PSP security protocol

intro for netdev developers

netdevconf 0x18
July 19th, 2024
willemb@google.com

PSP security protocol

encrypt every connection, hardware offloaded

netdevconf 0x18
July 19th, 2024
willemb@google.com

Goals of PSP

- Scale
  - 10M+ connections
  - 100K+ connections/sec
  - 100 usec key generation/insertion *tail* latency

Goals of PSP

- Scale
  - 10M+ connections
  - 100K+ connections/sec
  - 100 usec key generation/insertion *tail* latency
- Functionality
  - telemetry: maintain per-flow network monitoring in network
  - load balancing: multi-path flows across network and RSS

Goals of PSP

- Scale
  - 10M+ connections
  - 100K+ connections/sec
  - 100 usec key generation/insertion *tail* latency
- Functionality
  - telemetry: maintain per-flow network monitoring in network
  - load balancing: multi-path flows across network and RSS
- Simplicity
  - reduced feature set, especially crypto algorithms
  - simple header parsing

Terminology: IPsec

- SA: Security Association: a simplex connection
- SADB: Security Association Database: connection state indexed by ID
- SPI: Security Parameter Index: u32 connection ID

Features

- Scale
  - No per-connection state on device: O(1) scaling
- Functionality
  - Telemetry: AEAD with crypt_off: integrity only up to e.g., inner TCP ports
  - Load balancing: encapsulate in UDP, entropy in src port
- Simplicity
  - Reduced crypto: AES-GCM only, no AH, etc.
  - Simple header parsing: NextHdr at offset 0, HdrExtLen field, always IV field

Header Layout

Scale

Rx

- 1 device key
- Session key derivation: KDF in counter mode from { device key, SPI }
- Session key derivation: from SPI in packet, at line rate

Tx

- Key in descriptor

Alt

- Keys in SADB
  - Tx and/or Rx
  - storage cost
  - or on-device cache: latency to RAM

Key Exchange and Rotation

- Key exchange
  - Initial: out-of-band (e.g., IKEv2)
  - 2 connection OOB vs 1 connection upgrade model

- Key rotation
  - SPI exhaustion + policy (e.g., time based)
  - Two device keys (double buffering)
  - Connection notification API: must rekey SA before double rotation

Implementation Details

- Replay protection: dependent on inner protocol
  - TCP PAWS
- AES-GCM IV must be unique: high precision device clock timestamp
  - Embedded timestamp for RTT estimation, etc.
  - Clock adjustments are sensitive: could cause repeated IV
- NIC Offloads must continue to work when enabling PSP encapsulation
  - TCP Segmentation offload
  - Receive Segment Coalescing (HW-GRO)
- Bonding: multiple device secrets

IPsec development: meeting PSP Goals

RFC 3948: UDP Encapsulation of IPsec ESP Packets

datatracker.ietf.org/doc/html/rfc3948

RFC 9395 + RFC 7321 + RFC 4835 +

RFC 8221: Cryptographic Algorithm Implementation Requirements [...]

datatracker.ietf.org/doc/html/rfc8221

Wrapped ESP v2 proposal (WESPv2)

datatracker.ietf.org/doc/draft-klassert-ipsecme-wespv2/

Further Reading

PSP architecture spec

github.com/google/psp/tree/main/doc

[RFC net-next 00/15] add basic PSP encryption for TCP connections

lore.kernel.org/netdev/20240510030435.120935-1-kuba@kernel.org/

LWN: "Offload-friendly network encryption in the kernel", July 2024, Daroc Alden

lwn.net/Articles/980430/

PSP Crypto offload with IDPF

netdevconf 0x18

July 2024

Anjali Singhai Jain

Arun Acharya

Phani Burra

IDPF Crypto Offload Support

- Generic Host terminated Crypto offload
  1. IPSec (Preprocessing, Host terminated)
  2. **PSP**
  3. DTLS

- Different ways of programming the SA Context, Crypto key
  1. Indirect over mailbox (virtchannel 2.0)
  2. Direct using Config queue to HW
  3. Crypto key in the TX Descriptor

Note: Host terminated, no HW terminated Crypto tunnels supported on IDPF (No tunnel mode support)

PSP kernel offload support from Kuba

- [RFC net-next 00/15] add basic PSP encryption for TCP connections - Jakub Kicinski (kernel.org)
  - This is with Mellanox (Nvidia) driver RFC patches
  - This is Transport mode support. Not tunnel mode

