Michael Pfeiffer • Michael Rossberg

ABOUT THE CHALLENGES OF RUNNING SOFTWARE-DEFINED ESP IN DATA CENTERS

Motivation

- ESP + IKE:
  - **The** standard for network layer VPNs
  - Around for two decades, proven in theory and practice
- **But:**
  - Data rates evolved significantly
    - Fast Ethernet (1995) → 100/200 GbE
  - IPsec's use cases evolved significantly
    - Scaling data-centers (switched → routed networks) & "Zero-trust"
    - Necessary: Multicast (VXLAN!), QoS, Jumbo frames
  - Hardware evolved significantly
    - CPUs: Multicore & SIMD
    - NICs: Multi-Queueing, Receive-Side Scaling
    - ASICs: Increased Parallelism

ESP & Hardware Parallelism

- Scenario: High traffic SA between two VPN gateways (> 10 Gbit/s)
- Must use multiple cores / queues, but:
  - Cannot synchronize sequence counters and replay windows fast enough
  - Even without replay protection: No field for decrypting gateway to RSS upon → reordering disturbs TCP et al.
- Only possibility: Multiple SAs between two gateways, but:
  - Data plane properties (e.g. number of cores) pushed into control plane → Multiple IKE exchanges & DPD
  - Complicates configuration and monitoring
  - Problems gets worse with multicast and QoS (hang on…)

→ ESP should support parallel processing by design

ESP & Modern Hardware Features

- Alignment:
  - ESP header 4 or 8 byte aligned; ESP ICV 4 byte aligned
  - Modern SIMD instructions like more (SSE/Neon: 16 byte, AVX: 32 byte)
  - Header alignment can be influenced in implementations by manipulating headroom in packet buffers
  - But: Trailer position depends on packet length
- Jumbo frames & Fragmentation:
  - NICs place them into multiple (chained) packet buffers
  - Trailer may end up split among two segments
  - Costly copy operation required to restore
- AES-NI et. al. aggravate problems in complex packet handling:
  - Focus shifts from actual crypto to packet handling

→ ESP’s header/trailer structure hinders high-performance implementations

ESP & Local Area Networks

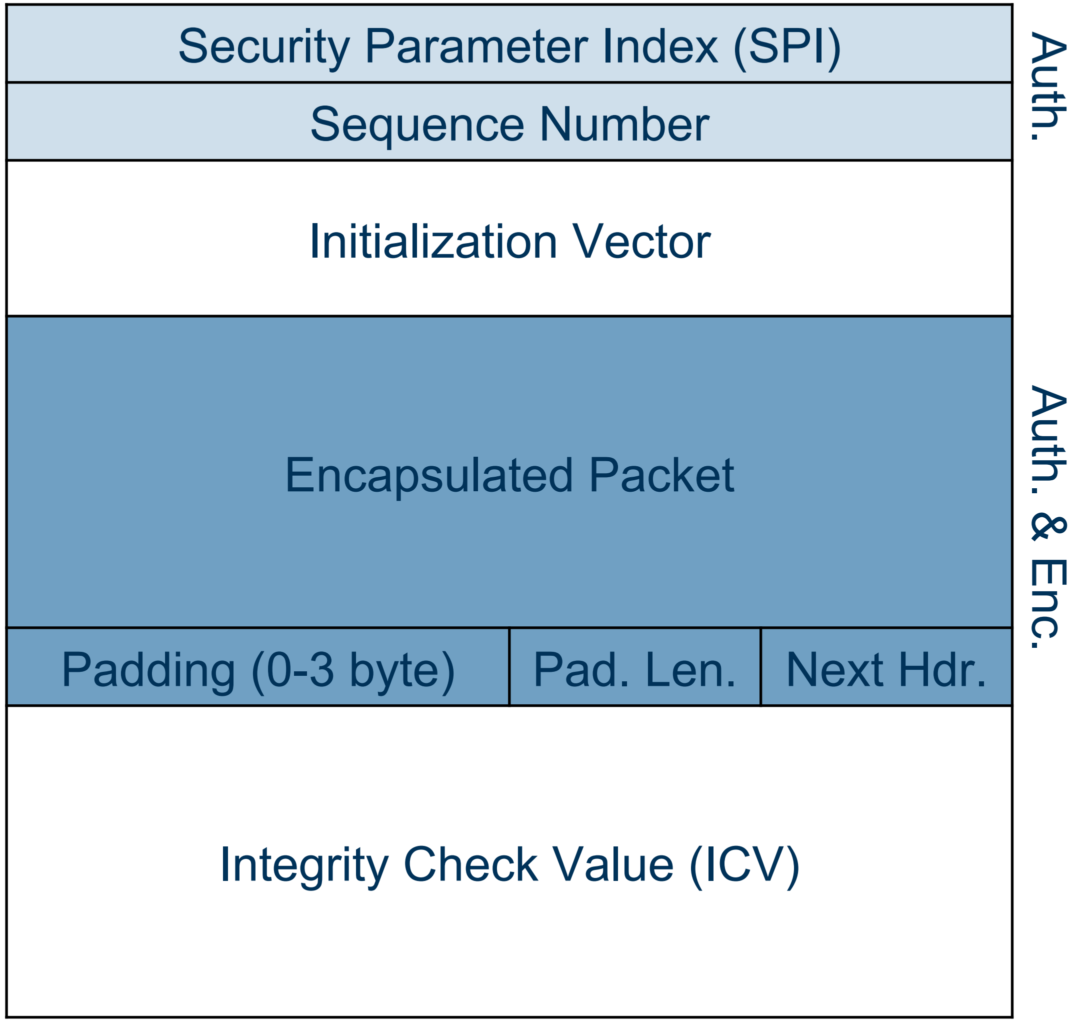
- Multicast (with multiple senders):
  - Replay protection does not work (sequence number synchronization)
  - 1 SA per sender? → „1 affects n“ scalability issue
  - ESNs transmit only least-significant 32 bit → Problem for receivers joining late
  - No built-in mechanism to avoid IV reusage (fatal in GCM)
- QoS:
  - QoS flags can be copied to outer header
  - But: Prioritized traffic leads to windows advancing before low-priority traffic arrives
  - Huge replay windows? → Performance issues
  - 1 SA per QoS class → See parallelism, multiplicates number of SAs!

