DOCA AES-GCM

Table of contents

Introduction

Prerequisites

Changes From Previous Releases

- Changes in 2.9.0

Environment

Architecture

- Objects
- Device and Representor
- Memory Buffers

Configuration Phase

- Configurations
- Mandatory Configurations
- Device Support
- Buffer Support

Execution Phase

- Tasks
- Encrypt Task
- Decrypt Task
- Events

State Machine

- Idle
- Starting

Running

Stopping

Alternative Datapath Options

DOCA AES-GCM Samples

Running the Samples

Samples

AES-GCM Encrypt

AES-GCM Decrypt

This guide provides instructions on building and developing applications that require data encryption and decryption using the AES-GCM algorithm.

Introduction

> **Note**
>
> The DOCA AES-GCM library is supported at alpha level and on NVIDIA® BlueField®-3 devices or higher.

The library provides an API for executing AES-GCM operations on DOCA buffers, where the buffers reside in either local memory (i.e., within the same host) or host memory accessible by the DPU (remote memory). Using DOCA AES-GCM, complex encrypt/decrypt operations can be easily executed in an optimized, hardware-accelerated manner.

This document is intended for software developers wishing to accelerate their application's encrypt/decrypt operations.

Prerequisites

This library follows the architecture of a DOCA Core context, it is recommended to read the following sections before:

- DOCA Core Execution Model
- DOCA Core Device
- DOCA Core Memory Subsystem

Changes From Previous Releases

Changes in 2.9.0

N/A

Environment

DOCA AES-GCM-based applications can run either on the host machine or DPU target.

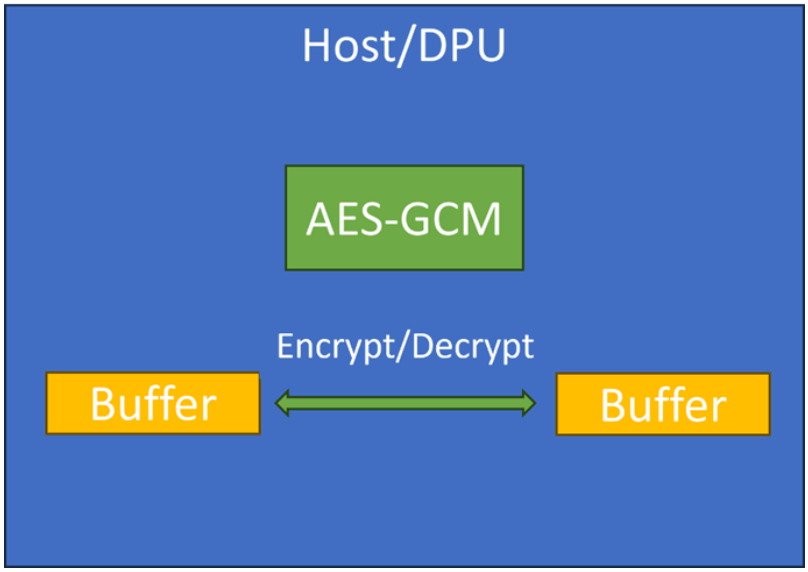
Encrypting/decrypting from the host to DPU and vice versa can only be run when the DPU is configured in DPU mode.

Architecture

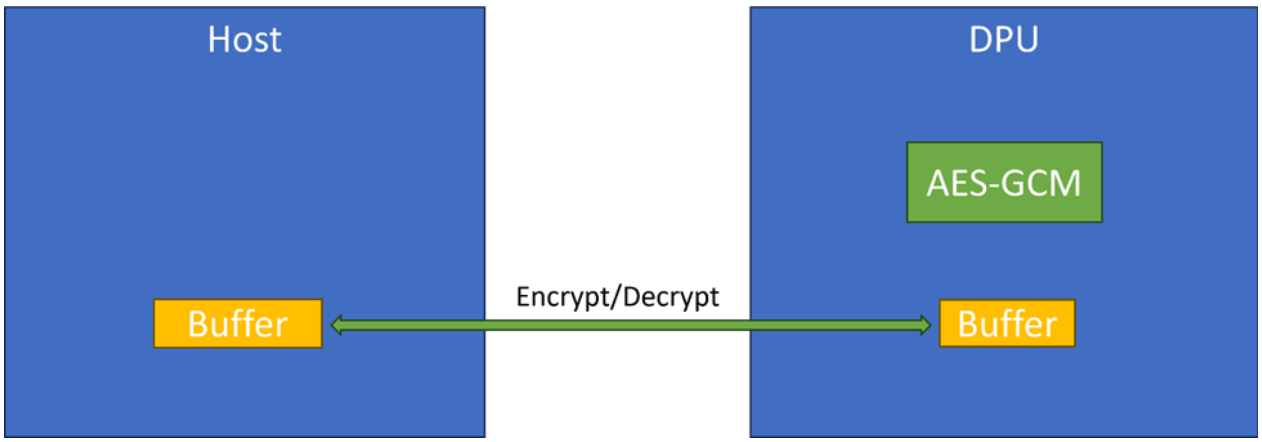
DOCA AES-GCM is a DOCA Core Context. This library leverages the DOCA Core architecture to expose asynchronous tasks/events that are offloaded to hardware.

