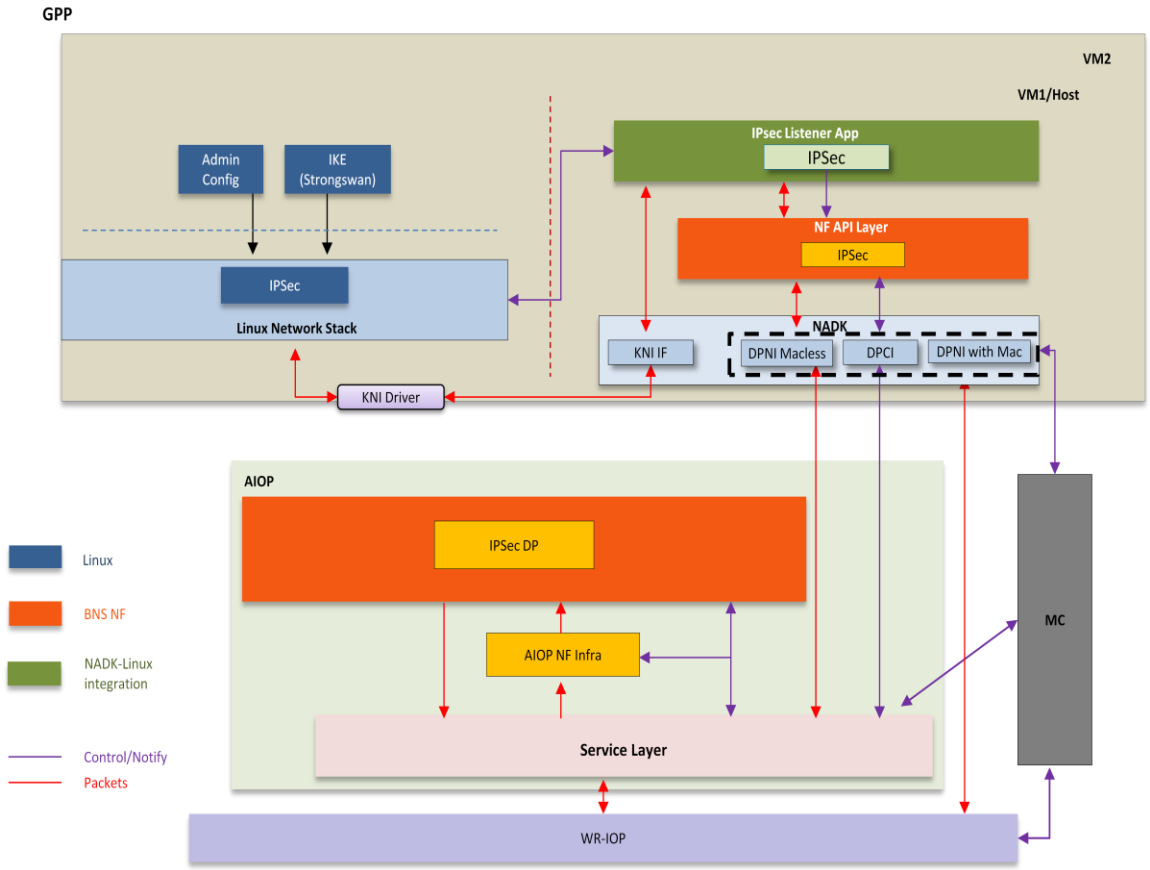
LS2085 NADK Based IPSEC Application Communicating with AIOP

1. Overview and Architecture of the AIOP-NADK Based IPSEC Application

On LS2085 platform, the basic networking product performs autonomous IP forwarding and IPSEC on the AIOP, data path functions run mostly independent of the GPP software and only involve GPP when necessary, GPP is usually involved in exception packet processing. AIOP resident software is used to perform data path operations for common networking and security applications. Networking functions(NF) is integrated with GPP control networking applications, users can integrate their networking applications with the respective NF APIs and immediately realize improved performance in their software using the LS2085A SoC. These make networking applications use of the underlying low level AIOP service layer libraries to provide common networking functions, AIOP can make use of these NF infrastructure elements to implement the networking data path.