→ ESP makes Multicast and QoS hard to use

Proposed Protocol Changes

- General approach:
  - Change ESP as little as possible
    → Keep existing security properties
  - IKEv2 not changed
  - Assume a modern AEAD cipher,
    i.e. AES-GCM → Conforms to RFC 8221
- Changes in:
  - Replay Protection
  - IVs
  - Trailer
- Working title: VPE
  - Vector Packet Encapsulation
  - Subject to debate ;-)
  - For today: *Focus on tunnel mode*

Current ESP packet format (AES-GCM):



Replay Protection

- Central idea: 1:N mapping between SA and replay windows
- Unicast:
  - *Window ID* (16 bit) allows steering traffic to distinct replay windows within a single SA
  - Simple case: Encrypting gateway inserts CPU core ID, receiving gateway performs RSS/Flow Steering based on Window ID
  - No need to coordinate number of cores between sender and receiver, but receivers must be able to track more than one replay window per core
  - Sequence counter/replay windows can reside in core-local memory → Solves parallelism
  - Some *Window ID* bits can be used for traffic classes → Solves QoS
- Multicast (with group key):
  - *Sender ID* (16 bit) unique to each sender (obtained from, e.g., certificate)
  - Replay window per (*Sender ID, Window ID)*
  - Distinct senders can increment their sequence numbers without coordination
  - Can be combined with parallelism and QoS

Replay Protection

- Keep protocol complexity low:
  - *SenderID* set to fixed value for unicast SAs
  - Implementations can always use (*Sender ID, Window ID)* as 32 bit lookup key
- RSS for encrypted traffic
  - Steer traffic based on (*Sender ID, Window ID)*
  - No need to perform RSS on SPI anymore
- Sequence Numbering:
  - Full ESN transmitted in each packet

