SessIon LayEr SecuriTy Approach SIESTA

Robert Moskowitz
Verizon
November 7, 2013

Problem

- The present end-to-end application security context is tightly coupled with the underlying communication context. This is problematic for at least three reasons.
  1. Not flexible: when the underlying communication context changes, the application security context must change, too.
  2. Overhead associated with such coupling is prohibitively expensive over constrained networks (such as sensor or cellular networks).
  3. Probably most important: an attack on the communication context immediately effects the application security.

Goals

- This work aims at a solution to the above problems, with the objective of providing security context and security message format(s) for an application,
  - which is fully decoupled from the underlying communication methodology and is thus resilient to attacks on the communication context.
  - With that, the security context may need to have basic understanding of the communication context to be efficient with datagram overhead and communication synchronization issues (such as sequence window management), and so it is desirable that solution supports the "hooks" into the underlying protocol

Why yet another Security Layer?

- What is the Session Layer? TCP/IP does not have a Session Layer
  - From Wikipedia

"The session layer provides the mechanism for opening, closing and managing a session between end-user application processes, i.e., a semi-permanent dialogue. Communication sessions consist of requests and responses that occur between applications. Session-layer services are commonly used in application environments that make use of remote procedure calls (RPCs)."

  - So every communications application can be seen to have a Session Layer abstraction.

What is not needed

- Another Key Management Protocol (KMP)?
  - These are hard to do
  - Instead, work out how to run existing independent KMPs
    - e.g. IKEv2 and HIP
- Changes to existing security layers
  - They each do the job they were invented for

Thus the Goals are

- Make network attacks uninteresting in that they do not disturb the security state
- Allow for very conservative message overhead while accommodating high performance networks
- Put the intelligence in the (independent) KMP, not the secure messaging format(s)
- Provide for security state survivablity and longevity
- Support mutli-flow, mulitcast, multipath, non-TCP/IP, and just about any other network design

Who is the Audience

- SDN
  - Being discussed in ONF for OpenFlow
    - Multiflows and Survivablity big concerns
    - And thus ETSI NFV as well
- M2M
  - SMS, particularly satellite SMS
  - Multicast sensor nets
- Expect other interest

Security State Survivability

- Security state loss is a significant threat
  - Loss of state most commonly from reboot
    - Thus rare in sensors, but expected on servers
  - Significant time lost to detect state loss at peer
  - Significant energy spent in sending packets that will be dropped due to loss of state at peer
  - Implementations MAY maintain state in a secure store
    - SHOULD when significant cost to peer to recover
    - SHOULD refresh keys with peers for full recovery
    - If Sequence # maintained, the key refresh MAY be delayed/scheduled

The State Machine

- An application using Session Security will
  - Start – call the KMP with the session identities and get the SPIs, key hierarchies, and ciphersuite
  - Key refresh – on seq # consumption, call the KMP for fresh keys
  - Secure – secure the message
  - Reveal – decrypt and/or authenticate the message
  - Teardown – tear down the security context
  - State machine needs to handle responder Security setup and state loss recovery (e.g. after reboot)
    - Some method is needed to connect a start or rekey with new SPIs to the SSE application that will use it. This could be internal to the KMP or KMP run over the eventual application port before any messages.

The Plan

- Use lessons learned with ESP to design a Session Security Envelope
  - Move ESP up the stack and adjust accordingly
- Only provide for 'modern' ciphersuites
  - e.g. AES with CCM, CMAC, GCM, GMAC
- Multiple formats tuned to differing requirements
- What about Additional Authenticated Data headers like HTTP stuff?

More the Plan

- This is so simple it SHOULD be done in 3 months!
  - No 'working group', just do it and submit
    - So far only slide deck describing SSE
    - Already claimed one implementation on NETBSD
  - But not exclusive design team
  - Mailing list
    - siesta@ietf.org
- 3? RFCS
  - SSE, IKEv2 support, HIP support
    - Others like Kerberos?