NADK(Network Acceleration Development Kit) is a complete user space development kit for networking applications. It provides high-level drivers for Freescale data paths and accelerators and a set of optimized run-time services. This IPSEC application is implemented in Linux user space using NADK framework, this application learns the Linux configuration through the netlink event notification and sends the configuration to AIOP DP using the respective NF APIs. The application invokes NF APIs to send configuration details to IPsec data path on the AIOP. The architecture of the IPsec application is described in the following diagram.



The IPSEC application data path features are summarized as the following.

IPSEC data path offers integrity, confidentiality, services access control, detection and rejection of replays.

Provides organization for Security Policy Database(SPD) and Security Association Database(SAD).

Provides support for IPv4 only.

Provides IP Encapsulating Security Payload(ESP) support, it uses SEC accelerator for crypto, encapsulation, and decapsulation operations.

Provides NF APIs for control plane IPsec application at GPP for configuring SPD policies, SAs.

2. NADK Based GPP Listener Program Design

2.1 NADK APIs Introduction Used in the Application

NADK is a library, the application program invokes NADK library APIs to use services provided by NADK framework.

Memory manger API is responsible for managing the overall memory for NADK applications, it allocates pools for objects in memory, for example APIs nadk_mpool_create and nadk_mpool_delete are used to create and destroy the memory pool of the DMA memory.

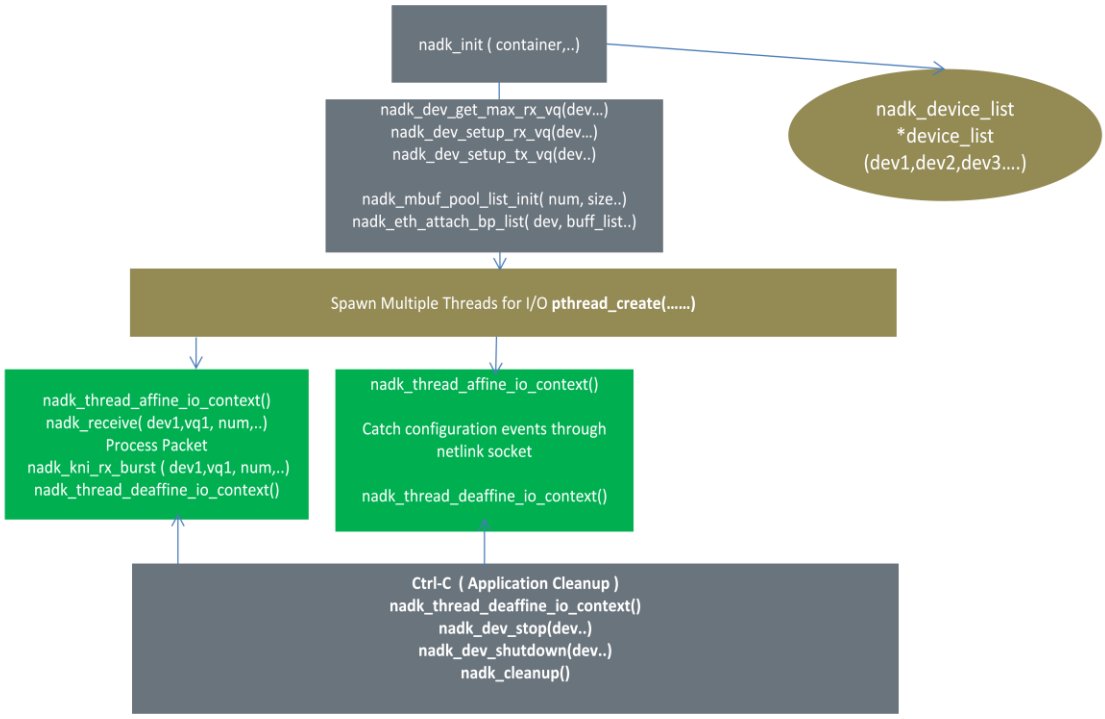
Packet Buffer management library provides the facility for NADK application to store message buffer, the message buffers are created at startup time using BMAN, APIs nadk_mbuf_pool_list_init and nadk_eth_attach_bp_list are used to initialize buffer pool and attach the buffer pool list to each Ethernet device.

