• Create a DPDK native high performance library for IPsec processing.
• Develop a modular library built around a core functionality of data-path processing and SA management.
• Optional modules which implement scalable and performant security association and security policy database, crypto load-balancing (host, lookaside, inline) and integration point for IKE clients.
• Application abstraction from hardware accelerators, library will handle accelerator allocation and resource usage.
/library/components

- **Core module:**
  - Data-path and SA management (create/destroy/update SA)

- **Optional modules:**
  - SA database with associated data path functions
  - SP database with associated data path functions
  - Crypto processing load-balancer
  - Shim layer for integration of library to existing external IKE solutions.
• Data path API is SA oriented and packet burst based.
• Low-level APIs handle the IPsec protocol processing only with no crypto processing and crypto device management.
• The high-level APIs plan to abstract all IPsec processing including crypto processing and hw acceleration.
• Datapath module can be used independently of the other modules to allow integration with existing applications which may want to use their existing infrastructure.
/library/supported_processing_modes

Host based Crypto Processing

Application

L3
librte_ipsec
L2
ethdev
cryptodev
NET PMD
HW CRYPTO PMD
NIC
Crypto Accelerator

Lookaside Hardware Crypto Processing

Application

L3
librte_ipsec
L2
ethdev
cryptodev
NET PMD
SW CRYPTO PMD
NIC
lcore

IO based Inline Crypto Processing

Application

L3
librte_ipsec
L2
ethdev
cryptodev
NET PMD
SmartNIC
SADB
crypto

Host based Crypto Processing

Lookaside Hardware Crypto Processing

IO based Inline Crypto Processing

SmartNIC
SADB
crypto
/library/19.02 features

- Transport/Tunnel ESP
- IPv4/(IPv6 partial)
- aes-cbc-128/hmac_sha1, aes-gcm-128, null/null
- rte_cryptodev - CPU/lookaside crypto (aesni_mb, aesni_gcm, qat)
- rte_security - inline crypto (ixgbe)
- ESN and anti-replay
  - Multi-threaded sequence handling capabilities
Standard APIs for SAD and SPD, allow deployment specific implementations to be used to optimize the use case.

High performance scalable databases implementation built around table libraries (hash, ACL etc.)

Inbound and outbound data path implementations for the security policy and security association databases.
Optional crypto load-balancer for selecting the best crypto processing option from the available processing models on a per SA basis based on user provided QoS parameters.

Support migration of SA from one processing model to another, i.e. hardware accelerator crypto processing to host CPU crypto processing in case of an oversubscribed accelerator.
/ike-client-shim

- NETLINK XFRM shim layer to allow external IKE client to integrate with the SA/SP databases.
lib/librte_ipsec

Datapath APIs
• SA structure is opaque to the application
• Memory allocation left up to the application.

size_t rte_ipsec_sa_size(uint32_t sz);

• SA initialization and hardware configuration based on sa_prm provided by user.

int rte_ipsec_sa_init(struct rte_ipsec_sa *sa,
    const struct rte_ipsec_sa_prm *prm);

• Query SA type, eg. egress tunnel/ESP

uint64_t rte_ipsec_sa_type(
    const struct rte_ipsec_sa *sa);

• Clear SA parameters, free related resources.

void rte_ipsec_sa_fini(struct rte_ipsec_sa *sa);

/* An opaque structure to reperesent Security Association (SA) */
struct rte_ipsec_sa;

/* SA initialisation parameters.*/
struct rte_ipsec_sa_prm {
    uint64_t userdata; /**< provided and interpreted by user */
    uint64_t flags; /**< see RTE_IPSEC_SAFLAG_* below */

    struct rte_security_ipsec_xform ipsec_xform;
    struct rte_crypto_sym_xform *crypto_xform;

    union {
        struct {
            uint8_t hdr_len; /**< tunnel header len */
            uint8_t hdr_l3_off; /**< offset for IPv4/IPv6 header */
            uint8_t next_proto; /**< next header protocol */
            const void *hdr; /**< tunnel header template */
        } tun; /**< tunnel mode repeated parameters */
        struct {
            uint8_t proto; /**< next header protocol */
        } trs; /**< transport mode repeated parameters */
    }

    uint32_t replay_win_sz;
    /**< window size to enable sequence replay attack handling. */
    /**< * Replay checking is disabled if the window size is 0. */
};
rte_ipsec_session is an aggregate structure that defines the particular SA on given security/crypto device:

