VPN Measurements on two host systems

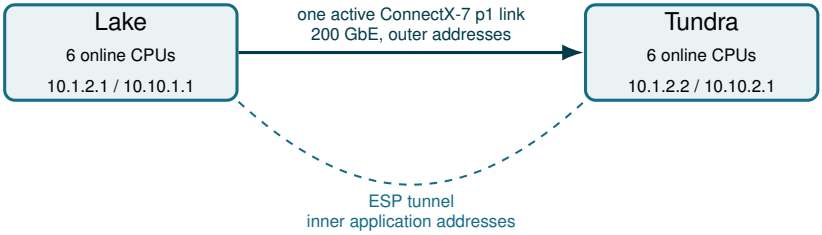
IPsec vs. WireGuard

(with updates during the workshop)

Paul Wouters - Hannes Tschofenig

IPsec Workshop 2026 – July 2026

Testbed



- Each adapter has two ports; only p1 is connected and measured.
- p0 is link-down; there is no link aggregation.
- SMT disabled on both systems.
- One NUMA node and CPUs 0–5 online.
- Applications use inner addresses for IPsec and WireGuard.

Test Overview

Test Phase	Nickname	Description
0	no VPN	UDP baseline, flow count, small packets, TCP RX rings
1	software XFRM	direct ESP tunnel, ESP-in-UDP/NAT-T, TCP $P = 1..8$
1N	anti-replay disabled	direct ESP tunnel, effective replay window 0, TCP $P = 1..8$
1R	replay window	tunnel replay windows 32–4096 packets
2	cloned Child SAs	one pinned process per flow, matching ports, $P = 1..8$
WG	WireGuard	one peer, same inner endpoints, TCP $P = 1..8$

No-VPN baseline: loss-free reference and scaling

Verified loss-free UDP reference

10.0 Gbit/s offered load

Large UDP, P1; three repetitions in both directions

0% datagram loss

0 UDP receive errors

0 physical NIC errors

Detailed UDP scaling matrix (three repetitions)

condition	Lake → Tundra	Tundra → Lake
large UDP, P7	43.0 / 7.2%	39.2 / 14.4%
IPv4 64 B, P6	1.18 / 6.0%	1.20 / 5.3%
IPv4 64 B, P8	0.87 / 30.7%	1.24 / 3.4%

Values are mean Gbit/s / mean loss; ranges across repetitions: P7 loss 2.1–12.0% and 9.1–21.8%; 64 B P8 varies strongly. High offered rates are therefore *not* loss-free link capacity. Receiver `rx_out_of_buffer` and/or UDP receive errors rise with loss; PHY errors remain zero.

10 Gbit/s is a verified loss-free reference, not a capacity limit. The scaling matrix locates a host receive-path loss boundary rather than physical link capacity.

Software XFRM: single-SA tunnel behaviour

Recorded TCP throughput in Gbit/s

Path	Lake → Tundra $P = 1$ / $P = 8$	Tundra → Lake $P = 1$ / $P = 8$
Software XFRM, tunnel / direct ESP	12.1 / 11.7	12.0 / 11.5
Software XFRM, ESP-in-UDP / NAT-T	9.54 / 8.93	9.70 / 8.93
WireGuard reference	8.45 / 8.51	7.17 / 7.36

Direct ESP

- IPsec `type=tunnel`; “direct ESP” means IP protocol 50, without an outer UDP encapsulation.
- Highest throughput among the encrypted paths.
- Higher parallelism causes TCP retransmits and XFRM failures.

ESP-in-UDP / NAT-T

- Still IPsec tunnel mode; only the outer encapsulation changes.
- Lower than direct ESP and kept as a separate condition.
- WireGuard is shown as a contextual tunnel reference.

