DPDK

DATA PLANE DEVELOPMENT KIT

Userspace 2015 | Dublin

A Symmetric Cryptography framework for DPDK

project scope

- mbuf burst oriented APIs for enqueuing and dequeuing of cryptographic workloads on devices.
- Creation of a symmetric Crypto API and device framework which is independent of the crypto device implementation.
- Support for chaining of crypto cipher and hash transforms in a single operation request.
- Session based and session-less crypto operations.

scheduling crypto workloads

- Fundamentally the DPDK crypto framework supports the scheduling of symmetric crypto operations using a mbuf burst oriented asynchronous APIs in the same vain as our ethdev rx/tx burst functions.

```
uint16_t rte_cryptodev_enqueue_burst (uint8_t dev_id, uint16_t qp_id,
                    struct rte_mbuf **pkts,  int16_t nb_pkts);
uint16_t rte_cryptodev_dequeue_burst (uint8_t dev_id, uint16_t qp_id,
                    struct rte_mbuf **pkts, uint16_t nb_pkts);
```

- A new crypto operation pointer has been added to the mbuf structure and a new offload flag PKT_TX_CRYPTO_OP which have to be set in the mbuf before a crypto operation can be requested.

crypto primitives

```
/** Symmetric Cipher Algorithms */
enum rte_crypto_cipher_algorithm {
	RTE_CRYPTO_SYM_CIPHER_NULL,
	RTE_CRYPTO_SYM_CIPHER_AES_CBC,
	RTE_CRYPTO_SYM_CIPHER_AES_GCM,
	...
}
```

```
/** Symmetric Cipher Direction */
enum rte_crypto_cipher_operation {
	RTE_CRYPTO_SYM_CIPHER_OP_ENCRYPT,
	RTE_CRYPTO_SYM_CIPHER_OP_DECRYPT
}
```

```
/** Symmetric Authentication / Hash Algorithms */
enum rte_crypto_auth_algorithm {
	RTE_CRYPTO_SYM_HASH_NONE,
	RTE_CRYPTO_SYM_HASH_SHA1,
	RTE_CRYPTO_SYM_HASH_SHA1_HMAC,
	RTE_CRYPTO_SYM_HASH_SHA224,
	....
}
```

```
/** Symmetric Authentication / Hash Operations */
enum rte_crypto_auth_operation {
	RTE_CRYPTO_SYM_HASH_OP_DIGEST_VERIFY,
	RTE_CRYPTO_SYM_HASH_OP_DIGEST_GENERATE
}
```

crypto transforms

```
/**Crypto transform structure. */
struct rte_crypto_xform {
        struct rte_crypto_xform *next;
        enum rte_crypto_xform_type type;
        union {
                struct rte_crypto_auth_xform auth;
                struct rte_crypto_cipher_xform cipher;
        };
};
```

```
/** Cipher Transform parameters */
struct rte_crypto_cipher_xform {
        enum rte_crypto_cipher_operation op;
        enum rte_crypto_cipher_algorithm algo;
        struct rte_crypto_key key;
};
```

```
/** Authentication Transform parameters */
struct rte_crypto_auth_xform {
        enum rte_crypto_auth_operation op;
        enum rte_crypto_auth_algorithm algo;
        struct rte_crypto_key key;
        uint32_t digest_length;
        uint32_t add_auth_data_length;
};
```

session management

- Sessions are used to manage information such as expand cipher keys and HMAC IPADs and OPADs, which need to calculated for a particular crypto operation, but are immutable on a packet to packet basis for a flow.
- Crypto sessions cache this immutable data in a optimal way for the underlying PMD and this allows further acceleration of the offload of crypto workloads.

struct rte_cryptodev_session *
rte_cryptodev_session_create(uint8_t dev_id, struct rte_crypto_xform *xform);

struct rte_cryptodev_session *
rte_cryptodev_session_free(struct rte_cryptodev_session *session);

