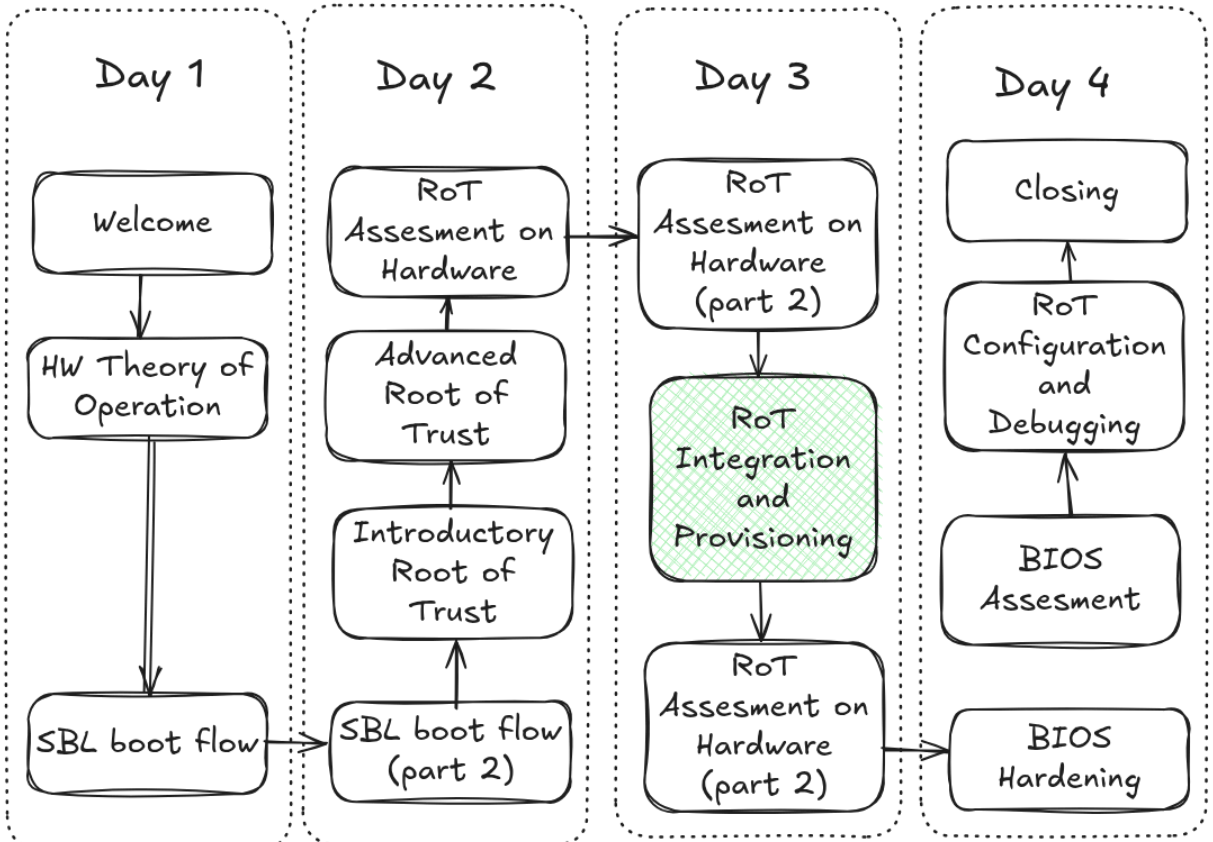
DS09SBL: Dasharo TrustRoot Training

Integration and Provisioning Root of Trust in Modern Hardware

Dasharo Trust Root Training

Where we are in the course



Goals of the Presentation

- Processes and procedures for provisioning Root of Trust in modern hardware platforms
- Case study: Intel Alder Lake N security features and provisioning Hands-On Lab: Provisioning Root of Trust and configuring security settings

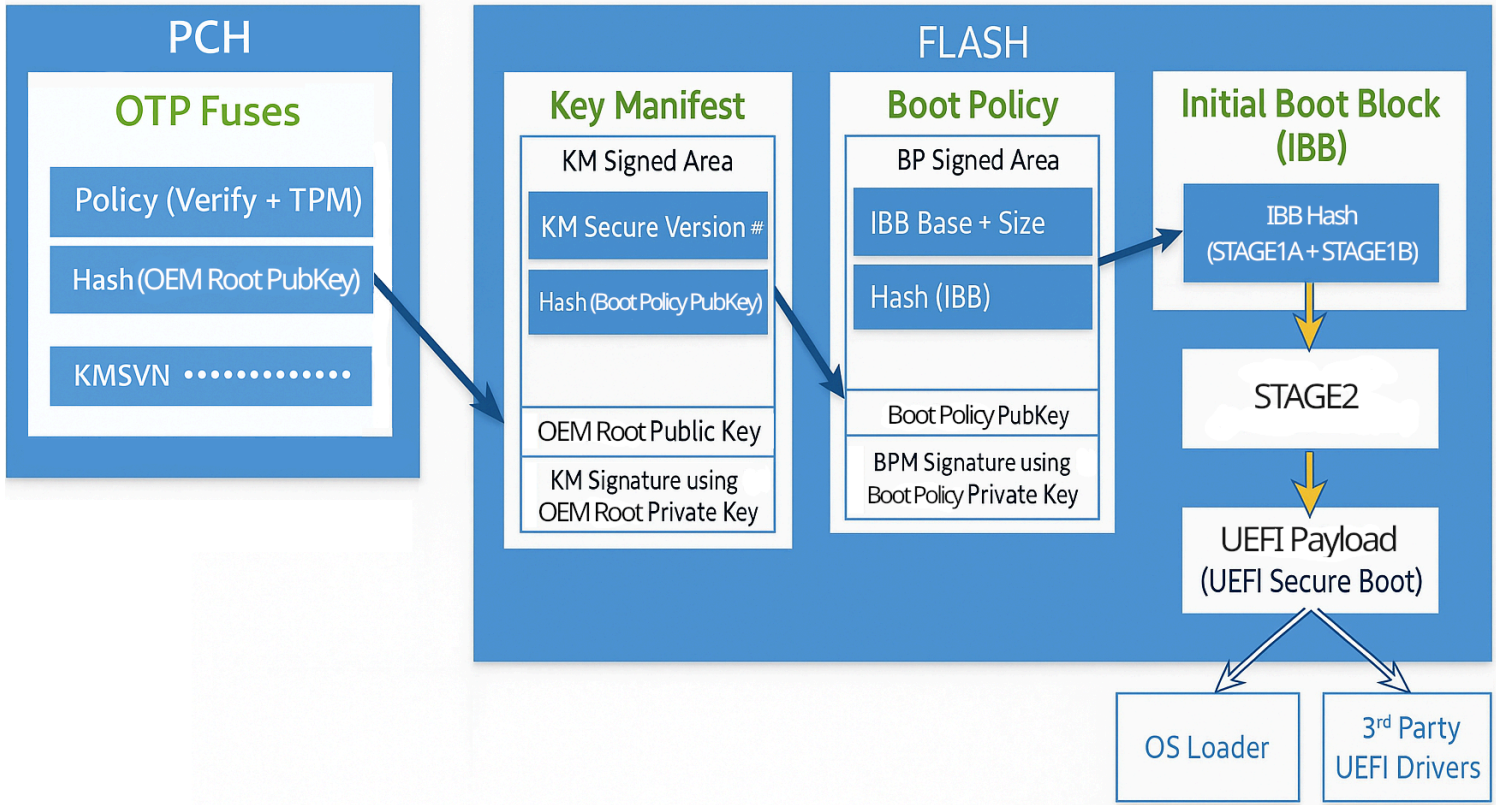
Tooling

- `mFIT`
- `MEInfo`
- `MEU`
- `BpmGen2`

Licenses

- Licenses are available in the `day3/licenses` directory in the shared link to the training materials:
  - `PV Intel OBL Software License Agreement_11.2.2017_1.pdf` - license accompanying the toolkit when downloading from R&DC
  - `SLA_TOOLS.pdf` - license accompanying the tools inside the ME toolkit package
  - `Intel OBL SDK and Tools License for MEU.pdf` - license accompanying the tools inside the ME toolkit package. For MEU tool.
  - `LICENSE-EULA` - license accompanying the BpmGen2 inside the BpmGen2 package

Boot Guard BIOS boot flow



Boot Guard provisioning steps

Boot Guard provisioning steps

1. Generate two pairs of keys:
   - OEM Root Key - its public part is being fused into the SOC/PCH during End of Manufacturing process. Authorizes Boot Policy Manifest key.
   - Boot Policy Manifest key - used to sign the firmware IBB regions

Boot Guard provisioning steps

1. Generate two pairs of keys:
   - OEM Root Key - its public part is being fused into the SOC/PCH during End of Manufacturing process. Authorizes Boot Policy Manifest key.
   - Boot Policy Manifest key - used to sign the firmware IBB regions
2. Generate OEM Key Manifest - MEU utility wraps OEM Root Key into a manifest structure used in signing ME firmware components

Boot Guard provisioning steps

1. Generate two pairs of keys:
   - OEM Root Key - its public part is being fused into the SOC/PCH during End of Manufacturing process. Authorizes Boot Policy Manifest key.
   - Boot Policy Manifest key - used to sign the firmware IBB regions
2. Generate OEM Key Manifest - MEU utility wraps OEM Root Key into a manifest structure used in signing ME firmware components
3. Configure ME and stitch it with OEM Key Manifest - mFIT utility is used to set the key hash to the SHA384 of OEM Root Key, configure Boot Guard policy and inject OEM Key Manifest into the ME firmware region.

Boot Guard provisioning steps

1. Generate two pairs of keys:
   - OEM Root Key - its public part is being fused into the SOC/PCH during End of Manufacturing process. Authorizes Boot Policy Manifest key.
   - Boot Policy Manifest key - used to sign the firmware IBB regions
2. Generate OEM Key Manifest - MEU utility wraps OEM Root Key into a manifest structure used in signing ME firmware components
3. Configure ME and stitch it with OEM Key Manifest - mFIT utility is used to set the key hash to the SHA384 of OEM Root Key, configure Boot Guard policy and inject OEM Key Manifest into the ME firmware region.
4. Put Boot Guard ACM into the firmware image (if not done so yet).

Boot Guard provisioning steps

1. Generate two pairs of keys:
   - OEM Root Key - its public part is being fused into the SOC/PCH during End of Manufacturing process. Authorizes Boot Policy Manifest key.
   - Boot Policy Manifest key - used to sign the firmware IBB regions
2. Generate OEM Key Manifest - MEU utility wraps OEM Root Key into a manifest structure used in signing ME firmware components
3. Configure ME and stitch it with OEM Key Manifest - mFIT utility is used to set the key hash to the SHA384 of OEM Root Key, configure Boot Guard policy and inject OEM Key Manifest into the ME firmware region.
4. Put Boot Guard ACM into the firmware image (if not done so yet).
5. Generate Boot Guard manifests and sign firmware - BpmGen2 utility generates Boot Guard Key Manifest and Boot Policy manifest, signs them and puts into the BIOS region. Manifests created in this step:
   - Key Manifest - signed with OEM Root Key, describes authorized Boot Policy Key
   - Boot Policy Manifest - signed with Boot Policy Key, includes the hash of the firmware iBB

Provisioning Boot Guard in Slim Bootloader

```
Platform/AlderlakeBoardPkg/
└── Script
    ├── GpioDataConfig.py
    ├── StitchIfwiConfig_adln.py
    ├── StitchIfwiConfig_adlp.py
    ├── StitchIfwiConfig_adlps.py
    ├── StitchIfwiConfig_adls.py
    ├── StitchIfwiConfig_azb.py
    ├── StitchIfwi.py
    └── StitchLoader.py
```

Provisioning Boot Guard in Slim Bootloader

```
Platform/AlderlakeBoardPkg/
└── Script
    ├── GpioDataConfig.py
    ├── StitchIfwiConfig_adln.py
    ├── StitchIfwiConfig_adlp.py
    ├── StitchIfwiConfig_adlps.py
    ├── StitchIfwiConfig_adls.py
    ├── StitchIfwiConfig_azb.py
    ├── StitchIfwi.py
    └── StitchLoader.py
```

Provisioning Boot Guard in Slim Bootloader

```
Platform/AlderlakeBoardPkg/
└── Script
    ├── GpioDataConfig.py
    ├── StitchIfwiConfig_adln.py
    ├── StitchIfwiConfig_adlp.py
    ├── StitchIfwiConfig_adlps.py
    ├── StitchIfwiConfig_adls.py
    ├── StitchIfwiConfig_azb.py
    ├── StitchIfwi.py
    └── StitchLoader.py
```

Provisioning Boot Guard in Slim Bootloader

```
Platform/AlderlakeBoardPkg/
└── Script
    ├── GpioDataConfig.py
    ├── StitchIfwiConfig_adln.py
    ├── StitchIfwiConfig_adlp.py
    ├── StitchIfwiConfig_adlps.py
    ├── StitchIfwiConfig_adls.py
    ├── StitchIfwiConfig_azb.py
    ├── StitchIfwi.py
    └── StitchLoader.py
```

Provisioning Boot Guard in Slim Bootloader

In case of Kontron Tiger Lake board the relevant scripts can be found in `Platform/TigerlakeBoardPkg/`:

```
Script
├── GpioDataConfig.py
├── StitchIfwiConfig_tglh.py
├── StitchIfwiConfig_tglu.py
├── StitchIfwi.py
└── StitchLoader.py
```

Provisioning Boot Guard in Slim Bootloader

```
user@OST2-VM:~$ python Platform/AlderlakeBoardPkg/Script/StitchIfwi.py -h
usage: StitchIfwi.py [-h] [-p PLATFORM] [-w WORK_DIR] -c CONFIG_FILE
                     [-s SBL_FILE] [-b {legacy,vm,fve,fvme}] [-d PLAT_DATA]
                     [-r] [-t {ptt,dtpm,none}] [-k KEY_DIR] [-o OPTION] [-op OUTPATH]

options:
  -h, --help            show this help message and exit
  -p PLATFORM           specify platform sku to stitch
  -w WORK_DIR           specify stitch workspace directory, CSME tools and ingredients should be here
  -c CONFIG_FILE        specify the platform specific stitch config file
  -s SBL_FILE           specify slim bootloader file or generate zip file
  -b {legacy,vm,fve,fvme}
                        specify Boot Guard profile type
  -d PLAT_DATA          Specify a platform specific data (HEX, DWORD) for customization
  -r                    delete temporary files after stitch
  -t {ptt,dtpm,none}    specify TPM type
  -k KEY_DIR            specify the path to Sbl Keys directory
  -o OPTION             Platform specific stitch option. Format: '-o option1;option2;...'
                        For each option its format is 'parameter:data'.
```

Provisioning Boot Guard in Slim Bootloader

```
user@OST2-VM:~$ python Platform/AlderlakeBoardPkg/Script/StitchIfwi.py -h
usage: StitchIfwi.py [-h] [-p PLATFORM] [-w WORK_DIR] -c CONFIG_FILE
                     [-s SBL_FILE] [-b {legacy,vm,fve,fvme}] [-d PLAT_DATA]
                     [-r] [-t {ptt,dtpm,none}] [-k KEY_DIR] [-o OPTION] [-op OUTPATH]

options:
  -h, --help            show this help message and exit
  -p PLATFORM           specify platform sku to stitch
  -w WORK_DIR           specify stitch workspace directory, CSME tools and ingredients should be here
  -c CONFIG_FILE        specify the platform specific stitch config file
  -s SBL_FILE           specify slim bootloader file or generate zip file
  -b {legacy,vm,fve,fvme}
                        specify Boot Guard profile type
  -d PLAT_DATA          Specify a platform specific data (HEX, DWORD) for customization
  -r                    delete temporary files after stitch
  -t {ptt,dtpm,none}    specify TPM type
  -k KEY_DIR            specify the path to Sbl Keys directory
  -o OPTION             Platform specific stitch option. Format: '-o option1;option2;...'
                        For each option its format is 'parameter:data'.
```

Provisioning Boot Guard in Slim Bootloader

```
user@OST2-VM:~$ python Platform/AlderlakeBoardPkg/Script/StitchIfwi.py -h
usage: StitchIfwi.py [-h] [-p PLATFORM] [-w WORK_DIR] -c CONFIG_FILE
                     [-s SBL_FILE] [-b {legacy,vm,fve,fvme}] [-d PLAT_DATA]
                     [-r] [-t {ptt,dtpm,none}] [-k KEY_DIR] [-o OPTION] [-op OUTPATH]

options:
  -h, --help            show this help message and exit
  -p PLATFORM           specify platform sku to stitch
  -w WORK_DIR           specify stitch workspace directory, CSME tools and ingredients should be here
  -c CONFIG_FILE        specify the platform specific stitch config file
  -s SBL_FILE           specify slim bootloader file or generate zip file
  -b {legacy,vm,fve,fvme}
                        specify Boot Guard profile type
  -d PLAT_DATA          Specify a platform specific data (HEX, DWORD) for customization
  -r                    delete temporary files after stitch
  -t {ptt,dtpm,none}    specify TPM type
  -k KEY_DIR            specify the path to Sbl Keys directory
  -o OPTION             Platform specific stitch option. Format: '-o option1;option2;...'
                        For each option its format is 'parameter:data'.
```

Provisioning Boot Guard in Slim Bootloader

```
user@OST2-VM:~$ python Platform/AlderlakeBoardPkg/Script/StitchIfwi.py -h
usage: StitchIfwi.py [-h] [-p PLATFORM] [-w WORK_DIR] -c CONFIG_FILE
                     [-s SBL_FILE] [-b {legacy,vm,fve,fvme}] [-d PLAT_DATA]
                     [-r] [-t {ptt,dtpm,none}] [-k KEY_DIR] [-o OPTION] [-op OUTPATH]

options:
  -h, --help            show this help message and exit
  -p PLATFORM           specify platform sku to stitch
  -w WORK_DIR           specify stitch workspace directory, CSME tools and ingredients should be here
  -c CONFIG_FILE        specify the platform specific stitch config file
  -s SBL_FILE           specify slim bootloader file or generate zip file
  -b {legacy,vm,fve,fvme}
                        specify Boot Guard profile type
  -d PLAT_DATA          Specify a platform specific data (HEX, DWORD) for customization
  -r                    delete temporary files after stitch
  -t {ptt,dtpm,none}    specify TPM type
  -k KEY_DIR            specify the path to Sbl Keys directory
  -o OPTION             Platform specific stitch option. Format: '-o option1;option2;...'
                        For each option its format is 'parameter:data'.
```

Provisioning Boot Guard in Slim Bootloader

```
user@OST2-VM:~$ python Platform/AlderlakeBoardPkg/Script/StitchIfwi.py -h
usage: StitchIfwi.py [-h] [-p PLATFORM] [-w WORK_DIR] -c CONFIG_FILE
                     [-s SBL_FILE] [-b {legacy,vm,fve,fvme}] [-d PLAT_DATA]
                     [-r] [-t {ptt,dtpm,none}] [-k KEY_DIR] [-o OPTION] [-op OUTPATH]

options:
  -h, --help            show this help message and exit
  -p PLATFORM           specify platform sku to stitch
  -w WORK_DIR           specify stitch workspace directory, CSME tools and ingredients should be here
  -c CONFIG_FILE        specify the platform specific stitch config file
  -s SBL_FILE           specify slim bootloader file or generate zip file
  -b {legacy,vm,fve,fvme}
                        specify Boot Guard profile type
  -d PLAT_DATA          Specify a platform specific data (HEX, DWORD) for customization
  -r                    delete temporary files after stitch
  -t {ptt,dtpm,none}    specify TPM type
  -k KEY_DIR            specify the path to Sbl Keys directory
  -o OPTION             Platform specific stitch option. Format: '-o option1;option2;...'
                        For each option its format is 'parameter:data'.
```

Provisioning Boot Guard in Slim Bootloader

```
user@OST2-VM:~$ python Platform/AlderlakeBoardPkg/Script/StitchIfwi.py -h
usage: StitchIfwi.py [-h] [-p PLATFORM] [-w WORK_DIR] -c CONFIG_FILE
                     [-s SBL_FILE] [-b {legacy,vm,fve,fvme}] [-d PLAT_DATA]
                     [-r] [-t {ptt,dtpm,none}] [-k KEY_DIR] [-o OPTION] [-op OUTPATH]

options:
  -h, --help            show this help message and exit
  -p PLATFORM           specify platform sku to stitch
  -w WORK_DIR           specify stitch workspace directory, CSME tools and ingredients should be here
  -c CONFIG_FILE        specify the platform specific stitch config file
  -s SBL_FILE           specify slim bootloader file or generate zip file
  -b {legacy,vm,fve,fvme}
                        specify Boot Guard profile type
  -d PLAT_DATA          Specify a platform specific data (HEX, DWORD) for customization
  -r                    delete temporary files after stitch
  -t {ptt,dtpm,none}    specify TPM type
  -k KEY_DIR            specify the path to Sbl Keys directory
  -o OPTION             Platform specific stitch option. Format: '-o option1;option2;...'
                        For each option its format is 'parameter:data'.
```

Provisioning Boot Guard in Slim Bootloader

```
user@OST2-VM:~$ python Platform/AlderlakeBoardPkg/Script/StitchIfwi.py -h
usage: StitchIfwi.py [-h] [-p PLATFORM] [-w WORK_DIR] -c CONFIG_FILE
                     [-s SBL_FILE] [-b {legacy,vm,fve,fvme}] [-d PLAT_DATA]
                     [-r] [-t {ptt,dtpm,none}] [-k KEY_DIR] [-o OPTION] [-op OUTPATH]

options:
  -h, --help            show this help message and exit
  -p PLATFORM           specify platform sku to stitch
  -w WORK_DIR           specify stitch workspace directory, CSME tools and ingredients should be here
  -c CONFIG_FILE        specify the platform specific stitch config file
  -s SBL_FILE           specify slim bootloader file or generate zip file
  -b {legacy,vm,fve,fvme}
                        specify Boot Guard profile type
  -d PLAT_DATA          Specify a platform specific data (HEX, DWORD) for customization
  -r                    delete temporary files after stitch
  -t {ptt,dtpm,none}    specify TPM type
  -k KEY_DIR            specify the path to Sbl Keys directory
  -o OPTION             Platform specific stitch option. Format: '-o option1;option2;...'
                        For each option its format is 'parameter:data'.
```

Provisioning Boot Guard in Slim Bootloader

```
user@OST2-VM:~$ python Platform/AlderlakeBoardPkg/Script/StitchIfwi.py -h
usage: StitchIfwi.py [-h] [-p PLATFORM] [-w WORK_DIR] -c CONFIG_FILE
                     [-s SBL_FILE] [-b {legacy,vm,fve,fvme}] [-d PLAT_DATA]
                     [-r] [-t {ptt,dtpm,none}] [-k KEY_DIR] [-o OPTION] [-op OUTPATH]

options:
  -h, --help            show this help message and exit
  -p PLATFORM           specify platform sku to stitch
  -w WORK_DIR           specify stitch workspace directory, CSME tools and ingredients should be here
  -c CONFIG_FILE        specify the platform specific stitch config file
  -s SBL_FILE           specify slim bootloader file or generate zip file
  -b {legacy,vm,fve,fvme}
                        specify Boot Guard profile type
  -d PLAT_DATA          Specify a platform specific data (HEX, DWORD) for customization
  -r                    delete temporary files after stitch
  -t {ptt,dtpm,none}    specify TPM type
  -k KEY_DIR            specify the path to Sbl Keys directory
  -o OPTION             Platform specific stitch option. Format: '-o option1;option2;...'
                        For each option its format is 'parameter:data'.
```

Provisioning Boot Guard in Slim Bootloader

```
user@OST2-VM:~$ python Platform/AlderlakeBoardPkg/Script/StitchIfwi.py -h
usage: StitchIfwi.py [-h] [-p PLATFORM] [-w WORK_DIR] -c CONFIG_FILE
                     [-s SBL_FILE] [-b {legacy,vm,fve,fvme}] [-d PLAT_DATA]
                     [-r] [-t {ptt,dtpm,none}] [-k KEY_DIR] [-o OPTION] [-op OUTPATH]

options:
  -h, --help            show this help message and exit
  -p PLATFORM           specify platform sku to stitch
  -w WORK_DIR           specify stitch workspace directory, CSME tools and ingredients should be here
  -c CONFIG_FILE        specify the platform specific stitch config file
  -s SBL_FILE           specify slim bootloader file or generate zip file
  -b {legacy,vm,fve,fvme}
                        specify Boot Guard profile type
  -d PLAT_DATA          Specify a platform specific data (HEX, DWORD) for customization
  -r                    delete temporary files after stitch
  -t {ptt,dtpm,none}    specify TPM type
  -k KEY_DIR            specify the path to Sbl Keys directory
  -o OPTION             Platform specific stitch option. Format: '-o option1;option2;...'
                        For each option its format is 'parameter:data'.
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
```