Replay-Window Experiment

Change exactly one parameter:

- Windows: 32, 64, 128, 256, 512, 1024, 2048, and 4096 packets

Keep these parameters unchanged:

- tunnel mode/encapsulation, endpoints, direction, TCP duration, and $P = 1..8$
- CPU/IRQ placement and offered workload

Compare these results:

- TCP throughput, `Retr`, and sender `TcpRetransSegs`

Receiver and link counters

- XFRM inbound `failed`
- `rx_out_of_buffer`
- UDP socket errors
- physical NIC errors

Interpretation

Only an otherwise-identical repeated comparison—same effective SAs, workload, CPU/IRQ placement, and RSS mapping—can attribute a change to the replay-window size.

Why this experiment matters

Counters distinguish anti-replay drops from socket, RX, and general XFRM pressure.

Replay-window: RFC and implementation range

Single-SA tunnel, P=6; one 10-second run per window

window	Lake → Tundra	Tundra → Lake	meaning
32	10.4 Gbit/s; SA `failed` +55026	10.7 Gbit/s; +66942	RFC 4303 minimum
64	11.2 Gbit/s; +32916	11.0 Gbit/s; +41312	RFC-preferred default
128	11.9 Gbit/s; +29069	10.4 Gbit/s; +48627	Libreswan default
256	11.7 Gbit/s; +28149	11.5 Gbit/s; +38183	intermediate
512	12.4 Gbit/s; +18970	11.8 Gbit/s; +17594	intermediate

Interpretation

Every requested inbound XFRM window was validated on both hosts before traffic (bitmap lengths 1, 2, 4, 8, and 16 words respectively). At this load, the RFC-/implementation-sized windows show tens of thousands of inbound-SA discards. The direction-specific variation at 128 means that this single-run series establishes a trend, not a precise ranking of nearby window sizes.

Replay window: 1024–4096 (Phase 1R)

Single-SA tunnel, P=6; one 10-second run per window

window	Lake → Tundra	Tundra → Lake
1024	11.8 Gbit/s; SA `failed` +16987	11.7 Gbit/s; +23226
2048	11.8 Gbit/s; SA `failed` +7174	11.6 Gbit/s; +11709
4096	12.0 Gbit/s; SA `failed` +553	12.1 Gbit/s; +1802

Interpretation

Both active inbound tunnel SAs were validated before traffic with the requested effective window. `failed` is the inbound-SA discard delta, not failed XFRM processing as a whole. At P6, discards and TCP retransmits fall across the larger-window range (Lake → Tundra Retr: 18512, 8010, 1227), while throughput stays near 12 Gbit/s. Together with the 32–512 measurements, this supports an anti-replay contribution to drops; receive-path pressure remains a limit. One run per window without CPU/IRQ pinning or RSS/XFRM correlation is not a controlled capacity comparison.

Phase 1N: anti-replay disabled (Replay Window = 0)

Separate, validated control condition

Direct ESP tunnel, software XFRM, AES-GCM-256, and the normal bidirectional TCP matrix ($P = 1..8$). Before traffic, the inbound XFRM SAs on both Lake and Tundra were validated with effective `replay_window 0`; this is anti-replay disabled, not a small replay window.

Direction	P=1	P=4	P=8
Lake → Tundra	13.6 Gbit/s	12.5 Gbit/s	12.0 Gbit/s
Tundra → Lake	12.8 Gbit/s	11.5 Gbit/s	10.5 Gbit/s

What disappeared

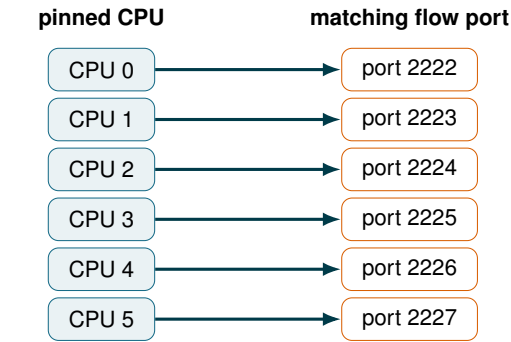
Receiver XFRM `failed` stayed at zero for every P and direction. This confirms that the test really bypassed anti-replay rejection.

What remained

From P=3, TCP retransmits and receiver `rx_out_of_buffer` rose together. Thus these losses are not caused by the replay check in this run.