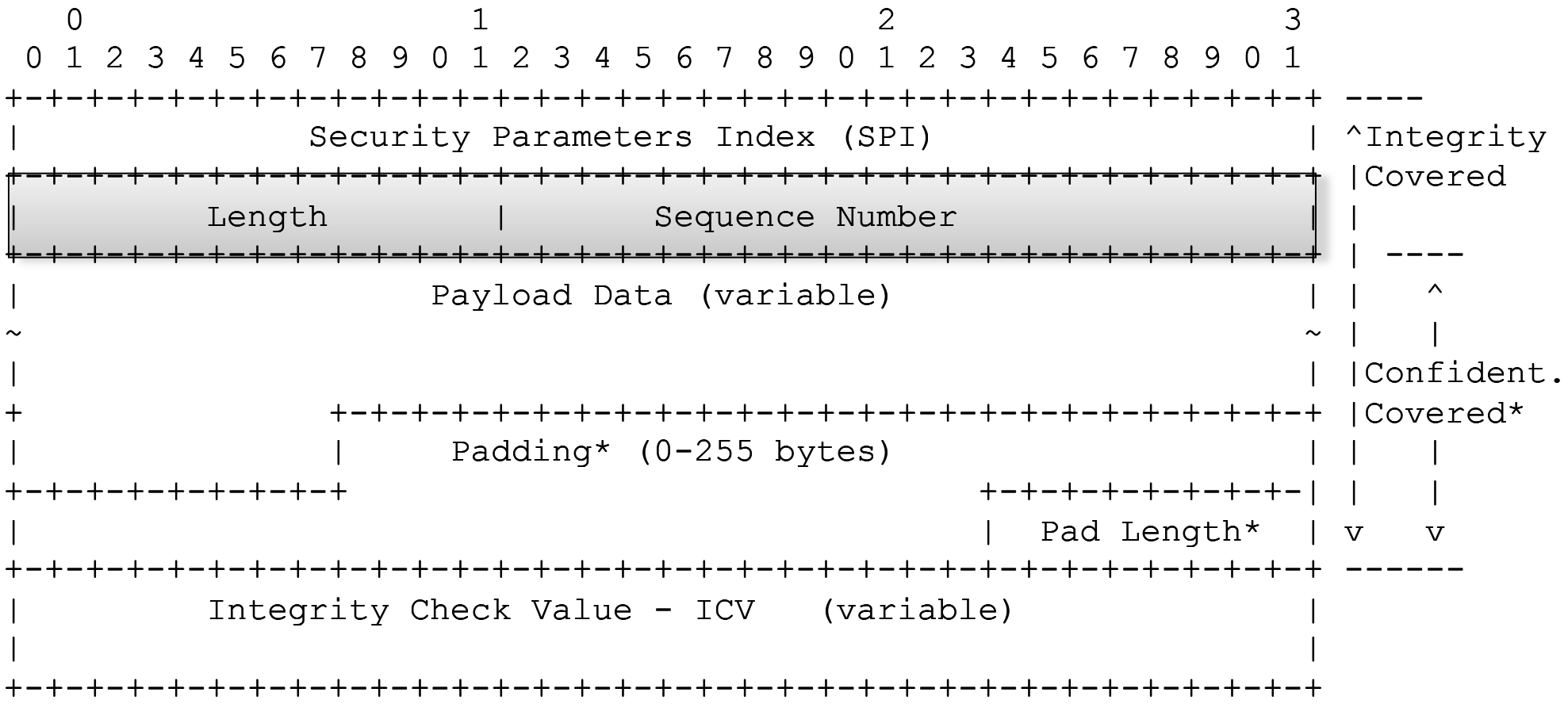
Disclaimer

- I am currently working on a number of internal products using an SSE
  - Why I am behind on my assignments for a number of standards works
  - Management and Legal insisted on a patent application filing
    - I will do IETF disclosure process once completed