Breakdown of Boot Guard provisioning in Slim Bootloader

StitchIfwi.py

```
def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
    temp_dir = os.path.abspath(os.path.join (stitch_dir, 'Temp'))
    if os.path.exists(temp_dir):
        shutil.rmtree(temp_dir, ignore_errors=True)
    shutil.copytree (os.path.join (stitch_dir, 'Input'), temp_dir)

    # Get bios region image ready
    sbl_image_ext = os.path.splitext(sbl_file)

    if sbl_image_ext[1] =/= ".zip":
        print ("\nCopy SBL image %s for stitch" % sbl_file)
        shutil.copy(sbl_file, os.path.join(temp_dir, "SlimBootloader.bin"))
    else:
        print ("\nUnpack files from zip file ...")
        zf = zipfile.ZipFile(sbl_file, 'r', zipfile.ZIP_DEFLATED)
        zf.extractall(temp_dir)
        zf.close()
...
def main():
...
work_dir = os.path.abspath (args.work_dir)
os.chdir(work_dir)
if stitch (work_dir, stitch_cfg_file, sbl_file, args.btg_profile, plt_params_list, args.plat_data, args.platform, args.tpm):
    raise Exception ('Stitching process failed !')
os.chdir(curr_dir)
```

Breakdown of Boot Guard provisioning in Slim Bootloader

StitchIfwi.py

```
def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
    temp_dir = os.path.abspath(os.path.join (stitch_dir, 'Temp'))
    if os.path.exists(temp_dir):
        shutil.rmtree(temp_dir, ignore_errors=True)
    shutil.copytree (os.path.join (stitch_dir, 'Input'), temp_dir)

    # Get bios region image ready
    sbl_image_ext = os.path.splitext(sbl_file)

    if sbl_image_ext[1] =/= ".zip":
        print ("\nCopy SBL image %s for stitch" % sbl_file)
        shutil.copy(sbl_file, os.path.join(temp_dir, "SlimBootloader.bin"))
    else:
        print ("\nUnpack files from zip file ...")
        zf = zipfile.ZipFile(sbl_file, 'r', zipfile.ZIP_DEFLATED)
        zf.extractall(temp_dir)
        zf.close()
...
def main():
...
work_dir = os.path.abspath (args.work_dir)
os.chdir(work_dir)
if stitch (work_dir, stitch_cfg_file, sbl_file, args.btg_profile, plt_params_list, args.plat_data, args.platform, args.tpm):
    raise Exception ('Stitching process failed !')
os.chdir(curr_dir)
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
```

StitchIfwi.py:

```
def replace_components (ifwi_src_path, stitch_cfg_file, plt_params_list):
    print ("Replacing components.......")
    replace_list = stitch_cfg_file.get_component_replace_list (plt_params_list)
    for flash_path, file_path, comp_alg, pri_key, svn in replace_list:
        replace_component (ifwi_src_path, flash_path, file_path, comp_alg, pri_key, svn)

...

def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    if platform_data:
        fd = open(os.path.join(temp_dir, "SlimBootloader.bin"), "rb")
        input_data = bytearray(fd.read())
        fd.close()
        print ("\n Adding platform data to Slimbootloader ...")
        data = add_platform_data(input_data, platform_data)
        fd = open(os.path.join(temp_dir, "SlimBootloader.bin"), "wb")
        fd.write(data)
        fd.close()

    print("Replace components in both partitions....")
    replace_components (os.path.join(temp_dir, "SlimBootloader.bin"), stitch_cfg_file, plt_params_list)
```

StitchIfwi.py:

```
def replace_components (ifwi_src_path, stitch_cfg_file, plt_params_list):
    print ("Replacing components.......")
    replace_list = stitch_cfg_file.get_component_replace_list (plt_params_list)
    for flash_path, file_path, comp_alg, pri_key, svn in replace_list:
        replace_component (ifwi_src_path, flash_path, file_path, comp_alg, pri_key, svn)

...

def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    if platform_data:
        fd = open(os.path.join(temp_dir, "SlimBootloader.bin"), "rb")
        input_data = bytearray(fd.read())
        fd.close()
        print ("\n Adding platform data to Slimbootloader ...")
        data = add_platform_data(input_data, platform_data)
        fd = open(os.path.join(temp_dir, "SlimBootloader.bin"), "wb")
        fd.write(data)
        fd.close()

    print("Replace components in both partitions....")
    replace_components (os.path.join(temp_dir, "SlimBootloader.bin"), stitch_cfg_file, plt_params_list)
```

StitchIfwi.py:

```
def replace_components (ifwi_src_path, stitch_cfg_file, plt_params_list):
    print ("Replacing components.......")
    replace_list = stitch_cfg_file.get_component_replace_list (plt_params_list)
    for flash_path, file_path, comp_alg, pri_key, svn in replace_list:
        replace_component (ifwi_src_path, flash_path, file_path, comp_alg, pri_key, svn)

...

def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    if platform_data:
        fd = open(os.path.join(temp_dir, "SlimBootloader.bin"), "rb")
        input_data = bytearray(fd.read())
        fd.close()
        print ("\n Adding platform data to Slimbootloader ...")
        data = add_platform_data(input_data, platform_data)
        fd = open(os.path.join(temp_dir, "SlimBootloader.bin"), "wb")
        fd.write(data)
        fd.close()

    print("Replace components in both partitions....")
    replace_components (os.path.join(temp_dir, "SlimBootloader.bin"), stitch_cfg_file, plt_params_list)
```

StitchIfwi.py:

```
def replace_components (ifwi_src_path, stitch_cfg_file, plt_params_list):
    print ("Replacing components.......")
    replace_list = stitch_cfg_file.get_component_replace_list (plt_params_list)
    for flash_path, file_path, comp_alg, pri_key, svn in replace_list:
        replace_component (ifwi_src_path, flash_path, file_path, comp_alg, pri_key, svn)

...

def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    if platform_data:
        fd = open(os.path.join(temp_dir, "SlimBootloader.bin"), "rb")
        input_data = bytearray(fd.read())
        fd.close()
        print ("\n Adding platform data to Slimbootloader ...")
        data = add_platform_data(input_data, platform_data)
        fd = open(os.path.join(temp_dir, "SlimBootloader.bin"), "wb")
        fd.write(data)
        fd.close()

    print("Replace components in both partitions....")
    replace_components (os.path.join(temp_dir, "SlimBootloader.bin"), stitch_cfg_file, plt_params_list)
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
```

StitchIfwiConfig_adln.py:

```
def get_component_replace_list(plt_params_list):
    replace_list = [
       #    Path                   file name              compress    Key                          SVN
      ('IFWI/BIOS/TS0/ACM0',      'Input/acm0.bin',        'dummy',    '',                            ''),
      ('IFWI/BIOS/TS1/ACM0',      'Input/acm0.bin',        'dummy',    '',                            ''),
       ]

    if 'tsn' in plt_params_list:
        if os.path.exists('IPFW/TsnSubRegion.bin'):
            print ("TsnSubRegion.bin found")
            replace_list.append (
                ('IFWI/BIOS/NRD/IPFW/TMAC', 'IPFW/TsnSubRegion.bin',     'lz4',     'KEY_ID_CONTAINER_COMP_RSA3072', 0),   # TSN MAC Address
            )

    return replace_list
```

StitchIfwiConfig_adln.py:

```
def get_component_replace_list(plt_params_list):
    replace_list = [
       #    Path                   file name              compress    Key                          SVN
      ('IFWI/BIOS/TS0/ACM0',      'Input/acm0.bin',        'dummy',    '',                            ''),
      ('IFWI/BIOS/TS1/ACM0',      'Input/acm0.bin',        'dummy',    '',                            ''),
       ]

    if 'tsn' in plt_params_list:
        if os.path.exists('IPFW/TsnSubRegion.bin'):
            print ("TsnSubRegion.bin found")
            replace_list.append (
                ('IFWI/BIOS/NRD/IPFW/TMAC', 'IPFW/TsnSubRegion.bin',     'lz4',     'KEY_ID_CONTAINER_COMP_RSA3072', 0),   # TSN MAC Address
            )

    return replace_list
```

StitchIfwiConfig_adln.py:

```
def get_component_replace_list(plt_params_list):
    replace_list = [
       #    Path                   file name              compress    Key                          SVN
      ('IFWI/BIOS/TS0/ACM0',      'Input/acm0.bin',        'dummy',    '',                           ''),
      ('IFWI/BIOS/TS1/ACM0',      'Input/acm0.bin',        'dummy',    '',                           ''),
       ]

    if 'tsn' in plt_params_list:
        if os.path.exists('IPFW/TsnSubRegion.bin'):
            print ("TsnSubRegion.bin found")
            replace_list.append (
                ('IFWI/BIOS/NRD/IPFW/TMAC', 'IPFW/TsnSubRegion.bin',     'lz4',     'KEY_ID_CONTAINER_COMP_RSA3072', 0),   # TSN MAC Address
            )

    return replace_list
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
```

StitchIfwi.py:

```
def gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm):
    print ("Generating xml file .........")

    fit_tool     = os.path.join (stitch_dir, 'Fit', 'mfit')
    new_xml_file = os.path.join (stitch_dir, 'Temp', 'new.xml')
    updated_xml_file = os.path.join (stitch_dir, 'Temp', 'updated.xml')
    sku = stitch_cfg_file.get_platform_sku().get(platform)
    cmd = [fit_tool, '-l', sku, '-s', new_xml_file, '--workingdir', os.path.join (stitch_dir, 'Temp')]
    run_process (cmd)

...

def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    # Generate xml
    gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm)
```

StitchIfwiConfig_adln.py:

```
def get_platform_sku():
    platform_sku ={
        'adln'     :   'Intel(R) TwinLake Chipset - Consumer - SPI',
    }
    return platform_sku
```

StitchIfwi.py:

```
def gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm):
    print ("Generating xml file .........")

    fit_tool     = os.path.join (stitch_dir, 'Fit', 'mfit')
    new_xml_file = os.path.join (stitch_dir, 'Temp', 'new.xml')
    updated_xml_file = os.path.join (stitch_dir, 'Temp', 'updated.xml')
    sku = stitch_cfg_file.get_platform_sku().get(platform)
    cmd = [fit_tool, '-l', sku, '-s', new_xml_file, '--workingdir', os.path.join (stitch_dir, 'Temp')]
    run_process (cmd)

...

def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    # Generate xml
    gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm)
```

StitchIfwiConfig_adln.py:

```
def get_platform_sku():
    platform_sku ={
        'adln'     :   'Intel(R) TwinLake Chipset - Consumer - SPI',
    }
    return platform_sku
```

StitchIfwi.py:

```
def gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm):
    print ("Generating xml file .........")

    fit_tool     = os.path.join (stitch_dir, 'Fit', 'mfit')
    new_xml_file = os.path.join (stitch_dir, 'Temp', 'new.xml')
    updated_xml_file = os.path.join (stitch_dir, 'Temp', 'updated.xml')
    sku = stitch_cfg_file.get_platform_sku().get(platform)
    cmd = [fit_tool, '-l', sku, '-s', new_xml_file, '--workingdir', os.path.join (stitch_dir, 'Temp')]
    run_process (cmd)

...

def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    # Generate xml
    gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm)
```

StitchIfwiConfig_adln.py:

```
def get_platform_sku():
    platform_sku ={
        'adln'     :   'Intel(R) TwinLake Chipset - Consumer - SPI',
    }
    return platform_sku
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
      StitchIfwiConfig_adln.py: get_xml_change_list()
```

StitchIfwi.py:

```
def gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm):
    print ("Generating xml file .........")

    fit_tool     = os.path.join (stitch_dir, 'Fit', 'mfit')
    new_xml_file = os.path.join (stitch_dir, 'Temp', 'new.xml')
    updated_xml_file = os.path.join (stitch_dir, 'Temp', 'updated.xml')
    sku = stitch_cfg_file.get_platform_sku().get(platform)
    cmd = [fit_tool, '-l', sku, '-s', new_xml_file, '--workingdir', os.path.join (stitch_dir, 'Temp')]
    run_process (cmd)

    tree = ET.parse(new_xml_file)

    xml_change_list = stitch_cfg_file.get_xml_change_list (platform, plt_params_list)
    for each in xml_change_list:
        for xml_path, value in each:
            node = tree.find('%s' % xml_path)
            if '$SourceDir' in value:
              value = value.replace ('$SourceDir', os.path.join (stitch_dir, 'Temp'))
            node.set('value', value)
            print (value)

    tree.write(updated_xml_file)
```

StitchIfwi.py:

```
def gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm):
    print ("Generating xml file .........")

    fit_tool     = os.path.join (stitch_dir, 'Fit', 'mfit')
    new_xml_file = os.path.join (stitch_dir, 'Temp', 'new.xml')
    updated_xml_file = os.path.join (stitch_dir, 'Temp', 'updated.xml')
    sku = stitch_cfg_file.get_platform_sku().get(platform)
    cmd = [fit_tool, '-l', sku, '-s', new_xml_file, '--workingdir', os.path.join (stitch_dir, 'Temp')]
    run_process (cmd)

    tree = ET.parse(new_xml_file)

    xml_change_list = stitch_cfg_file.get_xml_change_list (platform, plt_params_list)
    for each in xml_change_list:
        for xml_path, value in each:
            node = tree.find('%s' % xml_path)
            if '$SourceDir' in value:
              value = value.replace ('$SourceDir', os.path.join (stitch_dir, 'Temp'))
            node.set('value', value)
            print (value)

    tree.write(updated_xml_file)
```

StitchIfwi.py:

```
def gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm):
    print ("Generating xml file .........")

    fit_tool     = os.path.join (stitch_dir, 'Fit', 'mfit')
    new_xml_file = os.path.join (stitch_dir, 'Temp', 'new.xml')
    updated_xml_file = os.path.join (stitch_dir, 'Temp', 'updated.xml')
    sku = stitch_cfg_file.get_platform_sku().get(platform)
    cmd = [fit_tool, '-l', sku, '-s', new_xml_file, '--workingdir', os.path.join (stitch_dir, 'Temp')]
    run_process (cmd)

    tree = ET.parse(new_xml_file)

    xml_change_list = stitch_cfg_file.get_xml_change_list (platform, plt_params_list)
    for each in xml_change_list:
        for xml_path, value in each:
            node = tree.find('%s' % xml_path)
            if '$SourceDir' in value:
              value = value.replace ('$SourceDir', os.path.join (stitch_dir, 'Temp'))
            node.set('value', value)
            print (value)

    tree.write(updated_xml_file)
```

StitchIfwiConfig_adln.py:

```
def get_xml_change_list (platform, plt_params_list):
    xml_change_list = []
    xml_change_list.append ([
      # Path                                                  | value |
      # ============================================================================
      #Region Order
        ('./FlashLayout/BiosRegion/InputFile',                                  '$SourceDir\BiosRegion.bin'),
        ('./FlashLayout/Ifwi_IntelMePmcRegion/MeRegionFile',                    '$SourceDir\MeRegionFile.bin'),
        ('./FlashLayout/Ifwi_IntelMePmcRegion/ChipInitBinary',                  '$SourceDir\ChipInitBinary.bin'),
        ('./FlashLayout/Ifwi_IntelMePmcRegion/PmcBinary',                       '$SourceDir\PmcBinary.bin'),
        ('./FlashLayout/SubPartitions/PchcSubPartitionData/InputFile',          '$SourceDir\PchcSubPartitionData.bin'),
        ('./FlashLayout/SubPartitions/IunitSubPartition/InputFile',             '$SourceDir\Iunit.bin'),
...
        ('./PlatformProtection/PlatformIntegrity/OemExtInputFile',              '$SourceDir\OemExtInputFile.bin'),
...
        ('./FlexIO/Type-CSubsystemConfiguration/TypeCPort1Config',              'No Thunderbolt'),
        ('./FlexIO/Type-CSubsystemConfiguration/IomBinaryFile',                 '$SourceDir\Iom.bin'),
        ('./FlexIO/Type-CSubsystemConfiguration/PhyBinaryFile',                 '$SourceDir\TypeC_NorthPHYRegion.bin'),
```

StitchIfwiConfig_adln.py:

```
def check_parameter(para_list):
    print (para_list)
    para_supported = {
        'crb'    : {},
        'rvp_ddr5' : {},
        'quad'   : {},
        'debug'  : {},
        '16MB'   : {},
        'tsn'    : {}
       }

    para_help = \
        """
        'crb'    -- Stitch for CRB board
        'rvp_ddr5' -- Stitch for RVP DDR5 board
        'quad'   -- Stitch IFWI with SPI QUAD mode
        '16MB'   -- Stitch image set to 16MB, by default use 32MB.
        'debug'  -- Enable DAM and DCI configuration (Only use for debug purpose but not for final production!)
        'tsn'    -- Stitch sample Tsn Mac address binary along with TSN AIC softstraps
        """
...
def get_xml_change_list (platform, plt_params_list):
...
    if 'tsn' in plt_params_list:
        print ("Applying changes to enable TSN")
        xml_change_list.append ([
          ('./NetworkingConnectivity/TimeSensitiveNetworkingConfiguration/TsnEnabled', 'TSN MIPI Clock'),
        ])
```

StitchIfwiConfig_adln.py:

```
def check_parameter(para_list):
    print (para_list)
    para_supported = {
        'crb'    : {},
        'rvp_ddr5' : {},
        'quad'   : {},
        'debug'  : {},
        '16MB'   : {},
        'tsn'    : {}
       }

    para_help = \
        """
        'crb'    -- Stitch for CRB board
        'rvp_ddr5' -- Stitch for RVP DDR5 board
        'quad'   -- Stitch IFWI with SPI QUAD mode
        '16MB'   -- Stitch image set to 16MB, by default use 32MB.
        'debug'  -- Enable DAM and DCI configuration (Only use for debug purpose but not for final production!)
        'tsn'    -- Stitch sample Tsn Mac address binary along with TSN AIC softstraps
        """
...
def get_xml_change_list (platform, plt_params_list):
...
    if 'tsn' in plt_params_list:
        print ("Applying changes to enable TSN")
        xml_change_list.append ([
          ('./NetworkingConnectivity/TimeSensitiveNetworkingConfiguration/TsnEnabled', 'TSN MIPI Clock'),
        ])
```

StitchIfwiConfig_adln.py:

```
def check_parameter(para_list):
    print (para_list)
    para_supported = {
        'crb'    : {},
        'rvp_ddr5' : {},
        'quad'   : {},
        'debug'  : {},
        '16MB'   : {},
        'tsn'    : {}
       }

    para_help = \
        """
        'crb'    -- Stitch for CRB board
        'rvp_ddr5' -- Stitch for RVP DDR5 board
        'quad'   -- Stitch IFWI with SPI QUAD mode
        '16MB'   -- Stitch image set to 16MB, by default use 32MB.
        'debug'  -- Enable DAM and DCI configuration (Only use for debug purpose but not for final production!)
        'tsn'    -- Stitch sample Tsn Mac address binary along with TSN AIC softstraps
        """
...
def get_xml_change_list (platform, plt_params_list):
...
    if 'tsn' in plt_params_list:
        print ("Applying changes to enable TSN")
        xml_change_list.append ([
          ('./NetworkingConnectivity/TimeSensitiveNetworkingConfiguration/TsnEnabled', 'TSN MIPI Clock'),
        ])
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
      StitchIfwiConfig_adln.py: get_xml_change_list()
    StitchIfwi.py: patch_xml_file()
```

StitchIfwi.py:

```
def patch_xml_file(stitch_dir, ifwi_src_path):
    print ("Patching xml file .........")

    TOP_SWAP_SIZE = {
        0x00020000 : '64KB',
        0x00040000 : '128KB',
        0x00080000 : '512KB',
        0x00100000 : '1MB',
        0x00200000 : '2MB',
        0x00400000 : '4MB',
        0x00800000 : '8MB',
    }

    ifwi_bin = bytearray (get_file_data (ifwi_src_path))
    ifwi = IFWI_PARSER.parse_ifwi_binary (ifwi_bin)
    ts0_comp = IFWI_PARSER.locate_components (ifwi, 'IFWI/BIOS/TS0')
    if len(ts0_comp) == 0:
        raise Exception ("Could not locate component IFWI/BIOS/TS0!")

    updated_xml_file = os.path.join (stitch_dir, 'Temp', 'updated.xml')
    tree = ET.parse(updated_xml_file)
    node = tree.find('./FlashSettings/BiosConfiguration/TopSwapOverride')
    node.set('value', TOP_SWAP_SIZE[ts0_comp[0].length])
    tree.write(updated_xml_file)

...

def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    # Generate xml
    gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm)

    patch_xml_file(stitch_dir, os.path.join(temp_dir, "SlimBootloader.bin"))
```

StitchIfwi.py:

```
def patch_xml_file(stitch_dir, ifwi_src_path):
    print ("Patching xml file .........")

    TOP_SWAP_SIZE = {
        0x00020000 : '64KB',
        0x00040000 : '128KB',
        0x00080000 : '512KB',
        0x00100000 : '1MB',
        0x00200000 : '2MB',
        0x00400000 : '4MB',
        0x00800000 : '8MB',
    }

    ifwi_bin = bytearray (get_file_data (ifwi_src_path))
    ifwi = IFWI_PARSER.parse_ifwi_binary (ifwi_bin)
    ts0_comp = IFWI_PARSER.locate_components (ifwi, 'IFWI/BIOS/TS0')
    if len(ts0_comp) == 0:
        raise Exception ("Could not locate component IFWI/BIOS/TS0!")

    updated_xml_file = os.path.join (stitch_dir, 'Temp', 'updated.xml')
    tree = ET.parse(updated_xml_file)
    node = tree.find('./FlashSettings/BiosConfiguration/TopSwapOverride')
    node.set('value', TOP_SWAP_SIZE[ts0_comp[0].length])
    tree.write(updated_xml_file)

...

def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    # Generate xml
    gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm)

    patch_xml_file(stitch_dir, os.path.join(temp_dir, "SlimBootloader.bin"))
```

StitchIfwi.py:

```
def patch_xml_file(stitch_dir, ifwi_src_path):
    print ("Patching xml file .........")

    TOP_SWAP_SIZE = {
        0x00020000 : '64KB',
        0x00040000 : '128KB',
        0x00080000 : '512KB',
        0x00100000 : '1MB',
        0x00200000 : '2MB',
        0x00400000 : '4MB',
        0x00800000 : '8MB',
    }

    ifwi_bin = bytearray (get_file_data (ifwi_src_path))
    ifwi = IFWI_PARSER.parse_ifwi_binary (ifwi_bin)
    ts0_comp = IFWI_PARSER.locate_components (ifwi, 'IFWI/BIOS/TS0')
    if len(ts0_comp) == 0:
        raise Exception ("Could not locate component IFWI/BIOS/TS0!")

    updated_xml_file = os.path.join (stitch_dir, 'Temp', 'updated.xml')
    tree = ET.parse(updated_xml_file)
    node = tree.find('./FlashSettings/BiosConfiguration/TopSwapOverride')
    node.set('value', TOP_SWAP_SIZE[ts0_comp[0].length])
    tree.write(updated_xml_file)

...

def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    # Generate xml
    gen_xml_file(stitch_dir, stitch_cfg_file, btg_profile, plt_params_list, platform, tpm)

    patch_xml_file(stitch_dir, os.path.join(temp_dir, "SlimBootloader.bin"))
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
      StitchIfwiConfig_adln.py: get_xml_change_list()
    StitchIfwi.py: patch_xml_file()
    StitchIfwi.py: update_btGuard_manifests ()
```

StitchIfwi.py:

```
def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    patch_xml_file(stitch_dir, os.path.join(temp_dir, "SlimBootloader.bin"))

    if sign_bin_flag:
        update_btGuard_manifests(stitch_dir, stitch_cfg_file, btg_profile, tpm)
    else:
        shutil.copy(os.path.join(temp_dir, "SlimBootloader.bin"), os.path.join(temp_dir, "BiosRegion.bin"))
```

StitchIfwi.py:

```
def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    patch_xml_file(stitch_dir, os.path.join(temp_dir, "SlimBootloader.bin"))

    if sign_bin_flag:
        update_btGuard_manifests(stitch_dir, stitch_cfg_file, btg_profile, tpm)
    else:
        shutil.copy(os.path.join(temp_dir, "SlimBootloader.bin"), os.path.join(temp_dir, "BiosRegion.bin"))
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
      StitchIfwiConfig_adln.py: get_xml_change_list()
    StitchIfwi.py: patch_xml_file()
    StitchIfwi.py: update_btGuard_manifests ()
      security_stitch_help.py: update_btGuard_manifests ()
      security_stitch_help.py: gen_bpmgen2_params()
      security_stitch_help.py: sign_slimboot_binary()
```