- pointer to the SA object
- security session action type
- pointer to security/crypto session, plus other related data
- session/device specific functions to prepare/process IPsec packets

```c
int rte_ipsec_session_prepare(
    struct rte_ipsec_session *session);
```

```c
struct rte_ipsec_session {
    struct rte_ipsec_sa *sa;
    /** session action type */
    enum rte_security_session_action_type type;
    /** session and related data */
    union {
        struct {
            struct rte_cryptodev_sym_session *ses;
        } crypto;
        struct {
            struct rte_security_session *ses;
            struct rte_security_ctx *ctx;
            uint32_t ol_flags;
        } security;
    };
    /** functions to prepare/process IPsec packets */
    struct rte_ipsec_sa_pkt_func pkt_func;
} __rte_cache_aligned;
```
/apis/rte_ipsec_pkt_crypto_prepare

- Takes an array of packets with their associated rte_ipsec_session.
- Preforms all IPsec related pre-crypto processing on the packet burst.
- Generates a set of crypto operations for processing on a crypto device for the input packet burst.

```
static inline uint16_t rte_ipsec_pkt_crypto_prepare(const struct rte_ipsec_session *session,
                                                    struct rte_mbuf *mb[], struct rte_crypto_op *cop[], uint16_t num)
```
• Retrieve IPsec session pointer from crypto_op metadata

```c
static inline struct rte_ipsec_session *
rte_ipsec_ses_from_crypto(const struct rte_crypto_op *cop);
```

• Take as input an input array crypto ops associated with multiple IPsec sessions.

• Extract related mbufs and group them by the `rte_ipsec_session` that they belong to successfully.

```c
static inline uint16_t
rte_ipsec_pkt_crypto_group(  
    const struct rte_crypto_op *cop[],  
    struct rte_mbuf *mb[],  
    struct rte_ipsec_group grp[], uint16_t num);
```

• For mbuf which crypto-op wasn't completed a failure flag will be raised in ol_flags

```c
struct rte_ipsec_group {  
    union {  
        uint64_t val;  
        void *ptr;  
    } id; /**< grouped by value */  
    struct rte_mbuf **m; /**< start of the group */  
    uint32_t cnt; /**< number of entries in the group */  
    int32_t rc; /**< status code associated with the group */  
};
```
• Takes a burst of processed crypto operations associated with a single SA session.

• Performs all IPsec related post crypto processing and returns an array of processed mbufs.

static inline uint16_t
rte_ipsec_pkt_process(const struct rte_ipsec_session *session,
struct rte_mbuf *mb[], uint16_t num);
/datapath/low-level-pipeline

- **Input packet burst**
- **SA lookup**
- **IPsec pkt crypto prepare**
- **Crypto enq**
- **Crypto deq**
- **IPsec pkt crypto group**
- **IPsec pkt process**

- **Unprocessed crypto-op burst grouped by SA**
- **Processed ungrouped crypto-op burst**
- **Processed grouped crypto-op burst**

- **Packet burst grouped by SA**
- **Crypto processing**
- **IPsec processed packet burst**
Examples

IPsec Security GW
• Introduces new parallel processing path to existing application which uses the new library

• Will maintain existing data path in application until full feature parity
  • -I option used to select new codepath
lib/librte_ipsec
• IPsec Library Patches
  http://patches.dpdk.org/patch/48143/
  http://patches.dpdk.org/patch/48144/
  http://patches.dpdk.org/patch/48145/
  http://patches.dpdk.org/patch/48146/
  http://patches.dpdk.org/patch/48147/
  http://patches.dpdk.org/patch/48148/
  http://patches.dpdk.org/patch/48149/
  http://patches.dpdk.org/patch/48150/
  http://patches.dpdk.org/patch/48151/

• IPsec Security GW
  http://patches.dpdk.org/patch/48276/
  http://patches.dpdk.org/patch/48277/
  http://patches.dpdk.org/patch/48278/
  http://patches.dpdk.org/patch/48279/
  http://patches.dpdk.org/patch/48280/
  http://patches.dpdk.org/patch/48281/
  http://patches.dpdk.org/patch/48282/
• AH transport/tunnel mode.
• Full IPv6 support.
• Fully migrate examples/ipsec-secgw to use librte_ipsec.
• Data-path scaling, multicore processing of “Fat Flow” SA.
• High Level Data Path APIs.
• SAD APIs and database implementation.
• SPD APIs and database implementation.
• External IKE daemon integration.