f9596>] __get_free_pages+0xc/0x52
[ 2932.341920] [<000000003e8cccc8>] epm_init+0x158/0x1a0 [keystone_driver]
[ 2932.348502] [<0000000032e4188b>] create_enclave+0x56/0xb0 [keystone_driver]
[ 2932.355447] [<000000008a656a96>] keystone_create_enclave+0x16/0x40 [keystone_driver]
[ 2932.363174] [<000000003bbf2147>] keystone_ioctl+0x132/0x164 [keystone_driver]
[ 2932.370288] [<00000000755f7993>] do_vfs_ioctl+0x76/0x4f4
[ 2932.375582] [<00000000b88b9c1d>] SyS_ioctl+0x36/0x60
[ 2932.380533] [<00000000aae667a5>] check_syscall_nr+0x1e/0x22
[ 2932.386132] ---[ end trace 66814e3a8c80ec12 ]---
ker.c compiled at Feb 16 2021 11:17:21
uri = http://192.168.0.5:8888/api/tam_cbor, cose=0, talist=
[1970/01/01 00:48:56:0796] NOTICE: POST: http://192.168.0.5:8888/api/tam_cbor
[1970/01/01 00:48:56:0798] NOTICE: (hexdump: zero length)
[1970/01/01 00:48:56:0801] NOTICE: created client ssl context for default
[1970/01/01 00:48:56:0802] NOTICE: http://192.168.0.3:8888/api/tam_cbor
[1970/01/01 00:48:56:0861] NOTICE:
[1970/01/01 00:48:56:0862] NOTICE: 0000: 83 01 A4 01 81 01 03 81 00 14 1A 77 77 77 77 04
       ...........wwww.
[1970/01/01 00:48:56:0862] NOTICE: 0010: 43 01 02 03 02                                      C....

[1970/01/01 00:48:56:0862] NOTICE:
[1970/01/01 00:48:56:0871] NOTICE: POST: http://192.168.0.5:8888/api/tam_cbor
[1970/01/01 00:48:56:0871] NOTICE:
[1970/01/01 00:48:56:0871] NOTICE: 0000: 82 02 A4 14 1A 77 77 77 77 08 80 0E 80 0F 80
       .....wwww......
[1970/01/01 00:48:56:0872] NOTICE:
[1970/01/01 00:48:56:0873] NOTICE: created client ssl context for default
[1970/01/01 00:48:56:0874] NOTICE: http://192.168.0.5:8888/api/tam_cbor
[1970/01/01 00:48:56:0962] NOTICE:
[1970/01/01 00:48:56:0962] NOTICE: 0000: 82 03 A2 0A 81 59 01 37 A2 02 58 72 81 58 6F D2
       .....Y.7..Xr.Xo.
[1970/01/01 00:48:56:0963] NOTICE: 0010: 84 43 A1 01 26 A0 58 24 82 02 58 20 75 80 7C 54
       .C..&.X$..X u.|T
[1970/01/01 00:48:56:0963] NOTICE: 0020: 62 40 D2 14 E5 7B D5 C4 6A 7C E5 2D ED B0 3D 0E
       b@...{..j|.-..=.
[1970/01/01 00:48:56:0964] NOTICE: 0030: CC 80 75 F3 F7 E0 65 B3 60 CE AD 85 58 40 54 81
       ..u...e.`...X@T.
[1970/01/01 00:48:56:0964] NOTICE: 0040: 49 CD CA D8 17 72 CC EA 61 4A 19 99 05 AB 97 33
       I....r..aJ.....3
[1970/01/01 00:48:56:0965] NOTICE: 0050: EA 48 D7 1F 13 AE 33 0D 47 FF F5 B8 6C 5C 9B 7A
       .H....3.G...l\.z
[1970/01/01 00:48:56:0965] NOTICE: 0060: BB 12 BC 2D FE 9C 20 6A C8 7F E2 28 58 74 E0 74    ...-..
       j...(Xt.t
[1970/01/01 00:48:56:0965] NOTICE: 0070: A3 BD C4 DA B9 20 C4 37 35 8F 67 46 90 76 03 58    .....
       .75.gF.v.X
[1970/01/01 00:48:56:0966] NOTICE: 0080: BE A5 01 01 02 01 03 58 60 A2 02 44 81 81 41 00
       .......X`..D..A.
[1970/01/01 00:48:56:0966] NOTICE: 0090: 04 58 56 86 14 A4 01 50 FA 6B 4A 53 D5 AD 5F DF
       .XV....P.kJS.._.
[1970/01/01 00:48:56:0967] NOTICE: 00A0: BE 9D E6 63 E4 D4 1F FE 02 50 14 92 AF 14 25 69
       ...c.....P....%i
[1970/01/01 00:48:56:0967] NOTICE: 00B0: 5E 48 BF 42 9B 2D 51 F2 AB 45 03 58 24 82 02 58
```