Platform/CommonBoardPkg/Script/security_stitch_help.py

```
def update_btGuard_manifests(stitch_dir, stitch_cfg_file, btg_profile, tpm):
    output_dir  = os.path.join(stitch_dir, "Temp")
    sbl_file    = os.path.join (output_dir, 'SlimBootloader.bin')
    out_file    = os.path.join(output_dir, "BiosRegion.bin")
    bpm_gen2dir = os.path.join (stitch_dir, 'BpmGen2')
    bmpgen_params = os.path.join(output_dir, "bpmgen2.params")
    bpm_key_dir = os.path.join (bpm_gen2dir, 'keys')
    key_size    = '3072'
    hash_type   = 'sha384'
    key_manifest_hash_file = os.path.join(output_dir, "kmsigpubkey.hash")

    print ("Generating BPM GEN2 params file")
    gen_bpmgen2_params(stitch_cfg_file, os.path.join(bpm_gen2dir, "bpmgen2.params"), os.path.join(output_dir, "bpmgen2.params"))

    print("Sign partitions....")
    sign_slimboot_binary(sbl_file, bpm_gen2dir, bmpgen_params, bpm_key_dir, key_size, hash_type, out_file, output_dir, key_manifest_hash_file)

...
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
      StitchIfwiConfig_adln.py: get_xml_change_list()
    StitchIfwi.py: patch_xml_file()
    StitchIfwi.py: update_btGuard_manifests ()
      security_stitch_help.py: update_btGuard_manifests ()
        security_stitch_help.py: gen_bpmgen2_params()
          StitchIfwiConfig_adln.py: get_bpmgen2_params_change_list()
        security_stitch_help.py: sign_slimboot_binary()
```

Platform/CommonBoardPkg/Script/security_stitch_help.py

```
def gen_bpmgen2_params (stitch_cfg_file, InFile, OutFile):
    InFileptr = open(InFile, 'r', encoding='utf8')
    lines = InFileptr.readlines()
    InFileptr.close()

    params_change_list = stitch_cfg_file.get_bpmgen2_params_change_list()

    for item in params_change_list:
        for variable, value in item:
            for linenumber, line in enumerate(lines):
                if line.split(':')[0].strip() == variable:
                    lines[linenumber] = variable + ':    ' + value + '\n'
                    break

    if OutFile == '':
        OutFile = Infile

    Outfileptr = open(OutFile, 'w')
    Outfileptr.write("".join(lines))
    Outfileptr.close()
```

Platform/CommonBoardPkg/Script/security_stitch_help.py

```
def gen_bpmgen2_params (stitch_cfg_file, InFile, OutFile):
    InFileptr = open(InFile, 'r', encoding='utf8')
    lines = InFileptr.readlines()
    InFileptr.close()

    params_change_list = stitch_cfg_file.get_bpmgen2_params_change_list()

    for item in params_change_list:
        for variable, value in item:
            for linenumber, line in enumerate(lines):
                if line.split(':')[0].strip() == variable:
                    lines[linenumber] = variable + ':    ' + value + '\n'
                    break

    if OutFile == '':
        OutFile = Infile

    Outfileptr = open(OutFile, 'w')
    Outfileptr.write("".join(lines))
    Outfileptr.close()
```

Platform/CommonBoardPkg/Script/security_stitch_help.py

```
def gen_bpmgen2_params (stitch_cfg_file, InFile, OutFile):
    InFileptr = open(InFile, 'r', encoding='utf8')
    lines = InFileptr.readlines()
    InFileptr.close()

    params_change_list = stitch_cfg_file.get_bpmgen2_params_change_list()

    for item in params_change_list:
        for variable, value in item:
            for linenumber, line in enumerate(lines):
                if line.split(':')[0].strip() == variable:
                    lines[linenumber] = variable + ':    ' + value + '\n'
                    break

    if OutFile == '':
        OutFile = Infile

    Outfileptr = open(OutFile, 'w')
    Outfileptr.write("".join(lines))
    Outfileptr.close()
```

Platform/CommonBoardPkg/Script/security_stitch_help.py

```
def gen_bpmgen2_params (stitch_cfg_file, InFile, OutFile):
    InFileptr = open(InFile, 'r', encoding='utf8')
    lines = InFileptr.readlines()
    InFileptr.close()

    params_change_list = stitch_cfg_file.get_bpmgen2_params_change_list()

    for item in params_change_list:
        for variable, value in item:
            for linenumber, line in enumerate(lines):
                if line.split(':')[0].strip() == variable:
                    lines[linenumber] = variable + ':    ' + value + '\n'
                    break

    if OutFile == '':
        OutFile = Infile

    Outfileptr = open(OutFile, 'w')
    Outfileptr.write("".join(lines))
    Outfileptr.close()
```

StitchIfwiConfig_adln.py:

```
def get_bpmgen2_params_change_list ():
    params_change_list = []
    params_change_list.append ([
      # variable                | value |
      # ================================
      ('PlatformRules',         'ADL Client'),
      ('BpmStrutVersion',       '0x20'),
      ('BpmRevision',           '0x01'),
      ('BpmRevocation',         '1'),
      ('AcmRevocation',         '2'),
      ('NEMPages',              '3'),
      ('AcpiBase',              '0x1800'),
      ('IbbFlags',              '0x12'),
      ('IbbHashAlgID',          '0x0C:SHA384'),
      ('TxtInclude',            'FALSE'),
      ('PcdInclude',            'TRUE'),
      ('BpmSigScheme',          '0x16:RSAPSS'),
      ('BpmSigPubKey',          r'BpmGen2/keys/bpm_pubkey_3072.pem'),
      ('BpmSigPrivKey',         r'BpmGen2/keys/bpm_privkey_3072.pem'),
      ('BpmKeySizeBits',        '3072'),
      ('BpmSigHashAlgID',       '0x0C:SHA384'),
      ])
    return params_change_list
```

bpmgen2.params:

```
IbbFlags: 0x12
//    Bit0 : Enable DMA Protection;
//    Bit1 : Issue TPM Start-up from Locality 3;
//    Bit2 : Extend Authority Measurements into the Authority PCR;
//    Bit3 : On error: Leave TPM Hierarchies enabled. Cap all PCRs;
//    Bit4 : Top Swap Supported;
//    Bit5 : Force (MK)TME;
//    Bit6 : Enforce SPIRAL specification between BIOS/CSE
//    Bit7 : SRTM Attenstation Control
//    Bit8 : Force Communication NEM Buffer
```

bpmgen2.params:

```
IbbFlags: 0x12
//    Bit0 : Enable DMA Protection;
//    Bit1 : Issue TPM Start-up from Locality 3;
//    Bit2 : Extend Authority Measurements into the Authority PCR;
//    Bit3 : On error: Leave TPM Hierarchies enabled. Cap all PCRs;
//    Bit4 : Top Swap Supported;
//    Bit5 : Force (MK)TME;
//    Bit6 : Enforce SPIRAL specification between BIOS/CSE
//    Bit7 : SRTM Attenstation Control
//    Bit8 : Force Communication NEM Buffer
```

bpmgen2.params:

```
IbbFlags: 0x12
//    Bit0 : Enable DMA Protection;
//    Bit1 : Issue TPM Start-up from Locality 3;
//    Bit2 : Extend Authority Measurements into the Authority PCR;
//    Bit3 : On error: Leave TPM Hierarchies enabled. Cap all PCRs;
//    Bit4 : Top Swap Supported;
//    Bit5 : Force (MK)TME;
//    Bit6 : Enforce SPIRAL specification between BIOS/CSE
//    Bit7 : SRTM Attenstation Control
//    Bit8 : Force Communication NEM Buffer
```

bpmgen2.params:

```
IbbFlags: 0x12
//    Bit0 : Enable DMA Protection;
//    Bit1 : Issue TPM Start-up from Locality 3;
//    Bit2 : Extend Authority Measurements into the Authority PCR;
//    Bit3 : On error: Leave TPM Hierarchies enabled. Cap all PCRs;
//    Bit4 : Top Swap Supported;
//    Bit5 : Force (MK)TME;
//    Bit6 : Enforce SPIRAL specification between BIOS/CSE
//    Bit7 : SRTM Attenstation Control
//    Bit8 : Force Communication NEM Buffer
```

bpmgen2.params:

```
IbbFlags: 0x12
//    Bit0 : Enable DMA Protection;
//    Bit1 : Issue TPM Start-up from Locality 3;
//    Bit2 : Extend Authority Measurements into the Authority PCR;
//    Bit3 : On error: Leave TPM Hierarchies enabled. Cap all PCRs;
//    Bit4 : Top Swap Supported;
//    Bit5 : Force (MK)TME;
//    Bit6 : Enforce SPIRAL specification between BIOS/CSE
//    Bit7 : SRTM Attenstation Control
//    Bit8 : Force Communication NEM Buffer
```

bpmgen2.params:

```
IbbFlags: 0x12
//    Bit0 : Enable DMA Protection;
//    Bit1 : Issue TPM Start-up from Locality 3;
//    Bit2 : Extend Authority Measurements into the Authority PCR;
//    Bit3 : On error: Leave TPM Hierarchies enabled. Cap all PCRs;
//    Bit4 : Top Swap Supported;
//    Bit5 : Force (MK)TME;
//    Bit6 : Enforce SPIRAL specification between BIOS/CSE
//    Bit7 : SRTM Attenstation Control
//    Bit8 : Force Communication NEM Buffer
```

bpmgen2.params:

```
IbbFlags: 0x12
//    Bit0 : Enable DMA Protection;
//    Bit1 : Issue TPM Start-up from Locality 3;
//    Bit2 : Extend Authority Measurements into the Authority PCR;
//    Bit3 : On error: Leave TPM Hierarchies enabled. Cap all PCRs;
//    Bit4 : Top Swap Supported;
//    Bit5 : Force (MK)TME;
//    Bit6 : Enforce SPIRAL specification between BIOS/CSE
//    Bit7 : SRTM Attenstation Control
//    Bit8 : Force Communication NEM Buffer
```

bpmgen2.params:

```
IbbFlags: 0x12
//    Bit0 : Enable DMA Protection;
//    Bit1 : Issue TPM Start-up from Locality 3;
//    Bit2 : Extend Authority Measurements into the Authority PCR;
//    Bit3 : On error: Leave TPM Hierarchies enabled. Cap all PCRs;
//    Bit4 : Top Swap Supported;
//    Bit5 : Force (MK)TME;
//    Bit6 : Enforce SPIRAL specification between BIOS/CSE
//    Bit7 : SRTM Attenstation Control
//    Bit8 : Force Communication NEM Buffer
```

bpmgen2.params:

```
IbbFlags: 0x12
//    Bit0 : Enable DMA Protection;
//    Bit1 : Issue TPM Start-up from Locality 3;
//    Bit2 : Extend Authority Measurements into the Authority PCR;
//    Bit3 : On error: Leave TPM Hierarchies enabled. Cap all PCRs;
//    Bit4 : Top Swap Supported;
//    Bit5 : Force (MK)TME;
//    Bit6 : Enforce SPIRAL specification between BIOS/CSE
//    Bit7 : SRTM Attenstation Control
//    Bit8 : Force Communication NEM Buffer
```

bpmgen2.params:

```
IbbFlags: 0x12
//    Bit0 : Enable DMA Protection;
//    Bit1 : Issue TPM Start-up from Locality 3;
//    Bit2 : Extend Authority Measurements into the Authority PCR;
//    Bit3 : On error: Leave TPM Hierarchies enabled. Cap all PCRs;
//    Bit4 : Top Swap Supported;
//    Bit5 : Force (MK)TME;
//    Bit6 : Enforce SPIRAL specification between BIOS/CSE
//    Bit7 : SRTM Attenstation Control
//    Bit8 : Force Communication NEM Buffer
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
      StitchIfwiConfig_adln.py: get_xml_change_list()
    StitchIfwi.py: patch_xml_file()
    StitchIfwi.py: update_btGuard_manifests ()
      security_stitch_help.py: update_btGuard_manifests ()
        security_stitch_help.py: gen_bpmgen2_params()
          StitchIfwiConfig_adln.py: get_bpmgen2_params_change_list()
        security_stitch_help.py: sign_slimboot_binary()
          BtgSign.py: sign_slimboot_binary()
            BtgSign.py: bpm_sign_binary()
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
def sign_slimboot_binary(infile, bpm_gen2dir, bpmgen2_params, key_dir,  key_sz, hash_type, out_file, output_dir, oem_pub_key_hash_file):

    # get topswap size
    sbl_bin  = bytearray (get_file_data (infile))
    ifwi     = IFWI_PARSER.parse_ifwi_binary (sbl_bin)
    ifwi_comps = IFWI_PARSER.locate_components (ifwi, 'IFWI/BIOS/TS0')
    if len(ifwi_comps) == 0:
        raise Exception("Cannot find path 'IFWI/BIOS/TS0' in ifwi image!")
    for ifwi_comp in ifwi_comps:
        top_swap_size = ifwi_comp.length

    print("Sign primary partition....")
    bpm_sign_binary(infile, os.path.join(output_dir, "sbl_sec.bin"), output_dir, bpm_gen2dir, key_dir, bpmgen2_params, key_sz, oem_pub_key_hash_file, hash_type)

    print("Swap top swap block....")
    swap_ts_block(os.path.join(output_dir, "sbl_sec.bin"), os.path.join(output_dir, "SwappedA.bin"), top_swap_size)

    print("Sign backup partition....")
    bpm_sign_binary(os.path.join(output_dir,"SwappedA.bin"),  os.path.join(output_dir, "sbl_sec.bin"), output_dir, bpm_gen2dir, key_dir, bpmgen2_params, '3072', oem_pub_key_hash_file, 'sha384')
    os.remove(os.path.join(output_dir,"SwappedA.bin"))

    print("Swap to original top swap block....")
    swap_ts_block(os.path.join(output_dir, "sbl_sec.bin"), os.path.join(output_dir, "SwappedA.bin"), top_swap_size)
    shutil.copy(os.path.join(output_dir, "SwappedA.bin"), os.path.join(output_dir, "SlimBootloader.bin"))
    shutil.copy(os.path.join(output_dir, "SlimBootloader.bin"), out_file)
    os.remove(os.path.join(output_dir,"SwappedA.bin"))
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
def sign_slimboot_binary(infile, bpm_gen2dir, bpmgen2_params, key_dir,  key_sz, hash_type, out_file, output_dir, oem_pub_key_hash_file):

    # get topswap size
    sbl_bin  = bytearray (get_file_data (infile))
    ifwi     = IFWI_PARSER.parse_ifwi_binary (sbl_bin)
    ifwi_comps = IFWI_PARSER.locate_components (ifwi, 'IFWI/BIOS/TS0')
    if len(ifwi_comps) == 0:
        raise Exception("Cannot find path 'IFWI/BIOS/TS0' in ifwi image!")
    for ifwi_comp in ifwi_comps:
        top_swap_size = ifwi_comp.length

    print("Sign primary partition....")
    bpm_sign_binary(infile, os.path.join(output_dir, "sbl_sec.bin"), output_dir, bpm_gen2dir, key_dir, bpmgen2_params, key_sz, oem_pub_key_hash_file, hash_type)

    print("Swap top swap block....")
    swap_ts_block(os.path.join(output_dir, "sbl_sec.bin"), os.path.join(output_dir, "SwappedA.bin"), top_swap_size)

    print("Sign backup partition....")
    bpm_sign_binary(os.path.join(output_dir,"SwappedA.bin"),  os.path.join(output_dir, "sbl_sec.bin"), output_dir, bpm_gen2dir, key_dir, bpmgen2_params, '3072', oem_pub_key_hash_file, 'sha384')
    os.remove(os.path.join(output_dir,"SwappedA.bin"))

    print("Swap to original top swap block....")
    swap_ts_block(os.path.join(output_dir, "sbl_sec.bin"), os.path.join(output_dir, "SwappedA.bin"), top_swap_size)
    shutil.copy(os.path.join(output_dir, "SwappedA.bin"), os.path.join(output_dir, "SlimBootloader.bin"))
    shutil.copy(os.path.join(output_dir, "SlimBootloader.bin"), out_file)
    os.remove(os.path.join(output_dir,"SwappedA.bin"))
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
def sign_slimboot_binary(infile, bpm_gen2dir, bpmgen2_params, key_dir,  key_sz, hash_type, out_file, output_dir, oem_pub_key_hash_file):

    # get topswap size
    sbl_bin  = bytearray (get_file_data (infile))
    ifwi     = IFWI_PARSER.parse_ifwi_binary (sbl_bin)
    ifwi_comps = IFWI_PARSER.locate_components (ifwi, 'IFWI/BIOS/TS0')
    if len(ifwi_comps) == 0:
        raise Exception("Cannot find path 'IFWI/BIOS/TS0' in ifwi image!")
    for ifwi_comp in ifwi_comps:
        top_swap_size = ifwi_comp.length

    print("Sign primary partition....")
    bpm_sign_binary(infile, os.path.join(output_dir, "sbl_sec.bin"), output_dir, bpm_gen2dir, key_dir, bpmgen2_params, key_sz, oem_pub_key_hash_file, hash_type)

    print("Swap top swap block....")
    swap_ts_block(os.path.join(output_dir, "sbl_sec.bin"), os.path.join(output_dir, "SwappedA.bin"), top_swap_size)

    print("Sign backup partition....")
    bpm_sign_binary(os.path.join(output_dir,"SwappedA.bin"),  os.path.join(output_dir, "sbl_sec.bin"), output_dir, bpm_gen2dir, key_dir, bpmgen2_params, '3072', oem_pub_key_hash_file, 'sha384')
    os.remove(os.path.join(output_dir,"SwappedA.bin"))

    print("Swap to original top swap block....")
    swap_ts_block(os.path.join(output_dir, "sbl_sec.bin"), os.path.join(output_dir, "SwappedA.bin"), top_swap_size)
    shutil.copy(os.path.join(output_dir, "SwappedA.bin"), os.path.join(output_dir, "SlimBootloader.bin"))
    shutil.copy(os.path.join(output_dir, "SlimBootloader.bin"), out_file)
    os.remove(os.path.join(output_dir,"SwappedA.bin"))
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
def sign_slimboot_binary(infile, bpm_gen2dir, bpmgen2_params, key_dir,  key_sz, hash_type, out_file, output_dir, oem_pub_key_hash_file):

    # get topswap size
    sbl_bin  = bytearray (get_file_data (infile))
    ifwi     = IFWI_PARSER.parse_ifwi_binary (sbl_bin)
    ifwi_comps = IFWI_PARSER.locate_components (ifwi, 'IFWI/BIOS/TS0')
    if len(ifwi_comps) == 0:
        raise Exception("Cannot find path 'IFWI/BIOS/TS0' in ifwi image!")
    for ifwi_comp in ifwi_comps:
        top_swap_size = ifwi_comp.length

    print("Sign primary partition....")
    bpm_sign_binary(infile, os.path.join(output_dir, "sbl_sec.bin"), output_dir, bpm_gen2dir, key_dir, bpmgen2_params, key_sz, oem_pub_key_hash_file, hash_type)

    print("Swap top swap block....")
    swap_ts_block(os.path.join(output_dir, "sbl_sec.bin"), os.path.join(output_dir, "SwappedA.bin"), top_swap_size)

    print("Sign backup partition....")
    bpm_sign_binary(os.path.join(output_dir,"SwappedA.bin"),  os.path.join(output_dir, "sbl_sec.bin"), output_dir, bpm_gen2dir, key_dir, bpmgen2_params, '3072', oem_pub_key_hash_file, 'sha384')
    os.remove(os.path.join(output_dir,"SwappedA.bin"))

    print("Swap to original top swap block....")
    swap_ts_block(os.path.join(output_dir, "sbl_sec.bin"), os.path.join(output_dir, "SwappedA.bin"), top_swap_size)
    shutil.copy(os.path.join(output_dir, "SwappedA.bin"), os.path.join(output_dir, "SlimBootloader.bin"))
    shutil.copy(os.path.join(output_dir, "SlimBootloader.bin"), out_file)
    os.remove(os.path.join(output_dir,"SwappedA.bin"))
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
      StitchIfwiConfig_adln.py: get_xml_change_list()
    StitchIfwi.py: patch_xml_file()
    StitchIfwi.py: update_btGuard_manifests ()
      security_stitch_help.py: update_btGuard_manifests ()
        security_stitch_help.py: gen_bpmgen2_params()
          StitchIfwiConfig_adln.py: get_bpmgen2_params_change_list()
        security_stitch_help.py: sign_slimboot_binary()
          BtgSign.py: sign_slimboot_binary()
            BtgSign.py: bpm_sign_binary()
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
SIGNING_KEY = {
    # Key Id                                | Key File Name |
    # ========================================================
    # KEY_ID_PRIV_OEM is used for OEM signing
    "KEY_ID_PRIV_OEM_RSA2048"          :    "oem_privkey_2048.pem",
    "KEY_ID_PRIV_OEM_RSA3072"          :    "oem_privkey_3072.pem",

    # KEY_ID_PRIV_BPM is used for BPM signing
    "KEY_ID_PRIV_BPM_RSA2048"          :    "bpm_privkey_2048.pem",
    "KEY_ID_PRIV_BPM_RSA3072"          :    "bpm_privkey_3072.pem",

    "KEY_ID_PUB_OEM_RSA2048"           :    "oem_pubkey_2048.pem",
    "KEY_ID_PUB_OEM_RSA3072"           :    "oem_pubkey_3072.pem",

    "KEY_ID_PUB_BPM_RSA2048"           :    "bpm_pubkey_2048.pem",
    "KEY_ID_PUB_BPM_RSA3072"           :    "bpm_pubkey_3072.pem",
    }
...
def bpm_sign_binary(infile, out_file, output_dir, bpm_gen2dir, key_dir, bpmgen2_params, key_sz, oem_pub_key_hash_file, hash_type):

    shutil.copy(infile, os.path.join(output_dir,"sbl_sec_temp.bin"))


    oem_priv_key = SIGNING_KEY['KEY_ID_PRIV_OEM_RSA' + key_sz]
    oem_pub_key  = SIGNING_KEY['KEY_ID_PUB_OEM_RSA'  + key_sz]
    bpm_priv_key = SIGNING_KEY['KEY_ID_PRIV_BPM_RSA' + key_sz]
    bpm_pub_key  = SIGNING_KEY['KEY_ID_PUB_BPM_RSA'  + key_sz]
    if os.name == 'nt':
        bpmgen2_tool = os.path.join (bpm_gen2dir, 'bpmgen2.exe')
    else:
        bpmgen2_tool = os.path.join (bpm_gen2dir, 'bpmgen2')

    print (key_dir)
    if not os.path.exists(key_dir):
        print("Generating new keys....")
        generate_keys (key_dir, key_sz)
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
SIGNING_KEY = {
    # Key Id                                | Key File Name |
    # ============================================================
    # KEY_ID_PRIV_OEM is used for OEM signing
    "KEY_ID_PRIV_OEM_RSA2048"          :    "oem_privkey_2048.pem",
    "KEY_ID_PRIV_OEM_RSA3072"          :    "oem_privkey_3072.pem",

    # KEY_ID_PRIV_BPM is used for BPM signing
    "KEY_ID_PRIV_BPM_RSA2048"          :    "bpm_privkey_2048.pem",
    "KEY_ID_PRIV_BPM_RSA3072"          :    "bpm_privkey_3072.pem",

    "KEY_ID_PUB_OEM_RSA2048"           :    "oem_pubkey_2048.pem",
    "KEY_ID_PUB_OEM_RSA3072"           :    "oem_pubkey_3072.pem",

    "KEY_ID_PUB_BPM_RSA2048"           :    "bpm_pubkey_2048.pem",
    "KEY_ID_PUB_BPM_RSA3072"           :    "bpm_pubkey_3072.pem",
    }
...
def bpm_sign_binary(infile, out_file, output_dir, bpm_gen2dir, key_dir, bpmgen2_params, key_sz, oem_pub_key_hash_file, hash_type):

    shutil.copy(infile, os.path.join(output_dir,"sbl_sec_temp.bin"))


    oem_priv_key = SIGNING_KEY['KEY_ID_PRIV_OEM_RSA' + key_sz]
    oem_pub_key  = SIGNING_KEY['KEY_ID_PUB_OEM_RSA'  + key_sz]
    bpm_priv_key = SIGNING_KEY['KEY_ID_PRIV_BPM_RSA' + key_sz]
    bpm_pub_key  = SIGNING_KEY['KEY_ID_PUB_BPM_RSA'  + key_sz]
    if os.name == 'nt':
        bpmgen2_tool = os.path.join (bpm_gen2dir, 'bpmgen2.exe')
    else:
        bpmgen2_tool = os.path.join (bpm_gen2dir, 'bpmgen2')

    print (key_dir)
    if not os.path.exists(key_dir):
        print("Generating new keys....")
        generate_keys (key_dir, key_sz)
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
SIGNING_KEY = {
    # Key Id                                | Key File Name |
    # ========================================================
    # KEY_ID_PRIV_OEM is used for OEM signing
    "KEY_ID_PRIV_OEM_RSA2048"          :    "oem_privkey_2048.pem",
    "KEY_ID_PRIV_OEM_RSA3072"          :    "oem_privkey_3072.pem",

    # KEY_ID_PRIV_BPM is used for BPM signing
    "KEY_ID_PRIV_BPM_RSA2048"          :    "bpm_privkey_2048.pem",
    "KEY_ID_PRIV_BPM_RSA3072"          :    "bpm_privkey_3072.pem",

    "KEY_ID_PUB_OEM_RSA2048"           :    "oem_pubkey_2048.pem",
    "KEY_ID_PUB_OEM_RSA3072"           :    "oem_pubkey_3072.pem",

    "KEY_ID_PUB_BPM_RSA2048"           :    "bpm_pubkey_2048.pem",
    "KEY_ID_PUB_BPM_RSA3072"           :    "bpm_pubkey_3072.pem",
    }
...
def bpm_sign_binary(infile, out_file, output_dir, bpm_gen2dir, key_dir, bpmgen2_params, key_sz, oem_pub_key_hash_file, hash_type):

    shutil.copy(infile, os.path.join(output_dir,"sbl_sec_temp.bin"))


    oem_priv_key = SIGNING_KEY['KEY_ID_PRIV_OEM_RSA' + key_sz]
    oem_pub_key  = SIGNING_KEY['KEY_ID_PUB_OEM_RSA'  + key_sz]
    bpm_priv_key = SIGNING_KEY['KEY_ID_PRIV_BPM_RSA' + key_sz]
    bpm_pub_key  = SIGNING_KEY['KEY_ID_PUB_BPM_RSA'  + key_sz]
    if os.name == 'nt':
        bpmgen2_tool = os.path.join (bpm_gen2dir, 'bpmgen2.exe')
    else:
        bpmgen2_tool = os.path.join (bpm_gen2dir, 'bpmgen2')

    print (key_dir)
    if not os.path.exists(key_dir):
        print("Generating new keys....")
        generate_keys (key_dir, key_sz)
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
SIGNING_KEY = {
    # Key Id                                | Key File Name |
    # ========================================================
    # KEY_ID_PRIV_OEM is used for OEM signing
    "KEY_ID_PRIV_OEM_RSA2048"          :    "oem_privkey_2048.pem",
    "KEY_ID_PRIV_OEM_RSA3072"          :    "oem_privkey_3072.pem",

    # KEY_ID_PRIV_BPM is used for BPM signing
    "KEY_ID_PRIV_BPM_RSA2048"          :    "bpm_privkey_2048.pem",
    "KEY_ID_PRIV_BPM_RSA3072"          :    "bpm_privkey_3072.pem",

    "KEY_ID_PUB_OEM_RSA2048"           :    "oem_pubkey_2048.pem",
    "KEY_ID_PUB_OEM_RSA3072"           :    "oem_pubkey_3072.pem",

    "KEY_ID_PUB_BPM_RSA2048"           :    "bpm_pubkey_2048.pem",
    "KEY_ID_PUB_BPM_RSA3072"           :    "bpm_pubkey_3072.pem",
    }
...
def bpm_sign_binary(infile, out_file, output_dir, bpm_gen2dir, key_dir, bpmgen2_params, key_sz, oem_pub_key_hash_file, hash_type):

    shutil.copy(infile, os.path.join(output_dir,"sbl_sec_temp.bin"))


    oem_priv_key = SIGNING_KEY['KEY_ID_PRIV_OEM_RSA' + key_sz]
    oem_pub_key  = SIGNING_KEY['KEY_ID_PUB_OEM_RSA'  + key_sz]
    bpm_priv_key = SIGNING_KEY['KEY_ID_PRIV_BPM_RSA' + key_sz]
    bpm_pub_key  = SIGNING_KEY['KEY_ID_PUB_BPM_RSA'  + key_sz]
    if os.name == 'nt':
        bpmgen2_tool = os.path.join (bpm_gen2dir, 'bpmgen2.exe')
    else:
        bpmgen2_tool = os.path.join (bpm_gen2dir, 'bpmgen2')

    print (key_dir)
    if not os.path.exists(key_dir):
        print("Generating new keys....")
        generate_keys (key_dir, key_sz)
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
def bpm_sign_binary(infile, out_file, output_dir, bpm_gen2dir, key_dir, bpmgen2_params, key_sz, oem_pub_key_hash_file, hash_type):
 ...
    print("Generating Btg KeyManifest.bin....")
    run_process ([bpmgen2_tool,
        'KMGEN',
        '-KEY',        os.path.join (key_dir, bpm_pub_key), 'BPM',
        '-KM',         os.path.join (output_dir, 'KeyManifest.bin'),
        '-SIGNKEY',    os.path.join (key_dir,  oem_priv_key),
        '-SIGNPUBKEY', os.path.join (key_dir, oem_pub_key),
        '-KMID',       '0x01',
        '-KMKHASH',    hash_type,
        '-SCHEME',     'RSAPSS',
        '-SVN',        '0',
        '-d:2'])

    print("Generating Btg Boot Policy Manifest (BPM).bin....")
    run_process ([bpmgen2_tool,
        'GEN',
        os.path.join (output_dir, 'sbl_sec_temp.bin'),
        bpmgen2_params,
        '-BPM',        os.path.join (output_dir, 'BpmManifest.bin'),
        '-U',          out_file,
        '-KM',         os.path.join (output_dir, 'KeyManifest.bin'),
        '-d:2'])

    # Generate OemKeyHash
    if oem_pub_key_hash_file:
        gen_oem_key_hash(output_dir,  os.path.join (key_dir, oem_pub_key), oem_pub_key_hash_file, hash_type, key_sz)
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
      StitchIfwiConfig_adln.py: get_xml_change_list()
    StitchIfwi.py: patch_xml_file()
    StitchIfwi.py: update_btGuard_manifests ()
      security_stitch_help.py: update_btGuard_manifests ()
        security_stitch_help.py: gen_bpmgen2_params()
          StitchIfwiConfig_adln.py: get_bpmgen2_params_change_list()
        security_stitch_help.py: sign_slimboot_binary()
          BtgSign.py: sign_slimboot_binary()
            BtgSign.py: bpm_sign_binary()
        StitchIfwi.py: gen_sign_oem_key_manifest()
          BtgSign.py: gen_sign_oem_key_manifest()
```