Current work on 3 SSE Formats

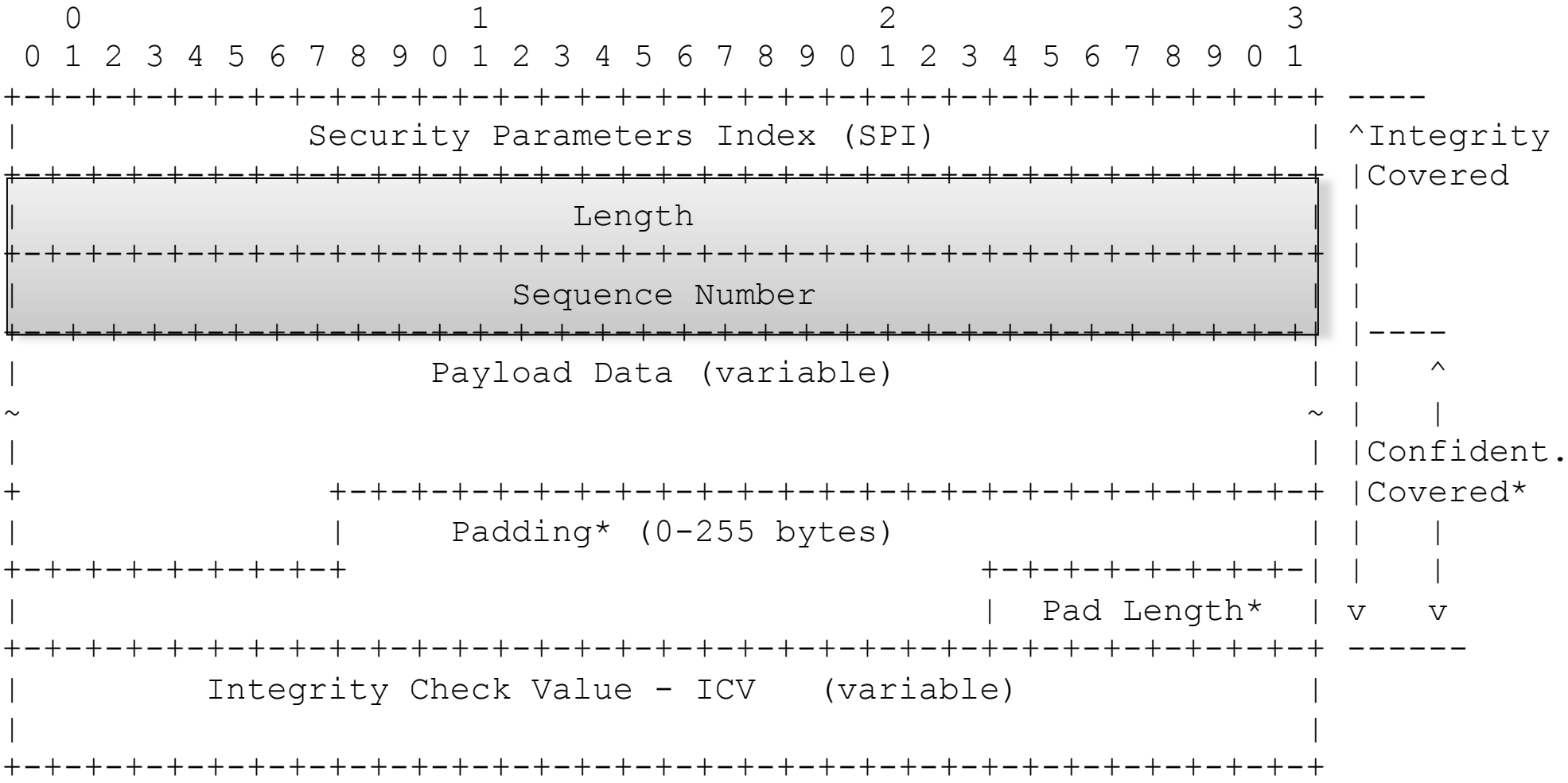
- Highly constrained networks have a high cost per byte
- Fast networks MUST be able to maintain message synchronization
- This dichotomy leads to two SSE formats
- Very high bandwidth networks in the future
  - e.g. 400gbs ethernet
    - Which exceeds 32 bit explicit sequence number capabilities
    - Thus a third format
- Let the market decide what gets used