This is a separate Phase-1N control, not a paired Replay-Window-1024 comparison: CPU/IRQ placement was not synchronized with the 1R runs. It supports causal diagnosis, not a numerical claim that disabling anti-replay improves throughput.

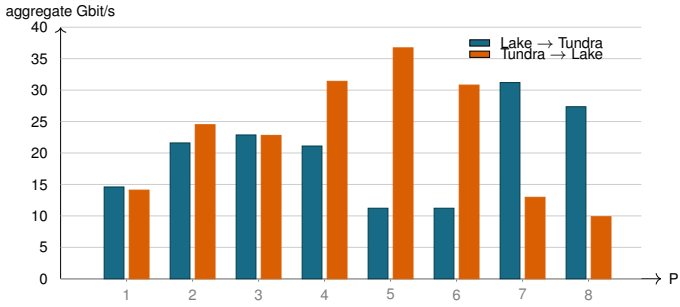
What `percpu_pN` means



one client process and one server process per flow
same port identifies the flow at both endpoints

- `percpu_p6` means six independent flows, not one `iperf3 -P 6` process.
- Each process is pinned with `taskset` or `numactl`.
- ‘P=7‘ and ‘P=8‘ reuse CPUs because only CPUs 0–5 are online.
- The mapping tests whether distinct flows can activate distinct Child SAs.

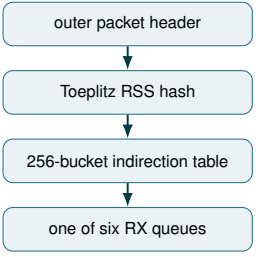
Pinned per-CPU Tests: controlled bidirectional repeat



Interpretation

One valid controlled run, not an average. Fresh sender-initiated SAs create different random SPI sets in the two directions; the NIC Toeplitz hash therefore produces different RSS collisions. Ordinary hardware RSS maps the six ESP SPIs to only four queues in either direction, so queue/IRQ/XFRM load remains direction-dependent despite symmetric application placement. $P = 7$ and $P = 8$ also reuse application CPUs and SAs, so they are oversubscription points rather than seven- or eight-CPU scaling results.

Hardware RSS and IPsec



Lake and Tundra: `mlx5_core`, six RX rings, Toeplitz RSS, hardware hashing enabled.

Direct ESP

Different ESP SPIs empirically reach different queues, although the mlx5 driver does not expose an `esp4` hash-field profile via `ethtool`. Different SPIs can still collide on one of six queues.

ESP-in-UDP / NAT-T

The visible RSS flow is the outer UDP tuple: source/destination IP address and UDP port. The encrypted inner TCP tuple is not visible; cloned SAs sharing UDP endpoints therefore share one hardware RSS flow.

Control boundary

`ntuple-filters` can be enabled, but mlx5 rejects an explicit `esp4 SPI` $\rightarrow$ `queue` rule. We can observe and select a favorable Direct-ESP SPI set, not program a deterministic SPI-to-queue map.

NAT-T: why per-SA RSS does not emerge automatically

What the NIC sees in the current single-IKE-SA setup

Inner flows differ by TCP port, but their headers are encrypted. All cloned Child SAs share one IKE/NAT-T socket and therefore normally expose the same outer tuple to hardware `udp4`-RSS:

`10.10.1.1:<TCP port>` → `10.10.2.1:<TCP port>` $\xrightarrow{\text{encrypt}}$ `10.1.2.1:4500` → `10.1.2.2:4500/UDP`

$\xrightarrow{\text{RSS}}$ one RX queue

Ways to add RSS entropy

- Multiple outer IP addresses and separate IKE/IPsec connections: ordinary NIC-RSS sees different IP tuples, but this is multi-tunnel sharding, not one per-resource IKE-SA.
- UDP multiport draft: one ephemeral UDP source port per resource-SA retains one IKE-SA and supplies UDP-port entropy. It requires coordinated daemon and XFRM support; not available in the current Libreswan setup.