Platform/CommonBoardPkg/Script/security_stitch_help.py:

```
def update_btGuard_manifests(stitch_dir, stitch_cfg_file, btg_profile, tpm):
...
    print("Sign partitions....")
    sign_slimboot_binary(sbl_file, bpm_gen2dir, bmpgen_params, bpm_key_dir, key_size, hash_type, out_file, output_dir, key_manifest_hash_file)

    # Generate OemKeyManifest bin if not available in inputs
    print("Generate Oem Key Manifest....")
    if os.name == 'nt':
        meu_path = os.path.join (stitch_dir, 'Meu', 'meu.exe')
    else:
        meu_path = os.path.join (stitch_dir, 'Meu', 'meu')
    oem_bin_input  = os.path.join (stitch_dir, 'Input', 'OemExtInputFile.bin')
    oem_bin_output = os.path.join (stitch_dir, 'Temp', 'OemExtInputFile.bin')
    gen_sign_oem_key_manifest(meu_path, oem_bin_input, bpm_key_dir, key_size, output_dir, oem_bin_output)
```

Platform/CommonBoardPkg/Script/security_stitch_help.py:

```
def update_btGuard_manifests(stitch_dir, stitch_cfg_file, btg_profile, tpm):
...
    print("Sign partitions....")
    sign_slimboot_binary(sbl_file, bpm_gen2dir, bmpgen_params, bpm_key_dir, key_size, hash_type, out_file, output_dir, key_manifest_hash_file)

    # Generate OemKeyManifest bin if not available in inputs
    print("Generate Oem Key Manifest....")
    if os.name == 'nt':
        meu_path = os.path.join (stitch_dir, 'Meu', 'meu.exe')
    else:
        meu_path = os.path.join (stitch_dir, 'Meu', 'meu')
    oem_bin_input  = os.path.join (stitch_dir, 'Input', 'OemExtInputFile.bin')
    oem_bin_output = os.path.join (stitch_dir, 'Temp', 'OemExtInputFile.bin')
    gen_sign_oem_key_manifest(meu_path, oem_bin_input, bpm_key_dir, key_size, output_dir, oem_bin_output)
```

Platform/CommonBoardPkg/Script/security_stitch_help.py:

```
def update_btGuard_manifests(stitch_dir, stitch_cfg_file, btg_profile, tpm):
...
    print("Sign partitions....")
    sign_slimboot_binary(sbl_file, bpm_gen2dir, bmpgen_params, bpm_key_dir, key_size, hash_type, out_file, output_dir, key_manifest_hash_file)

    # Generate OemKeyManifest bin if not available in inputs
    print("Generate Oem Key Manifest....")
    if os.name == 'nt':
        meu_path = os.path.join (stitch_dir, 'Meu', 'meu.exe')
    else:
        meu_path = os.path.join (stitch_dir, 'Meu', 'meu')
    oem_bin_input  = os.path.join (stitch_dir, 'Input', 'OemExtInputFile.bin')
    oem_bin_output = os.path.join (stitch_dir, 'Temp', 'OemExtInputFile.bin')
    gen_sign_oem_key_manifest(meu_path, oem_bin_input, bpm_key_dir, key_size, output_dir, oem_bin_output)
```

Platform/CommonBoardPkg/Script/security_stitch_help.py:

```
def update_btGuard_manifests(stitch_dir, stitch_cfg_file, btg_profile, tpm):
...
    print("Sign partitions....")
    sign_slimboot_binary(sbl_file, bpm_gen2dir, bmpgen_params, bpm_key_dir, key_size, hash_type, out_file, output_dir, key_manifest_hash_file)

    # Generate OemKeyManifest bin if not available in inputs
    print("Generate Oem Key Manifest....")
    if os.name == 'nt':
        meu_path = os.path.join (stitch_dir, 'Meu', 'meu.exe')
    else:
        meu_path = os.path.join (stitch_dir, 'Meu', 'meu')
    oem_bin_input  = os.path.join (stitch_dir, 'Input', 'OemExtInputFile.bin')
    oem_bin_output = os.path.join (stitch_dir, 'Temp', 'OemExtInputFile.bin')
    gen_sign_oem_key_manifest(meu_path, oem_bin_input, bpm_key_dir, key_size, output_dir, oem_bin_output)
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
      StitchIfwiConfig_adln.py: get_xml_change_list()
    StitchIfwi.py: patch_xml_file()
    StitchIfwi.py: update_btGuard_manifests ()
      security_stitch_help.py: update_btGuard_manifests ()
        security_stitch_help.py: gen_bpmgen2_params()
          StitchIfwiConfig_adln.py: get_bpmgen2_params_change_list()
        security_stitch_help.py: sign_slimboot_binary()
          BtgSign.py: sign_slimboot_binary()
            BtgSign.py: bpm_sign_binary()
        StitchIfwi.py: gen_sign_oem_key_manifest()
          BtgSign.py: gen_sign_oem_key_manifest()
            StitchIfwiConfig_adln.py: get_oemkeymanifest_change_list ()
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
def gen_sign_oem_key_manifest(meu_path, oem_bin_input, key_dir, key_sz, output_dir, oem_bin_output):

    oem_bin_sign   = oem_bin_output
    oem_priv_key = SIGNING_KEY['KEY_ID_PRIV_OEM_RSA' + key_sz]

    openssl_path = get_openssl_path()

    if not os.path.exists(oem_bin_input):
        print("Generate and sigm OemKeyManifest binary....")
        #create an dummy meu_config.xml
        run_process ([meu_path, '-gen', 'meu_config', '-save', os.path.join(output_dir, 'meu_config.xml')])
        #Generate default OEMKeyManifest config xml
        run_process ([meu_path, '-gen', 'OEMKeyManifest', '-save', os.path.join(output_dir, 'oemkeymanifest_sample_config.xml')])

        #Update OEMKeyManifest config xml for sample/test params
        tree = ET.parse(os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'))

        xml_change_list = get_oemkeymanifest_change_list()
        for each in xml_change_list:
            for xml_path, value in each:
                node = tree.find('%s' % xml_path)
                node.set('value', value)

        tree.write(os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'), xml_declaration=True, encoding='utf-8')

        #Generate signed OEMKeyManifest binary
        #meu.exe -f decomp_km1.xml -o OemExtInputFile.bin -key oem_privkey_3072.pem -stp C:\Openssl\openssl.exe

        run_process ([meu_path, '-f', os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'), '-o', oem_bin_sign,
            '-key',  os.path.join (key_dir, oem_priv_key),
            '-cfg', os.path.join(output_dir, 'meu_config.xml'),
            '-stp', openssl_path])
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
def gen_sign_oem_key_manifest(meu_path, oem_bin_input, key_dir, key_sz, output_dir, oem_bin_output):

    oem_bin_sign   = oem_bin_output
    oem_priv_key = SIGNING_KEY['KEY_ID_PRIV_OEM_RSA' + key_sz]

    openssl_path = get_openssl_path()

    if not os.path.exists(oem_bin_input):
        print("Generate and sigm OemKeyManifest binary....")
        #create an dummy meu_config.xml
        run_process ([meu_path, '-gen', 'meu_config', '-save', os.path.join(output_dir, 'meu_config.xml')])
        #Generate default OEMKeyManifest config xml
        run_process ([meu_path, '-gen', 'OEMKeyManifest', '-save', os.path.join(output_dir, 'oemkeymanifest_sample_config.xml')])

        #Update OEMKeyManifest config xml for sample/test params
        tree = ET.parse(os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'))

        xml_change_list = get_oemkeymanifest_change_list()
        for each in xml_change_list:
            for xml_path, value in each:
                node = tree.find('%s' % xml_path)
                node.set('value', value)

        tree.write(os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'), xml_declaration=True, encoding='utf-8')

        #Generate signed OEMKeyManifest binary
        #meu.exe -f decomp_km1.xml -o OemExtInputFile.bin -key oem_privkey_3072.pem -stp C:\Openssl\openssl.exe

        run_process ([meu_path, '-f', os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'), '-o', oem_bin_sign,
            '-key',  os.path.join (key_dir, oem_priv_key),
            '-cfg', os.path.join(output_dir, 'meu_config.xml'),
            '-stp', openssl_path])
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
def gen_sign_oem_key_manifest(meu_path, oem_bin_input, key_dir, key_sz, output_dir, oem_bin_output):

    oem_bin_sign   = oem_bin_output
    oem_priv_key = SIGNING_KEY['KEY_ID_PRIV_OEM_RSA' + key_sz]

    openssl_path = get_openssl_path()

    if not os.path.exists(oem_bin_input):
        print("Generate and sigm OemKeyManifest binary....")
        #create an dummy meu_config.xml
        run_process ([meu_path, '-gen', 'meu_config', '-save', os.path.join(output_dir, 'meu_config.xml')])
        #Generate default OEMKeyManifest config xml
        run_process ([meu_path, '-gen', 'OEMKeyManifest', '-save', os.path.join(output_dir, 'oemkeymanifest_sample_config.xml')])

        #Update OEMKeyManifest config xml for sample/test params
        tree = ET.parse(os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'))

        xml_change_list = get_oemkeymanifest_change_list()
        for each in xml_change_list:
            for xml_path, value in each:
                node = tree.find('%s' % xml_path)
                node.set('value', value)

        tree.write(os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'), xml_declaration=True, encoding='utf-8')

        #Generate signed OEMKeyManifest binary
        #meu.exe -f decomp_km1.xml -o OemExtInputFile.bin -key oem_privkey_3072.pem -stp C:\Openssl\openssl.exe

        run_process ([meu_path, '-f', os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'), '-o', oem_bin_sign,
            '-key',  os.path.join (key_dir, oem_priv_key),
            '-cfg', os.path.join(output_dir, 'meu_config.xml'),
            '-stp', openssl_path])
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
def gen_sign_oem_key_manifest(meu_path, oem_bin_input, key_dir, key_sz, output_dir, oem_bin_output):

    oem_bin_sign   = oem_bin_output
    oem_priv_key = SIGNING_KEY['KEY_ID_PRIV_OEM_RSA' + key_sz]

    openssl_path = get_openssl_path()

    if not os.path.exists(oem_bin_input):
        print("Generate and sigm OemKeyManifest binary....")
        #create an dummy meu_config.xml
        run_process ([meu_path, '-gen', 'meu_config', '-save', os.path.join(output_dir, 'meu_config.xml')])
        #Generate default OEMKeyManifest config xml
        run_process ([meu_path, '-gen', 'OEMKeyManifest', '-save', os.path.join(output_dir, 'oemkeymanifest_sample_config.xml')])

        #Update OEMKeyManifest config xml for sample/test params
        tree = ET.parse(os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'))

        xml_change_list = get_oemkeymanifest_change_list()
        for each in xml_change_list:
            for xml_path, value in each:
                node = tree.find('%s' % xml_path)
                node.set('value', value)

        tree.write(os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'), xml_declaration=True, encoding='utf-8')

        #Generate signed OEMKeyManifest binary
        #meu.exe -f decomp_km1.xml -o OemExtInputFile.bin -key oem_privkey_3072.pem -stp C:\Openssl\openssl.exe

        run_process ([meu_path, '-f', os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'), '-o', oem_bin_sign,
            '-key',  os.path.join (key_dir, oem_priv_key),
            '-cfg', os.path.join(output_dir, 'meu_config.xml'),
            '-stp', openssl_path])
```

Platform/CommonBoardPkg/Script/BtgSign.py:

```
def gen_sign_oem_key_manifest(meu_path, oem_bin_input, key_dir, key_sz, output_dir, oem_bin_output):

    oem_bin_sign   = oem_bin_output
    oem_priv_key = SIGNING_KEY['KEY_ID_PRIV_OEM_RSA' + key_sz]

    openssl_path = get_openssl_path()

    if not os.path.exists(oem_bin_input):
        print("Generate and sigm OemKeyManifest binary....")
        #create an dummy meu_config.xml
        run_process ([meu_path, '-gen', 'meu_config', '-save', os.path.join(output_dir, 'meu_config.xml')])
        #Generate default OEMKeyManifest config xml
        run_process ([meu_path, '-gen', 'OEMKeyManifest', '-save', os.path.join(output_dir, 'oemkeymanifest_sample_config.xml')])

        #Update OEMKeyManifest config xml for sample/test params
        tree = ET.parse(os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'))

        xml_change_list = get_oemkeymanifest_change_list()
        for each in xml_change_list:
            for xml_path, value in each:
                node = tree.find('%s' % xml_path)
                node.set('value', value)

        tree.write(os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'), xml_declaration=True, encoding='utf-8')

        #Generate signed OEMKeyManifest binary
        #meu.exe -f decomp_km1.xml -o OemExtInputFile.bin -key oem_privkey_3072.pem -stp C:\Openssl\openssl.exe

        run_process ([meu_path, '-f', os.path.join(output_dir, 'oemkeymanifest_sample_config.xml'), '-o', oem_bin_sign,
            '-key',  os.path.join (key_dir, oem_priv_key),
            '-cfg', os.path.join(output_dir, 'meu_config.xml'),
            '-stp', openssl_path])
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
      StitchIfwiConfig_adln.py: get_xml_change_list()
    StitchIfwi.py: patch_xml_file()
    StitchIfwi.py: update_btGuard_manifests ()
      security_stitch_help.py: update_btGuard_manifests ()
        security_stitch_help.py: gen_bpmgen2_params()
          StitchIfwiConfig_adln.py: get_bpmgen2_params_change_list()
        security_stitch_help.py: sign_slimboot_binary()
          BtgSign.py: sign_slimboot_binary()
            BtgSign.py: bpm_sign_binary()
        StitchIfwi.py: gen_sign_oem_key_manifest()
          BtgSign.py: gen_sign_oem_key_manifest()
            StitchIfwiConfig_adln.py: get_oemkeymanifest_change_list ()
        security_stitch_help.py: update_btGuard_xml()
        security_stitch_help.py: update_tpm_type()
```

Platform/CommonBoardPkg/Script/security_stitch_help.py:

```
def update_btGuard_manifests(stitch_dir, stitch_cfg_file, btg_profile, tpm):
...
    gen_sign_oem_key_manifest(meu_path, oem_bin_input, bpm_key_dir, key_size, output_dir, oem_bin_output)

    # Update xml for Btg
    updated_xml_file = os.path.join (stitch_dir, 'Temp', 'updated.xml')

    tree = ET.parse(updated_xml_file)
    update_btGuard_xml(btg_profile, stitch_dir, tree)
    update_tpm_type(tpm, tree)

    tree.write(updated_xml_file)
```

Platform/CommonBoardPkg/Script/security_stitch_help.py:

```
def update_btGuard_manifests(stitch_dir, stitch_cfg_file, btg_profile, tpm):
 ...
    gen_sign_oem_key_manifest(meu_path, oem_bin_input, bpm_key_dir, key_size, output_dir, oem_bin_output)

    # Update xml for Btg
    updated_xml_file = os.path.join (stitch_dir, 'Temp', 'updated.xml')

    tree = ET.parse(updated_xml_file)
    update_btGuard_xml(btg_profile, stitch_dir, tree)
    update_tpm_type(tpm, tree)

    tree.write(updated_xml_file)
```

Platform/CommonBoardPkg/Script/security_stitch_help.py:

```
def update_btGuard_xml(btg_profile, stitch_dir, tree):
    output_dir = os.path.join(stitch_dir, "Temp")
    kmsigpubkeyhashfile = os.path.join(output_dir, "kmsigpubkey.hash")

    if btg_profile == 'vm':
        btguardprofile = 3
    elif btg_profile == 'fve':
        btguardprofile = 4
    elif btg_profile == 'fvme':
        btguardprofile = 5
    else:
       print ("Boot Guard is NOT enabled.....")
       btguardprofile = 0

    # Convert OEM key hash into hexadecimal for Fit tool consumption
    with open(kmsigpubkeyhashfile,"rb") as kmsigpubkeyhash_fh:
        hash = bytearray (kmsigpubkeyhash_fh.read())
        oemkeyhash = ""
        for b in hash:
            oemkeyhash = oemkeyhash + "%02X " % b
        oemkeyhash = oemkeyhash[:-1]
    print("oemkeyhash:")
    print(oemkeyhash)

    node = tree.find('./PlatformProtection/PlatformIntegrity/OemPublicKeyHash')
    node.attrib['value'] = oemkeyhash

    node = tree.find('./PlatformProtection/BootGuardConfiguration/BtGuardKeyManifestId')
    node.attrib['value'] = '0x1'

    node = tree.find('./PlatformProtection/PlatformIntegrity/OemExtInputFile')
    node.attrib['value'] = os.path.join(output_dir, "OemExtInputFile.bin")

    node = tree.find('./PlatformProtection/BootGuardConfiguration/BtGuardProfileConfig')
    node.attrib['value'] = btg_profile_values[btguardprofile]
```

Platform/CommonBoardPkg/Script/security_stitch_help.py:

```
def update_btGuard_xml(btg_profile, stitch_dir, tree):
    output_dir = os.path.join(stitch_dir, "Temp")
    kmsigpubkeyhashfile = os.path.join(output_dir, "kmsigpubkey.hash")

    if btg_profile == 'vm':
        btguardprofile = 3
    elif btg_profile == 'fve':
        btguardprofile = 4
    elif btg_profile == 'fvme':
        btguardprofile = 5
    else:
       print ("Boot Guard is NOT enabled.....")
       btguardprofile = 0

    # Convert OEM key hash into hexadecimal for Fit tool consumption
    with open(kmsigpubkeyhashfile,"rb") as kmsigpubkeyhash_fh:
        hash = bytearray (kmsigpubkeyhash_fh.read())
        oemkeyhash = ""
        for b in hash:
            oemkeyhash = oemkeyhash + "%02X " % b
        oemkeyhash = oemkeyhash[:-1]
    print("oemkeyhash:")
    print(oemkeyhash)

    node = tree.find('./PlatformProtection/PlatformIntegrity/OemPublicKeyHash')
    node.attrib['value'] = oemkeyhash

    node = tree.find('./PlatformProtection/BootGuardConfiguration/BtGuardKeyManifestId')
    node.attrib['value'] = '0x1'

    node = tree.find('./PlatformProtection/PlatformIntegrity/OemExtInputFile')
    node.attrib['value'] = os.path.join(output_dir, "OemExtInputFile.bin")

    node = tree.find('./PlatformProtection/BootGuardConfiguration/BtGuardProfileConfig')
    node.attrib['value'] = btg_profile_values[btguardprofile]
```