AES-GCM can be used to encrypt/decrypt data as illustrated in the following diagrams:

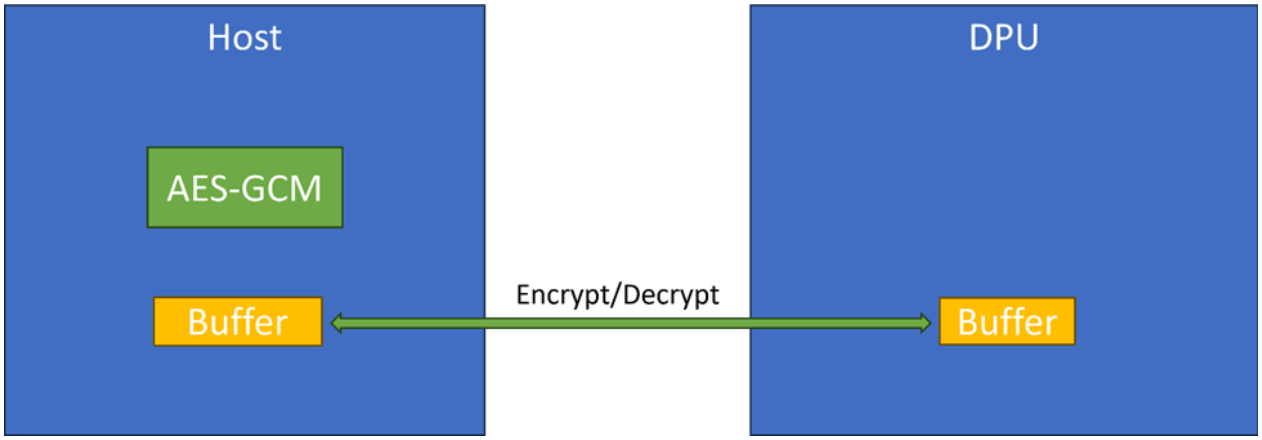
- Encrypt/decrypt from local memory to local memory:



- Using the DPU to copy memory between the host and the DPU:



- Using the host to copy memory between the host and the DPU:



Objects

Device and Representor

The library requires a DOCA device to operate. The device is used to access memory and perform the actual encrypt/decrypt. See DOCA Core Device Discovery.

For the same BlueField DPU, it does not matter which device is used (i.e., PF/VF/SF) as all these devices utilize the same hardware component. If there are multiple DPUs, then it is possible to create a AES-GCM instance per DPU, providing each instance with a device from a different DPU.

To access memory that is not local (i.e., from the host to DPU or vice versa), the DPU side of the application must pick a device with an appropriate representor (see DOCA Core Device Representor Discovery). The device must stay valid as long as AES-GCM instance is not destroyed.

Memory Buffers

The encrypt/decrypt task requires two DOCA buffers containing the destination and the source.

Depending on the allocation pattern of the buffers, consider the DOCA Core Inventory Types table.

To find what kind of memory is supported, refer to the buffer support table.

Note

Buffers must not be modified or read during the encrypt/decrypt operation.

Configuration Phase

To start using the library users must go through a configuration phase as described in DOCA Core Context Configuration Phase.

This section describes how to configure and start the context to allow execution of tasks and retrieval of events.

Configurations

The context can be configured to match the application use case.

To find if a configuration is supported or its min/max value, refer to Device Support.