The Kernel NIC Interface(KNI) is an NADK control plane solution that allow user space application to exchange packets with the Kernel networking stack, NADK creates a FIFO queue for packet ingress and egress to the KNI kernel module for each device created, Kernel Network Interface provides a copy-based mechanism between Kerner and user-space components to send and receive control packets to the Linux network stack, nadk_kni_rx_burst, nadk_kni_rx_burst and nadk_kni_handle_request are used in the application to received, send and handle packets from and to KNI devices.

I/O portals are not specific to any device, but are required to perform I/O to any device, each application thread needs to affine a portal to itself with nadk_thread_affine_io_context.

2.2 Packet Processing in Multiple Threads Mode NADK Application

The application is designed in two threads modes, one thread is used for monitoring NI and KNI devices for packet processing, the other monitors netlink message and uses cmdif to communicate with AIOP. The following diagram describes the packets processing of the application.



2.3 IPSEC XFRM Events monitored by the Listener

The IPSEC application sets up xfrm Netlink sock, receiving and detecting the following events and invokes NF IPSEC APIs to process the configuration event to organize Security Policy Database.
XFRM_MSG_NEWPOLICY: handled by ipsec_new_policy_event invoking nf_ipsec_spd_add(ns_id,&in, API_CTRL_FLAG_NO_RESP_EXPECTED,&out, NULL).

XFRM_MSG_UPDPOLICY: handled by function ipsec_update_policy_event invoking API nf_ipsec_spd_mod(ns_id,&in, API_CTRL_FLAG_NO_RESP_EXPECTED, &out, NULL).

XFRM_MSG_DELPOLICY: handled by function ipsec_del_policy_event invoking NFAPI nf_ipsec_spd_del(ns_id, &in , API_CTRL_FLAG_NO_RESP_EXPECTED, &out, NULL).

XFRM_MSG_FLUSHPOLICY: handled by function ipsec_flush_policy_event invoking NF API nf_ipsec_spd_del(ns_id, &in , API_CTRL_FLAG_NO_RESP_EXPECTED, &out, NULL).

XFRM_MSG_POLEXPIRE: handled by function ipsec_expire_policy_event invoking NF API nf_ipsec_spd_del(ns_id, &in , API_CTRL_FLAG_NO_RESP_EXPECTED, &out, NULL).

XFRM_MSG_NEWSA: handled by function ipsec_new_sa_event invoking NF API nf_ipsec_sa_add(ns_id, &in, API_CTRL_FLAG_NO_RESP_EXPECTED, &out, NULL), the encapsulating modes TUNNEL and TRANSPORT are supported.

XFRM_MSG_DELSA: handled by function ipsec_del_sa_event invoking nf_ipsec_sa_del(ns_id,&in, API_CTRL_FLAG_NO_RESP_EXPECTED, &out, NULL).

XFRM_MSG_UPDSA: handled by function ipsec_update_sa_event invoking nf_ipsec_sa_del(ns_id, &in, API_CTRL_FLAG_NO_RESP_EXPECTED, &out, NULL).
XFRM_MSG_EXPIRE: handled by ipsec_expire_sa_event invoking nf_ipsec_sa_del(ns_id, &in, API_CTRL_FLAG_NO_RESP_EXPECTED, &out, NULL).

XFRM_MSG_FLUSHSA: handled by ipsec_flush_sa_event invoking nf_ipsec_sa_del(ns_id, &in , API_CTRL_FLAG_NO_RESP_EXPECTED, &out, NULL).

3. IPSEC Application communicates with AIOP through NF APIs

3.1 IPSEC provided features implemented at AIOP

The followings are IPSEC provided features which are implemented at AIOP.
Tunnel Mode: provides IPSec services to IP and higher level protocols.