Platform/CommonBoardPkg/Script/security_stitch_help.py:

```
def update_btGuard_xml(btg_profile, stitch_dir, tree):
    output_dir = os.path.join(stitch_dir, "Temp")
    kmsigpubkeyhashfile = os.path.join(output_dir, "kmsigpubkey.hash")

    if btg_profile == 'vm':
        btguardprofile = 3
    elif btg_profile == 'fve':
        btguardprofile = 4
    elif btg_profile == 'fvme':
        btguardprofile = 5
    else:
       print ("Boot Guard is NOT enabled.....")
       btguardprofile = 0

    # Convert OEM key hash into hexadecimal for Fit tool consumption
    with open(kmsigpubkeyhashfile,"rb") as kmsigpubkeyhash_fh:
        hash = bytearray (kmsigpubkeyhash_fh.read())
        oemkeyhash = ""
        for b in hash:
            oemkeyhash = oemkeyhash + "%02X " % b
        oemkeyhash = oemkeyhash[:-1]
    print("oemkeyhash:")
    print(oemkeyhash)

    node = tree.find('./PlatformProtection/PlatformIntegrity/OemPublicKeyHash')
    node.attrib['value'] = oemkeyhash

    node = tree.find('./PlatformProtection/BootGuardConfiguration/BtGuardKeyManifestId')
    node.attrib['value'] = '0x1'

    node = tree.find('./PlatformProtection/PlatformIntegrity/OemExtInputFile')
    node.attrib['value'] = os.path.join(output_dir, "OemExtInputFile.bin")

    node = tree.find('./PlatformProtection/BootGuardConfiguration/BtGuardProfileConfig')
    node.attrib['value'] = btg_profile_values[btguardprofile]
```

Platform/CommonBoardPkg/Script/security_stitch_help.py:

```
def update_btGuard_xml(btg_profile, stitch_dir, tree):
    output_dir = os.path.join(stitch_dir, "Temp")
    kmsigpubkeyhashfile = os.path.join(output_dir, "kmsigpubkey.hash")

    if btg_profile == 'vm':
        btguardprofile = 3
    elif btg_profile == 'fve':
        btguardprofile = 4
    elif btg_profile == 'fvme':
        btguardprofile = 5
    else:
       print ("Boot Guard is NOT enabled.....")
       btguardprofile = 0

    # Convert OEM key hash into hexadecimal for Fit tool consumption
    with open(kmsigpubkeyhashfile,"rb") as kmsigpubkeyhash_fh:
        hash = bytearray (kmsigpubkeyhash_fh.read())
        oemkeyhash = ""
        for b in hash:
            oemkeyhash = oemkeyhash + "%02X " % b
        oemkeyhash = oemkeyhash[:-1]
    print("oemkeyhash:")
    print(oemkeyhash)

    node = tree.find('./PlatformProtection/PlatformIntegrity/OemPublicKeyHash')
    node.attrib['value'] = oemkeyhash

    node = tree.find('./PlatformProtection/BootGuardConfiguration/BtGuardKeyManifestId')
    node.attrib['value'] = '0x1'

    node = tree.find('./PlatformProtection/PlatformIntegrity/OemExtInputFile')
    node.attrib['value'] = os.path.join(output_dir, "OemExtInputFile.bin")

    node = tree.find('./PlatformProtection/BootGuardConfiguration/BtGuardProfileConfig')
    node.attrib['value'] = btg_profile_values[btguardprofile]
```

Platform/CommonBoardPkg/Script/security_stitch_help.py:

```
def update_btGuard_xml(btg_profile, stitch_dir, tree):
    output_dir = os.path.join(stitch_dir, "Temp")
    kmsigpubkeyhashfile = os.path.join(output_dir, "kmsigpubkey.hash")

    if btg_profile == 'vm':
        btguardprofile = 3
    elif btg_profile == 'fve':
        btguardprofile = 4
    elif btg_profile == 'fvme':
        btguardprofile = 5
    else:
       print ("Boot Guard is NOT enabled.....")
       btguardprofile = 0

    # Convert OEM key hash into hexadecimal for Fit tool consumption
    with open(kmsigpubkeyhashfile,"rb") as kmsigpubkeyhash_fh:
        hash = bytearray (kmsigpubkeyhash_fh.read())
        oemkeyhash = ""
        for b in hash:
            oemkeyhash = oemkeyhash + "%02X " % b
        oemkeyhash = oemkeyhash[:-1]
    print("oemkeyhash:")
    print(oemkeyhash)

    node = tree.find('./PlatformProtection/PlatformIntegrity/OemPublicKeyHash')
    node.attrib['value'] = oemkeyhash

    node = tree.find('./PlatformProtection/BootGuardConfiguration/BtGuardKeyManifestId')
    node.attrib['value'] = '0x1'

    node = tree.find('./PlatformProtection/PlatformIntegrity/OemExtInputFile')
    node.attrib['value'] = os.path.join(output_dir, "OemExtInputFile.bin")

    node = tree.find('./PlatformProtection/BootGuardConfiguration/BtGuardProfileConfig')
    node.attrib['value'] = btg_profile_values[btguardprofile]
```

Provisioning script callstack

```
StitchIfwi.py: main()
  StitchIfwi.py: stitch()
    StitchIfwi.py: add_platform_data()
    StitchIfwi.py: replace_components()
      StitchIfwiConfig_adln.py: get_component_replace_list()
    StitchIfwi.py: gen_xml_file()
      StitchIfwiConfig_adln.py: get_platform_sku()
      StitchIfwiConfig_adln.py: get_xml_change_list()
    StitchIfwi.py: patch_xml_file()
    StitchIfwi.py: update_btGuard_manifests ()
      security_stitch_help.py: update_btGuard_manifests ()
        security_stitch_help.py: gen_bpmgen2_params()
          StitchIfwiConfig_adln.py: get_bpmgen2_params_change_list()
        security_stitch_help.py: sign_slimboot_binary()
          BtgSign.py: sign_slimboot_binary()
            BtgSign.py: bpm_sign_binary()
        StitchIfwi.py: gen_sign_oem_key_manifest()
          BtgSign.py: gen_sign_oem_key_manifest()
            StitchIfwiConfig_adln.py: get_oemkeymanifest_change_list ()
        security_stitch_help.py: update_btGuard_xml()
        security_stitch_help.py: update_tpm_type()
    StitchIfwiConfig_adln.py: get_platform_sku()
```

StitchIfwi.py:

```
def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    if sign_bin_flag:
        update_btGuard_manifests(stitch_dir, stitch_cfg_file, btg_profile, tpm)
    else:
        shutil.copy(os.path.join(temp_dir, "SlimBootloader.bin"), os.path.join(temp_dir, "BiosRegion.bin"))

    print ("Run fit tool to generate ifwi.........")
    sku = stitch_cfg_file.get_platform_sku().get(platform)
    run_process (['./Fit/mfit', '-l', sku, '--loadconfig', os.path.join (temp_dir, 'updated.xml'), '--build', 'Temp/Ifwi.bin',
        '--workingdir', temp_dir, '--destdir', temp_dir])
    return 0
```

StitchIfwi.py:

```
def stitch (stitch_dir, stitch_cfg_file, sbl_file, btg_profile, plt_params_list, platform_data, platform, tpm, full_rdundant = True):
...
    if sign_bin_flag:
        update_btGuard_manifests(stitch_dir, stitch_cfg_file, btg_profile, tpm)
    else:
        shutil.copy(os.path.join(temp_dir, "SlimBootloader.bin"), os.path.join(temp_dir, "BiosRegion.bin"))

    print ("Run fit tool to generate ifwi.........")
    sku = stitch_cfg_file.get_platform_sku().get(platform)
    run_process (['./Fit/mfit', '-l', sku, '--loadconfig', os.path.join (temp_dir, 'updated.xml'), '--build', 'Temp/Ifwi.bin',
        '--workingdir', temp_dir, '--destdir', temp_dir])
    return 0
```

Preparing the dependencies for the provisioning

So far we have gathered the dependencies based on our study of Python scripts:

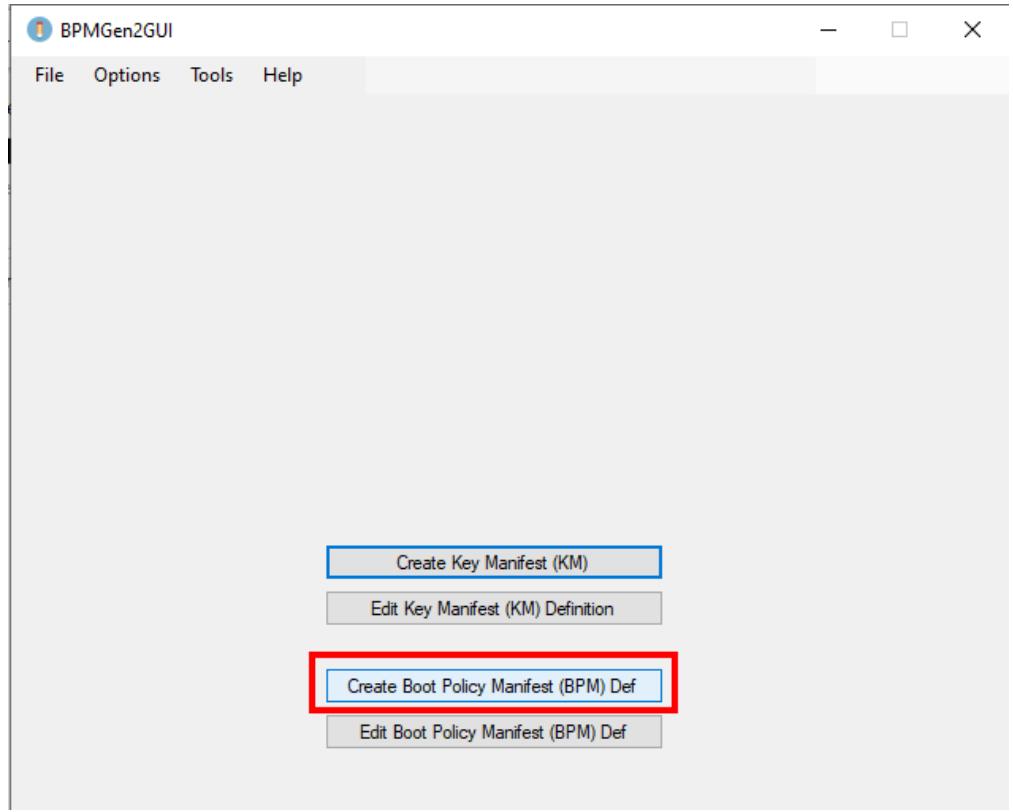
```
# Components from get_xml_change_list
Input/PmcBinary.bin
Input/PchcSubPartitionData.bin
Input/Iunit.bin
Input/ChipInitBinary.bin
Input/Iom.bin
Input/TypeC_NorthPHYRegion.bin
Input/MeRegionFile.bin
Input/acm0.bin
# Tools
Meu/meu
Fit/mfit
BpmGen2/bpmgen2
BpmGen2/bpmgen2.params
```

We know how to get the tools, but what about the rest?

Generating `bpmgen2.params`

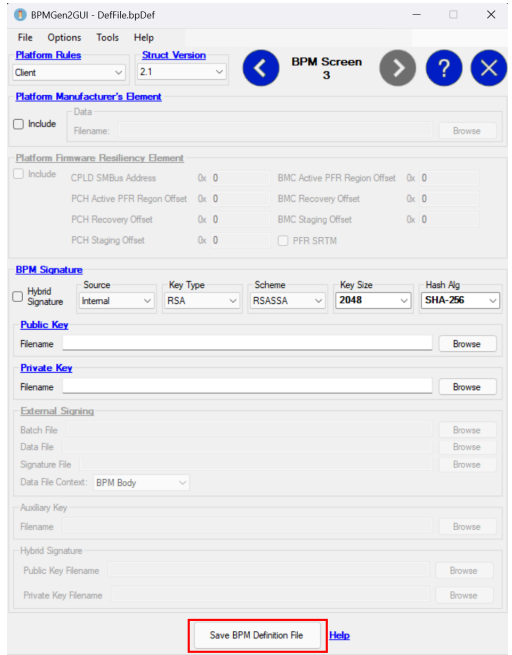
- Generating `bpmgen2.params` is tricky
- Only possible using the GUI version for Windows
- Download BpmGen2 from Intel R&DC (kit number 573188)
  - Currently `release2025-08-04` version available and used in the training
- Unpack the ZIP and run `BpmGen2GUI.exe` on Windows
- Select the working directory to be the one where BpmGen2 was unpacked

Practice #304 Exercise #1



Practice #304 Exercise #1

Practice #304 Exercise #1



Resulting file should be placed in the selected working directory as `DefFile.bpDef` (if the name was not changed).

Transfer it to the VM to `~/training_materials/src/slimbootloader/BpmGen2/bpmgen2.params` and the `bpmgen2` Linux utility to `~/training_materials/src/slimbootloader/BpmGen2/bpmgen2`.

```
# FILEHEADER
FileID: _BPMDEF_
FileVersion:  1
ToolVersion:  7
ToolDate: 20000101
FileDate: 20250811
//
# BPM_DEF
PlatformRules:  Client
BpmStrutVersion:  0x21
BpmRevision:  1
BpmRevocation:  1
AcmRevocation:  2
NEMPages: 0x4
IbbSetCount:  1
CurrentIbbSet:  0
//
# IBB_SET
IbbSetType: 0:ColdBoot
IbbSetInclude:  TRUE
PBETValue:  0xF
MCHBAR: 0x00000000FEDC0000
VTD_BAR:  0x00000000FED91000
DmaProtBase0: 0x0
DmaProtLimit0:  0x0
DmaProtBase1: 0x0
DmaProtLimit1:  0x0
IbbFlags: 0x3
//    Bit0 : Enable DMA Protection;
//    Bit1 : Issue TPM Start-up from Locality 3;
//    Bit2 : Extend Authority Measurements into the Authority PCR;
//    Bit3 : On error: Leave TPM Hierarchies enabled. Cap all PCRs;
//    Bit4 : Top Swap Supported;
//    Bit5 : Force (MK)TME;
//    Bit6 : Enforce SPIRAL specification between BIOS/CSE
```

```
DmaProtAutoCalc:  TRUE
IbbHashAlgID: 0x0B
IbbEntry: 0xFFFFFFF0
PostIbbHashAlgID: 0x10
PostIBBHashSource:  Calculate
PostIbbHashFile:
IbbSegSource: FIT
IbbStubHash:  False
IbbSegFile:
IbbGuid:  4a4ca1c6-871c-45bb-8801-6910a7aa5807
ObbHashAlgID: 0x10
ObbFullFvHash:  FALSE
ObbHashSource:  File
ObbHashFile:
//
# TXT_ELEMENT
TxtInclude: FALSE
MinSvn: 0x0
TxtFlags: 0x0
// [4:0] = TXT execution profile
//     00000b – Use Default based on HW
//     00001b - Server Profile
//     00010b - Client Profile
// [6:5] = “Memory scrubbing” policy
//     00b - Trust Verified BIOS
//     01b - Trust Any BIOS
//     10b - Trust No BIOS
// [8:7] = Backup Policy
//     00b - Default
//     01b - Power Down
//     10b - Unbreakable Shutdown
//     11b - PFR Recovery
// [31] = Reset AUX control (1=AUX Reset leaf will delete AUX Index)
```

```
//MemoryDepletion Power Down
StrideSize: 0×0
AcpiBase: 0×400
PwrmBase: 0×FE000000
PdUseDefault: TRUE
PdMinutes:  5
PdSeconds:  0
PttCmosOffset0: 0×7E
PttCmosOffset1: 0×7F
//TXTE Segments
TxtSegSource: IBB
TxtSegGuid: 4a4ca1c6-871c-45bb-8801-6910a7aa5807
TxtSegHashAlgID:  0×10
//
# PLATFORM_CONFIG_ELEMENT
PcdInclude: TRUE
PdReqLocation:  TPM
//    Power down request location for CMOS
CmosIndexRegister:  0×70
CmosDataRegister: 0×71
CmosIndexOffset:  125
CmosBitFieldWidth:  3
CmosBitFieldPosition: 0
//
# TPM1.2_LOCATION
TpmIndexHandle: 0×50000004
TpmByteOffset:  7
TpmBitFieldWidth: 3
TpmBitFieldPosition:  0
//
# TPM2.0_LOCATION
TpmIndexHandle: 0×1C10104
TpmByteOffset:  7
TpmBitFieldWidth: 3
TpmBitFieldPosition:  0
//
```

```
# PTT_LOCATION
TpmIndexHandle: 0×1C10104
TpmByteOffset:  7
TpmBitFieldWidth: 3
TpmBitFieldPosition:  0
//
# COMMUNICATION_NEM_BUFFER
CnbsInclude:  False
CnbsBase: 0×FF000000
CnbsSize: 0×1000
//
# STACK_DATA_BUFFER
SdbsInclude:  False
SdbsBase: 0×0
SdbsSize: 0×0
//
# PLATFORM_MANUFACTURERS_ELEMENT
PmdeInclude:  FALSE
PmdeFile:
//
# BPM_SIGNATURE
BpmSigSource: Internal
BpmSigHashAlgID:  0×0B
BpmHybridSigMode: FALSE
BpmSigKeyType:  0×01
BpmSigScheme: 0×14
BpmKeySizeBits: 2048
BpmSigPubKey:
BpmSigPrivKey:
BpmSigAux:
BpmSigBatch:
BpmSigData:
BpmSigDataType: BPM Body
BpmSigXSig:
//
#EOF
```

Preparing Intel ME components.

1. Download Intel CSE ADL-N 16.50.10.1351v7 A0 Consumer toolkit (kit number 793965).
2. Transfer it to the VM and unpack the ZIP.
3. Navigate to `cd <toolkit_dir>/Tools/System_Tools/MFIT/Linux64/`.
4. Make the binary executable `chmod +x mfit`.
5. Decompose ODROID-H4 image:

```
./mfit --decompose ~/slimbootloader/Outputs/odroid_h4/ifwi-debug.bin
```

The decomposed image will be placed in `<toolkit_dir>/Tools/System_Tools/MFIT/Linux64/ifwi-debug`.

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Results of the decomposition:

```
ls ifwi-debug/Decompose/
BiosPlugin#BiosRegion.bin
CsePlugin#IOM.bin
CsePlugin#IUNIT.bin
CsePlugin#NPHY.bin
CsePlugin#OEM_KM.bin
CsePlugin#PCHC.bin
CsePlugin#PMC.bin
CsePlugin#UTOK.bin
DescriptorPlugin#EcRegionPointer.bin
DescriptorPlugin#OEM.bin
ifwi-debug.map
ifwi-debug.txt
'ME Sub Partition.bin'
NVARS#IshPdt#IshPdtBinary.bin
NVARS#MphyTable#ChipInitBinary.bin
NVARS#TraceHubFiltering#RomTraceFiltering.bin
```

Now let's copy the dependencies:

```
cd ifwi-debug/Decompose/
cp CsePlugin#PMC.bin                    ~/training_materials/src/slimbootloader/Input/PmcBinary.bin
cp CsePlugin#PCHC.bin                   ~/training_materials/src/slimbootloader/Input/PchcSubPartitionData.bin
cp CsePlugin#IUNIT.bin                  ~/training_materials/src/slimbootloader/Input/Iunit.bin
cp NVARS#MphyTable#ChipInitBinary.bin   ~/training_materials/src/slimbootloader/Input/ChipInitBinary.bin
cp CsePlugin#IOM.bin                    ~/training_materials/src/slimbootloader/Input/Iom.bin
cp CsePlugin#NPHY.bin                   ~/training_materials/src/slimbootloader/Input/TypeC_NorthPHYRegion.bin
cp ME\ Sub\ Partition.bin               ~/training_materials/src/slimbootloader/Input/MeRegionFile.bin
```

Final structure of directories and dependencies for provisioning:

```
slimbootloader
├── BpmGen2
│   ├── bpmgen2
│   └── bpmgen2.params
├── Fit
│   └── mfit
├── Input
│   ├── acm0.bin
│   ├── ChipInitBinary.bin
│   ├── Iom.bin
│   ├── Iunit.bin
│   ├── MeRegionFile.bin
│   ├── PchcSubPartitionData.bin
│   ├── PmcBinary.bin
│   └── TypeC_NorthPHYRegion.bin
└── Meu
    └── meu
```

Practice #304 Exercise #3

Obtaining the Boot Guard ACM

1. Search Intel R&DC for `Alder Lake Startup ACM`.
2. Download the kit 836699 (BIOS ACM 1.18.19 and SINIT ACM 1.18.18).
3. Transfer the ZIP to VM and unpack it. Copy the ACM to SBL directory:

```
unzip ADL_RPL_StartupACM_1.18.19_SINITACM_1.18.18.zip
cd ADL_RPL_StartupACM_1.18.19_SINITACM_1.18.18/BIOSACM
cp ADL_BIOSAC_v1_18_19_20240903_REL_NT_O1.PW_signed_256K.bin \
    ~/training_materials/src/slimbootloader/Input/acm0.bin
```

Practice #304 Exercise #4

Provisioning Slim Bootloader for Boot Guard.