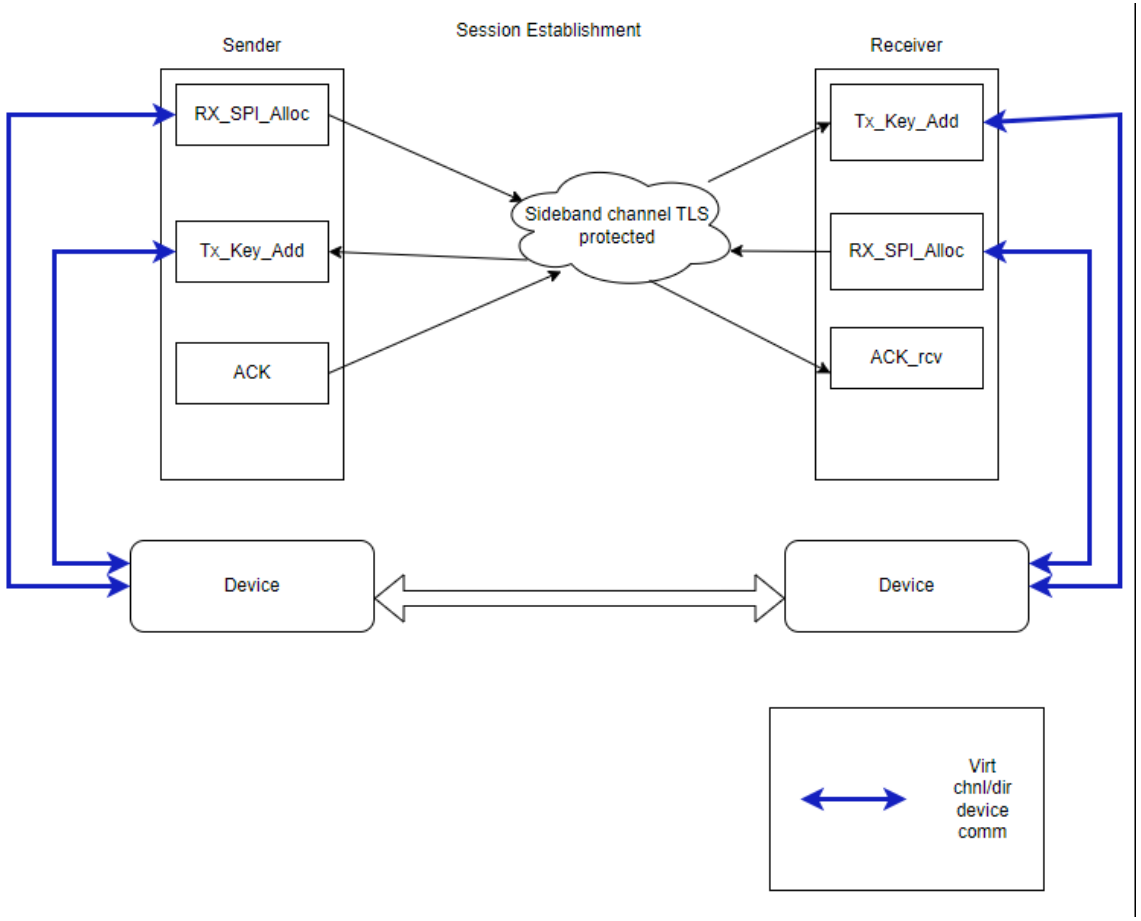
Kernel hooks to IDPF driver:

SL	psp_dev_ops	Comments	Virtual channel commands
1	int (*set_config)(struct psp_dev *psd, struct psp_dev_config *conf,struct netlink_ext_ack *extack);		No specific device config
2	int (*key_rotate)(struct psp_dev *psd, struct netlink_ext_ack *extack);	Upper layer SW triggers a key rotate	VIRTCHNL2_OP_PSP_ROTATE_KEY
3	int (*rx_spi_alloc)(struct psp_dev *psd, u32 version, struct psp_key_parsed *assoc, struct netlink_ext_ack *extack);	The PF/VF request for a spi and session to device.	VIRTCHNL2_OP_PSP_ALLOC_RX_SA
4	int (*tx_key_add)(struct psp_dev *psd, struct psp_assoc *pas, struct netlink_ext_ack *extack);	The PF/VF adds the session key and security association for egress.	VIRTCHNL2_OP_PSP_ADD_TX_SA
5	void (*tx_key_del)(struct psp_dev *psd, struct psp_assoc *pas);	The PF/VF deletes the session key and security association for egress.	VIRTCHNL2_OP_PSP_DEL_TX_SA
6	void (*get_stats)(struct psp_dev *psd, struct psp_dev_stats *stats);	Fetch the PSP statistics from device.	VIRTCHNL2_OP_PSP_GET_STATS