Transport Mode: provides IPSEC services mainly to higher level.

UDP Nat Traversal: UDP Encapsulation for the traffic to successfully pass through the NAT devices.

ESP: Encapsulation Security Protocol provides confidentiality and limited traffic flow confidentiality.

ESN: Extended Sequence Number.

Redside Fragmentation: Configuration on a SPD policy basis.

Reassemble before IPSEC: IP reassembly on inbound packet.

Life Time Seconds/Kilobytes/Number of Packets: Upon soft/hard life time expiry, and indication to normal path will be given to re-negotiate/establish IPSEC SA.

Path MTU: IPSec receives an unauthenticated message having size greater than SA MTU, on some conditions it drops the packet and sends PMTU ICMP message.

SPD policy Table organization: Support for SPD Policy Organization to apply/bypass/drop the packets which are entering into IPSEC.

SA Organization: Support for SA organization, which is looked up to select matching SA during packet processing to apply IPSec (Encrypt/De-crypt).

SA Per DSCP: Separate SA is used/established for the matching DSCP range.

ICMP Error Message processing: ICMP Error Messages directed to the Security Device Process/Discard if the configuration is available for Type and Code.

Algorithms support:
ESP-Encryption Algorithms(AES, AES Counter, DES, 3DES).
ESP-Authentication Algorithms(MD5, SHA1 SHA2 Variants).

3.2 IPSEC NF APIs Used to Configure AIOP

nf_ipsec_spd_add: This API is used to add Inbound/Outbound SPD policy to SPD policy database.
nf_ipsec_spd_del: This API is used to delete Inbound/Outbound SPD policy from SPD policy database.

nf_ipsec_spd_mod: This API is used to modify Inbound/Outbound SPD policy.
nf_ipsec_spd_get: This API is used to fetch Inbound/Outbound SPD information.
nf_ipsec_spd_flush: This API is used to flush/delete all Inbound and Outbound SPD policies in a given name space.

nf_ipsec_icmp_err_msg_typecode_add: This API is used to add ICMP Error Message Type and Code configuration to ICMP database.
nf_ipsec_icmp_err_msg_typecode_del: This API is used to delete ICMP Error Message Type and Code policy from ICMP database.

nf_ipsec_sa_add: This API is used to add Inbound/Outbound SA to SA database.
nf_ipsec_sa_del: This API is used to delete Inbound/Outbound SA from SA database.
nf_ipsec_sa_mod: This API is used to modify Inbound/Outbound SA.
nf_ipsec_sa_get: This API is used to fetch Inbound/Outbound SA information.

nf_ipsec_global_stats_get: This API fetches global statistics for given Name Space.