1. If you haven't produced a binary yet, build a debug image: `DEBUG=1 ./build.sh odroid_h4`
2. Patch the scripts to make the provisioning pass:

```
diff --git a/Platform/AlderlakeBoardPkg/Script/StitchIfwiConfig_adln.py b/Platform/AlderlakeBoardPkg/Script/StitchIfwiConfig_adln.py
index 58c64cff..96bb68a8 100644
--- a/Platform/AlderlakeBoardPkg/Script/StitchIfwiConfig_adln.py
+++ b/Platform/AlderlakeBoardPkg/Script/StitchIfwiConfig_adln.py
@@ -68,7 +68,8 @@ def get_bpmgen2_params_change_list ():

 def get_platform_sku():
     platform_sku ={
-        'adln'    :   'Intel(R) TwinLake Chipset - Consumer - SPI',
+        'adln'    :   'Intel(R) AlderLake N Chipset - Consumer - SPI',
+        'twl'     :   'Intel(R) TwinLake Chipset - Consumer - SPI',
     }
     return platform_sku

@@ -95,6 +96,7 @@ def check_parameter(para_list):
         'rvp_ddr5' : {},
         'quad'   : {},
         'debug'  : {},
+        '16MB'   : {},
         'tsn'    : {}
       }

@@ -103,6 +105,7 @@ def check_parameter(para_list):
         'crb'    -- Stitch for CRB board
         'rvp_ddr5' -- Stitch for RVP DDR5 board
         'quad'   -- Stitch IFWI with SPI QUAD mode
+        '16MB'   -- Stitch image set to 16MB, by default use 32MB.
         'debug'  -- Enable DAM and DCI configuration (Only use for debug purpose but not for final production!)
         'tsn'    -- Stitch sample Tsn Mac address binary along with TSN AIC softstraps
         """
```

Practice #304 Exercise #4

```
@@ -140,16 +143,16 @@ def get_xml_change_list (platform, plt_params_list):
      # Path                                                  | value |
      # ============================================================================
      #Region Order
-       ('./FlashLayout/BiosRegion/InputFile',                                '$SourceDir\BiosRegion.bin'),
-       ('./FlashLayout/Ifwi_IntelMePmcRegion/MeRegionFile',                  '$SourceDir\MeRegionFile.bin'),
-       ('./FlashLayout/Ifwi_IntelMePmcRegion/ChipInitBinary',                '$SourceDir\ChipInitBinary.bin'),
-       ('./FlashLayout/Ifwi_IntelMePmcRegion/PmcBinary',                     '$SourceDir\PmcBinary.bin'),
-       ('./FlashLayout/SubPartitions/PchcSubPartitionData/InputFile',        '$SourceDir\PchcSubPartitionData.bin'),
-       ('./FlashLayout/SubPartitions/IunitSubPartition/InputFile',           '$SourceDir\Iunit.bin'),
+       ('./FlashLayout/BiosRegion/InputFile',                                '$SourceDir/BiosRegion.bin'),
+       ('./FlashLayout/Ifwi_IntelMePmcRegion/MeRegionFile',                  '$SourceDir/MeRegionFile.bin'),
+       ('./FlashLayout/Ifwi_IntelMePmcRegion/ChipInitBinary',                '$SourceDir/ChipInitBinary.bin'),
+       ('./FlashLayout/Ifwi_IntelMePmcRegion/PmcBinary',                     '$SourceDir/PmcBinary.bin'),
+       ('./FlashLayout/SubPartitions/PchcSubPartitionData/InputFile',        '$SourceDir/PchcSubPartitionData.bin'),
+       ('./FlashLayout/SubPartitions/IunitSubPartition/InputFile',           '$SourceDir/Iunit.bin'),
        ('./FlashSettings/VsccTable/VsccEntries/VsccEntry/VsccEntryName',     'VsccEntry0'),
        ('./FlashSettings/BiosConfiguration/BiosRedAssistance',               'Disabled'),
-       ('./PlatformProtection/PlatformIntegrity/OemExtInputFile',            '$SourceDir\OemExtInputFile.bin'),
+       ('./PlatformProtection/PlatformIntegrity/OemExtInputFile',            '$SourceDir/OemExtInputFile.bin'),
        ('./PlatformProtection/BootGuardConfiguration/BtGuardKeyManifestId',  '0x1'),
        ('./PlatformProtection/TpmOverSpiBusConfiguration/SpiOverTpmBusEnable', 'Yes'),
        ('./Icc/IccPolicies/Profiles/Profile/ClockOutputConfiguration/ClkoutSRC0', 'Enabled'),
```

Practice #304 Exercise #4

```
@@ -157,9 +160,9 @@ def get_xml_change_list (platform, plt_params_list):
        ('./Icc/IccPolicies/Profiles/Profile/ClockOutputConfiguration/ClkoutSRC3',   'Enabled'),
        ('./NetworkingConnectivity/WiredLanConfiguration/MELanPowerWell',           'SLP_LAN#'),
        ('./InternalPchBuses/EspiConfiguration/EspiEcSlvAtchdFlshMor',              'Multiple Outstanding Requests'),
-       ('./IntegratedSensorHub/IshImage/InputFile',                                '$SourceDir\IshImage.bin'),
-       ('./IntegratedSensorHub/IshImage/Length',                                   '0x58000'),
-       ('./IntegratedSensorHub/IshData/PdtBinary',                                 '$SourceDir\PdtBinary.bin'),
+     #  ('./IntegratedSensorHub/IshImage/InputFile',                                '$SourceDir/IshImage.bin'),
+     #  ('./IntegratedSensorHub/IshImage/Length',                                   '0x58000'),
+     #  ('./IntegratedSensorHub/IshData/PdtBinary',                                 '$SourceDir/PdtBinary.bin'),
        ('./Debug/DelayedAuthenticationModeConfiguration/DelayedAuthMode',          'No'),
        ('./Debug/IntelMeFirmwareDebuggingOverrides/DbgOverridePreProdSi',          '0x7'),
        ('./Debug/DirectConnectInterfaceConfiguration/DciDbcEnable',                'No'),
@@ -167,8 +170,8 @@ def get_xml_change_list (platform, plt_params_list):
        ('./FlexIO/SataPcieComboPortConfiguration/SataPCIeComboPort1',              'GPIO Polarity PCIe'),
        ('./FlexIO/Usb2PortConfiguration/USB2Prt8ConTypeSel',                       'Express Card / M.2 S2'),
        ('./FlexIO/Type-CSubsystemConfiguration/TypeCPort1Config',                  'No Thunderbolt'),
-       ('./FlexIO/Type-CSubsystemConfiguration/IomBinaryFile',                     '$SourceDir\Iom.bin'),
-       ('./FlexIO/Type-CSubsystemConfiguration/PhyBinaryFile',                     '$SourceDir\TypeC_NorthPHYRegion.bin'),
+       ('./FlexIO/Type-CSubsystemConfiguration/IomBinaryFile',                     '$SourceDir/Iom.bin'),
+       ('./FlexIO/Type-CSubsystemConfiguration/PhyBinaryFile',                     '$SourceDir/TypeC_NorthPHYRegion.bin'),
        ('./FlexIO/PowerDelivery_PdControllerConfiguration/TypeCPort1RetimerEnabled','No'),
        ('./FlexIO/PowerDelivery_PdControllerConfiguration/TypeCPort2RetimerEnabled','No'),
        ('./Gpio/GpioVccioVoltageControl/GppD10voltSelect',                         '1.8Volts'),
```

Practice #304 Exercise #4

```
@@ -227,8 +230,8 @@ def get_xml_change_list (platform, plt_params_list):
          ('./BuildSettings/HarnessGlobalData/HarnessRevision',                        '#25'),
          ('./BuildSettings/HarnessGlobalData/SelectedRvp',                            'TWL / ADL-N DDR5 (ADP-P + ADL-N / TWL)'),
          ('./FlashLayout/EcRegion/Enabled',                                           'Enabled'),
-         ('./FlashLayout/EcRegion/InputFile',                                         '$SourceDir\EcRegion.bin'),
-         ('./FlashLayout/EcRegion/EcRegionPointer',                                   '$SourceDir\EcRegionPointer.bin'),
+         ('./FlashLayout/EcRegion/InputFile',                                         '$SourceDir/EcRegion.bin'),
+         ('./FlashLayout/EcRegion/EcRegionPointer',                                   '$SourceDir/EcRegionPointer.bin'),
          ('./FlashSettings/FlashConfiguration/SpiDualIoReadEnable',                   'Yes'),
          ('./FlashSettings/FlashConfiguration/SpiDualOutReadEnable',                  'Yes'),
          ('./IntelMeKernel/IntelMeFirmwareUpdate/HmrfpoEnable',                       'Yes'),
@@ -283,6 +286,12 @@ def get_xml_change_list (platform, plt_params_list):
          ('./NetworkingConnectivity/TimeSensitiveNetworkingConfiguration/TsnEnabled', 'TSN MIPI Clock'),
        ])

+   if '16MB' in plt_params_list:
+       print ("Applying changes to generate 16MB image")
+       xml_change_list.append ([
+           ('./BuildSettings/BuildResults/FlashComponentsSizes',                    '16'),
+           ])
+
    if 'quad' in plt_params_list:
        print ("Applying changes to enable spi quad")
        xml_change_list.append ([
```

Once done, run the provisioning:

```
python Platform/AlderlakeBoardPkg/Script/StitchIfwi.py \
    -b vm \
    -p adln \
    -w ~/training_materials/src/slimbootloader \
    -s Outputs/odroid_h4/SlimBootloader.bin \
    -c Platform/AlderlakeBoardPkg/Script/StitchIfwiConfig_adln.py \
    -d 0×AAFFFF0C \
    -k SblTestKeys/ \
    -o 16MB;
```

The process should end with:

```
IFWI Stitching completed successfully !
Boot Guard Profile: VM
IFWI image: sbl_ifwi_adln.bin
```

Extract the ME configuration XML from a firmware image files.

This exercise aims to show what differences were made compared to the unprovisioned image and whether they are dangerous for the platform hardware.

1. Navigate to: `<toolkit_dir>/Tools/System_Tools/MFIT/Linux64`
2. Run:

```
./mfit --decompose ~/training_materials/src/slimbootloader/Output/odroid_h4/ifwi-debug.bin` \
    --saveconfig config_unprovisioned.xml
```

> Each mfit command should end with:
>
> ```
> Decompose '<image_path>' image completed successfully
> 'config_unprovisioned.xml' generated successfully
> ```

3. Run:

```
./mfit --decompose ~/training_materials/src/slimbootloader/sbl_ifwi_adln.bin` \
    --saveconfig config_provisioned.xml
```

4. Run `diff config_unprovisioned.xml config_provisioned.xml`

XML differences:

```
113,116c113,116
<          <SpiDualOutReadEnable value="Yes" value_list="['No', 'Yes']" ...
<          <SpiDualIoReadEnable value="Yes" value_list="['No', 'Yes']" ...
<          <QuadOutReadEnable value="No" value_list="['No', 'Yes']" ...
<          <QuadIoReadEnable value="No" value_list="['No', 'Yes']" ...
---
>          <SpiDualOutReadEnable value="No" value_list="['No', 'Yes']" ...
>          <SpiDualIoReadEnable value="No" value_list="['No', 'Yes']" ...
>          <QuadOutReadEnable value="Yes" value_list="['No', 'Yes']" ...
>          <QuadIoReadEnable value="Yes" value_list="['No', 'Yes']" ...
448c448
<          <BtGuardProfileConfig value="Boot Guard Profile 0 - No_FVME" ...
---
>          <BtGuardProfileConfig value="Boot Guard Profile 3 - VM" ...
```

XML differences:

```
113,116c113,116
<          <SpiDualOutReadEnable value="Yes" value_list="['No', 'Yes']" ...
<          <SpiDualIoReadEnable value="Yes" value_list="['No', 'Yes']" ...
<          <QuadOutReadEnable value="No" value_list="['No', 'Yes']" ...
<          <QuadIoReadEnable value="No" value_list="['No', 'Yes']" ...
---
>          <SpiDualOutReadEnable value="No" value_list="['No', 'Yes']" ...
>          <SpiDualIoReadEnable value="No" value_list="['No', 'Yes']" ...
>          <QuadOutReadEnable value="Yes" value_list="['No', 'Yes']" ...
>          <QuadIoReadEnable value="Yes" value_list="['No', 'Yes']" ...
448c448
<          <BtGuardProfileConfig value="Boot Guard Profile 0 - No_FVME" ...
---
>          <BtGuardProfileConfig value="Boot Guard Profile 3 - VM" ...
```

XML differences:

```
113,116c113,116
<          <SpiDualOutReadEnable value="Yes" value_list="['No', 'Yes']" ...
<          <SpiDualIoReadEnable value="Yes" value_list="['No', 'Yes']" ...
<          <QuadOutReadEnable value="No" value_list="['No', 'Yes']" ...
<          <QuadIoReadEnable value="No" value_list="['No', 'Yes']" ...
---
>          <SpiDualOutReadEnable value="No" value_list="['No', 'Yes']" ...
>          <SpiDualIoReadEnable value="No" value_list="['No', 'Yes']" ...
>          <QuadOutReadEnable value="Yes" value_list="['No', 'Yes']" ...
>          <QuadIoReadEnable value="Yes" value_list="['No', 'Yes']" ...
448c448
<          <BtGuardProfileConfig value="Boot Guard Profile 0 - No_FVME" ...
---
>          <BtGuardProfileConfig value="Boot Guard Profile 3 - VM" ...
```

Practice #304 Exercise #5

```
718,720c718,720
<          <TcssPortEnMask value="0x1" ...
<          <TypeCPort1Config value="DP Fixed Connection" ...
<          <TypeCPort2Config value="No Thunderbolt" ...
---
>          <TcssPortEnMask value="0xF" ...
>          <TypeCPort1Config value="No Thunderbolt" ...
>          <TypeCPort2Config value="No Restrictions" ...
786c786
<          <GppE16voltSelect value="1.8Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_E16 Individual Voltage Selec
---
>          <GppE16voltSelect value="3.3Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_E16 Individual Voltage Selec
814,815c814,815
<          <GppD1voltSelect value="3.3Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D1 Individual Voltage Select"
<          <GppD2voltSelect value="3.3Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D2 Individual Voltage Select"
---
>          <GppD1voltSelect value="1.8Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D1 Individual Voltage Select"
>          <GppD2voltSelect value="1.8Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D2 Individual Voltage Select"
```

Practice #304 Exercise #5

```
718,720c718,720
<           <TcssPortEnMask value="0x1" ...
<           <TypeCPort1Config value="DP Fixed Connection" ...
<           <TypeCPort2Config value="No Thunderbolt" ...
---
>           <TcssPortEnMask value="0xF" ...
>           <TypeCPort1Config value="No Thunderbolt" ...
>           <TypeCPort2Config value="No Restrictions" ...
786c786
<           <GppE16voltSelect value="1.8Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_E16 Individual Voltage Selec
---
>           <GppE16voltSelect value="3.3Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_E16 Individual Voltage Selec
814,815c814,815
<           <GppD1voltSelect value="3.3Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D1 Individual Voltage Select"
<           <GppD2voltSelect value="3.3Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D2 Individual Voltage Select"
---
>           <GppD1voltSelect value="1.8Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D1 Individual Voltage Select"
>           <GppD2voltSelect value="1.8Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D2 Individual Voltage Select"
```

Practice #304 Exercise #5

```
718,720c718,720
<          <TcssPortEnMask value="0x1" ...
<          <TypeCPort1Config value="DP Fixed Connection" ...
<          <TypeCPort2Config value="No Thunderbolt" ...
---
>          <TcssPortEnMask value="0xF" ...
>          <TypeCPort1Config value="No Thunderbolt" ...
>          <TypeCPort2Config value="No Restrictions" ...
786c786
<          <GppE16voltSelect value="1.8Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_E16 Individual Voltage Selec
---
>          <GppE16voltSelect value="3.3Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_E16 Individual Voltage Selec
814,815c814,815
<          <GppD1voltSelect value="3.3Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D1 Individual Voltage Select"
<          <GppD2voltSelect value="3.3Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D2 Individual Voltage Select"
---
>          <GppD1voltSelect value="1.8Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D1 Individual Voltage Select"
>          <GppD2voltSelect value="1.8Volts" value_list="['3.3Volts', '1.8Volts']" label="GPP_D2 Individual Voltage Select"
```

Results of provisioning script

- The provisioning script otuptus a lot of information from the tools being executed
  - Some of the outptu are simpyl duplicated due to BpmGen2 being executed two times (to generate Key Manifest and Boot Policy Manfiest) and for each Slim Bootlaoder Top Swap region (yet another two times).
- Most of the output is not so much interesting, except when an error happens.
- I will mainly focus on the output from BpmGen2, because most of the ME configuration was already discussed or will be discussed in the next presentation about fuses and ME configuration.
  - BpmGen2 besides the signign has a capability to parse and dump information about FIT table and Boot Guard manifests in the image (if present)
  - We will leverage it to explain what are the field of manfiests used for
- Most of it is explained in the CBnT BIOS Writers Guide (doc 575623) and Boot Guard BIOS specification (doc 557867)
- Similar output that will be discussed, can be achieved by running:

```
./bpmgen2 INFO sbl_ifwi_adln.bin
```

BpmGen2 output - FIT

```
###############
# FIT Table: #
###############
    FIT Pointer Offset: 0x40
    FIT Table Address:  0xffff5280
======  ================  ======  ========  ==============  ===  ========  (======  ===  ======  ===  ======)
Index:      Address       Size    Version       Type       C_V  Checksum  (Index  Data Width  Bit  Offset)
======  ================  ======  ========  ==============  ===  ========  (======  ===  ======  ===  ======)
 00:   2020205f5449465f 00000b   0100   00-'_FIT_   '   01     65
 01:   00000000ffc2d000 000000   0100   01-MICROCODE    00     00
 02:   00000000ffc68000 000000   0100   01-MICROCODE    00     00
 03:   00000000ffca3000 000000   0100   01-MICROCODE    00     00
 04:   00000000ffcde000 000000   0100   01-MICROCODE    00     00
 05:   00000000fff80000 000000   0100   02-STARTUP_ACM  00     00
 06:   00000000ffff5340 000acc   0100   07-BIOS_MODULE  00     00
 07:   00000000fffe5000 001028   0100   07-BIOS_MODULE  00     00
 08:   00000000ffe00000 010000   0100   07-BIOS_MODULE  00     00
 09:   00000000fffe4600 000400   0100   0b-KEYMANIFEST  00     00
 10:   00000000fffe4a00 000600   0100   0c-BP_MANIFEST  00     00
======  ================  ======  ========  ==============  ===  ========  (======  ===  ======  ===  ======)
Index:      Address       Size    Version       Type       C_V  Checksum  (Index  Data Width  Bit  Offset)
======  ================  ======  ========  ==============  ===  ========  (======  ===  ======  ===  ======)
```

BpmGen2 output - FIT

```
###############
# FIT Table: #
###############
    FIT Pointer Offset: 0x40
    FIT Table Address:  0xffff5280
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
Index:      Address      Size  Version       Type      C_V  Checksum (Index  Data Width  Bit  Offset)
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
 00:   2020205f5449465f 00000b   0100   00-'_FIT_   '   01     65
 01:   00000000ffc2d000 000000   0100   01-MICROCODE    00     00
 02:   00000000ffc68000 000000   0100   01-MICROCODE    00     00
 03:   00000000ffca3000 000000   0100   01-MICROCODE    00     00
 04:   00000000ffcde000 000000   0100   01-MICROCODE    00     00
 05:   00000000fff80000 000000   0100   02-STARTUP_ACM  00     00
 06:   00000000ffff5340 000acc   0100   07-BIOS_MODULE  00     00
 07:   00000000fffe5000 001028   0100   07-BIOS_MODULE  00     00
 08:   00000000ffe00000 010000   0100   07-BIOS_MODULE  00     00
 09:   00000000fffe4600 000400   0100   0b-KEYMANIFEST  00     00
 10:   00000000fffe4a00 000600   0100   0c-BP_MANIFEST  00     00
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
Index:      Address      Size  Version       Type      C_V  Checksum (Index  Data Width  Bit  Offset)
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
```

BpmGen2 output - FIT

```
###############
# FIT Table: #
###############
    FIT Pointer Offset: 0x40
    FIT Table Address:  0xffff5280
```

Index:	Address	Size	Version	Type	C_V	Checksum	(Index	Data Width	Bit	Offset)
00:	2020205f5449465f	00000b	0100	00-'_FIT_ '	01	65				
01:	00000000ffc2d000	000000	0100	01-MICROCODE	00	00				
02:	00000000ffc68000	000000	0100	01-MICROCODE	00	00				
03:	00000000ffca3000	000000	0100	01-MICROCODE	00	00				
04:	00000000ffcde000	000000	0100	01-MICROCODE	00	00				
05:	00000000fff80000	000000	0100	02-STARTUP_ACM	00	00				
06:	00000000ffff5340	000acc	0100	07-BIOS_MODULE	00	00				
07:	00000000fffe5000	001028	0100	07-BIOS_MODULE	00	00				
08:	00000000ffe00000	010000	0100	07-BIOS_MODULE	00	00				
09:	00000000fffe4600	000400	0100	0b-KEYMANIFEST	00	00				
10:	00000000fffe4a00	000600	0100	0c-BP_MANIFEST	00	00				
Index:	Address	Size	Version	Type	C_V	Checksum	(Index	Data Width	Bit	Offset)

BpmGen2 output - FIT

```
###############
# FIT Table: #
###############
    FIT Pointer Offset: 0x40
    FIT Table Address:  0xffff5280
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
Index:      Address      Size  Version       Type      C_V  Checksum (Index  Data Width  Bit  Offset)
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
 00:   2020205f5449465f 00000b   0100   00-'_FIT_   '   01     65
 01:   00000000ffc2d000 000000   0100   01-MICROCODE    00     00
 02:   00000000ffc68000 000000   0100   01-MICROCODE    00     00
 03:   00000000ffca3000 000000   0100   01-MICROCODE    00     00
 04:   00000000ffcde000 000000   0100   01-MICROCODE    00     00
 05:   00000000fff80000 000000   0100   02-STARTUP_ACM  00     00
 06:   00000000ffff5340 000acc   0100   07-BIOS_MODULE  00     00
 07:   00000000fffe5000 001028   0100   07-BIOS_MODULE  00     00
 08:   00000000ffe00000 010000   0100   07-BIOS_MODULE  00     00
 09:   00000000fffe4600 000400   0100   0b-KEYMANIFEST  00     00
 10:   00000000fffe4a00 000600   0100   0c-BP_MANIFEST  00     00
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
Index:      Address      Size  Version       Type      C_V  Checksum (Index  Data Width  Bit  Offset)
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
```

BpmGen2 output - FIT

```
###############
# FIT Table: #
###############
    FIT Pointer Offset: 0x40
    FIT Table Address:  0xffff5280
```

Index:	Address	Size	Version	Type	C_V	Checksum	(Index	Data Width	Bit	Offset)
00:	2020205f5449465f	00000b	0100	00-'_FIT_ '	01	65				
01:	00000000ffc2d000	000000	0100	01-MICROCODE	00	00				
02:	00000000ffc68000	000000	0100	01-MICROCODE	00	00				
03:	00000000ffca3000	000000	0100	01-MICROCODE	00	00				
04:	00000000ffcde000	000000	0100	01-MICROCODE	00	00				
05:	00000000fff80000	000000	0100	02-STARTUP_ACM	00	00				
06:	00000000ffff5340	000acc	0100	07-BIOS_MODULE	00	00				
07:	00000000fffe5000	001028	0100	07-BIOS_MODULE	00	00				
08:	00000000ffe00000	010000	0100	07-BIOS_MODULE	00	00				
09:	00000000fffe4600	000400	0100	0b-KEYMANIFEST	00	00				
10:	00000000fffe4a00	000600	0100	0c-BP_MANIFEST	00	00				

Index:	Address	Size	Version	Type	C_V	Checksum	(Index	Data Width	Bit	Offset)

BpmGen2 output - FIT

```
###############
# FIT Table: #
###############
    FIT Pointer Offset: 0x40
    FIT Table Address:  0xffff5280
======  ================  ======  ========  ==============  ===  ========  (======  ===  ======  ===  ======)
Index:      Address       Size   Version       Type       C_V  Checksum  (Index   Data Width   Bit   Offset)
======  ================  ======  ========  ==============  ===  ========  (======  ===  ======  ===  ======)
 00:   2020205f5449465f 00000b   0100   00-'_FIT_   '   01     65
 01:   00000000ffc2d000 000000   0100   01-MICROCODE    00     00
 02:   00000000ffc68000 000000   0100   01-MICROCODE    00     00
 03:   00000000ffca3000 000000   0100   01-MICROCODE    00     00
 04:   00000000ffcde000 000000   0100   01-MICROCODE    00     00
 05:   00000000fff80000 000000   0100   02-STARTUP_ACM  00     00
 06:   00000000ffff5340 000acc   0100   07-BIOS_MODULE  00     00
 07:   00000000fffe5000 001028   0100   07-BIOS_MODULE  00     00
 08:   00000000ffe00000 010000   0100   07-BIOS_MODULE  00     00
 09:   00000000fffe4600 000400   0100   0b-KEYMANIFEST  00     00
 10:   00000000fffe4a00 000600   0100   0c-BP_MANIFEST  00     00
======  ================  ======  ========  ==============  ===  ========  (======  ===  ======  ===  ======)
Index:      Address       Size   Version       Type       C_V  Checksum  (Index   Data Width   Bit   Offset)
======  ================  ======  ========  ==============  ===  ========  (======  ===  ======  ===  ======)
```

BpmGen2 output - FIT

```
###############
# FIT Table: #
###############
    FIT Pointer Offset: 0x40
    FIT Table Address:  0xffff5280
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
Index:      Address      Size  Version       Type      C_V  Checksum (Index  Data Width  Bit  Offset)
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
 00:   2020205f5449465f 00000b   0100   00-'_FIT_   '   01     65
 01:   00000000ffc2d000 000000   0100   01-MICROCODE    00     00
 02:   00000000ffc68000 000000   0100   01-MICROCODE    00     00
 03:   00000000ffca3000 000000   0100   01-MICROCODE    00     00
 04:   00000000ffcde000 000000   0100   01-MICROCODE    00     00
 05:   00000000fff80000 000000   0100   02-STARTUP_ACM  00     00
 06:   00000000ffff5340 000acc   0100   07-BIOS_MODULE  00     00
 07:   00000000fffe5000 001028   0100   07-BIOS_MODULE  00     00
 08:   00000000ffe00000 010000   0100   07-BIOS_MODULE  00     00
 09:   00000000fffe4600 000400   0100   0b-KEYMANIFEST  00     00
 10:   00000000fffe4a00 000600   0100   0c-BP_MANIFEST  00     00
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
Index:      Address      Size  Version       Type      C_V  Checksum (Index  Data Width  Bit  Offset)
====== ================ ====== ======== ============== === ======== (====== === ====== === ======)
```

BpmGen2 - Key Manifest

```
#################
# Key Manifest #
#################
  StructureID:    __KEYM__
  StructVersion:  0x21
  Reserved:       0x00 00 00
  KeySigOffset:   0x0044
  Reserved:       0x00 00 00
  KeyManifestVer: 0x01
  KMSVN:          0x00
  KeyManifestID:  0x01
  KmPubKey Alg:   0x000c - 0x0C:SHA384
  Number of Manifest Key Digests:   1
  KeyHashes:
  [1] Usage:      0x1 For: Boot Policy Manifest,
      HashAlg:    0x000b - 0x0B:SHA256
      Size:       0x0020
      HashBuffer: 33bd9465ed9548c0ec62d2d5c1ab4ffaa164d469efc470733a3469000ceef571
  Signature Structure:
    Version:      0x10
     KeyAlg:      0x0001 0x01:RSA
    RsaPublicKeyStructure:
       Version:   0x10
       KeySize:   0x0c00
```