Initialization Vectors

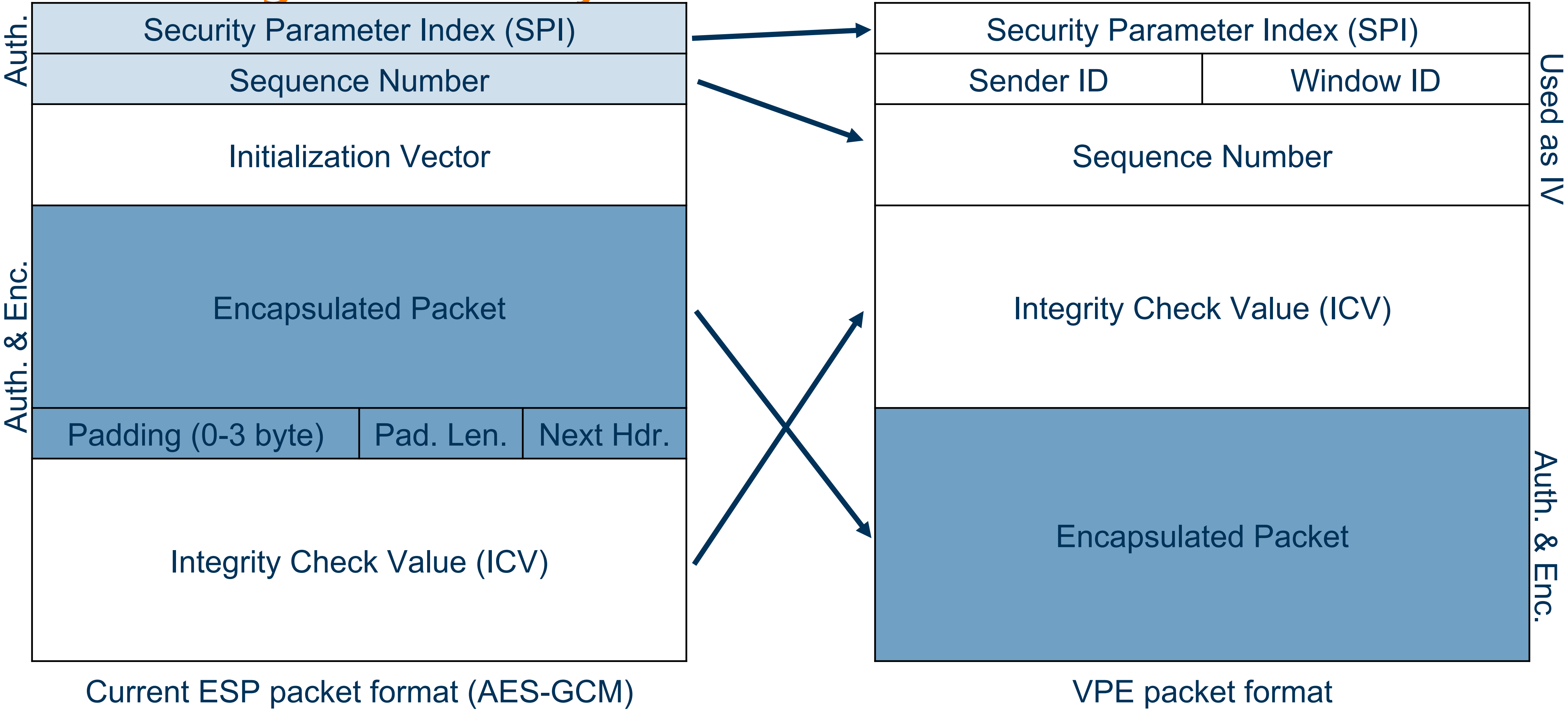
- For a given SA:
  - IVs must not repeat
  - Synchronizing IVs across cores is too costly (just as sequence numbers)
  - Distinct number spaces for each core and sender (multicast)
- Approach:
  - Use (*SenderID, WindowID, ESN*) as IV (96 bit)
  - In spirit of RFC 8750 and IEEE 802.1AE
  - Unique by design
  - No salt used anymore (no need for IVs to be secret or random)
- Place ESN directly after (*SenderID, WindowID*):
  - Reading 12 bytes from the beginning of sender *SenderID* returns IV

Trailer

- Padding:
  - Modern AEAD ciphers do not require padding
  - Proposal: Drop explicit padding entirely
  - Implicit padding still possible in tunnel mode (IP header contains length) → No impact on traffic flow confidentiality
- Next header field:
  - Superfluous in tunnel mode (v4/v6 discernible by first nibble)
  - Proposal: Drop next header
  - Transport mode would require different approach here… (not covered in this talk)
- Integrity Check Value:
  - Moved to header → No danger of being placed in two segment buffers
  - Aligned to 16 byte (respective to the start of the header) → Independent from packet length

