NVIDIA DOCA with OpenSSL
This guide provides instructions on using DOCA SHA for OpenSSL implementations.

Introduction

The `doca_sha_offload_engine` is an OpenSSL dynamic engine with the ability of offloading SHA calculation. It can offload the OpenSSL one-shot SHA-1, SHA-256, and SHA-512. It supports synchronous mode and asynchronous mode by leveraging the OpenSSL `async_jobs` library. For more information on the `async_jobs` library, please refer to official OpenSSL documentation.

This engine is based on the `doca_sha` library and the OpenSSL dynamic engine interface API. For more information on the OpenSSL dynamic engine, please refer to official OpenSSL documentation.

This engine can be called by an OpenSSL application through the OpenSSL high-level algorithm call interface, `EVP_Digest`. For more information on the `EVP_Digest`, please refer to official OpenSSL documentation.

Prerequisites

- Hardware-based `doca_sha` engine which can be verified by calling `doca_sha_get_hardware_supported()`
- Installed OpenSSL version ≥ 1.1.1

Architecture

The following diagram shows the software hierarchy of `doca_sha_offload_engine` and its location in the whole DOCA repository.

From the perspective of OpenSSL, this engine is an instantiation of the OpenSSL dynamic engine interface API by leveraging the `doca_sha` library.
Capabilities and Limitations

- Only one-shot OpenSSL SHA is supported
- The maximum message length ≤ 2GB, the same as doca_sha library

OpenSSL Command Line Verification

Verify that the engine can be loaded:

```
$ openssl engine dynamic -pre NO_VCHECK:1 -pre
SO_PATH:${DOCA_DIR}/infrastructure/doca_sha_offload_engine/libdoca_sha_offload_engine.so -pre
LOAD -vvv -t -c
(dynamic) Dynamic engine loading support
[Success]:
SO_PATH:${DOCA_DIR}/infrastructure/doca_sha_offload_engine/libdoca_sha_offload_engine.so
[Success]: LOAD
Loaded: (doca_sha_offload_engine) Openssl SHA offloading engine based on doca_sha
[SHA1, SHA256, SHA512]
```
NVIDIA DOCA with OpenSSL

For SHA-1:

```bash
$ echo "hello world" | openssl dgst -sha1 -engine
{DOCA_DIR}/infrastructure/doca_sha_offload_engine/libdoca_sha_offload_engine.so -engine_impl
```

For SHA-256:

```bash
$ echo "hello world" | openssl dgst -sha256 -engine
{DOCA_DIR}/infrastructure/doca_sha_offload_engine/libdoca_sha_offload_engine.so -engine_impl
```

For SHA-512:

```bash
$ echo "hello world" | openssl dgst -sha512 -engine
{DOCA_DIR}/infrastructure/doca_sha_offload_engine/libdoca_sha_offload_engine.so -engine_impl
```

**OpenSSL Throughput Test**

openssl-speed is the OpenSSL throughput benchmark tool. For more information, consult official OpenSSL documentation. doca_sha_offload_engine throughput can also be measured using openssl-speed.

- SHA-1, each job 10000 bytes, using engine:

  ```bash
  $ openssl speed -evp sha1 -bytes 10000 -elapsed --engine
  {DOCA_DIR}/infrastructure/doca_sha_offload_engine/libdoca_sha_offload_engine.so
  ```

- SHA-256, each job 10000 bytes, using engine, async_jobs=256:

  ```bash
  $ openssl speed -evp sha256 -bytes 10000 -elapsed --engine
  {DOCA_DIR}/infrastructure/doca_sha_offload_engine/libdoca_sha_offload_engine.so
  ```
NVIDIA DOCA with OpenSSL

Using DOCA SHA Offload Engine in OpenSSL Application

More information on the dynamic engine usage can be found in the official OpenSSL documentation.

1. To load the `doca_sha_offload_engine` (optionally, set engine PCIe address):

   ```c
   ENGINE *e;
   const char *doca_engine_path = 
   "${DOCA_DIR}/infrastructure/docaShaOffloadEngine/libdocaShaOffloadEngine.so";
   const char *default_doca_pci_addr = "03:00.0";
   ENGINE_load_dynamic();
   e = ENGINE_by_id(doca_engine_path);
   ENGINE_ctrl_cmd_string(e, "set_pci_addr", doca_engine_pci_addr, 0);
   ENGINE_init(e);
   ENGINE_set_default_digests(e);
   ```

2. To perform SHA calculation by calling the OpenSSL high-level function `EVP_XXX`:

   ```c
   const EVP_MD *evp_md = EVP_sha1();
   EVP_MD_CTX *mdctx = EVP_MD_CTX_create();
   EVP_DigestInit_ex(mdctx, evp_md, e);
   EVP_DigestUpdate(mdctx, msg, msg_len);
   EVP_DigestFinal_ex(mdctx, digest, digest_len);
   ```

- SHA-512, each job 10000 bytes, using engine, async_jobs=256, threads=8:
3. To unload the engine:

```c
EVP_MD_CTX_destroy(mdctx);
ENGINE_unregister_digests(e);
ENGINE_finish(e);
ENGINE_free(e);
```