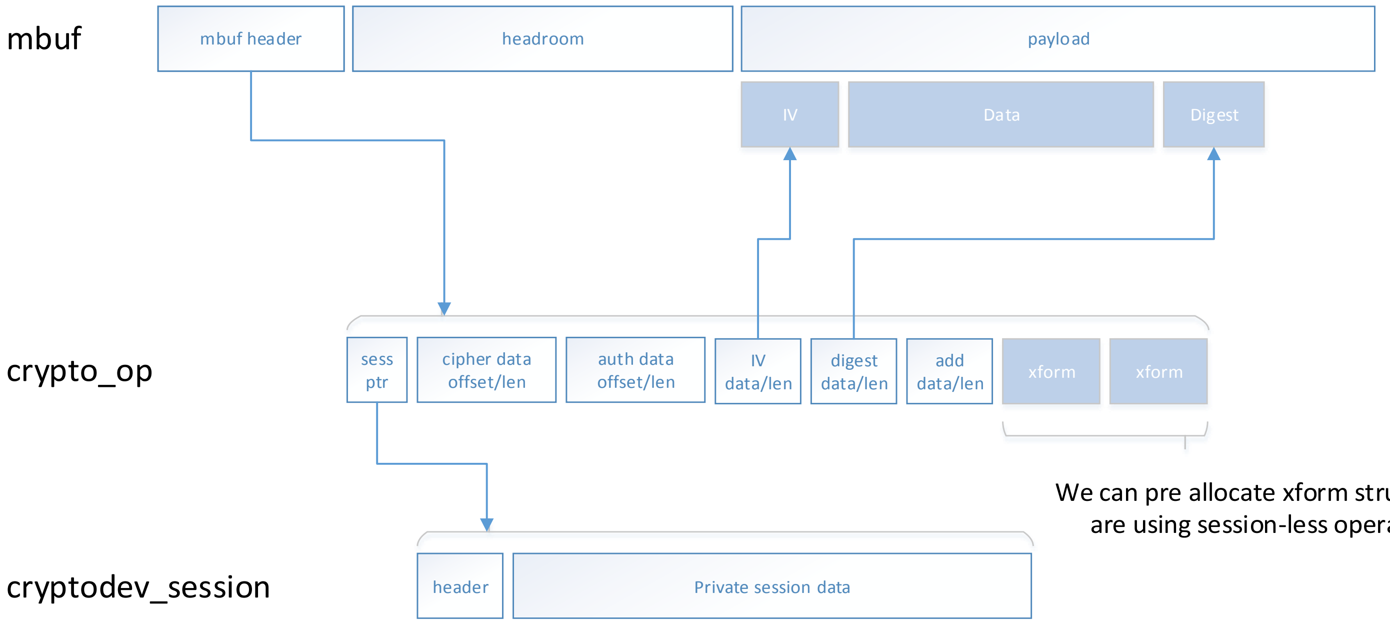
session pool management

- The crypto device framework provides a set of session pool management APIs for the creation and freeing of the sessions
- The framework also provides hooks so the PMDs can pass the amount of memory required for that PMDs private session parameters, as well as initialization functions for the configuration of the session parameters and freeing function so the PMD can managed the memory on destruction of a session
- Sessions created on a particular device can only be used on crypto devices of the same type, and if you try to use a session on a device different to that on which it was created then the crypto operation will fail

crypto operations

- Crypto operation data structures must be attached to each mbuf which you wish to apply a crypto transform to.
- It specifies the offsets and length of the data into the mbuf payload which is to be operated on.
- It contains pointers to IV, digest and additional authentication data, set as required, which can be in the mbuf or at a different memory location. When using a hw accelerators the physical addresses must be set for these parameters.
- Finally the crypto operation contains either a pointer to the crypto session or in the case of a session-less operation a pointer to the first element of a xform chain.

crypto operations



We can pre allocate xform structs if we are using session-less operations

operation pools

- As crypto operations are assigned on a per packet basis, and therefore need to be allocated in the data path. We have create some pktmbuf like functions for managing per allocated crypto operations mempools.
- Note that the pool create function takes a nb_xforms parameter, this can be used to allocate memory for xform chains if you are planning on using session-less operations.

struct rte_mempool ***rte_crypto_op_pool_create** (const char *name, unsigned nb_ops, unsigned cache_size, unsigned nb_xforms, int socket_id);

struct rte_crypto_op_data ***rte_crypto_op_alloc** (struct rte_mempool *mp);

void **rte_crypto_op_free** (struct rte_crypto_op_data *op);

session-less operations

- This allows crypto operations to be submitted to a crypto device without the need to have created a cached session.

```
struct rte_crypto_op_data * rte_crypto_op_alloc_sessionless (struct rte_mempool *mp,
        unsigned nb_xforms);
```