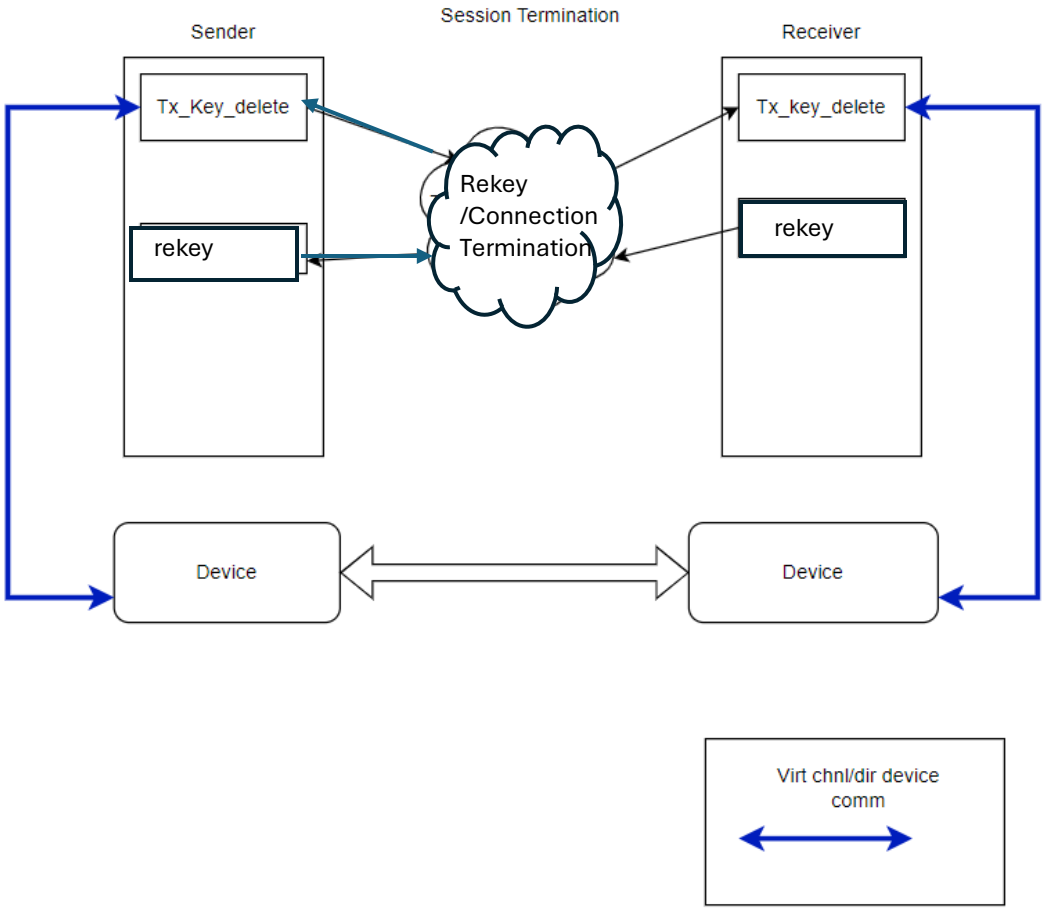
Driver flow and Device Interface Details

- Device Init
- Session Initialization
- Runtime Flows (Inline Encrypt, Decrypt)
- Session termination
- Master Key Rotation
- Getting stats
- Error Handling