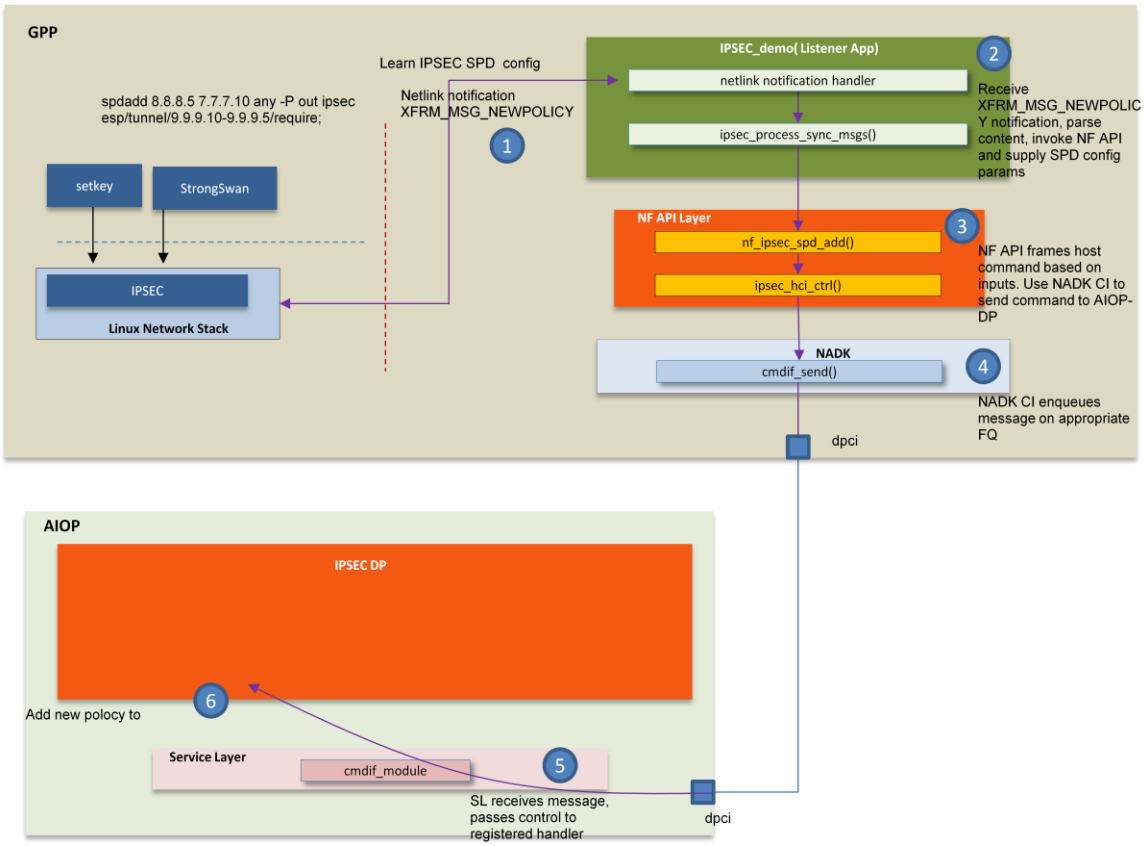
nf_ipsec_capabilities_get: This API fetches IPSec module Capabilities.

nf_ipsec_dp_set_status: This API is used to set IPSec-DP.

nf_ipsec_set_icmp_err_msg_process_status: This API is used to enable/disable ICMP error message processing for given name space.

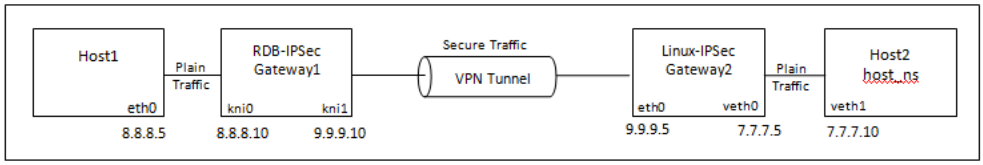
3.3 The process of add SPD policy in the IPSEC Application

The following is the whole process of adding SPD policy in the IPSEC system.



4. Setup Networking Environment to Verify the IPSEC application

In the section, IPSEC data path will be verified in the networking environment setting up with LS2085RDB using security protocol ESP for tunnel mode. LS2085RDB boards are used as security gateways which encapsulate the outgoing traffic and de-capsulate the incoming traffic, forming a secured tunnel between them. In the use case, IPv4 traffic with 3-DES encryption and HMAC-SHA1 authentication algorithms are used with following configuration. The follow is a diagram for the system.



(1)Build aiop_app.elf image with aiop-bnsp enabled, which includes IPSEC function support in AIOP image, then flash this image on NOR Flash, 0x580900000 for current bank and 0x584900000 for the alternate bank. In addition, aiop-bnsp needs to be rebuilt with CodeWarrior for Advanced Packet Processing license.