- Returns crypto op with session-less flag set and transform chain pointers setup.
- User is required to set transform type and populate the parameters needed.

```
crypto_op->xform->type = RTE_CRYPTO_XFORM_CIPHER

crypto_op->xform->next->type = RTE_CRYPTO_XFORM_HASH
```

DPDK

Implemented PMD’s

AES-NI multi-buffer PMD

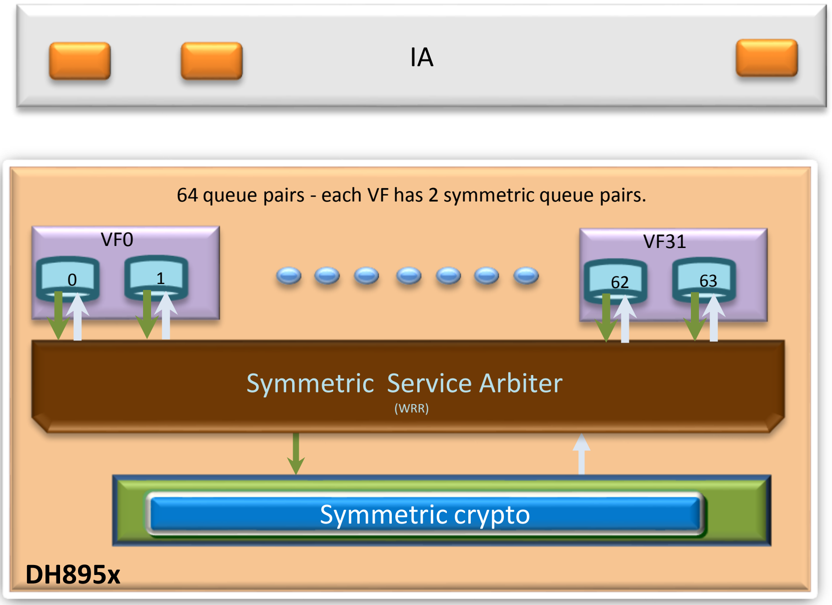
- A purely software based PMD.
- Takes advantage of **Advanced Encryption Standard New Instructions (AES-NI)** instructions to improve the speed of performing AES encryption and decryption on core.
- The PMD is a light weight wrapper around the multi-buffer library
- It also leverages the vectorised instructions to further accelerate both cipher and authenatication processing.
- Whitepaper: http://www.intel.com/content/www/us/en/intelligent-systems/intel-technology/fast-multi-buffer-ipsec-implementations-ia-processors-paper.html
- Download: https://downloadcenter.intel.com/download/22972

QAT PMD

- PMD is a data path driver for Intel's QuickAssist Technology specificly supporting the DH89xx series (Coleto Creek) of accelerators.
- Provides up to 50 Gbps of bulk crypto.
- Cryptographic Primitives Supported
  - **Symmetric ciphers**: **AES**, 3DES/DES, RC4, Kasumi, Snow3G …
  - **Message Digest/Hash** (MD5, **SHA1**, **SHA2**) and Authentication (**HMAC**, **AES-XCBC**)
  - **Algorithm Chaining** (One Cipher and one Hash in a single operation) and **Authenticated Encryption** (**AES-GCM**, AES-CCM)
  - Public key cryptography: RSA, DSA, DH, ECDSA, ECDH
- Data Compression Primitives Supported
  - Compression and Decompression
  - Algorithms: Deflate (LZ77 plus Huffman coding with gzip or zlib header)
  - Stateful and stateless compression and decompression

QAT PMD

- Still requires the PF kernel driver
    - Enabling SR-IOV on the QAT device to expose multiple VFs.
    - Can support up to 32 VFs.
- Supports 2 queue pairs per VF.
- Reserved space to allow compression and asymmetric queue pairs to be added at a later date.