```
       ^H.B.-Q..E.X$..X
[1970/01/01 00:48:56:0968] NOTICE: 00C0: 20 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE
       .."3DUfw.......
[1970/01/01 00:48:56:0968] NOTICE: 00D0: FF 01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32
       ..#Eg........vT2
[1970/01/01 00:48:56:0969] NOTICE: 00E0: 10 0E 19 87 D0 01 F6 02 F6 09 58 4E 86 13 A1 15
       ..........XN....
[1970/01/01 00:48:56:0969] NOTICE: 00F0: 78 44 68 74 74 70 3A 2F 2F 31 39 32 2E 31 36 38
       xDhttp://192.168
[1970/01/01 00:48:56:0970] NOTICE: 0100: 2E 31 31 2E 33 3A 38 38 38 38 2F 54 41 73 2F 38
       .0.5:8888/TAs/8
[1970/01/01 00:48:56:0970] NOTICE: 0110: 64 38 32 35 37 33 61 2D 39 32 36 64 2D 34 37 35
       d82573a-926d-475
[1970/01/01 00:48:56:0971] NOTICE: 0120: 34 2D 39 33 35 33 2D 33 32 64 63 32 39 39 39 37
       4-9353-32dc29997
[1970/01/01 00:48:56:0971] NOTICE: 0130: 66 37 34 2E 74 61 15 F6 03 F6 0A 43 82 03 F6 14
       f74.ta.....C....
[1970/01/01 00:48:56:0972] NOTICE: 0140: 1A 77 77 77 78                                     .wwwx

[1970/01/01 00:48:56:0972] NOTICE:
[1970/01/01 00:48:56:0983] NOTICE: GET: http://192.168.0.5:8888/TAs/8d82573a-926d-4754-9353-32dc29997f74.ta
[1970/01/01 00:48:56:0984] NOTICE: created client ssl context for default
[1970/01/01 00:48:56:0985] NOTICE: http://192.168.0.5:8888/TAs/8d82573a-926d-4754-9353-32dc29997f74.ta
teep_message_unwrap_ta_image: msg len 234110
Decrypt
Decrypt OK: length 174887
Verify
Signature OK 0 130552
ta_store_install: ta_image_len = 130552 ta_name=8d82573a-926d-4754-9353-32dc29997f74
[1970/01/01 00:49:01:9453] NOTICE: POST: http://192.168.0.5:8888/api/tam_cbor
[1970/01/01 00:49:01:9454] NOTICE:
[1970/01/01 00:49:01:9454] NOTICE: 0000: 82 05 A1 14 1A 77 77 77 77
       .....wwww
[1970/01/01 00:49:01:9454] NOTICE:
[1970/01/01 00:49:01:9456] NOTICE: created client ssl context for default
[1970/01/01 00:49:01:9457] NOTICE: http://192.168.0.5:8888/api/tam_cbor
[1970/01/01 00:49:01:9505] NOTICE: (hexdump: zero length)
```

5.4 OPTEE

Build `teep-device` with OPTEE. So make sure OPTEE and its supporting sources have been build already.

5.4.1 Clone and Build

Prepare the environment setup

```
export TEE=optee
export OPTEE_DIR=<optee_3.9.0_rpi3 dir>
export PATH=$PATH:$OPTEE_DIR/toolchains/aarch64/bin:$OPTEE_DIR/toolchains/aarch32/bin
```

Clone and Build

```
git clone https://192.168.100.100/rinkai/teep-device.git
cd teep-device
git submodule sync --recursive
git submodule update --init --recursive
make
```

5.4.2 Check teep-device by running hello-app and teep-broker-app on RPI3

To check teep-device on RPI3, we need to run TAM server on PC and networking with RPI3 board

5.4.3 Run Tamproto (TAM Server)

First start the TAM server on PC. Make sure IP address configured on PC and RPI3 board.