(2)Configure NADK and IPSEC environment to setup the application.

```
#/usr/nadk/nadk-static/bin/bind_dprc.sh
#insmod /usr/nadk/drv/nadk_kni.ko
insmod /lib/modules/`uname -r`/kernel/net/xfrm/xfrm_algo.ko
insmod /lib/modules/`uname -r`/kernel/net/ipv4/esp4.ko
insmod /lib/modules/`uname -r`/kernel/crypto/authenc.ko
insmod /lib/modules/`uname -r`/kernel/crypto/cbc.ko
insmod /lib/modules/`uname -r`/kernel/crypto/hmac.ko
insmod /lib/modules/`uname -r`/kernel/crypto/des_generic.ko
insmod /lib/modules/`uname -r`/kernel/crypto/sha1_generic.ko
insmod /lib/modules/`uname -r`/kernel/crypto/md5.ko
insmod /lib/modules/`uname -r`/kernel/crypto/authencesn.ko
insmod /lib/modules/`uname -r`/kernel/net/key/af_key.ko
insmod /lib/modules/`uname -r`/kernel/net/xfrm/xfrm_user.ko
# ./ipsec_demo –g dprc.2
```

(3)On ls2085rdb board for gateway 1 enter the following commands.

```
ip netns exec bnsp_ns bash
ifconfig kni0 hw ether 00:00:00:00:00:06
ifconfig kni1 hw ether 00:00:00:00:00:07
ip addr add 8.8.8.10/24 dev kni0
ip link set dev kni0 up
ip addr add 9.9.9.10/24 dev kni1
ip link set dev kni1 up
route add –net 7.7.7.0/24 gw 9.9.9.5
```

(4) Configuration of IPSec Policies and SAs on LS2085RDB Gateway1.

Create the file /etc/setkey.conf with following content, then run the command “setkey -f /etc/setkey.conf”.

```
flush;
spdflush;
spdadd 8.8.8.5 7.7.7.10 any -P out ipsec esp/tunnel/9.9.9.10-9.9.9.5/require;
spdadd 7.7.7.10 8.8.8.5 any -P in ipsec esp/tunnel/9.9.9.5-9.9.9.10/require;
add 9.9.9.10 9.9.9.5 esp 5674 -m tunnel -E 3des-cbc "abcdabcdabcdabcdabcdabcd" -A hmac-sha1 "abcdabcdabcdabcdabcd";
```

add 9.9.9.5 9.9.9.10 esp 4765 -m tunnel -E 3des-cbc "abcdabcdabcdabcdabcdabcd" -A hmac-sha1 "abcdabcdabcdabcdabcd";

(5)On LS2085RDB board for gateway 2 enter the following commands.
ip link add veth0 type veth peer name veth1
ifconfig eth0 9.9.9.5/24 up
ifconfig veth0 7.7.7.5/24 up
route add –net 8.8.8.0/24 gw 9.9.9.10
echo 1 > /proc/sys/net/ip4/ip_forward

(6) Configuration of IPSec Policies and SAs on LS2085RDB Gateway2.
Create the file /etc/setkey.conf with following content, then run the command “setkey -f /etc/setkey.conf”.
flush;
spdflush;
add 9.9.9.10 9.9.9.5 esp 5674 -m tunnel -E 3des-cbc "abcdabcdabcdabcdabcdabcd" -A hmac-sha1 "abcdabcdabcdabcdabcd";
add 9.9.9.5 9.9.9.10 esp 4765 -m tunnel -E 3des-cbc "abcdabcdabcdabcdabcdabcd" -A hmac-sha1 "abcdabcdabcdabcdabcd";
spdadd 8.8.8.5 7.7.7.10 any -P in ipsec esp/tunnel/9.9.9.10-9.9.9.5/require;
spdadd 7.7.7.10 8.8.8.5

(7)Configure Host1 and Host2 networking interfaces.
Host1:
ifconfig eth0 8.8.8.5/24 up
route add –net 7.7.7.0/24 gw 8.8.8.10
Host2:
ip netns add host_ns
ip link set veth1 netns host_ns
ip netns exec host_ns ifconfig veth1 7.7.7.10/24 up
ip netns exec host_ns route add –net 8.8.8.0/24 gw 7.7.7.5

(8) From Host 1 (8.8.8.5):
ping 7.7.7.10 –c 5
Run tcpdump at 9.9.9.5 to confirm IPSec ESP traffic from both of gateways.