SSE Compact Format



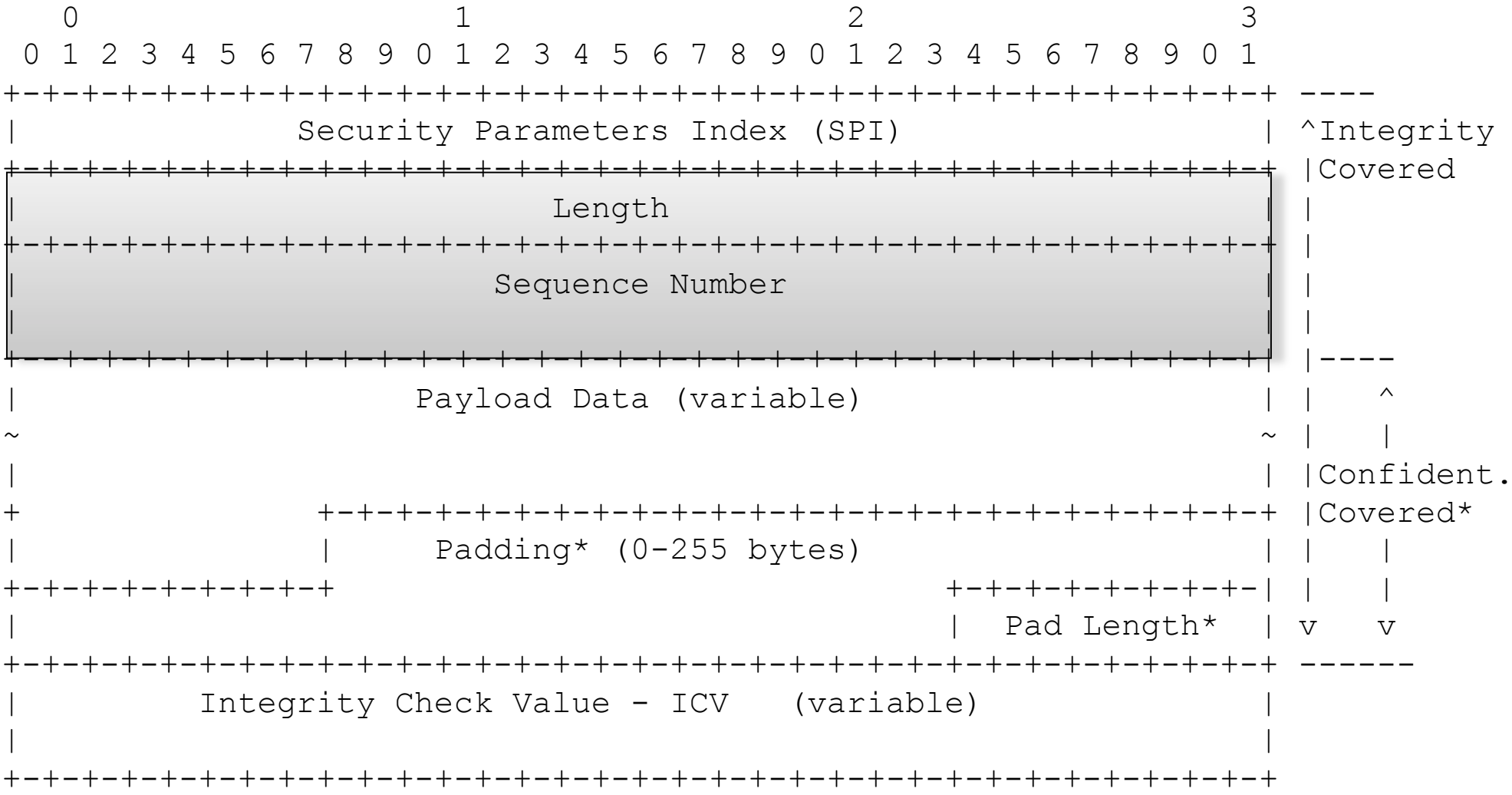
* Optional

SSE Large Format



* Optional

SSE Extreme Format



* Optional

Thank you for your time