BpmGen2 - Key Manifest

```
#################
# Key Manifest #
#################
  StructureID:    __KEYM__
  StructVersion:  0x21
  Reserved:       0x00 00 00
  KeySigOffset:   0x0044
  Reserved:       0x00 00 00
  KeyManifestVer: 0x01
  KMSVN:          0x00
  KeyManifestID:  0x01
  KmPubKey Alg:   0x000c - 0x0C:SHA384
  Number of Manifest Key Digests:   1
  KeyHashes:
  [1] Usage:      0x1 For: Boot Policy Manifest,
      HashAlg:    0x000b - 0x0B:SHA256
      Size:       0x0020
      HashBuffer: 33bd9465ed9548c0ec62d2d5c1ab4ffaa164d469efc470733a3469000ceef571
  Signature Structure:
    Version:      0x10
     KeyAlg:      0x0001 0x01:RSA
    RsaPublicKeyStructure:
       Version:   0x10
       KeySize:   0x0c00
```

BpmGen2 - Key Manifest

```
#################
# Key Manifest #
#################
  StructureID:    __KEYM__
  StructVersion:  0x21
  Reserved:       0x00 00 00
  KeySigOffset:   0x0044
  Reserved:       0x00 00 00
  KeyManifestVer: 0x01
  KMSVN:          0x00
  KeyManifestID:  0x01
  KmPubKey Alg:   0x000c - 0x0C:SHA384
  Number of Manifest Key Digests:   1
  KeyHashes:
  [1] Usage:      0x1 For: Boot Policy Manifest,
      HashAlg:    0x000b - 0x0B:SHA256
      Size:       0x0020
      HashBuffer: 33bd9465ed9548c0ec62d2d5c1ab4ffaa164d469efc470733a3469000ceef571
  Signature Structure:
    Version:      0x10
     KeyAlg:      0x0001 0x01:RSA
    RsaPublicKeyStructure:
       Version:   0x10
       KeySize:   0x0c00
```

BpmGen2 - Key Manifest

```
#################
# Key Manifest #
#################
  StructureID:    __KEYM__
  StructVersion:  0x21
  Reserved:       0x00 00 00
  KeySigOffset:   0x0044
  Reserved:       0x00 00 00
  KeyManifestVer: 0x01
  KMSVN:          0x00
  KeyManifestID:  0x01
  KmPubKey Alg:   0x000c - 0x0C:SHA384
  Number of Manifest Key Digests:   1
  KeyHashes:
  [1] Usage:      0x1 For: Boot Policy Manifest,
      HashAlg:    0x000b - 0x0B:SHA256
      Size:       0x0020
      HashBuffer: 33bd9465ed9548c0ec62d2d5c1ab4ffaa164d469efc470733a3469000ceef571
  Signature Structure:
    Version:      0x10
     KeyAlg:      0x0001 0x01:RSA
    RsaPublicKeyStructure:
       Version:   0x10
       KeySize:   0x0c00
```

BpmGen2 - Key Manifest

- At the end of Key Manifest output the tool prints he hashes of the OEM Root Key.
- It is informative content. The hash of modulus and exponent should match the hash being provisioned in the ME emulated fuses.
- The additional modulus only hash was used in older microarchitectures (Skylake/Kaby Lake), that required only hash of the modulus to be configured in the emulated fuses.

```
# FYI: KM Public Key Hash Digest (Modulus+Exponent)#
  63 48 70 62 a7 06 92 68 55 6b 81 a6 b3 50 97 2c
  ae b6 3f d1 51 01 eb f3 a0 58 bb 20 de 08 6b 56
  34 bc b5 b1 b6 56 37 c6 5e 4b fa c7 6c 0a e7 09

# FYI: KM Public Key Hash Digest (Modulus Only)#
  50 b4 c9 2b 63 b1 82 22 a8 b3 57 6d d2 1c a3 f1
  78 db 74 47 03 e7 dd 26 15 5d 52 ab 7a 2e 29 57
  0a b4 23 63 2d bd 57 93 f2 d4 ec dd a5 0c 60 1a
```

BpmGen2 - Boot Policy Manifest

```
######################
# BootPolicyManifest #
######################
BootPolicyManifestHeader:
  StructureID:    __ACBP__
  StructVersion:  0x21
  HdrStructVer:   0x20
  HdrSize:        0x0014
  KeySigOffset:   0x015c
  BpmRevision:    0x01
  BpmRevocation:  0x01
  AcmRevocation:  0x02
  Reserved:       00
  NEMPages:       0x0003
```

BpmGen2 - Boot Policy Manifest

```
#########################
# BootPolicyManifest #
#########################
BootPolicyManifestHeader:
  StructureID:    __ACBP__
  StructVersion:  0x21
  HdrStructVer:   0x20
  HdrSize:        0x0014
  KeySigOffset:   0x015c
  BpmRevision:    0x01
  BpmRevocation:  0x01
  AcmRevocation:  0x02
  Reserved:       00
  NEMPages:       0x0003
```

BpmGen2 - Boot Policy Manifest

```
#######################
# BootPolicyManifest #
#######################
BootPolicyManifestHeader:
  StructureID:    __ACBP__
  StructVersion:  0×21
  HdrStructVer:   0×20
  HdrSize:        0×0014
  KeySigOffset:   0×015c
  BpmRevision:    0×01
  BpmRevocation:  0×01
  AcmRevocation:  0×02
  Reserved:       00
  NEMPages:       0×0003
```

BpmGen2 - Boot Policy Manifest

```
IbbElement:
  StructureID:    __IBBS__
  StructVersion:  0x20
  Reserved:       00
  Element Size:   0108 (264)
  Reserved:       00
  SetType:        0x00
  Reserved:       00
  PBETValue:      0x0f
  Flags:          0x00000013
    Enable VT-d:  1
    InitMeasLoc3: 1
    AuthorityMeas:0
    TPM on Error: 0
    Top Swap:     1
    Force (MK)TME:0
    SpiralEnforce:0
    SRTM Attest:  0
    Force CNB:    0
    SVN_OVR:      0
  IBB_MCHBAR:     0x00000000fedc0000
  VTD_BAR:        0x00000000fed91000
  DmaProtBase0:   0xffe00000
  DmaProtLimit0:  0xffff0000
```

BpmGen2 - Boot Policy Manifest

```
IbbElement:
  StructureID:    __IBBS__
  StructVersion:  0x20
  Reserved:       00
  Element Size:   0108 (264)
  Reserved:       00
  SetType:        0x00
  Reserved:       00
  PBETValue:      0x0f
  Flags:          0x00000013
    Enable VT-d:  1
    InitMeasLoc3: 1
    AuthorityMeas:0
    TPM on Error: 0
    Top Swap:     1
    Force (MK)TME:0
    SpiralEnforce:0
    SRTM Attest:  0
    Force CNB:    0
    SVN_OVR:      0
  IBB_MCHBAR:     0x00000000fedc0000
  VTD_BAR:        0x00000000fed91000
  DmaProtBase0:   0xffe00000
  DmaProtLimit0:  0xffff0000
```

BpmGen2 - Boot Policy Manifest

```
IbbElement:
  StructureID:    __IBBS__
  StructVersion:  0x20
  Reserved:       00
  Element Size:   0108 (264)
  Reserved:       00
  SetType:        0x00
  Reserved:       00
  PBETValue:      0x0f
  Flags:          0x00000013
    Enable VT-d:  1
    InitMeasLoc3: 1
    AuthorityMeas:0
    TPM on Error: 0
    Top Swap:     1
    Force (MK)TME:0
    SpiralEnforce:0
    SRTM Attest:  0
    Force CNB:    0
    SVN_OVR:      0
  IBB_MCHBAR:     0x00000000fedc0000
  VTD_BAR:        0x00000000fed91000
  DmaProtBase0:   0xffe00000
  DmaProtLimit0:  0xffff0000
```

BpmGen2 - Boot Policy Manifest

```
IbbElement:
  StructureID:    __IBBS__
  StructVersion:  0x20
  Reserved:       00
  Element Size:   0108 (264)
  Reserved:       00
  SetType:        0x00
  Reserved:       00
  PBETValue:      0x0f
  Flags:          0x00000013
    Enable VT-d:  1
    InitMeasLoc3: 1
    AuthorityMeas:0
    TPM on Error: 0
    Top Swap:     1
    Force (MK)TME:0
    SpiralEnforce:0
    SRTM Attest:  0
    Force CNB:    0
    SVN_OVR:      0
  IBB_MCHBAR:     0x00000000fedc0000
  VTD_BAR:        0x00000000fed91000
  DmaProtBase0:   0xffe00000
  DmaProtLimit0:  0xffff0000
```

BpmGen2 - Boot Policy Manifest

```
PostIbbHash:
    HashAlg:    0x0010 - 0x10:NULL
    Size:       0x0000
    HashBuffer:
IbbEntry:       0xfffffff0
HashList        (Number of Digests: 4, Total Size: 152)
  [0] HashAlg:    0x000c - 0x0C:SHA384
      Size:       0x0030
      HashBuffer: e7455c48371559adc5b472a61c0b72caf4740a0d5b3ab1cec0ff1bad3fb05ca3f4b6821389ade643fc83656b62b2658b
  [1] HashAlg:    0x0004 - 0x04:SHA1
      Size:       0x0014
      HashBuffer: 8161f9c870f1fde4bba44296b9060fa80d43ba82
  [2] HashAlg:    0x000b - 0x0B:SHA256
      Size:       0x0020
      HashBuffer: e7e722061c48d76a69b14cb5908fbbc9c8175bd4df49a216269de0c5db59d0c5
  [3] HashAlg:    0x0012 - 0x12:SM3
      Size:       0x0020
      HashBuffer: 934eda2ff5ca4cb1fffb7426ab51bf8207c00fa38f3ee21d1cd430d2d6db8f86
OBB Digest:
    HashAlg:    0x0010 - 0x10:NULL
    Size:       0x0000
    HashBuffer:
Reserved:       00  00 00
Segment Count:  0x03
 Seg#  Base------  Size------  Flags-  Reserved  Measured  Cache Type---
 [ 0]  0xffe00000  0x00100000  0x0000  00, 00    Yes       Write Protect
 [ 1]  0xfffe5000  0x00010280  0x0000  00, 00    Yes       Write Protect
 [ 2]  0xffff5340  0x0000acc0  0x0000  00, 00    Yes       Write Protect
```

BpmGen2 - Boot Policy Manifest

```
PostIbbHash:
    HashAlg:    0x0010 - 0x10:NULL
    Size:       0x0000
    HashBuffer:
IbbEntry:       0xfffffff0
HashList        (Number of Digests: 4, Total Size: 152)
  [0] HashAlg:    0x000c - 0x0C:SHA384
      Size:       0x0030
      HashBuffer: e7455c48371559adc5b472a61c0b72caf4740a0d5b3ab1cec0ff1bad3fb05ca3f4b6821389ade643fc83656b62b2658b
  [1] HashAlg:    0x0004 - 0x04:SHA1
      Size:       0x0014
      HashBuffer: 8161f9c870f1fde4bba44296b9060fa80d43ba82
  [2] HashAlg:    0x000b - 0x0B:SHA256
      Size:       0x0020
      HashBuffer: e7e722061c48d76a69b14cb5908fbbc9c8175bd4df49a216269de0c5db59d0c5
  [3] HashAlg:    0x0012 - 0x12:SM3
      Size:       0x0020
      HashBuffer: 934eda2ff5ca4cb1fffb7426ab51bf8207c00fa38f3ee21d1cd430d2d6db8f86
OBB Digest:
    HashAlg:    0x0010 - 0x10:NULL
    Size:       0x0000
    HashBuffer:
Reserved:       00  00 00
Segment Count:  0x03
 Seg#  Base------  Size------  Flags-  Reserved  Measured  Cache Type---
 [ 0]  0xffe00000  0x00100000  0x0000  00, 00    Yes       Write Protect
 [ 1]  0xfffe5000  0x00010280  0x0000  00, 00    Yes       Write Protect
 [ 2]  0xffff5340  0x0000acc0  0x0000  00, 00    Yes       Write Protect
```

BpmGen2 - Boot Policy Manifest

```
PostIbbHash:
    HashAlg:    0x0010 - 0x10:NULL
    Size:       0x0000
    HashBuffer:
IbbEntry:       0xfffffff0
HashList        (Number of Digests: 4, Total Size: 152)
  [0] HashAlg:    0x000c - 0x0C:SHA384
      Size:       0x0030
      HashBuffer: e7455c48371559adc5b472a61c0b72caf4740a0d5b3ab1cec0ff1bad3fb05ca3f4b6821389ade643fc83656b62b2658b
  [1] HashAlg:    0x0004 - 0x04:SHA1
      Size:       0x0014
      HashBuffer: 8161f9c870f1fde4bba44296b9060fa80d43ba82
  [2] HashAlg:    0x000b - 0x0B:SHA256
      Size:       0x0020
      HashBuffer: e7e722061c48d76a69b14cb5908fbbc9c8175bd4df49a216269de0c5db59d0c5
  [3] HashAlg:    0x0012 - 0x12:SM3
      Size:       0x0020
      HashBuffer: 934eda2ff5ca4cb1fffb7426ab51bf8207c00fa38f3ee21d1cd430d2d6db8f86
OBB Digest:
    HashAlg:    0x0010 - 0x10:NULL
    Size:       0x0000
    HashBuffer:
Reserved:       00  00 00
Segment Count:  0x03
 Seg#  Base------  Size------  Flags-  Reserved  Measured  Cache Type---
 [ 0]  0xffe00000  0x00100000  0x0000  00, 00    Yes       Write Protect
 [ 1]  0xfffe5000  0x00010280  0x0000  00, 00    Yes       Write Protect
 [ 2]  0xffff5340  0x0000acc0  0x0000  00, 00    Yes       Write Protect
```

BpmGen2 - Boot Policy Manifest

```
PostIbbHash:
    HashAlg:    0x0010 - 0x10:NULL
    Size:       0x0000
    HashBuffer:
IbbEntry:       0xfffffff0
HashList        (Number of Digests: 4, Total Size: 152)
  [0] HashAlg:    0x000c - 0x0C:SHA384
      Size:       0x0030
      HashBuffer: e7455c48371559adc5b472a61c0b72caf4740a0d5b3ab1cec0ff1bad3fb05ca3f4b6821389ade643fc83656b62b2658b
  [1] HashAlg:    0x0004 - 0x04:SHA1
      Size:       0x0014
      HashBuffer: 8161f9c870f1fde4bba44296b9060fa80d43ba82
  [2] HashAlg:    0x000b - 0x0B:SHA256
      Size:       0x0020
      HashBuffer: e7e722061c48d76a69b14cb5908fbbc9c8175bd4df49a216269de0c5db59d0c5
  [3] HashAlg:    0x0012 - 0x12:SM3
      Size:       0x0020
      HashBuffer: 934eda2ff5ca4cb1fffb7426ab51bf8207c00fa38f3ee21d1cd430d2d6db8f86
OBB Digest:
    HashAlg:    0x0010 - 0x10:NULL
    Size:       0x0000
    HashBuffer:
Reserved:       00  00 00
Segment Count:  0x03
 Seg#  Base------  Size------  Flags-  Reserved  Measured  Cache Type---
 [ 0]  0xffe00000  0x00100000  0x0000  00, 00    Yes       Write Protect
 [ 1]  0xfffe5000  0x00010280  0x0000  00, 00    Yes       Write Protect
 [ 2]  0xffff5340  0x0000acc0  0x0000  00, 00    Yes       Write Protect
```

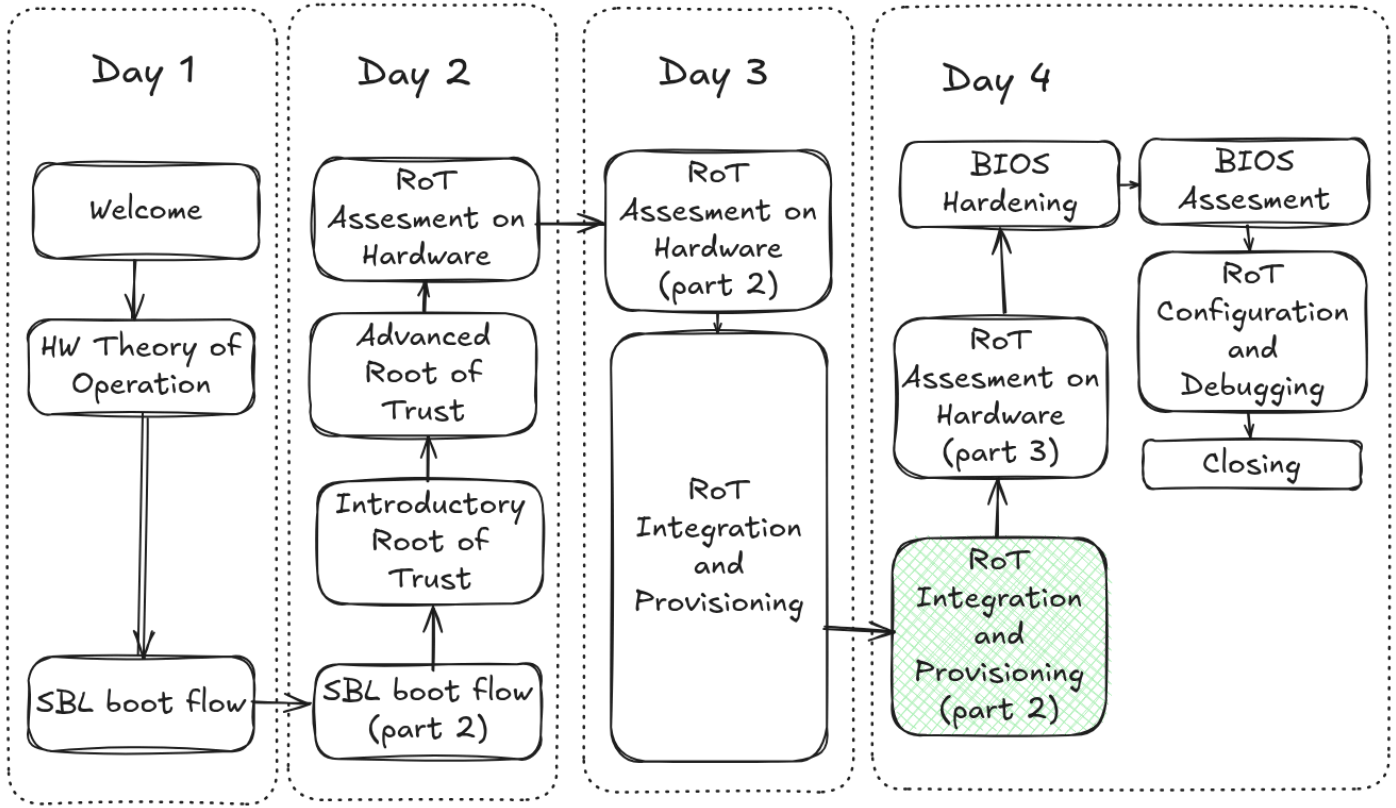
BPmGen2 - final FIT

```
Final FIT table
###############
# FIT Table: #
###############
    FIT Pointer Offset: 0x40
    FIT Table Address:  0xffff5280
═══════ ═══════════════ ═══════ ═══════ ═══════════════ ═══ ═══════ (═══════ ═══ ═══════ ═══ ═══════)
Index:      Address      Size  Version       Type      C_V  Checksum (Index  Data Width  Bit  Offset)
═══════ ═══════════════ ═══════ ═══════ ═══════════════ ═══ ═══════ (═══════ ═══ ═══════ ═══ ═══════)
 00:   2020205f5449465f 00000b   0100   00-'_FIT_   '   01     78
 01:   00000000ff95a000 000000   0100   01-MICROCODE    00     00
 02:   00000000ff995000 000000   0100   01-MICROCODE    00     00
 03:   00000000ff9d0000 000000   0100   01-MICROCODE    00     00
 04:   00000000ffa0b000 000000   0100   01-MICROCODE    00     00
 05:   00000000fff80000 000000   0100   02-STARTUP_ACM  00     00
 06:   00000000ffff5340 000acc   0100   07-BIOS_MODULE  00     00
 07:   00000000fffe5000 001028   0100   07-BIOS_MODULE  00     00
 08:   00000000ffb2d000 010000   0100   07-BIOS_MODULE  00     00
 09:   00000000fffe4600 000355   0100   0b-KEYMANIFEST  00     00
 10:   00000000fffe4a00 00046d   0100   0c-BP_MANIFEST  00     00
═══════ ═══════════════ ═══════ ═══════ ═══════════════ ═══ ═══════ (═══════ ═══ ═══════ ═══ ═══════)
Index:      Address      Size  Version       Type      C_V  Checksum (Index  Data Width  Bit  Offset)
═══════ ═══════════════ ═══════ ═══════ ═══════════════ ═══ ═══════ (═══════ ═══ ═══════ ═══ ═══════)
```

BPmGen2 - final FIT

```
Final FIT table
###############
# FIT Table: #
###############
    FIT Pointer Offset: 0×40
    FIT Table Address:  0×ffff5280
═══════ ═══════════════ ═══════ ═══════ ═══════════════ ═══ ═══════ (═══════ ═══ ═══════ ═══ ═══════)
Index:      Address      Size  Version       Type      C_V  Checksum (Index  Data Width  Bit  Offset)
═══════ ═══════════════ ═══════ ═══════ ═══════════════ ═══ ═══════ (═══════ ═══ ═══════ ═══ ═══════)
 00:   2020205f5449465f 00000b   0100   00-'_FIT_   '   01     78
 01:   00000000ff95a000 000000   0100   01-MICROCODE    00     00
 02:   00000000ff995000 000000   0100   01-MICROCODE    00     00
 03:   00000000ff9d0000 000000   0100   01-MICROCODE    00     00
 04:   00000000ffa0b000 000000   0100   01-MICROCODE    00     00
 05:   00000000fff80000 000000   0100   02-STARTUP_ACM  00     00
 06:   00000000ffff5340 000acc   0100   07-BIOS_MODULE  00     00
 07:   00000000fffe5000 001028   0100   07-BIOS_MODULE  00     00
 08:   00000000ffb2d000 010000   0100   07-BIOS_MODULE  00     00
 09:   00000000fffe4600 000355   0100   0b-KEYMANIFEST  00     00
 10:   00000000fffe4a00 00046d   0100   0c-BP_MANIFEST  00     00
═══════ ═══════════════ ═══════ ═══════ ═══════════════ ═══ ═══════ (═══════ ═══ ═══════ ═══ ═══════)
Index:      Address      Size  Version       Type      C_V  Checksum (Index  Data Width  Bit  Offset)
═══════ ═══════════════ ═══════ ═══════ ═══════════════ ═══ ═══════ (═══════ ═══ ═══════ ═══ ═══════)
```

Where we are in the course



Practice #304 Exercise #6

Running the Boot Guard provisioned Slim Bootloader image.

1. Set the jumper to boot from flash containing vendor firmware.
2. Transfer the provisioned `sbl_ifwi_adln.bin` and `ifwi-debug.bin` to the USB stick with DTS.
3. Put the stick into the ODROID and boot from it.
4. Switch the flash chip jumper back.
5. Flash the `sbl_ifwi_adln.bin` using flashrom in DTS shell:

```
flashrom -p internal -w sbl_ifwi_adln.bin --ifd -i me -i bios
```

Important

We flash only the ME region and BIOS region. It is enough to prove that Boot Guard works and avoid risks mentioend earlier. However, in usual provisioned image deployment you should flash the whole binary, just in case the flash descriptor has been modified too. But then you will need to patch the XML properly by adding your platform specific settings to `StitchIfwiConfig_<platform>.py` (`adln` in case of ODROID-H4).

6. Additionally flash the descriptor from Dasharo image, to ensure we have the correct top swap size:

```
flashrom -p internal -w ifwi-debug.bin --ifd -i fd -N
```

7. Power cycle the board.

> **Note**
>
> Power cycle is required in this situation so that the ME configuration will be applied properly. ME is always picky when overwriting its region or the flash descriptor.

Slim Bootloader output on serial port:

```
============= Intel Slim Bootloader STAGE1B =============
[Boot Guard] AcmStatus : 0×C0008000
[Boot Guard] BootStatus: 0×98400000
[Boot Guard] Boot Guard is Enabled Successfully.
[Boot Guard] Acm Info: 0×730000006F
[Boot Guard] Verified Boot Status: Enabled
[Boot Guard] Measured Boot Status: Enabled
```

Conclusion: IT WORKS!

