Improvements in the OpenBSD IPsec stack

Mike Belopuhov <mikeb@openbsd.org>

AES-GCM: Overview

- is a combined authentication/encryption transformation
- encrypts and generates MAC in one pass
- 128 bit MAC (not truncated)
- AES-GMAC is an authentication only version
- is essentially an AES-CTR + GHASH

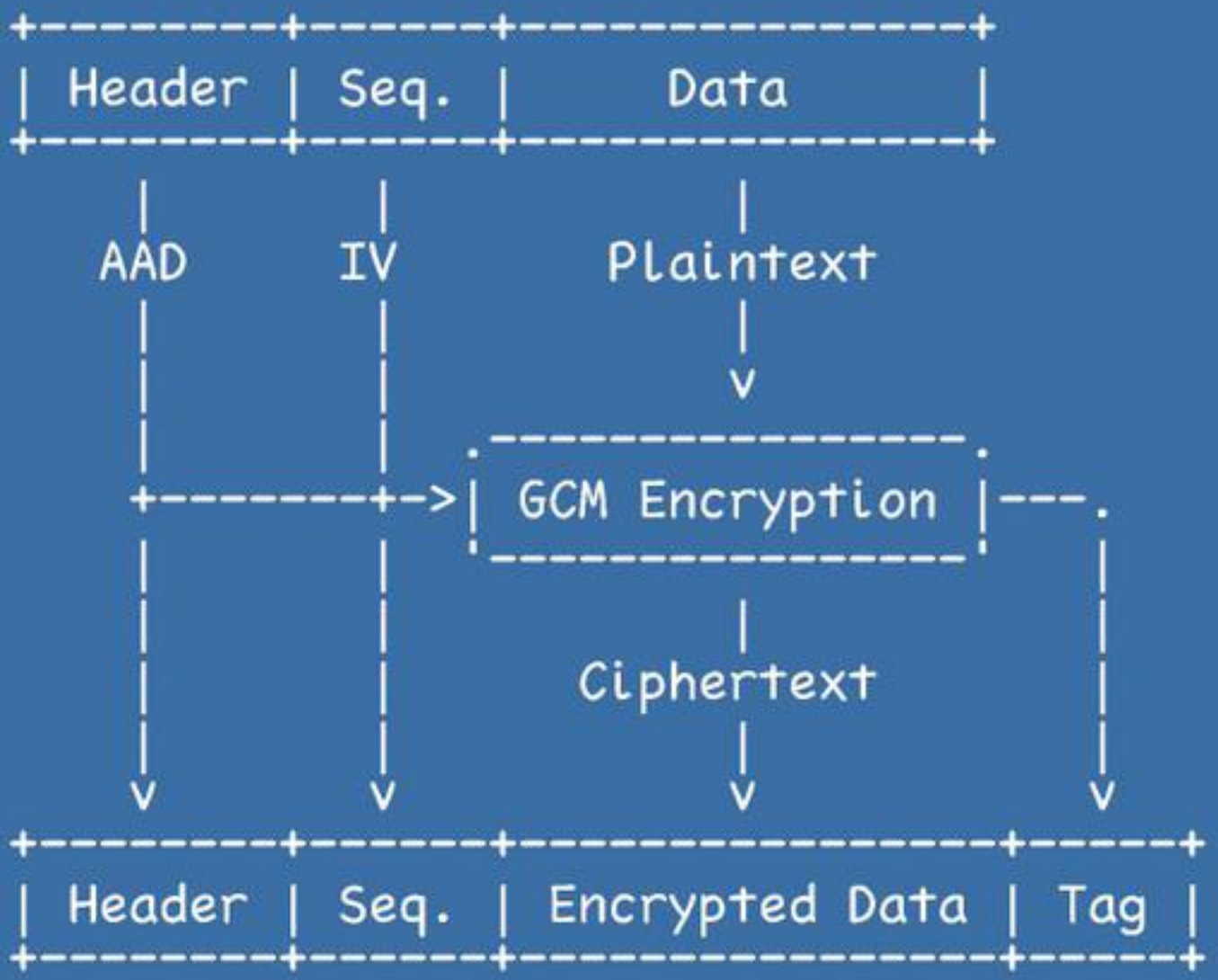
AES-GCM: Why?