→ Trailer can be dropped entirely

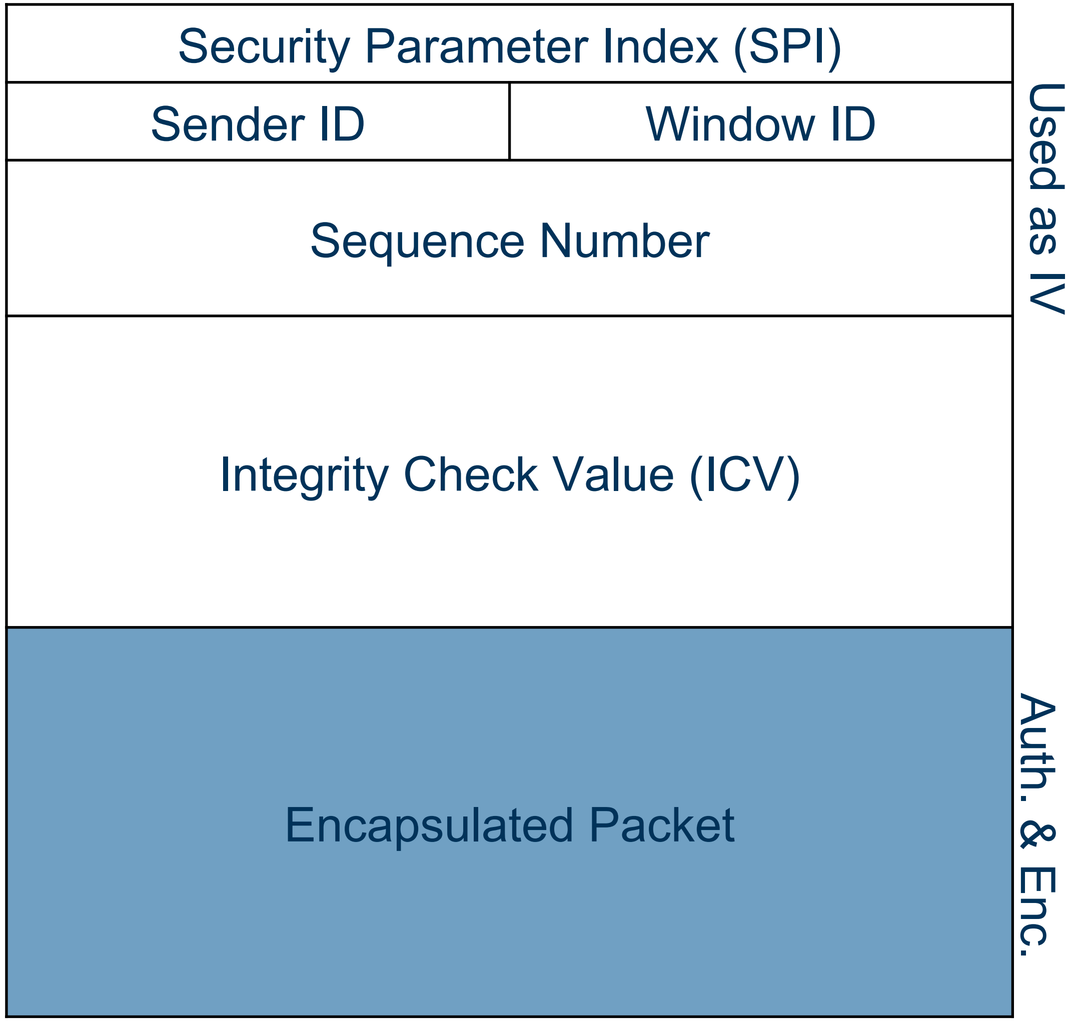
Resulting Packet Layout



Resulting Packet Layout

- No need to authenticate
  - Sender ID, Window ID, or Sequence Number → Modification changes IV → ICV mismatch
  - SPI → Modification routes packet to wrong or invalid cryptographic context
  - ICV → Modification causes ICV mismatch (obviously)