PSP Session Establishment Flow



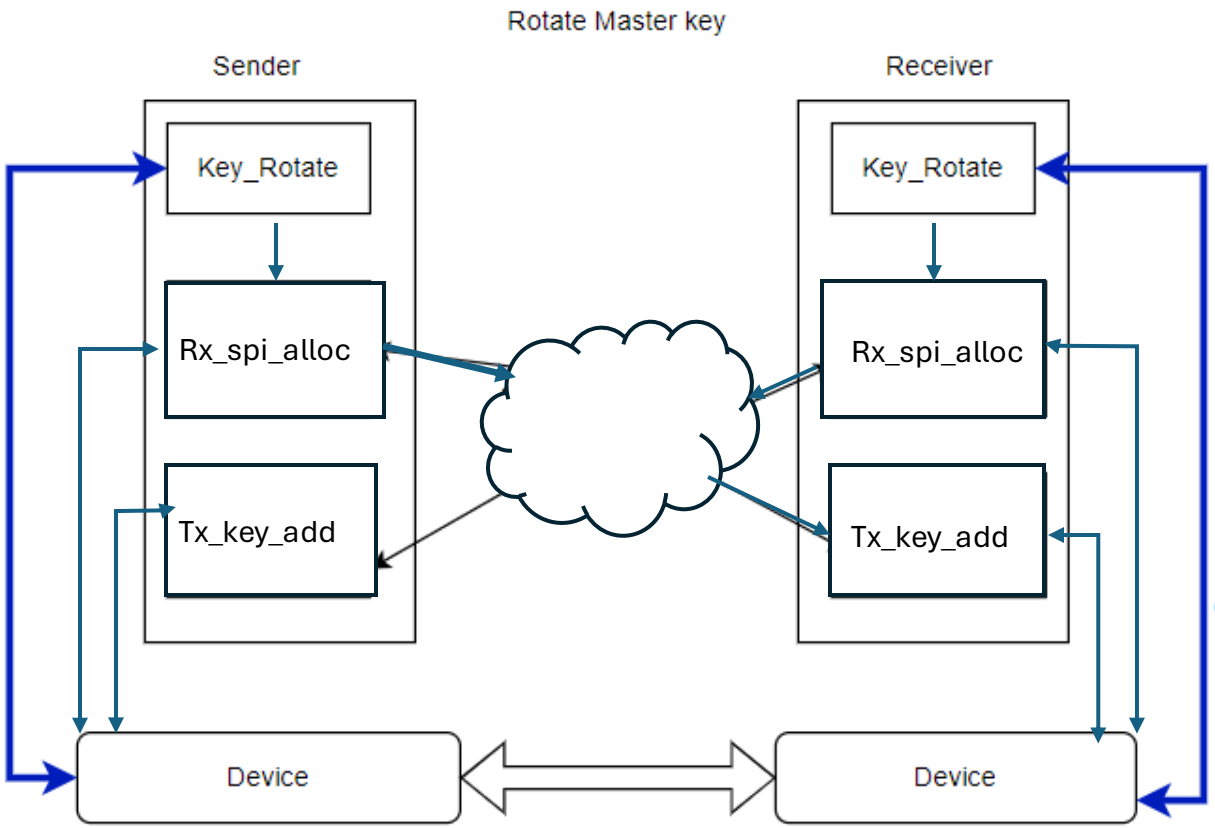
PSP Session Termination



Master Key Rotation

Session: PSP crypto Session

MK Timeout or
SPI rollover



MK Timeout or
SPI rollover

Enhancements needed in the kernel hooks

Current	Extensions that are needed
Single key add/del/get	Bulk add/del/get ndo_ops as the scale is very large and some prefetch can help as well to get better performance.
IKE application triggers Master key rotate	Allow Device to trigger master key rotate events as well based on time or SPI overflow.
Driver adds PSP related headers	This to move into the kernel for all drivers to benefit.
set_config in driver	Can move up to kernel
Driver removes the PSP headers	This to move into the kernel for all drivers to benefit.

Future use with Confidential Compute

- No knowledge or changes in the driver code to work in a TVM.
- Device Control plane and Device changes to create a secure isolated environment for each TVM
  - PSP a good protocol of choice for this.
  - The driver to Device ad Device control plane designed with the above in mind.

Crypto protocols for HW offload

- kTLS a bad example for offload.

- Key points to keep in mind:
  - Avoid per flow state on device, have all info in the packet header on Rx.
  - Crypto offset, good/bad?
- Anti-Replay window?

Conclusion

- Patches to come soon for IDPF and kernel enhancements.
- Help with IETF WESPv2 Spec from HW offload perspective.