NAME

ibv_flow_action_esp – Flow action esp for verbs

SYNOPSIS

```
#include <infiniband/verbs.h>

struct ibv_flow_action *
ibv_create_flow_action_esp(struct ibv_context *ctx,
                struct ibv_flow_action_esp *esp);
int
ibv_modify_flow_action_esp(struct ibv_flow_action *action,
                struct ibv_flow_action_esp *esp);

int ibv_destroy_flow_action(struct ibv_flow_action *action);
```

DESCRIPTION

An IPSEC ESP flow steering action allows a flow steering rule to decrypt or encrypt a packet after matching. Each action contains the necessary information for this operation in the *params* argument.

After the crypto operation the packet will continue to be processed by flow steering rules until it reaches a final action of discard or delivery.

After the action is created, then it should be associated with a *struct ibv_flow_attr* using *struct ibv_flow_spec_action_handle* flow specification. Each action can be associated with multiple flows, and *ibv_modify_flow_action_esp* will alter all associated flows simultaneously.

ARGUMENTS

ctx RDMA device context to create the action on.

esp ESP parameters and key material for the action.

action Existing action to modify ESP parameters.

action **Argument**

```
struct ibv_flow_action_esp {
    struct ibv_flow_action_esp_attr *esp_attr;

    /* See Key Material */
    uint16_t        keymat_proto;
    uint16_t        keymat_len;
    void           *keymat_ptr;

    /* See Replay Protection */
    uint16_t        replay_proto;
    uint16_t        replay_len;
    void           *replay_ptr;

    struct ibv_flow_action_esp_encap *esp_encap;

    uint32_t        comp_mask;
    uint32_t        esn;
};
```

comp_mask

Bitmask specifying what fields in the structure are valid.

esn The starting value of the ESP extended sequence number. Valid only if *IBV_FLOW_ACTION_ESP_MASK_ESN* is set in *comp_mask*.

The 32 bits of *esn* will be used to compute the full 64 bit ESN required for the AAD construction.

When in *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_INLINE_CRYPTO* mode, the implementa-

tion will automatically track rollover of the lower 32 bits of the ESN. However, an update of the window is required once every 2^31 sequences.

When in *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_FULL_OFFLOAD* mode this value is automatically incremended and it is also used for anti−replay checks.

esp_attr
See *ESP Attributes*. May be NULL on modify.

keymat_proto, *keymat_len*, *keymat_ptr*
Describe the key material and encryption standard to use. May be NULL on modify.

replay_proto, *replay_len*, *replay_ptr*
Describe the replay protection scheme used to manage sequence numbers and prevent replay attacks. This field is only valid in full offload mode. May be NULL on modify.

esp_encap
Describe the encapsulation of ESP packets such as the IP tunnel and/or UDP encapsulation. This field is only valid in full offload mode. May be NULL on modify.

ESP attributes

```
struct ibv_flow_action_esp_attr {
    uint32_t   spi;
    uint32_t   seq;
    uint32_t   tfc_pad;
    uint32_t   flags;
    uint64_t   hard_limit_pkts;
};
```

flags A bitwise OR of the various *IB_UVERBS_FLOW_ACTION_ESP_FLAGS* described below.

IB_UVERBS_FLOW_ACTION_ESP_FLAGS_DECRYPT, *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_ENCRYPT*
The action will decrypt or encrypt a packet using the provided keying material.

The implementation may require that encrypt is only used with an egress flow steering rule, and that decrypt is only used with an ingress flow steering rule.

Full Offload Mode

When *esp_attr* flag *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_FULL_OFFLOAD* is set the ESP header and trailer are added and removed automatically during the cipher operation. In this case the *esn* and *spi* are used to populate and check the ESP header, and any information from the *keymat* (eg a IV) is placed in the headers and otherwise handled automatically.

For decrypt the hardware will perform anti−replay.

Decryption failure will cause the packet to be dropped.

This action must be combined with the flow steering that identifies the packets protected by the SA defined in this action.

The following members of the esp_attr are used only in full offload mode:

spi The value for the ESP Security Parameters Index. It is only used for *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_FULL_OFFLAOD*.

seq The initial 32 lower bytes of the sequence number. This is the value of the ESP sequence number. It is only used for *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_FULL_OFFLAOD*.

tfc_pad The length of Traffic Flow Confidentiality Padding as specified by RFC4303. If it is set to zero no additional padding is added. It is only used for *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_FULL_OFFLAOD*.

hard_limit_pkts
: The hard lifetime of the SA measured in number of packets. As specified by RFC4301. After this limit is reached the action will drop future packets to prevent breaking the crypto. It is only used for *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_FULL_OFFLAOD*.

Inline Crypto Mode

When *esp_attr* flag *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_INLINE_CRYPTO* is set the user must providate packets with additional headers.

For encrypt the packet must contain a fully populated IPSEC packet except the data payload is left un−encrypted and there is no IPsec trailer. If the IV must be unpredictable, then a flag should indicate the transofrmation such as *IB_UVERBS_FLOW_ACTION_IV_ALGO_SEQ*.

IB_UVERBS_FLOW_ACTION_IV_ALGO_SEQ means that the IV is incremented sequentually. If the IV algorithm is supported by HW, then it could provide support for LSO offload with ESP inline crypto.

Finally, the IV used to encrypt the packet replaces the IV field provided, the payload is encrypted and authenticated, a trailer with padding is added and the ICV is added as well.