IA
64 queue pairs - each VF has 2 symmetric queue pairs.
VF0
0
1
VF31
62
63
Symmetric Service Arbiter
(WRR)
Symmetric crypto
DH895x

DPDK

Performance

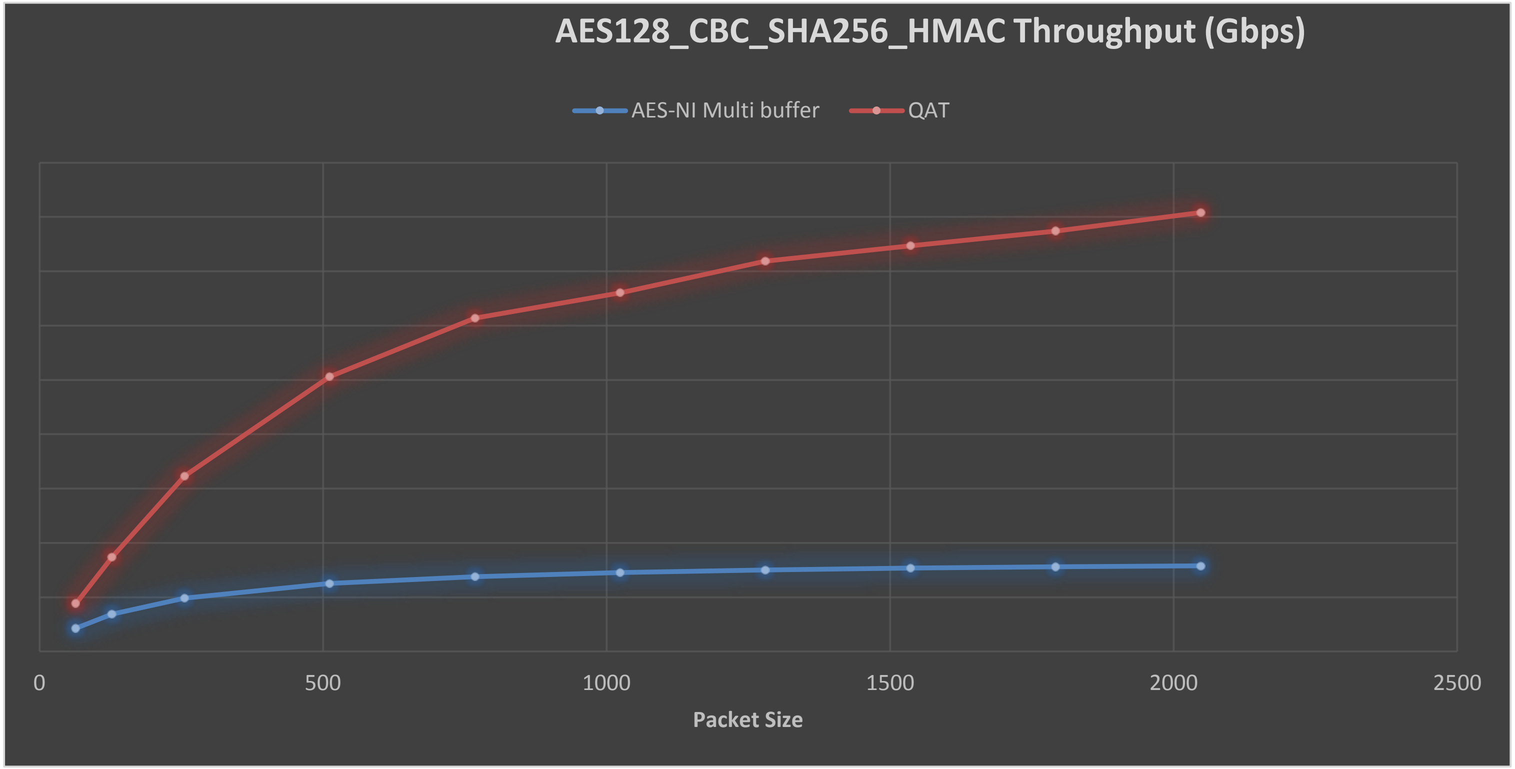
Throughput performance

- Created performance tests to the examples/test applications to allow measurement of baseline performance on your platform.
  - RTE>>cryptodev_qat_perftest
  - RTE>>cryptodev_aesni_mb_perftest

- Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz
  - Intel® QuickAssist Adapter 8950 (PCIe Gen 3 x8)

Single core throughput test

Userspace 2015



DPDK

Future Work

future work

- Adding asymmetric crypto to data path.
- Development of an DPDK accelerated IPsec solution based on the BSD kernel stack.