What we can test today

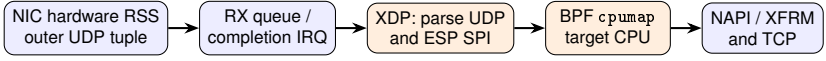
- XDP/eBPF can read the visible ESP SPI behind UDP and redirect to a chosen CPU. This is software steering *after* the NIC has selected its queue and IRQ.
- SPI-aware ESP-in-UDP hardware RSS would be ideal, but is not supported/proven on Lake and Tundra; normal `udp4`-RSS does not hash the SPI.

The current `ENCAPSULATION=yes` test uses NAT-T encapsulation without an intervening NAT gateway; the RSS limitation already applies because the outer UDP endpoints are shared.

NAT-T: XDP/eBPF SPI-to-CPU software steering

Purpose and measured setup

Six cloned NAT-T Child SAs ($P = 6$, AES-GCM, replay window 1024) used the same outer UDP endpoints. We measured the same SA set twice with controlled completion-IRQ affinity: once without XDP and once with a BPF map from each inbound ESP SPI to a selected CPU.



What the test validated

- Both baseline and steered runs completed successfully.
- XFRM CPU validation passed: every mapped SPI was observed on its configured `cpumap` CPU.
- The BPF map and per-CPU counters were saved with the run.

What it does *not* prove

- It does not alter hardware RSS, the original RX queue, or its NIC IRQ: all packets first share the outer UDP-RSS path.
- It is software steering with extra copy/redirect work, not SPI-aware NIC RSS; it is not a hardware-RSS capacity result.

NAT-T XDP steering: recorded P6 result

One paired 3-second P6 run, same NAT-T SA lifecycle and IRQ control

The baseline and XDP modes completed. Throughput below is the sum of six pinned `iperf3 -P 1` flows, using the receiver totals.

direction	no XDP	XDP `cpumap`	change	inbound SPI / target CPUs
Lake → Tundra	12.76	13.34	+4.5%	6 SPIs / CPUs 0–5
Tundra → Lake	13.01	6.14	−52.8%	1 SPI / CPU 0

Throughput in Gbit/s. Every configured SPI-to-CPU mapping that existed in the run was confirmed by the XFRM eBPF probe (100% match).

Why the directions differ

In Lake → Tundra, the receiver had six distinct inbound SPIs and XDP distributed them across six target CPUs. In the reverse direction, the run exposed one inbound SPI; all six flows were consequently redirected to CPU 0.

Correct conclusion

This run proves the XDP-to-XFRM CPU handoff, not a symmetric throughput improvement. It retains the shared outer UDP RX queue and adds redirect work. A capacity claim needs fresh, six-distinct-SPI SAs in *both* directions and repeated, longer measurements.

Phase 2 correlation: direct ESP

Controlled experiment

Six isolated flows, pinned to application CPUs 0–5, were measured in both directions with three repetitions. Each direction used a fresh IPsec lifecycle initiated by its bulk-data sender; completion IRQs were pinned explicitly. Per-SA, queue, IRQ/softirq, and XFRM eBPF samples were correlated for every flow.

Direction	outbound SAs	RX queues	across repetitions
Lake → Tundra	6 (one per app CPU)	4: rx1,2,3,5	6/6 stable
Tundra → Lake	6 (one per app CPU)	4: rx0,3,4,5	6/6 stable

Interpretation

Valid repeat: all 36 flow samples are `CORRELATED`; every group is stable across three repetitions. The former shared-SA result in the reverse direction was a lifecycle/acquisition artifact, not an inherent direction property. Both senders now create six stable SAs, but ordinary RSS still produces two collisions per direction; a full one-SA/one-queue mapping is not guaranteed.

Direct ESP: effect of `irqbalance`

Baseline: `irqbalance` active

- Completion-IRQ affinities changed during the run on both hosts.
- Only 7 of 12 bidirectional CPU groups were stable.
- The receive-side CPU can move although the SA and flow remain stable.

Controlled: explicit IRQ affinity

- No IRQ-affinity change occurred during the run.
- All 12 bidirectional CPU groups were stable across three repetitions.
- Queue-matched IRQ CPU and XFRM lookup CPU were deterministic.

