# Privacy Preserving Image Feature Extraction in TEE



## 00、Introduction

This document provides guidance on configuring Intel SGX to run a **Python-based feature extraction algorithm**. We leverage the libOS Occlum to execute an unmodified Python program. We use a server with 64-bit Ubuntu 20.04 LTS operating system, Intel(R) Core(TM) i7-9750H CPU with 12 physical cores. We provides step-to-step guideline as follows.

## 01、Install Intel SGX

Intel(R) Software Guard Extensions (Intel(R) SGX) is an Intel technology for application developers seeking to protect secret code and data from disclosure or modification. Refer to https://github.com/intel/linux-sgx for more details.

**Prerequisites**

1. Check whether your machine support Intel SGX (you can also run the attached project: check_sgx.c);

```
aptx4869-tee@WP:~/attachments$ gcc -o check_sgx check_sgx.c
aptx4869-tee@WP:~/attachments$ ./check_sgx
eax: 906ea ebx: 6100800 ecx: 7ffafbbf edx: bfebfbff
stepping 10
model 14
family 6
processor type 0
extended model 9
extended family 0
smx: 0

Extended feature bits (EAX=07H, ECX=0H)
eax: 0 ebx: 29c67af ecx: 40000000 edx: bc002e00
sgx available: 1
sgx launch control: 1

CPUID Leaf 12H, Sub-Leaf 0 of Intel SGX Capabilities (EAX=12H,ECX=0)
eax: 1 ebx: 0 ecx: 0 edx: 241f
sgx 1 supported: 1
sgx 2 supported: 0
MaxEnclaveSize_Not64: 1f
MaxEnclaveSize_64: 24
```

   PS: As shown in the figure, sgx 1 supported: 1 indicates support, while 0 indicates lack of support.

2. OS: Ubuntu 18.04 LTS, Ubuntu 20.04 LTS, or Ubuntu 22.04 LTS;

**Installation**

1. find the attachments install_deps.sh and install_sgx_driver.sh;
2. install the dependencies and Intel SGX SDK;

```
sudo ./install_deps.sh
```

PS: The command `source /opt/intel/sgxsdk/environment` represents the SGX source path. To automate its execution every time the terminal starts, consider adding this command to the ~/.bashrc file.

3. install the Intel SGX Driver

```
sudo ./install_sgx_driver.sh
```

4. startup the Intel SGX services

```
sudo /opt/intel/sgx-aesm-service/startup.sh
```

If any issues arise during the installation process, you can refer to https://github.com/intel/linux-sgx for guidance.

5. test the installation

- compile the code (refer to the attached project: simple_exp)

```
make  # hardware mode
make SGX_MODE=SIM  # simulation mode
```

```
CXX  <=  Enclave/Enclave.cpp
LINK =>  enclave.so
There is no enclave test key<Enclave_private_test.pem>.
The project will generate a key<Enclave_private_test.pem> for test.
Generating RSA private key, 3072 bit long modulus (2 primes)
.....................++++
.....................................................................................++++*.....................
.....................................................................................++++*.....................
.....................................................................................++++
e is 3 (0x03)
<EnclaveConfiguration>
    <ProdID>0</ProdID>
    <ISVSVN>0</ISVSVN>
    <StackMaxSize>0x40000</StackMaxSize>
    <HeapMaxSize>0x100000</HeapMaxSize>
    <TCSNum>10</TCSNum>
    <TCSPolicy>1</TCSPolicy>
    <!-- Recommend changing 'DisableDebug' to 1 to make the enclave undebuggable for enclave release -->
    <DisableDebug>0</DisableDebug>
    <MiscSelect>0</MiscSelect>
    <MiscMask>0xFFFFFFFF</MiscMask>
</EnclaveConfiguration>
tcs_num 10, tcs_max_num 10, tcs_min_pool 1
INFO: Enclave configuration 'MiscSelect' and 'MiscSelectMask' will prevent enclave from using dynamic features. To use the
dynamic features on SGX2 platform, suggest to set MiscMask[0]=0 and MiscSelect[0]=1.
The required memory is 3969024B.
The required memory is 0x3c9000, 3876 KB.
handle_compatible_metadata: Overwrite with metadata version 0x100000004
Succeed.
SIGN =>  enclave.signed.so
The project has been built in debug hardware mode.
```

- run the binary

```
./app
```

```
aptx4869-tee@WP:~/attachments/simple_exp$ ./app
a=2,b=3
c=5
Info: exp successfully returned.
```