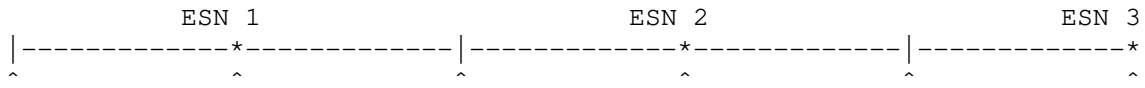
For decrypt the packet is authenticated and decrypted in−place, resulting in a decrypted IPSEC packet with no trailer. The result of decryption and authentication can be retrieved from an extended CQ via the *ibv_wc_read_XXX(3)* function.

This mode must be combined with the flow steering including *IBV_FLOW_SPEC_IPV4* and *IBV_FLOW_SPEC_ESP* to match the outer packet headers to ensure that the action is only applied to IPSEC packets with the correct identifiers.

For inline crypto, we have some special requirements to maintain a stateless ESN while maintaining the same parameters as software. The system supports offloading a portion of the IPSEC flow, enabling a single flow to be split between multiple NICs.

Determining the ESN for Ingress Packets

We require a "modify" command once every 2^31 packets. This modify command allows the implementation in HW to be stateless, as follows:



^ − marks where command invoked to update the SA ESN state machine.
| − marks the start of the ESN scope (0−2^32−1). At this point move SA ESN "new_window" bit to zero and increment ESN.
* − marks the middle of the ESN scope (2^31). At this point move SA ESN "new_window" bit to one.

For decryption the implementation uses the following state machine to determine ESN:

```
if (!overlap) {
    use esn // regardless of packet.seq
} else { // new_window
    if (packet.seq >= 2^31)
        use esn
    else // packet.seq < 2^31
        use esn+1
}
```

This mechanism is controlled by the *esp_attr* flag:

IB_UVERBS_FLOW_ACTION_ESP_FLAGS_ESN_NEW_WINDOW
: This flag is only used to provide stateless ESN support for inline crypto. It is used only for *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_INLINE_CRYPTO* and *IBV_FLOW_ACTION_ESP_MASK_ESN*.

  Setting this flag indicates that the bottom of the replay window is between 2^31 − 2^32.

Key Material for AES GCM (*IBV_ACTION_ESP_KEYMAT_AES_GCM*)

The AES GCM crypto algorithm as defined by RFC4106. This struct is to be provided in *keymat_ptr* when *keymat_proto* is set to *IBV_ACTION_ESP_KEYMAT_AES_GCM*.

```
struct ibv_flow_action_esp_aes_keymat_aes_gcm {
    uint64_t   iv;
    uint32_t   iv_algo; /* Use enum ib_uverbs_flow_action_esp_aes_gcm_keymat

    uint32_t   salt;
    uint32_t   icv_len;

    uint32_t   key_len;
    uint32_t   aes_key[256 / 32];
};
```

iv The starting value for the initialization vector used only with *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_FULL_OFFLOAD* encryption as defined in RFC4106. This field is ignored for *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_INLINE_CRYPTO*.

For a given key, the IV MUST NOT be reused.

iv_algo The algorithm used to transform/generate new IVs with *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_FULL_OFFLOAD* encryption.

The only supported value is *IB_UVERBS_FLOW_ACTION_IV_ALGO_SEQ* to generate sequantial IVs.

salt The salt as defined by RFC4106.

icv_len The length of the Integrity Check Value in bytes as defined by RFC4106.

aes_key, *key_len*
The cipher key data. It must be either 16, 24 or 32 bytes as defined by RFC4106.

Bitmap Replay Protection (*IBV_FLOW_ACTION_ESP_REPLAY_BMP*)

A shifting bitmap is used to identify which packets have already been transmitted. Each bit in the bitmap represents a packet, it is set if a packet with this ESP sequence number has been received and it passed authentication. If a packet with the same sequence is received, then the bit is already set, causing replay protection to drop the packet. The bitmap represents a window of *size* sequence numbers. If a newer sequence number is received, then the bitmap will shift to represent this as in RFC6479. The replay window cannot shift more than 2^31 sequence numbers forward.

This struct is to be provided in *replay_ptr* when *reply_proto* is set to *IBV_FLOW_ACTION_ESP_REPLAY_BMP*. In this mode reply_ptr and reply_len should point to a struct ibv_flow_action_esp_replay_bmp containing: *size* : The size of the bitmap.

ESP Encapsulation

An *esp_encap* specification is required when *eps_attr* flags *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_TUNNEL* is set. It is used to provide the fields for the encapsulation header that is added/removed to/from packets. Tunnel and Transport mode are defined as in RFC4301. UDP encapsulation of ESP can be specified by providing the appropriate UDP header.

This setting is only used in *IB_UVERBS_FLOW_ACTION_ESP_FLAGS_FULL_OFFLOAD* mode.

```
struct ibv_flow_action_esp_encap {
    void        *val;        /* pointer to struct ibv_flow_xxxx_filter */
    struct ibv_flow_action_esp_encap   *next_ptr;
    uint16_t    len;            /* Len of mask and pointer (separately) */
    uint16_t    type;           /* Use flow_spec enum */
};
```

Each link in the list specifies a network header in the same manner as the flow steering API. The header should be selected from a supported header in 'enum ibv_flow_spec_type'.

RETURN VALUE

Upon success *ibv_create_flow_action_esp* will return a new *struct ibv_flow_action* object, on error NULL will be returned and errno will be set.

Upon success *ibv_modify_action_esp* will return 0. On error the value of errno will be returned. If ibv_modify_flow_action fails, it is guarnteed that the last action still holds. If it succeeds, there is a point in the future where the old action is applied on all packets until this point and the new one is applied on all packets from this point and on.

SEE ALSO

ibv_create_flow(3), *ibv_destroy_action(3)*, *RFC 4106*