→ No AAD required



VPE packet format

Evaluation

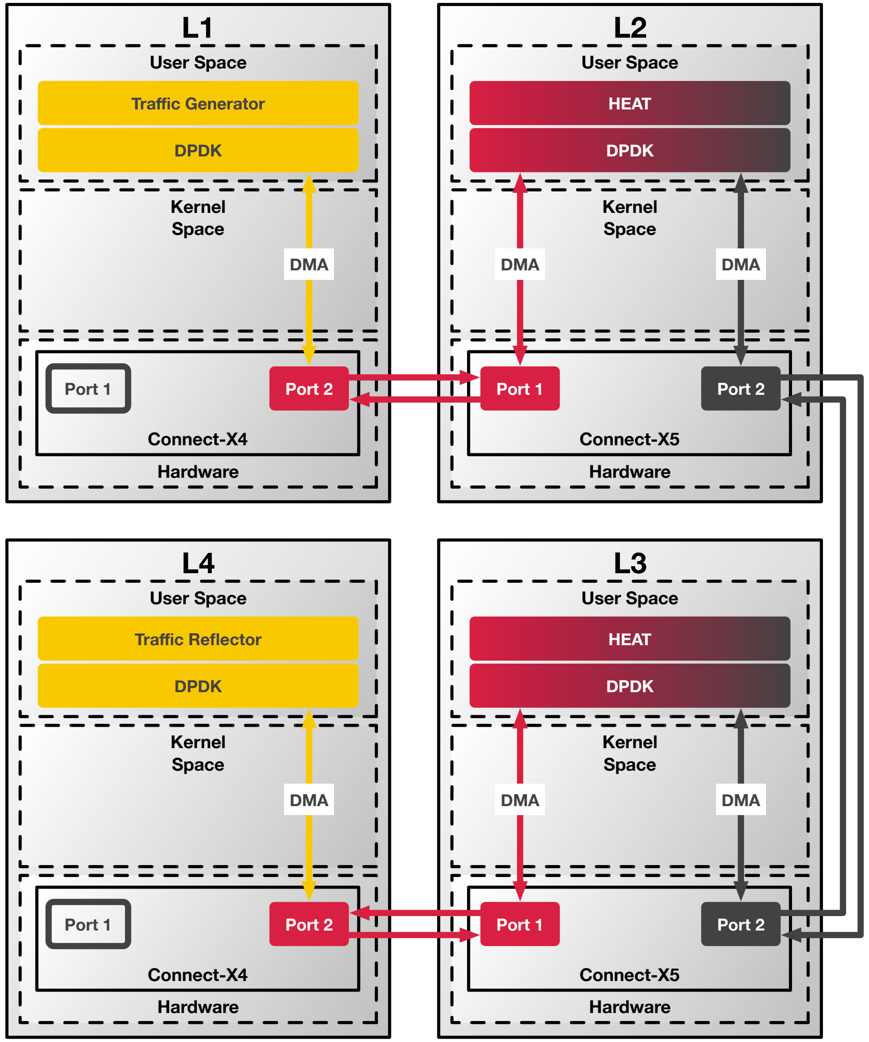
- Security discussion:
  - Entire encapsulated packet still covered by AEAD
  - Authentication of SPI & sequence number not required
  - TFC may require regular RSS rekeying (if TFC padding not used)
  - Improved security due to replay protection in multicast environments & IV reuse less likely
- Implementation Prototype:
  - Highly optimized C++ application
  - DPDK for NIC access
  - intel-ipsec-mb for vectorized crypto