# 02、Install Occlum

1. Quick Start with Docker

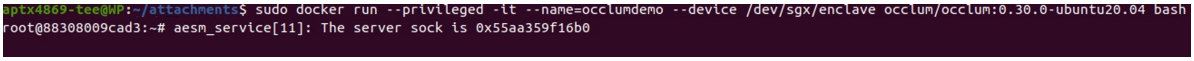
Many Occlum versions are available on Docker Hub, making it convenient to pull images directly. Pull the image and start the container using following command.

```
docker pull occlum/occlum:0.30.0-ubuntu20.04
docker run --privileged -it --name=occlumdemo --device /dev/sgx/enclave
occlum/occlum:0.30.0-ubuntu20.04 bash
```

**Parameters**

- --privileged: increase privileges.
- --name: specify the container name.
- --device: specify mapping the host's /dev/sgx/enclave device to the container's /dev/sgx/enclave device for running SGX-dependent applications inside the container.

Success as shown in the figure.




If it reports following errors, it means that the BIOS has not enabled SGX. You should check whether your machine supports Intel SGX. If yes, you can enable Intel SGX via the "software enable" procedure quickly. Please refer to https://github.com/intel/sgx-software-enable for more details.

```
root@6d532c1d688a:~# aesm_service[11]: [get_driver_type edmm_utility.cpp:116] Failed to open Intel SGX device.

aesm_service[11]: [get_driver_type edmm_utility.cpp:116] Failed to open Intel SGX device.

aesm_service[11]: [load_qe ../qe_logic.cpp:719] Error, call sgx_create_enclave QE fail [load_qe], SGXError:2006.

aesm_service[11]: Failed to load QE3: 0x2006
aesm service[11]: The server sock is 0x55bd1fcc2090
```

2. Install with Official Specification

```
echo 'deb [arch=amd64] https://occlum.io/occlum-package-repos/debian focal main'
| tee /etc/apt/sources.list.d/occlum.list
wget -qO - https://occlum.io/occlum-package-repos/debian/public.key | apt-key
add -
apt-get update
apt-get install -y occlum
echo "source /etc/profile" >> $HOME/.bashrc
```

Refer to https://github.com/occlum/occlum/blob/master/docs/install_occlum_packages.md for more details.

3. Test the installation

- **step-0 :** download the file hello_word.c

```
wget
https://raw.githubusercontent.com/occlum/occlum/master/demos/hello_c/hello_world
.c
```

- **step-1 :** compile the user program using the Occlum toolchain (e.g., occlum-gcc).

```
occlum-gcc -o hello_world hello_world.c
```

This will result in an executable file. You can test if it compiled successfully by running ./hello_world, which should output "Hello World."

```
aptx4869-tee@WP:~/Occlum/Occlum_exp$ occlum-gcc -o hello_world hello_world.c
aptx4869-tee@WP:~/Occlum/Occlum_exp$ ./hello_world
Hello World!
```

- **step-2 :** initialize an Occlum instance

```
mkdir occlum-instance && cd occlum-instance && occlum init
occlum new occlum-instance # alternatively
```

```
aptx4869-tee@WP:~/Occlum$ mkdir occlum-instance && cd occlum-instance && occlum init
/home/aptx4869/Occlum/occlum-instance initialized as an Occlum instance
```

- **step-3 :** copy the compiled file (executable hello_world) into the occlum-instance/image/bin/ folder

```
cp ../hello_world image/bin
```

- **step-4 :** build the Occlum instance

```
occlum build  # hardware mode
occlum build --sgx-mode SIM  # simulation mode
```

```
aptx4869-tee@WP:~/Occlum/occlum-instance$ occlum build
Enclave sign-tool: /opt/occlum/sgxsdk-tools/bin/x64/sgx_sign
Enclave sign-key: /opt/occlum/etc/template/Enclave.pem
SGX mode: HW
Enable EDMM: No
rm -rf /home/aptx4869/Occlum/occlum-instance/build
Building the initfs...
[+] Home dir is /home/aptx4869
[+] Open token file success!
[+] Token file valid!
[+] Init Enclave Successful 150839251435522!
Generate the SEFS image successfully
Building new image...
[+] Home dir is /home/aptx4869
[+] Open token file success!
[+] Token file valid!
[+] Init Enclave Successful 150860726272002!
Generate the SEFS image successfully
Building libOS...
Signing the enclave...
<EnclaveConfiguration>
    <ProdID>0</ProdID>
    <ISVSVN>0</ISVSVN>
    <StackMaxSize>1048576</StackMaxSize>
    <StackMinSize>1048576</StackMinSize>
    <HeapInitSize>33554432</HeapInitSize>
    <HeapMaxSize>33554432</HeapMaxSize>
    <HeapMinSize>33554432</HeapMinSize>
    <TCSNum>32</TCSNum>
    <TCSMaxNum>32</TCSMaxNum>
    <TCSMinPool>32</TCSMinPool>
    <TCSPolicy>1</TCSPolicy>
    <DisableDebug>0</DisableDebug>
    <MiscSelect>0</MiscSelect>
    <MiscMask>0</MiscMask>
    <ReservedMemMaxSize>314572800</ReservedMemMaxSize>
    <ReservedMemMinSize>314572800</ReservedMemMinSize>
    <ReservedMemInitSize>314572800</ReservedMemInitSize>
    <ReservedMemExecutable>1</ReservedMemExecutable>
    <EnableKSS>0</EnableKSS>
    <ISVEXTPRODID_H>0</ISVEXTPRODID_H>
    <ISVEXTPRODID_L>0</ISVEXTPRODID_L>
    <ISVFAMILYID_H>0</ISVFAMILYID_H>
    <ISVFAMILYID_L>0</ISVFAMILYID_L>
    <PKRU>0</PKRU>
    <AMX>0</AMX>
</EnclaveConfiguration>
```