Interpretation

`irqbalance` explains the changing receive-side CPU in the baseline. IRQ pinning fixes where a selected RX queue is processed; it neither selects the outbound SA nor prevents RSS collisions between different ESP SPIs.

Direct ESP: selecting a collision-free hardware-RSS set

Separate Fixed-SPI-RSS experiment: Lake → Tundra

The complete six-SA set was rebuilt until its random ESP SPIs covered all six hardware queues. RSS key and indirection table were unchanged; XDP only observed the already selected queue and returned `XDP_PASS`.

selection result	value	meaning
SA builds required	13	new random SPI set per build
selected SAs / RX queues	6 / 6	no queue collision in this set
XDP during workloads	off	normal hardware RSS measurement

P6 reference workload, one 3-second repetition

UDP at 20 Gbit/s offered: 19.98 Gbit/s received, 0% loss. TCP P6: 13.75 Gbit/s, zero retransmits. The separate P3 CPU-isolation condition is shown on the next slide.

This proves that 6/6 Direct-ESP RSS is attainable but not programmable. The run has no XFRM eBPF trace and is not a bidirectional throughput comparison; it cannot prove a one-SA-per-XFRM-CPU mapping.

Fixed SPI RSS: P3 CPU-isolation condition

Question

Once six Direct-ESP SAs occupy six different hardware queues, does separating application work from receive IRQ/NAPI work reduce CPU contention?

condition	iperf3 client/server CPUs	active RX IRQ/NAPI CPUs
P6 reference	0, 1, 2, 3, 4, 5	same CPUs: shared resources
P3 isolation	3, 4, 5	0, 1, 2: disjoint from applications

Observed Lake → Tundra

- UDP P3: 19.44 Gbit/s at 20 Gbit/s offered; 0% loss.
- TCP P3: 29.86 Gbit/s; zero retransmits.
- TCP P6 reference: 13.75 Gbit/s; zero retransmits.

Interpretation and limit

The receiver samples show NET_RX/softirq load on CPUs 0–2 while the P3 applications run on 3–5. The TCP difference is evidence for application/network-path contention, but this was one short repetition from the pre-XFRM-trace runner; it is not a final capacity claim.

Phase 2 correlation: ESP-in-UDP / NAT-T

Valid controlled run

Six isolated CPU-pinned flows in both directions, with three repetitions and explicit completion-IRQ affinity. The eBPF probe observes the parsed SPI at `xfrm_input_state_lookup`.

Direction	outbound SAs	RX queue	IRQ / XFRM CPU
Lake → Tundra	6 (one per app CPU)	`rx5`	5 / 5
Tundra → Lake	1 (shared)	`rx0`	0 / 0

Interpretation

Dominant TX/RX SPI, per-SA bytes, RX queue, queue-matched IRQ CPU, and XFRM lookup CPU all reached 100%. Unlike direct ESP, six distinct Lake outbound SPIs did not spread over multiple queues. This is consistent with RSS hashing the common outer UDP 5-tuple, not the encapsulated SPI or encrypted inner TCP tuple.

NIC crypto offload: paired Direct-ESP comparison

Controlled, validated condition

Direct ESP in `type=tunnel`, AES-GCM-256, replay window 1024, 10 seconds, and $P = 1..8$. The same matrix ran first with `nic-offload=no`, then with freshly created SAs and `nic-offload=crypto`.

direction	P	software	crypto	gain
Lake → Tundra	1	12.5	19.8	58%
	6	11.7	24.1	106%
	8	12.2	24.0	97%
Tundra → Lake	1	12.2	19.4	59%
	6	12.0	23.8	98%
	8	11.6	25.6	121%

Data-path evidence

Software mode had no offloaded XFRM states and no NIC-IPsec counter deltas. Crypto mode had inbound and outbound offload states on both hosts; matching NIC RX/TX IPsec counters increased and IPsec NIC drop counters stayed zero.

Interpretation

Crypto offload approximately doubles throughput and sharply reduces TCP retransmissions. It offloads ESP cryptography only: XFRM, TCP, queue/IRQ/NAPI, and other host-path work remain in software. This is not packet offload.