```
cd tamproto
npm app.js
JWKBaseKeyObject {
  keystore: JWKStore {},
  length: 4096,
  kty: 'RSA',
  kid: 'sWpWma0lDp_RfHKdtkGSVTYQaMIVQaKhESVmzjaW9jc',
  use: ",
  alg: " }
192.168.0.5
Express HTTP server listening on port 8888
Express HTTPS server listening on port 8443
```

Once TAM server is up, you see above messages

5.4.4 Copy the hello-app and teep-broker-app binaries to RPI3

5.4.4.1 Copy binaries over SSH to RPI3

- Connect to RPI3 over serial console(/dev/ttryUSB0) then assign IP address `ifconfig eth0 192.168.0.7`
- Copy the binaries from build PC over SSH (user:root) to RPI3

```
TODO - Further update required
```

5.4.4.2 Write to SD card

Please follow below below steps to write the teep-device binaries to SD-card

- Insert SD card to your PC for Unleashed
- Copy the binaries to SD card
- Move the sd to RPI3 board and boot it

```
TODO - Further update required
```

5.4.5 Check hello-app and teep-broker-app on RPI3

There are two methods to connect to RPI3.

- Serial Port using minicom (/dev/ttyUSB0)
- Over SSH: `ssh root@192.168.0.7`

```
TODO - Further update required
```

5.4.5.1 Run hello-app

```
TODO - Further update required
```

5.4.5.2 Run teep-broker-app

Use the TAM server IP address (i.e 192.168.0.3)

```
./teep-broker-app --tamurl http://192.168.0.3:8888/api/tam_cbor
```

Execution logs

```
TODO - Further update required
```

5.5 SGX

Build `teep-device` with SGX. Make sure SGX and its supporting sources have been build already.

5.5.1 Clone and Build on SGX

Prepare the environment setup

```
export TEE=pc
source /opt/intel/sgxsdk/environment
```

Clone and Build

```
git clone https://192.168.100.100/rinkai/teep-device.git
cd teep-device
git submodule sync --recursive
git submodule update --init --recursive
make
```

5.5.2 Check teep-device by running hello-app & teep-broker-app on SGX

To check teep-device on SGX, we need to run TAM server on PC and networking with SGX machine

5.5.3 Run Tamproto (TAM Server)

First start the TAM server on PC. Make sure IP address configured on PC and SGX machine.

```
<p />
cd tamproto
npm app.js
JWKBaseKeyObject {
  keystore: JWKStore {},
  length: 4096,
  kty: 'RSA',
  kid: 'sWpWma0lDp_RfHKdtkGSVTYQaMIVQaKhESVmzjaW9jc',
  use: '',
  alg: '' }
192.168.0.5
Express HTTP  server listening on port 8888
Express HTTPS server listening on port 8443
<p />
```

Once TAM server is up, you see above messages

5.5.4 Copy hello-app & teep-broker-app binaries to SGX

Copy the binaries to SGX/NUC machine over SSH

```
TODO - Further update required
```

If source is build natively on the SGX/NUC machine, then just copy the binaries to test PATH.

```
TODO - Further update required
```

5.5.5 Check hello-app and teep-broker-app on SGX

```
TODO - Further update required
```

5.5.5.1 Run hello-app

```
TODO - Further update required
```

5.5.5.2 Run teep-broker-app

If your TAM server IP address is 192.168.0.3, then you

```
./teep-broker-app --tamurl http://192.168.0.3:8888/api/tam_cbor
```

Execution logs

```
TODO - Further update required
```