Mandatory Configurations

These configurations must be set by the application before attempting to start the context:

- At least one task/event type must be configured. See configuration of Tasks and/or Events.
- A device with appropriate support must be provided upon creation

Device Support

DOCA AES-GCM requires a device to operate. For picking a device, see DOCA Core Device Discovery.

As device capabilities may change in the future (see DOCA Core Device Support) it is recommended to select your device using the following method:

- `doca_aes_gcm_cap_task_encrypt_is_supported`
- `doca_aes_gcm_cap_task_decrypt_is_supported`

Some devices can allow different capabilities as follows:

- The maximum number of tasks
- The maximum buffer size
- The maximum supported number of elements in DOCA linked-list buffer
- The maximum initialization vector length
- Check if authentication tag of size 96-bit is supported
- Check if authentication tag of size 128-bit is supported
- Check if a given AES-GCM key type is supported

Buffer Support

Tasks support buffers with the following features:

Buffer Type	Source Buffer	Destination Buffer
Local mmap buffer	Yes	Yes
Mmap from PCIe export buffer	Yes	Yes
Mmap from RDMA export buffer	No	No
Linked list buffer	Yes	No

Execution Phase

This section describes execution on the CPU using DOCA Core Progress Engine.

Tasks

DOCA AES-GCM exposes asynchronous tasks that leverage DPU hardware according to the DOCA Core architecture.

Encrypt Task

The encrypt task allows data encryption using buffers as described in Buffer Support.

Task Configuration

Description	API to Set Configuration	API to Query Support
Enable the task	`doca_aes_gcm_task_encrypt_set_conf`	`doca_aes_gcm_cap_task_encrypt_is_supported`
Number of tasks	`doca_aes_gcm_task_encrypt_set_conf`	`doca_aes_gcm_cap_get_max_num_tasks`
Maximal buffer size	–	`doca_aes_gcm_cap_task_encrypt_get_max_buf_size`
Maximum buffer list size	–	`doca_aes_gcm_cap_task_encrypt_get_max_list_buf_num_elem`
Maximum initialization vector length	–	`doca_aes_gcm_cap_task_encrypt_get_max_iv_length`
Enable authentication tag size	–	`doca_aes_gcm_cap_task_encrypt_is_tag_96_supported` `doca_aes_gcm_cap_task_encrypt_is_tag_128_supported`
Enable key type	–	`doca_aes_gcm_cap_task_encrypt_is_key_type_supported`

Task Input

Common input as described in DOCA Core Task

Name	Description	Notes
source buffer	Buffer pointing to the memory to be encrypted	Only the data residing in the data segment is encrypted
destination buffer	Buffer pointing to where memory is encrypted to	The encrypted data is appended to the tail segment
key	Key to encrypt the data	Created by the function `doca_aes_gcm_key_create` Users should use the same key to encrypt and decrypt the data
initialization vector (IV)	Initialization vector to be used by the AES-GCM algorithm	Users should use the same IV to encrypt and decrypt the data
initialization vector length	Initialization vector length that must be supplied for the AES-GCM algorithm	Represented in bytes, 0B-12B values are supported
authentication tag size	Authentication tag size to be supplied for the AES-GCM algorithm. The tag is automatically calculated and appended to the result buffer.	Represented in bytes, only 12B and 16B values are supported
additional authenticated data size	Additional authenticated data size to be supplied for the AES-GCM algorithm. This data, which should be present at the beginning of the source buffer, is will not encrypted but is authenticated.	Represented in bytes

Task Output

Common output as described in DOCA Core Task.

Task Completion Success

After the task completes successfully, the following happens:

- The data from the source buffer is encrypted and written to the destination buffer
- The destination buffer data segment is extended to include the encrypted data

Task Completion Failure

If the task fails midway:

- The context may enter stopping state if a fatal error occurs
- The source and destination `doca_buf` objects are not modified
- The destination buffer contents may be modified

Task Limitations

- The operation is not atomic
- Once the task is submitted, the source and destination should not be read/written to
- Other limitations are described in DOCA Core Task

Decrypt Task

The decrypt task allows data decryption. Using buffers as described in Buffer Support.