  - Tons of template inlining and intrinsics for speed ;-)
  - Three modes:
    - Simple & non-parallel ESP
    - Quite complex, but parallel ESP
    - Simple & parallel VPE

Performance Evaluation

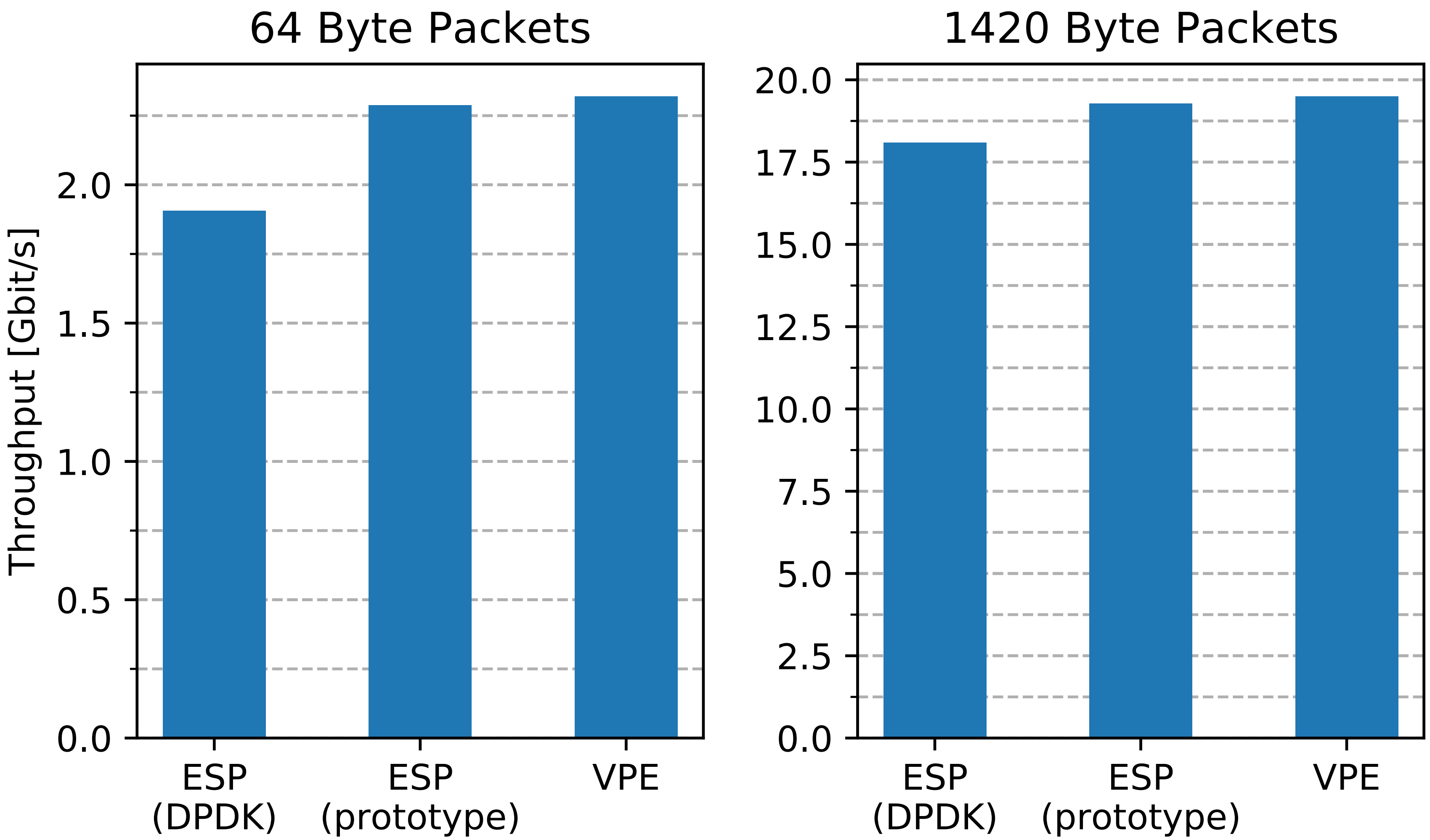
- Testbed consisting of 4 boxes:
  - Traffic Generator/Receiver
  - Gateway 1
  - Gateway 2
  - Traffic Reflector
- Decent, but general-purpose hardware:
  - Two-Socket Broadwell Xeons
  - Connect-X4/X5 NICs
- Measured throughput at receiver:

  → Without crypto headers (ESP, Outer IP)

  → Gateway handles same throughput in opposite direction!
- Baseline: DPDK's ipsec-secgw



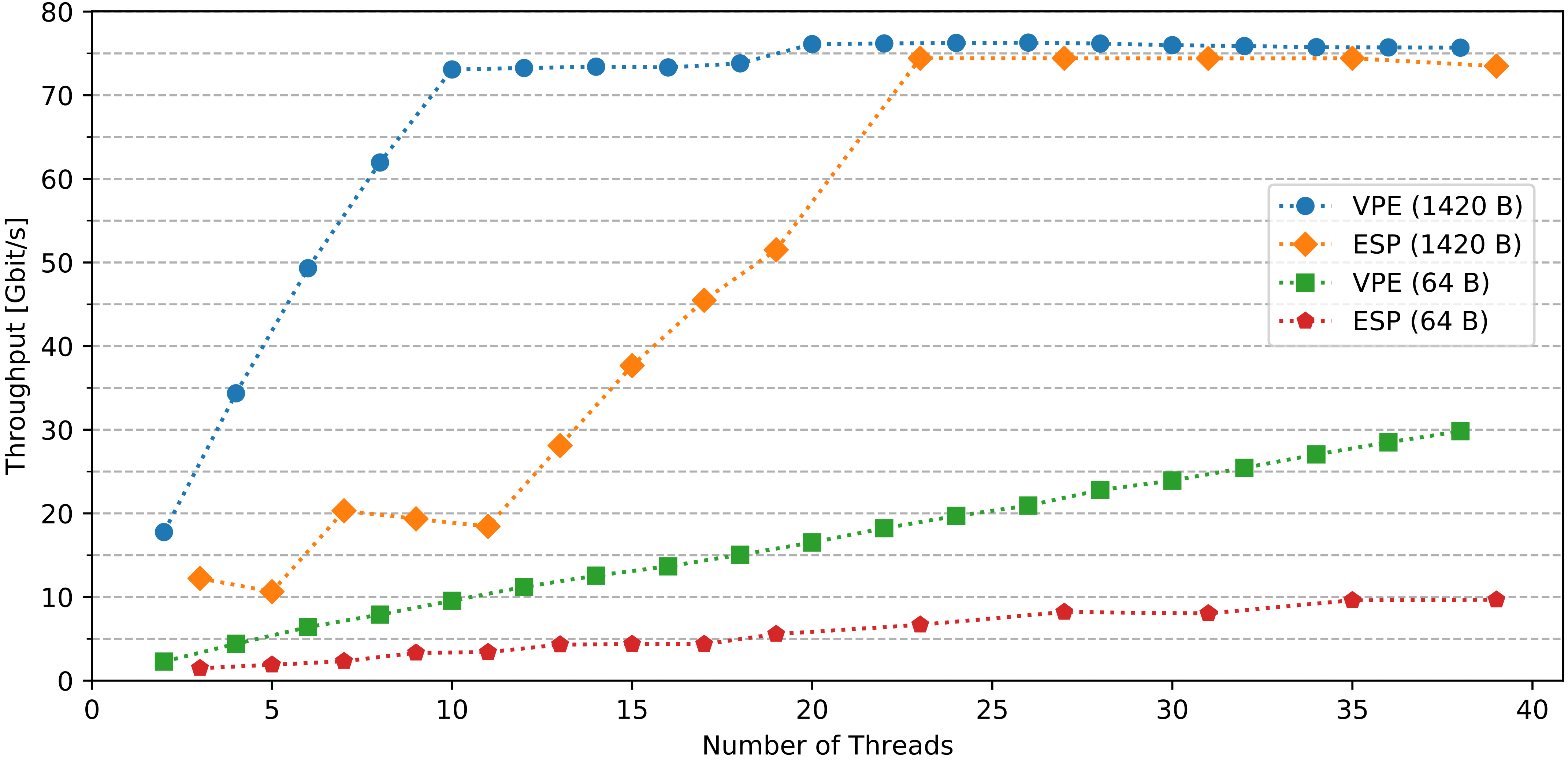
Single-Core Throughput



The SPIRIT of science

TECHNISCHE UNIVERSITÄT ILMENAU

Multi-Core Throughput



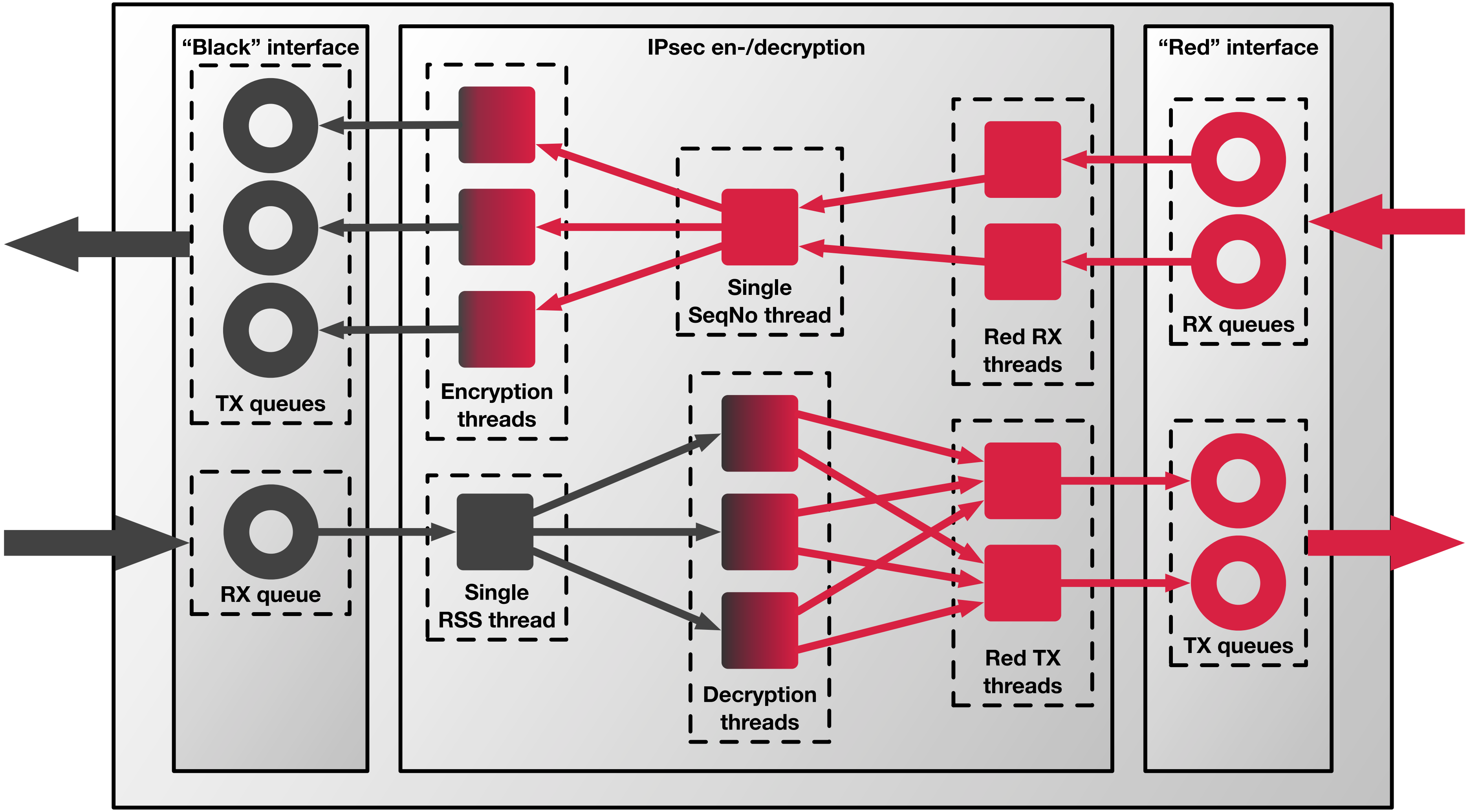
Conclusion

- Wrap-up: Presented VPE as modern companion to ESP with
  - N replay windows per SA to simplify
    - parallel processing,
    - multicast, and
    - QoS
  - Getting rid of the trailer makes software implementations faster/simpler
  - Implicit IVs to reduce risk of inadvertent reusage