- **step-5 :** run the Occlum instance

```
occlum run /bin/hello_world
```

```
tcs_num 32, tcs_max_num 32, tcs_min_pool 32
INFO: SGX1 only enclave, which will run on all platforms.
The required memory is 387334144B.
The required memory is 0x17164000, 378256 KB.
handle_compatible_metadata: Overwrite with metadata version 0x100000004
Succeed.
Built the Occlum image and enclave successfully
aptx4869-tee@WP:~/Occlum/occlum-instance$ occlum run /bin/hello_world
Hello World!
```

## 03、Run Python in Occlum

In the occlum/demos directory, there are many examples provided by the official documentation, including but not limited to C++, Java, Python, etc. Here, we take Python as an example.

1. install miniconda with the following script

```
#!/bin/bash
set -e
script_dir="$( cd "$( dirname "${BASH_SOURCE[0]}"  )" >/dev/null 2>&1 && pwd )"


# Install python and dependencies to specified position
[ -f Miniconda3-latest-Linux-x86_64.sh ] || wget
https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
[ -d miniconda ] || bash ./Miniconda3-latest-Linux-x86_64.sh -b -p
/root/miniconda

echo "source /root/miniconda/etc/profile.d/conda.sh" >> ~/.bashrc
source ~/.bashrc
```

If the download speed is extremely slow, consider trying a different mirror using the command conda config --add channels.

```
conda config --add channels
http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
conda config --add channels
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
conda config --add channels
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/msys2/
conda config --add channels
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch/
conda config --add channels fastai
conda config --set show_channel_urls yes
```

2. occlum init、build、run

Using the attached project, i.e., np, as an example.

**directory struct in an occlum project**

- np.yaml：YAML file specifying the directory structure to be copied into Occlum images.
- init.sh：script for creating a Python environment using conda (run only once if dependencies don't change).
- build.sh：script for initializing the Occlum instance and generating Occlum images.
- run.sh：script for initializing the Occlum instance and generating Occlum images.
- code：source code directory (modify if there are relative paths; adjust based on the Occlum images' paths).

```
root-docker@88308009cad3:~/occlum/np# tree
.
|-- build.sh
|-- code
|   `-- test_np.py
|-- conda.sh
|-- init.sh
|-- np.yaml
`-- run.sh

1 directory, 6 files
```

- **step-1 :** modify the .yaml file (shown in the following red box), that indicates copying the `../code` directory to `/bin` (relative to path `occlum_instance/images`).

```
includes:
  - base.yaml
targets:
  - target: /bin
    createlinks:
      - src: /opt/python-occlum/bin/python3
        linkname: python3
  # python packages
  - target: /opt
    copy:
      - dirs:
          - ../python-occlum
  # below are python code and data,you also can cp this demo.py to image/bin/ by hand before occlum build
  - target: /bin
    copy:
      - dirs:
          - ../code
~
~
```

- **step-2 :** run the script init.sh

```
./init.sh
```

```
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate /root/occlum/np/python-occlum
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

- **step-3 :** run the script build.sh