Task Configuration

Description	API to Set Configuration	API to Query Support
Enable the task	`doca_aes_gcm_task_decrypt_set_conf`	`doca_aes_gcm_cap_task_decrypt_is_supported`

Description	API to Set Configuration	API to Query Support
Number of tasks	`doca_aes_gcm_task_decrypt_set_conf`	`doca_aes_gcm_cap_get_max_num_tasks`
Maximal buffer size	–	`doca_aes_gcm_cap_task_decrypt_get_max_buf_size`
Maximum buffer list size	–	`doca_aes_gcm_cap_task_decrypt_get_max_list_buf_num_elem`
Maximum initialization vector length	–	`doca_aes_gcm_cap_task_decrypt_get_max_iv_length`
Enable authentication tag size	–	`doca_aes_gcm_cap_task_decrypt_is_tag_96_supported` `doca_aes_gcm_cap_task_decrypt_is_tag_128_supported`
Enable key type	–	`doca_aes_gcm_cap_task_decrypt_is_key_type_supported`

Task Input

Common input as described in DOCA Core Task.

Name	Description	Notes
Source buffer	Buffer pointing to the memory to be decrypted	Only the data residing in the data segment is decrypted
Destination buffer	Buffer pointing to where memory is decrypted to	The decrypted data is appended to the tail segment extending the data segment
Key	Key to decrypt the data	Created by the function `doca_aes_gcm_key_create` The user should use the same key to encrypt and decrypt the data

Name	Description	Notes
Initialization vector (IV)	Initialization vector to be used by the AES-GCM algorithm	The user should use the same IV to encrypt and decrypt the data
Initialization vector length	Initialization vector length that must be supplied for the AES-GCM algorithm	Represented in bytes, 0B-12B values are supported
Authentication tag size	Authentication tag size to be supplied for the AES-GCM algorithm. The tag, present at the end of the source buffer, is verified and is not present in the destination buffer.	Represented in bytes, only 12B and 16B values are supported
Additional authenticated data size	Additional authenticated data size to be supplied for the AES-GCM algorithm. This data, present at the beginning of the source buffer, is not encrypted but is authenticated.	Represented in bytes

Task Output

Common output as described in DOCA Core Task.

Task Completion Success

After the task completes successfully, the following happens:

- The data from the source buffer is decrypted and written to the destination buffer
- The destination buffer data segment is extended to include the decrypted data

Task Completion Failure

If the task fails midway:

- The context may enter stopping state if a fatal error occurs
- The source and destination `doca_buf` objects is not modified

- The destination buffer contents may be modified

Task Limitations

- The operation is not atomic
- Once the task is submitted, the source and destination should not be read/written to
- Other limitations are described in DOCA Core Task

Events

DOCA AES-GCM exposes asynchronous events to notify about changes that happen unexpectedly according to the DOCA Core architecture.

The only events AES-GCM exposes are common events as described in DOCA Core Event.

State Machine

The DOCA AES-GCM library follows the Context state machine as described in DOCA Core Context State Machine.

The following section describes moving states and what is allowed in each state.

Idle

In this state, it is expected that the application either:

- Destroys the context
- Starts the context

Allowed operations:

- Configuring the context according to Configurations

- Starting the context

It is possible to reach this state as follows:

Previous State	Transition Action
None	Create the context
Running	Call stop after making sure all tasks have been freed
Stopping	Call progress until all tasks are completed and freed

Starting

This state cannot be reached.

Running

In this state, it is expected that the application:

- Allocates and submits tasks
- Calls progress to complete tasks and/or receive events

Allowed operations:

- Allocating previously configured task
- Submitting a task
- Calling stop

It is possible to reach this state as follows:

Previous State	Transition Action
Idle	Call start after configuration

Stopping

In this state, it is expected that the application:

- Calls progress to complete all inflight tasks (tasks complete with failure)
- Frees any completed tasks

Allowed operations:

- Calling progress

It is possible to reach this state as follows:

Previous State	Transition Action
Running	Call progress and fatal error occurs
Running	Call stop without freeing all tasks

Alternative Datapath Options

DOCA AES-GCM only supports datapath on the CPU. See Execution Phase.

DOCA AES-GCM Samples

This section describes DOCA AES-GCM samples based on the DOCA AES-GCM library.

The samples in this section illustrate how to use the DOCA AES-GCM API to do the following:

- Encrypt contents of a buffer to another buffer
- Decrypt contents of a buffer to another buffer

Info

All the DOCA samples described in this section are governed under the BSD-3 software license agreement.