Unprovisioned binary

```
[Boot Guard] AcmStatus : 0×00000000
[Boot Guard] BootStatus: 0×00000000
[Boot Guard] Boot Guard Failed or is Disabled!
[Boot Guard] Acm Info: 0×7000000000
[Boot Guard] Verified Boot Status: Disabled
[Boot Guard] Measured Boot Status: Disabled
```

Provisioned binary

```
[Boot Guard] AcmStatus : 0×C0008000
[Boot Guard] BootStatus: 0×98400000
[Boot Guard] Boot Guard is Enabled Successfully.
[Boot Guard] Acm Info: 0×730000006F
[Boot Guard] Verified Boot Status: Enabled
[Boot Guard] Measured Boot Status: Enabled
```

Running `MEInfo` to analyze changes to the ME configuration

1. Transfer `<toolkit_dir>/Tools/System_Tools/MEInfo/Linux64/MEInfo` to USB stick with DTS.
2. Plug the sitck to ODROID and boot DTS to shell.
3. Run ME Info:

```
chmod +x MEInfo
./MEINfo
```

4. Compare output with the one from previous presentation.

FW Supported FPFs	FPF	UEP
		*In Use
	---	---
1st OEM Key Hash Revoked	Not set	Disabled
1st OEM Key Hash size	Not set	Enabled
1st OEM RSA Key size	Not set	Enabled
2nd OEM Key Hash Revoked	Not set	Disabled
2nd OEM Key Hash size	Not set	Enabled
2nd OEM RSA Key size	Not set	Enabled
BSMM Firmware Version Control	Not set	Enabled
CSE FW Firmware Version Control	Not set	Enabled
CSME Bootstrap Firmware Version Control	Not set	Enabled
DNX Firmware Version Control	Not set	Enabled
Error Enforcement Policy 0	Not set	Disabled
Error Enforcement Policy 1	Not set	Disabled
Flash Descriptor Verification	Not set	Disabled
Glitch Detection Disabled	Not set	Enabled
IDLM Firmware Version Control	Not set	Enabled
Intel PTT Encryption Key	Not set	Not Revoked
Intel(R) Manageability HW Fuse Status	Not set	Enabled
Intel(R) PTT	Not set	Enabled
OEM ID	Not set	0×00
OEM KM Firmware Version Control	Not set	Enabled
OEM Key Manifest	Not set	Enabled
OEM Key Revocation State	Not set	Disabled
OEM Platform ID	Not set	0×00
OEM Secure Boot Policy	Not set	0×7A
CPU Debugging	Not set	Disabled
BSP Initialization	Not set	Enabled
Protect BIOS Environment	Not set	Enabled
Measured Boot	Not set	Enabled
Verified Boot	Not set	Enabled
Key Manifest ID	Not set	0×01
Force Boot Guard ACM	Not set	Disabled

OEM key Hash RSA key size	Not set	Enabled
PID Refurbish Counter	Not set	0×00
PMC Firmware Version Control	Not set	Enabled
PTT Lockout Override Counter	Not set	0×00
Persistent PRTC Backup Power	Not set	Enabled
ROT Firmware Version Control	Not set	Enabled
RPMB Monotonic Counters	Not set	0×00
RPMC Rebinding	Not set	Enabled
RPMC Support	Not set	Enabled
SOC Config Lock State	Not set	Disabled
SPI Boot Source	Not set	Enabled
SPIRAL CPU	Not set	Enabled
Secure boot KM Firmware Version Control	Not set	Enabled
TXT Supported	Not set	Disabled
UFS Boot Source	Not set	Disabled
uCode Firmware Version Control	Not set	Enabled
DNX SVN	Not set	0×00
IDLM SVN	Not set	0×00
OEM KM SVN	Not set	0×00
PMC SVN	Not set	0×00
ROT KM SVN	Not set	0×00
Secure boot ACM SVN	Not set	0×00
Secure boot BSMM SVN	Not set	0×00
Secure boot KM SVN	Not set	0×00
Ucode SVN	Not set	0×00
1st OEM Public Key Hash FPF	Not set	
1st OEM Public Key Hash UEP		
63487062A7069268556B81A6B350972CAEB63FD15101EBF3		
A058BB20DE086B5634BCB5B1B65637C65E4BFAC76C0AE709		
2nd OEM Public Key Hash FPF	Not set	
2nd OEM Public Key Hash UEP		
0000000000000000000000000000000000000000000000000000		
0000000000000000000000000000000000000000000000000000		

Difference between unprovisioned and provisioned:

```
27c27
< OEM Secure Boot Policy                          Not set     0×42
---
> OEM Secure Boot Policy                          Not set     0×7A
30,32c30,32
< Protect BIOS Environment                        Not set     Disabled
< Measured Boot                                   Not set     Disabled
< Verified Boot                                   Not set     Disabled
---
> Protect BIOS Environment                        Not set     Enabled
> Measured Boot                                   Not set     Enabled
> Verified Boot                                   Not set     Enabled
46c46
< SPIRAL CPU                                      Not set     Disabled
---
> SPIRAL CPU                                      Not set     Enabled
64,65c64,65
<       0638EFCBB322E655C912D984070450BD22151A6BA96E53E8
<       6DDBAD48273EEE2BAB15DBE819696D989ABDC5EB9EB68A11
---
>       63487062A7069268556B81A6B350972CAEB63FD15101EBF3
>       A058BB20DE086B5634BCB5B1B65637C65E4BFAC76C0AE709
```

```
user@OST2-VM:~$ sha384sum  ~/training_materials/src/slimbootloader/Temp/kmsigpubkey.bin
```

Difference between unprovisioned and provisioned:

```
27c27
< OEM Secure Boot Policy                        Not set     0x42
---
> OEM Secure Boot Policy                        Not set     0x7A
30,32c30,32
< Protect BIOS Environment                      Not set     Disabled
< Measured Boot                                 Not set     Disabled
< Verified Boot                                 Not set     Disabled
---
> Protect BIOS Environment                      Not set     Enabled
> Measured Boot                                 Not set     Enabled
> Verified Boot                                 Not set     Enabled
46c46
< SPIRAL CPU                                    Not set     Disabled
---
> SPIRAL CPU                                    Not set     Enabled
64,65c64,65
<       0638EFCBB322E655C912D984070450BD22151A6BA96E53E8
<       6DDBAD48273EEE2BAB15DBE819696D989ABDC5EB9EB68A11
---
>       63487062A7069268556B81A6B350972CAEB63FD15101EBF3
>       A058BB20DE086B5634BCB5B1B65637C65E4BFAC76C0AE709
```

```
user@OST2-VM:~$ sha384sum  ~/training_materials/src/slimbootloader/Temp/kmsigpubkey.bin
```

Difference between unprovisioned and provisioned:

```
27c27
< OEM Secure Boot Policy                          Not set     0×42
---
> OEM Secure Boot Policy                          Not set     0×7A
30,32c30,32
< Protect BIOS Environment                        Not set     Disabled
< Measured Boot                                   Not set     Disabled
< Verified Boot                                   Not set     Disabled
---
> Protect BIOS Environment                        Not set     Enabled
> Measured Boot                                   Not set     Enabled
> Verified Boot                                   Not set     Enabled
46c46
< SPIRAL CPU                                      Not set     Disabled
---
> SPIRAL CPU                                      Not set     Enabled
64,65c64,65
<       0638EFCBB322E655C912D984070450BD22151A6BA96E53E8
<       6DDBAD48273EEE2BAB15DBE819696D989ABDC5EB9EB68A11
---
>       63487062A7069268556B81A6B350972CAEB63FD15101EBF3
>       A058BB20DE086B5634BCB5B1B65637C65E4BFAC76C0AE709
```

```
user@OST2-VM:~$ sha384sum  ~/training_materials/src/slimbootloader/Temp/kmsigpubkey.bin
```

Difference between unprovisioned and provisioned:

```
27c27
< OEM Secure Boot Policy                          Not set     0x42
---
> OEM Secure Boot Policy                          Not set     0x7A
30,32c30,32
< Protect BIOS Environment                        Not set     Disabled
< Measured Boot                                   Not set     Disabled
< Verified Boot                                   Not set     Disabled
---
> Protect BIOS Environment                        Not set     Enabled
> Measured Boot                                   Not set     Enabled
> Verified Boot                                   Not set     Enabled
46c46
< SPIRAL CPU                                      Not set     Disabled
---
> SPIRAL CPU                                      Not set     Enabled
64,65c64,65
<       0638EFCBB322E655C912D984070450BD22151A6BA96E53E8
<       6DDBAD48273EEE2BAB15DBE819696D989ABDC5EB9EB68A11
---
>       63487062A7069268556B81A6B350972CAEB63FD15101EBF3
>       A058BB20DE086B5634BCB5B1B65637C65E4BFAC76C0AE709
```

```
user@OST2-VM:~$ sha384sum  ~/training_materials/src/slimbootloader/Temp/kmsigpubkey.bin
```

Difference between unprovisioned and provisioned:

```
27c27
< OEM Secure Boot Policy                          Not set     0x42
---
> OEM Secure Boot Policy                          Not set     0x7A
30,32c30,32
< Protect BIOS Environment                        Not set     Disabled
< Measured Boot                                   Not set     Disabled
< Verified Boot                                   Not set     Disabled
---
> Protect BIOS Environment                        Not set     Enabled
> Measured Boot                                   Not set     Enabled
> Verified Boot                                   Not set     Enabled
46c46
< SPIRAL CPU                                      Not set     Disabled
---
> SPIRAL CPU                                      Not set     Enabled
64,65c64,65
<       0638EFCBB322E655C912D984070450BD22151A6BA96E53E8
<       6DDBAD48273EEE2BAB15DBE819696D989ABDC5EB9EB68A11
---
>       63487062A7069268556B81A6B350972CAEB63FD15101EBF3
>       A058BB20DE086B5634BCB5B1B65637C65E4BFAC76C0AE709
```

```
user@OST2-VM:~$ sha384sum  ~/training_materials/src/slimbootloader/Temp/kmsigpubkey.bin
```

Difference between unprovisioned and provisioned:

```
27c27
< OEM Secure Boot Policy                          Not set     0x42
---
> OEM Secure Boot Policy                          Not set     0x7A
30,32c30,32
< Protect BIOS Environment                        Not set     Disabled
< Measured Boot                                   Not set     Disabled
< Verified Boot                                   Not set     Disabled
---
> Protect BIOS Environment                        Not set     Enabled
> Measured Boot                                   Not set     Enabled
> Verified Boot                                   Not set     Enabled
46c46
< SPIRAL CPU                                      Not set     Disabled
---
> SPIRAL CPU                                      Not set     Enabled
64,65c64,65
<       0638EFCBB322E655C912D984070450BD22151A6BA96E53E8
<       6DDBAD48273EEE2BAB15DBE819696D989ABDC5EB9EB68A11
---
>       63487062A7069268556B81A6B350972CAEB63FD15101EBF3
>       A058BB20DE086B5634BCB5B1B65637C65E4BFAC76C0AE709
```

```
user@OST2-VM:~$ sha384sum  ~/training_materials/src/slimbootloader/Temp/kmsigpubkey.bin
```

Breaking Boot Guard verification.

Lets modify our firmware to check if Boot Guard will notice changes. We will modify the platform data which lives near the reset vector.

1. Lets modify `sbl_ifwi_adln.bin` platform to make sure we don't damage firmware. The platform data lives at fixed address of `0×fffffff4` (4 bytes after reset vector, 12 bytes from the end of flash).

   ```
   bash-5.2# hexdump -C sbl_ifwi_adln.bin |tail
   ```

   Output:

   ```
   00ffffc0  80 52 ff ff 00 00 00 00  00 00 00 00 00 00 00 00  |.R..............|
   00ffffd0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
   00ffffe0  00 00 00 00 00 00 00 00  00 00 00 00 56 54 46 00  |............VTF.|
   00fffff0  90 90 eb b9 0c ff ff aa  a4 50 ff ff 00 50 ff ff  |.........P...P..|
   01000000
   ```

   We will modify the 0xff byte at offset 0x00fffff6, which is the reserved field of the platform data and should not be used anywhere (but is still part of the IBB verified by Boot Guard).

1. Before modification

```
bash-5.2# dd if=sbl_ifwi_adln.bin bs=1 skip=$((0×00fffff4)) count=4 2>/dev/null | xxd -e
```

Output

```
00000000: aafff0c                          ....
```

2. We will replace `aafff0c` with `aa00ff0c`.

```
bash-5.2# cp sbl_ifwi_adln.bin{,.backup}
bash-5.2# echo -ne '\x00' | dd of=sbl_ifwi_adln.bin bs=1 seek=$((0×00fffff6)) conv=notrunc
```

3. After modification

```
bash-5.2# dd if=sbl_ifwi_adln.bin bs=1 skip=$((0×00fffff4)) count=4 2>/dev/null | xxd -e
```

Output

```
00000000: aa00ff0c                          ....
```

4. We can also compare differences between our binaries

```
bash-5.2# diff -u1 <(xxd sbl_ifwi_adln.bin.backup) <(xxd sbl_ifwi_adln.bin)
```

Output:

```
--- /dev/fd/63  2025-08-11 20:18:15.853717824 +0200
+++ /dev/fd/62  2025-08-11 20:18:15.854717810 +0200
@@ -1048575,2 +1048575,2 @@
 00fffe0: 0000 0000 0000 0000 0000 0000 5654 4600  ............VTF.
-00fffff0: 9090 ebb9 0cff ffaa a450 ffff 0050 ffff  .........P...P..
+00fffff0: 9090 ebb9 0cff 00aa a450 ffff 0050 ffff  .........P...P..
```

1. Let's flash our modified binary (remember to switch to vendor firmware first, then boot DTS, switch jumper again and flash):

```
bash-5.2# flashrom -p internal -w sbl_ifwi_adln.bin --ifd -i bios -N
```

2. Shutdown platform and remove power for few seconds
3. Enable capturing the output from serial port: `minicom -D /dev/ttyUSB0 -C /tmp/sbl.log`
4. Power platform again
5. Check Boot Guard state in the `/tmp/sbl.log`

```
user@OST2-VM:~$ cat /tmp/sbl.log | grep "Boot Guard"
```

Output:

```
[Boot Guard] AcmStatus : 0×C0008000
[Boot Guard] BootStatus: 0×98400000
[Boot Guard] Boot Guard is Enabled Successfully.
[Boot Guard] Acm Info: 0×730000006F
[Boot Guard] Verified Boot Status: Enabled
[Boot Guard] Measured Boot Status: Enabled
```

The platform booted as if nothing happened. Why? Boot Guard automatically switched to the redundant firmware part, activating the top swap mechanism. Remember the IbbFlags in `bpmgen2.params` file setting the top swap support? Boot partition swap can be detected by grepping for `BOOT: BP`:

```
user@OST2-VM:~$ cat /tmp/sbl.log | grep "BOOT: BP"
BOOT: BP1
```

For normal boots without Boot Guard enabled or boots with Boot Guard successful verification, the Slim Bootlaoder loads firmware from BP0 (boot partition 0), but now that the BP0 failed verification, Boot Guard switched to the redundant part, BP1 (boot partition 1).

Corrupting both top swap regions to cause Boot Guard failure.

Let's take the modified `sbl_ifwi_adln.bin` from previosu exercies and locate second top swap.

1. Verify where the `TOP SWAP A` starts. Use `FlashMap.txt`:

```
cat Outputs/odroid_h4/FlashMap.txt
...
  +-------------------------------------------------------------------------+
  |                              TOP SWAP A                                 |
  +-------------------------------------------------------------------------+
  |   SG1A   |  0x9e5000(0xFFFE5000)  |  0x01b000  |  Uncompressed, TS_A   |
  |   ACM0   |  0x980000(0xFFF80000)  |  0x065000  |  Uncompressed, TS_A   |
  +-------------------------------------------------------------------------+
  |                              TOP SWAP B                                 |
  +-------------------------------------------------------------------------+
  |   SG1A   |  0x965000(0xFFF65000)  |  0x01b000  |  Uncompressed, TS_B   |
  |   ACM0   |  0x900000(0xFFF00000)  |  0x065000  |  Uncompressed, TS_B   |
  +-------------------------------------------------------------------------+
...
```

2. Modify the platform data the same way as in previous exercies usign offet of TOP SWAP B platform data:

```
bash-5.2# echo -ne '\x00' | dd of=sbl_ifwi_adln.bin bs=1 seek=$((0x00f7fff4)) conv=notrunc
```

Result:

```
user@OST2-VM:~$ cat /tmp/sbl.log | grep -E "Boot Guard|: BP"
[Boot Guard] AcmStatus : 0×C0039910
[Boot Guard] BootStatus: 0×10410000
[Boot Guard] Boot Guard Failed or is Disabled!
[Boot Guard] Acm Info: 0×730000006E
[Boot Guard] Verified Boot Status: Enabled
[Boot Guard] Measured Boot Status: Enabled
Processor supports Boot Guard.
Boot Guard ACM Status = C0039910
Boot Guard Boot Status = 1041000000000000
NEM is not initiated by Boot Guard ACM
Boot Guard Support status: 1
BOOT: BP1
```

Note

If you flash ME region from unprovisioned `ifwi-debug.bin`, Boot Guard will become disabled, the image will still boot normally (even with corrupted IBB and properly signed manifests present), because we modified only unused reserved field of the platform data.

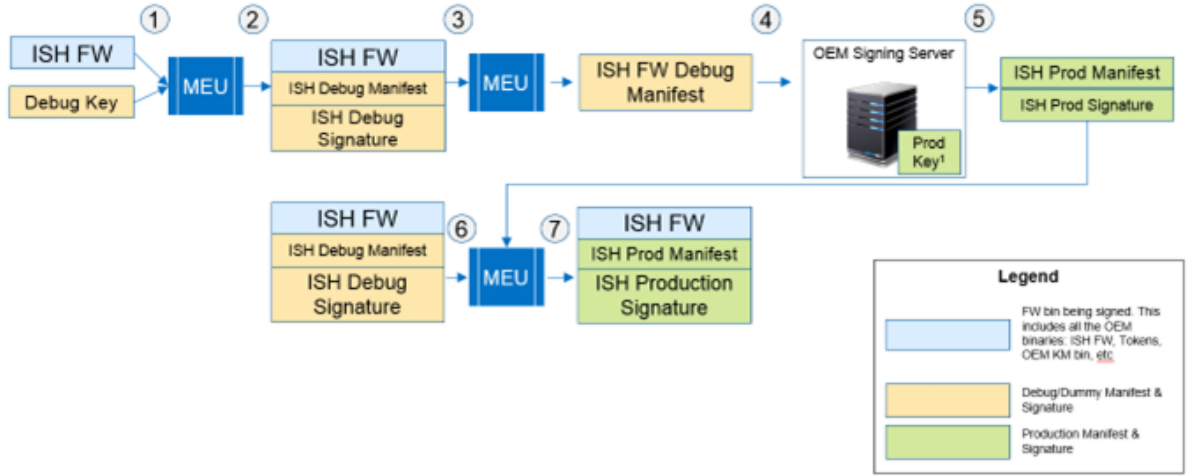
Provisioned and failed

```
[Boot Guard] AcmStatus : 0×C0039910
[Boot Guard] BootStatus: 0×10410000
[Boot Guard] Boot Guard Failed or is Disabled!
[Boot Guard] Acm Info: 0×730000006E
[Boot Guard] Verified Boot Status: Enabled
[Boot Guard] Measured Boot Status: Enabled
```

Provisioned and working

```
[Boot Guard] AcmStatus : 0×C0008000
[Boot Guard] BootStatus: 0×98400000
[Boot Guard] Boot Guard is Enabled Successfully.
[Boot Guard] Acm Info: 0×730000006F
[Boot Guard] Verified Boot Status: Enabled
[Boot Guard] Measured Boot Status: Enabled
```

Production

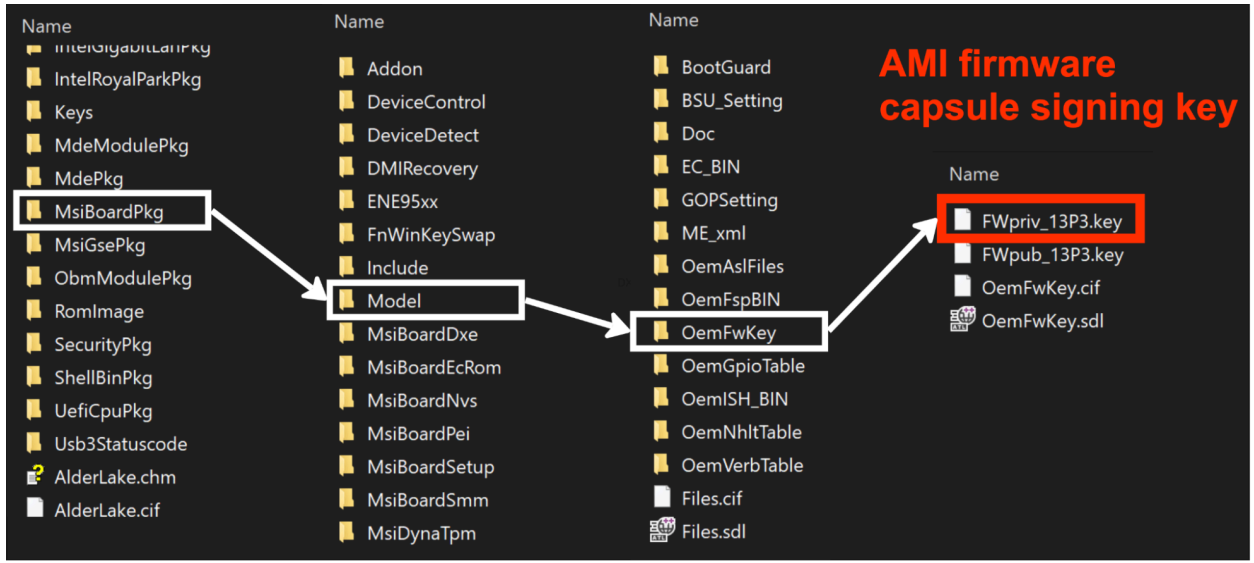


Signing on an OEM Secure Server

1. Create an IP FW binary signed with a placeholder key. This placeholder key is a local temporary key in the non-secure server.
2. Export the manifest section of the binary.
3. Parse the IP manifest header and remove the crypto block from the manifest binary. This is done based on fixed offsets or parsing of the binary with a hex editor. No other fields in the manifest should be changed.
4. Send the manifest binary (without crypto block) to the secure server.
5. Generate the production of private and public key pairs.
6. Calculate the signature for the manifest binary (without the crypto blocks).
7. A tool, such as OpenSSL, can be used to extract the public key modulus and exponent values.
8. Add a new crypto block to the manifest binary, filling in the appropriate data.
9. Send the manifest binary with the updated crypto block to a non-secure server.

Protection of Private Keys

> **Note**
>
> It should be noted, however, that if the exposed key is the private key used to sign the OEM Key Manifest (OEM KM), then unless the HW provides a method to disable the exposed key and enable a second key, platforms from that OEM that use the key that was exposed are potentially vulnerable to firmware modification and are not able to reestablish the CoT. It is therefore highly recommended that industry best practices be applied to ensure, at minimum, the OEM KM is properly protected from unauthorized access, such as storing private keys in a Hardware Security Module (HSM) and implementing a signing server as explained in the Secure Server Signing section.



> **Quote**
>
> Recent ransomware attacks on some OEMs have involved BIOS source code and build environments being taken that also included one or more private keys for those platforms. To reduce the chances of such exposure, private keys should never be stored with the BIOS source code.

> Quote
>
> You can point to a different key file using the PREGENERATED_SIGNING_KEY_SLIMBOOT_KEY_SHA256 variable in targets/kas/opt-fusa.yml. Use the same key file instead of ./SblPlatform/SblKeys/OS1_TestKey_Pub_RSA2048_priv.pem in Intel® Elkhart Lake Slim Bootloader (SBL) (Intel RDC#616714) to build SBL and IFWI binary file.

https://eci.intel.com/docs/2.6/components/fusasdk.html

Possible improvements

- SBL provisioning scripts are overly complicated and do more things than necessary, especially aroudn XML patching. This could be simplified, like we have done it for Kontron TGL board.
- Some actions need only to be done once:
  - MEU config and manifest XML
  - OEM Key MAnfiest generatio nwith MEU (if OEM Root Key does not change)
  - BpmGen2 parameters file patching
  - Key Manifest generation (if OEM Root Key and Boot Policy keys remain the same)
- Some actions should be better done offline in air gap environemt or through HSMs, mainly the signing process. MEU and BpmGen2 allow external signing and OpenSSL should be able to udnerstand URI with HSMs.
  - Having private keys lingering as regular file in the repo is a bad security practice
- Ditching the whole set of Intel tools in favor of own implementations (to be discussed in following presentation abotu ME configuration and fuses)

Q&A