- There will be paper with more details at this year's ARES conference
  - On https://www.tu-ilmenau.de/telematik/mitarbeiter/michael-rossberg/ once published
  - Or just email us for a preprint ;-)
  - Note: It describes a previous version of the protocol with the SPI after the sequence number
- Questions, comments, or angry rants?
  - michael.rossberg@tu-ilmenau.de
  - michael.pfeiffer@tu-ilmenau.de

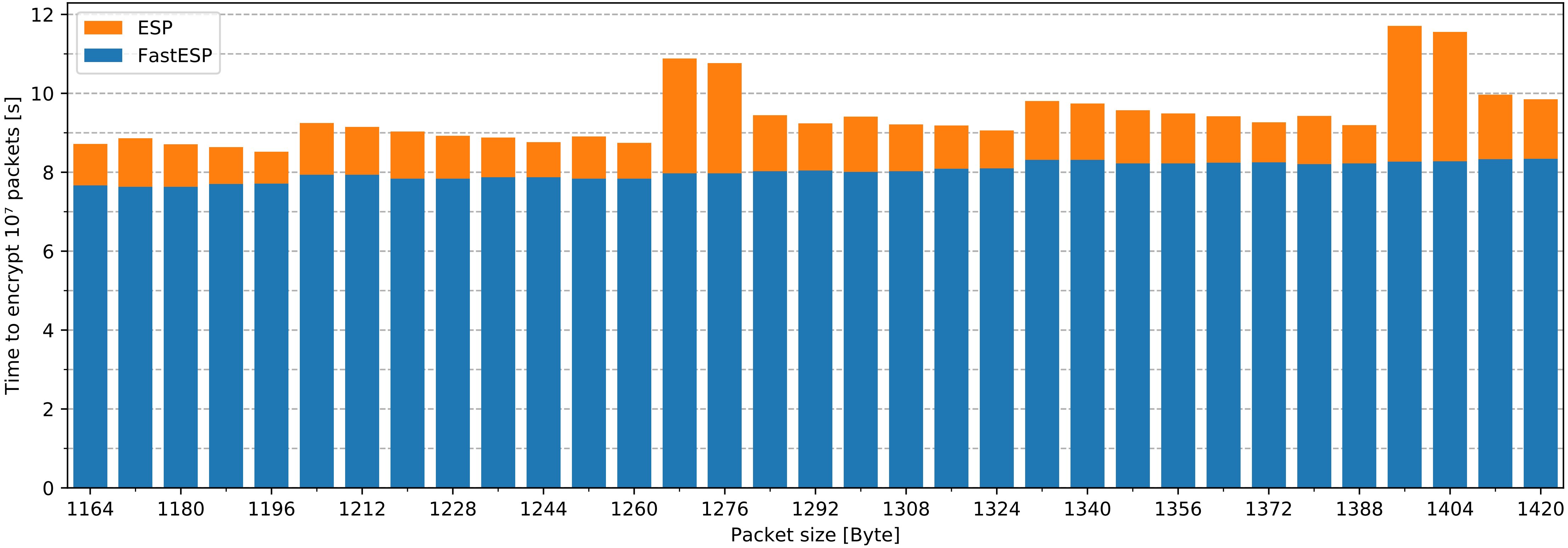
Michael Pfeiffer • Michael Rossberg

ABOUT THE CHALLENGES OF RUNNING SOFTWARE-DEFINED ESP IN DATA CENTERS

Parallel ESP processing: Threading model



Encryption Time vs. Packet Size



Encryption Time vs. Additional Headroom