```
./build.sh
```

```
Building new image...
[+] Home dir is /root
[+] Open token file success!
[+] Token file valid!
[+] Init Enclave Successful 1743756722178!
Generate the SEFS image successfully
Building libOS...
Signing the enclave...
<EnclaveConfiguration>
    <ProdID>0</ProdID>
    <ISVSVN>0</ISVSVN>
    <StackMaxSize>1048576</StackMaxSize>
    <StackMinSize>1048576</StackMinSize>
    <HeapInitSize>2147483648</HeapInitSize>
    <HeapMaxSize>2147483648</HeapMaxSize>
    <HeapMinSize>2147483648</HeapMinSize>
    <TCSNum>32</TCSNum>
    <TCSMaxNum>32</TCSMaxNum>
    <TCSMinPool>32</TCSMinPool>
    <TCSPolicy>1</TCSPolicy>
    <DisableDebug>0</DisableDebug>
    <MiscSelect>0</MiscSelect>
    <MiscMask>0</MiscMask>
    <ReservedMemMaxSize>314572800</ReservedMemMaxSize>
    <ReservedMemMinSize>314572800</ReservedMemMinSize>
    <ReservedMemInitSize>314572800</ReservedMemInitSize>
    <ReservedMemExecutable>1</ReservedMemExecutable>
    <EnableKSS>0</EnableKSS>
    <ISVEXTPRODID_H>0</ISVEXTPRODID_H>
    <ISVEXTPRODID_L>0</ISVEXTPRODID_L>
    <ISVFAMILYID_H>0</ISVFAMILYID_H>
    <ISVFAMILYID_L>0</ISVFAMILYID_L>
    <PKRU>0</PKRU>
    <AMX>0</AMX>
</EnclaveConfiguration>
tcs_num 32, tcs_max_num 32, tcs_min_pool 32
INFO: SGX1 only enclave, which will run on all platforms.
The required memory is 2501210112B.
The required memory is 0x95157000, 2442588 KB.
handle_compatible_metadata: Overwrite with metadata version 0x100000004
Succeed.
Built the Occlum image and enclave successfully
```

- **step-4 :** run the script run.sh

```
./run.sh
```

```
root-docker@88308009cad3:~/occlum/np# ./run.sh
/opt/python-occlum/lib/python3.7/site-packages/numpy/__init__.py
[[1 2 3]
 [4 5 6]]
sum: 21
mean: 3.5
max: 6
min: 1
[[1 4]
 [2 5]
 [3 6]]
```

## 04、Run the Image Feature Extraction Algorithm

Using the attached project, i.e., sgx_demo. Note that sgx_demo is an simple demo for image feature extraction, which uses the library openCV. The steps are similar to the above.

- **step-1 :** modify the source code to limit thread count and change the relative path of the image.

```
# -*- coding: utf-8 -*-
# @Author  : qiaohezhe
# @github : https://github.com/fengduqianhe
# @Date    :  10/22/2023
# version~Z Python 3.7.8
# @File : sgx_demo.py
# @Software: PyCharm
from __init__ import *
import os
os.environ['OMP_NUM_THREADS'] = '1'
from feature_extract_set import batch_extractor, extract_features


def run():
    images_path = 'data/card_dataset/'
    # batch_extractor(images_path)
    images = '/bin/opencv_exp/data/card_dataset/train.jpg'
    result = {}
    print('Extracting features from image %s' % images)
    name = images.split('/')[-1].lower()
    result[name] = extract_features(images)
    print(result[name])

run()
~
```

- **step-2 :** write the scripts for openCV

```
includes:
  - base.yaml
targets:
  - target: /bin
    createlinks:
      - src: /opt/python-occlum/bin/python3
        linkname: python3
  # python packages
  - target: /opt
    copy:
      - dirs:
          - ../python-occlum
  # below are python code and data,you also can cp this demo.py to image/bin/ by
hand before occlum build
  - target: /bin
    copy:
      - dirs:
          - ../opencv_exp
```

- **step-3 :** update the init.sh (refer to the attached `project sgx_demo/init.sh`) and run the script

```
#!/bin/bash
set -e
script_dir="$( cd "$( dirname "${BASH_SOURCE[0]}"  )" >/dev/null 2>&1 && pwd )"

#  Install python and dependencies to specified position
/root/miniconda/bin/conda create --prefix $script_dir/python-occlum -y
python=3.7.8 matplotlib imageio numpy opencv-python-headless scipy
```

```
./init.sh
```

- **step-4 :** update the build.sh (refer to the attached `project sgx_demo/build.sh`) and run the script

```
./build.sh
```

- **step-4 :** update the run.sh (refer to the attached project `sgx_demo/run.sh`) and run the script

```
./run.sh
```

```
INFO: SGX1 only enclave, which will run on all platforms.
The required memory is 3982508032B.
The required memory is 0xed604000, 3889168 KB.
handle_compatible_metadata: Overwrite with metadata version 0x100000004
Succeed.
Built the Occlum image and enclave successfully
root-docker@88308009cad3:~/occlum/sgx_demo# ./run.sh
Extracting features from image /bin/opencv_exp/data/card_dataset/train.jpg
run is ok!
[-0.0102374  -0.01206957  0.0789395  ...  0.00968362  0.03977215
  0.01300801]
```