WireGuard Comparison

Path	Lake → Tundra	Tundra → Lake	Observation
No-VPN reference	–	–	baseline is substantially higher
Software XFRM, direct ESP	≈12.1	≈12.0	Throughput is highly sensitive to receiver-side processing pressure.
ChaCha ESP	5.99 / 5.41	5.47 / 5.24	P=1 / P=8; separate software-cipher run.
Software XFRM, ESP/UDP	≈8.9	≈8.9	lower, separate NAT-T mode
WireGuard, $P = 1$	8.45	7.17	stable across $P = 1..8$

Scope of the comparison

WireGuard and IPsec are compared as tunnel mechanisms using inner application addresses. For application payloads, software ESP used `AES-GCM-16-256`, while WireGuard used its fixed `ChaCha20-Poly1305`. The IKE suite is control-plane protection only. Direct ESP and ESP-in-UDP remain separate wire-format conditions.

ChaCha20-Poly1305 ESP: recorded results

Controlled condition

- ‘type=tunnel‘, direct ESP, ‘nic-offload=no‘
- replay window 1024; TCP for $P = 1..8$
- 10 seconds per direction and stream count
- six online CPUs on each host
- both CPUs expose the AES CPU flag; AES-GCM can use AES-NI
- negotiated ESP: `CHACHA20_POLY1305`
- negotiated IKE: `AES_GCM_16_256-HMAC_SHA2_512-DH19`

Throughput: AES-GCM versus ChaCha20-Poly1305

	Lake → Tundra		Tundra → Lake	
P	AES-GCM	ChaCha	AES-GCM	ChaCha
1	12.1	5.99	12.0	5.47
4	12.0	5.79	12.5	5.80
6	11.8	5.33	12.3	5.50
8	11.7	5.41	11.5	5.24

ChaCha reaches about half the AES-GCM throughput in this pair of runs. Both algorithms degrade as P increases; the ChaCha run also shows increasing TCP retransmits and XFRM failures, with zero UDP socket and physical NIC error deltas.

ChaCha20-Poly1305 ESP: Interpretation

- The AES-GCM values are the earlier direct-ESP matrix with the same tunnel mode, replay window, and $P = 1..8$ workload.
- The runs were not simultaneous, so the table demonstrates a large observed cipher/path difference but does not by itself isolate CPU frequency, system state, or scheduling effects.
- Both CPUs advertise AES support, so the AES-GCM path can use AES-NI; this is consistent with its large performance advantage.
- On CPUs without AES acceleration, ChaCha20-Poly1305 may be competitive or faster.
- For this testbed, however, the result supports AES-GCM as the preferred IPsec cipher.

Conclusions

- The 10 Gbit/s UDP baseline is loss-free; the physical link is not the immediate limit.
- Observed bottlenecks are in the host/XFRM/TCP receive path; physical-NIC error counters do not explain them.
- With validated replay window 0, XFRM `failed` drops disappear but TCP retransmits and `rx_out_of_buffer` can remain: anti-replay contributes, but is not the sole limit.
- State-validated Phase-1R: 32–128 packet windows (RFC minimum, RFC-preferred, and Libreswan default) show tens of thousands of inbound-SA discards at P6. From 512 through 4096, discards and retransmits fall substantially while throughput remains near 12 Gbit/s. A repeated controlled replay-window comparison remains future work.
- Validated NIC crypto offload approximately doubles Direct-ESP AES-GCM throughput; it offloads crypto, not the complete IPsec packet path.
- With fresh sender-initiated lifecycles, both Lake and Tundra select six stable outbound Child SAs for six pinned flows.
- A valid controlled repeat has 36/36 correlated and all 12 stable flow groups. Normal Direct-ESP RSS uses only 4/6 queues; a separate SA search found a random 6/6 set, not programmable SPI steering.
- With NAT-T, Child SAs of one IKE-SA share the outer UDP tuple and thus one normal `udp4`-RSS queue. XDP can steer after RX-queue selection; SPI-aware hardware RSS is not proven on this NIC.