Running the Samples

1. Refer to the following documents:
   - DOCA Installation Guide for Linux for details on how to install BlueField-related software.
   - DOCA Troubleshooting for any issue you may encounter with the installation, compilation, or execution of DOCA samples.
2. To build a given sample:

```
cd /opt/mellanox/doca/samples/doca_aes_gcm/<sample_name>
meson/tmp/build
ninja -C/tmp/build
```

> **Info**
>
> The binary `doca_<sample_name>` is created under `/tmp/build/`.

3. Sample (e.g., `doca_aes_gcm_encrypt`) usage:

```
Usage: doca_aes_gcm_encrypt [DOCA Flags] [Program Flags]

DOCA Flags:
  -h, --help                        Print a help synopsis
```

```
  -v, --version                    Print program version
information
  -l, --log-level                  Set the (numeric) log
level for the program <10=DISABLE, 20=CRITICAL, 30=ERROR,
40=WARNING, 50=INFO, 60=DEBUG, 70=TRACE>
  --sdk-log-level                  Set the SDK (numeric) log
level for the program <10=DISABLE, 20=CRITICAL, 30=ERROR,
40=WARNING, 50=INFO, 60=DEBUG, 70=TRACE>
  -j, --json <path>                Parse all command flags
from an input json file

Program Flags:
  -p, --pci-addr                   DOCA device PCI device
address - default: 03:00.0
  -f, --file                       Input file to
encrypt/decrypt
  -o, --output                     Output file - default:
/tmp/out.txt
  -k, --key                        Raw key to
encrypt/decrypt with, represented in hex format (32
characters for 128-bit key, and 64 for 256-bit key) - default: 256-
bit key, equals to zero
  -i, --iv                         Initialization vector,
represented in hex format (0-24 characters for 0-96-bit IV) -
default: 96-bit IV, equals to zero
  -t, --tag size                   Authentication tag size.
Tag size is in bytes and can be 12B or 16B - default: 12
  -a, --aad size                   Additional authenticated
data size - default: 0
```

4. For additional information per sample, use the `-h` option:

```
/tmp/build/doca_<sample_name>-h
```

Samples

AES-GCM Encrypt

This sample illustrates how to encrypt data with AES-GCM.

The sample logic includes:

1. Locating DOCA device.
2. Initializing required DOCA Core structures.
3. Setting the AES-GCM encrypt tasks configuration.
4. Populating DOCA memory map with two relevant buffers.
5. Allocating element in DOCA buffer inventory for each buffer.
6. Creating DOCA AES-GCM key.
7. Allocating and initializing AES-GCM encrypt task.
8. Submitting AES-GCM encrypt task.
9. Retrieving AES-GCM encrypt task once it is done.
10. Checking task result.
11. Destroying all AES-GCM and DOCA Core structures.

Reference:

- `/opt/mellanox/doca/samples/doca_aes_gcm/aes_gcm_encrypt/aes_gcm_e`
- `/opt/mellanox/doca/samples/doca_aes_gcm/aes_gcm_encrypt/aes_gcm_e`
- `/opt/mellanox/doca/samples/doca_aes_gcm/aes_gcm_encrypt/meson.bui`

AES-GCM Decrypt

This sample illustrates how to decrypt data with AES-GCM.

The sample logic includes:

1. Locating DOCA device.
2. Initializing needed DOCA Core structures.
3. Setting the AES-GCM decrypt tasks configuration.
4. Populating DOCA memory map with two relevant buffers.
5. Allocating element in DOCA buffer inventory for each buffer.
6. Creating DOCA AES-GCM key.
7. Allocating and initializing AES-GCM decrypt task.
8. Submitting AES-GCM decrypt task.
9. Retrieving AES-GCM decrypt task once it is done.
10. Checking task result.
11. Destroying all AES-GCM and DOCA Core structures.

Reference:

- `/opt/mellanox/doca/samples/doca_aes_gcm/aes_gcm_decrypt/aes_gcm_d`
- `/opt/mellanox/doca/samples/doca_aes_gcm/aes_gcm_decrypt/aes_gcm_d`
- `/opt/mellanox/doca/samples/doca_aes_gcm/aes_gcm_decrypt/meson.bui`

order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.