- efficient use of an instruction pipeline
- easily parallelized even on instruction level
- unencumbered by patents
- accelerated in hardware: 2+ Gbps per Intel core

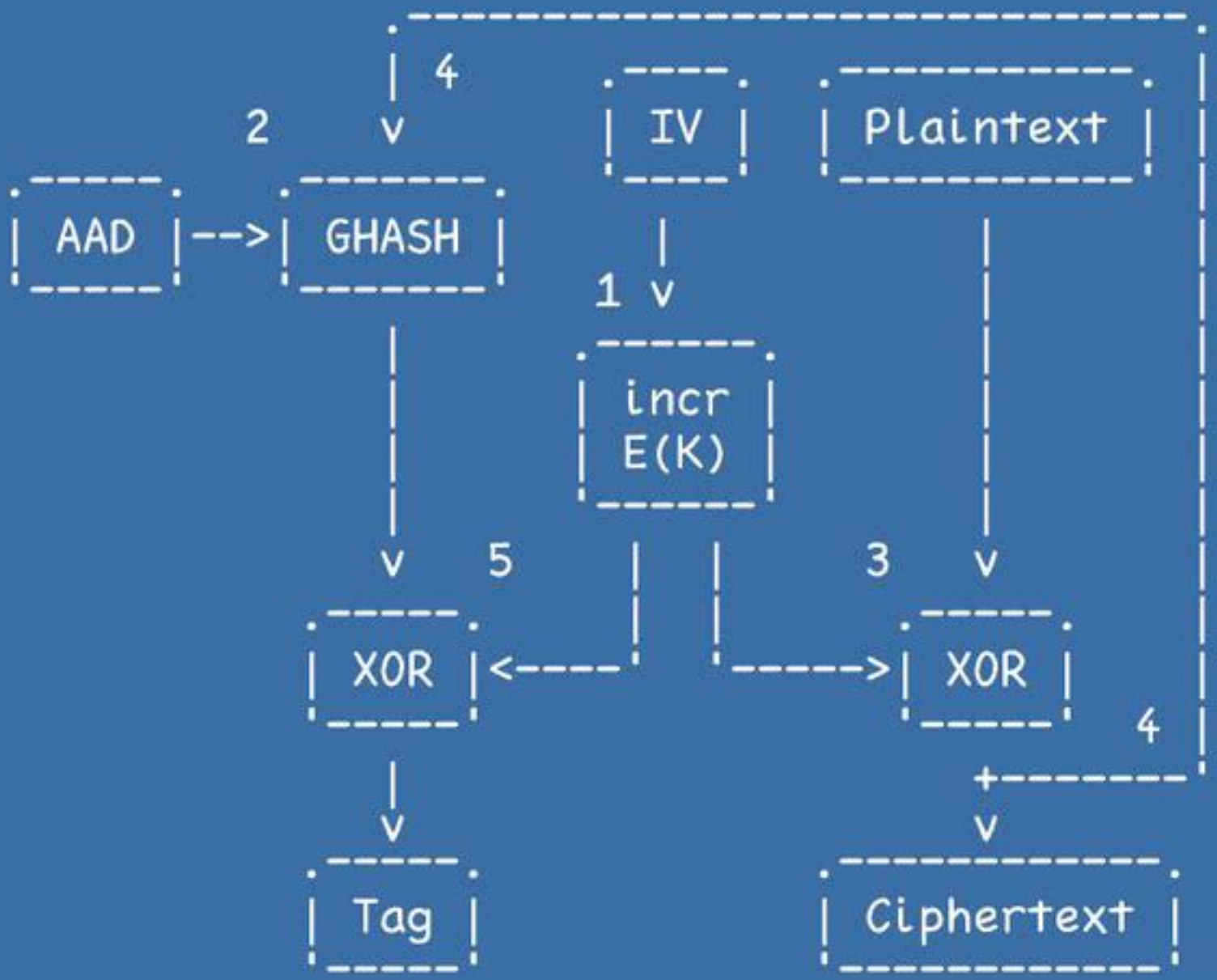
AES-GCM: Use

- MACsec, FC-SP, IPsec, SSH, TLS/SSL
- NSA Suite B endorses it as a preferred mode of AES
- Optional in the USGv6
- NIST standard

AES-GCM: Operation



AES-GCM: Operation



AES-GCM: Implementation in the kernel

Portable implementation written in C:

- reuses existing AES-CTR code
- AES_GMAC_{Init,Setkey,Reinit,Update,Final}
- swcr_authenc for combined transformations
- a straightforward implementation
- but slow

AMD64 specific written in asm and C:

- based on the BSD licensed code by Intel
- 650-750 Mbps in the IPsec tunnel mode

AES-NI & CLMUL: SSE instructions

Available in the Intel Westmere and newer:

- aeskeygenassist, aesimc - key expansion
- aesenc, aesdec - encryption/decryption round
- aesenclast, aesdeclast - final round
- pclmulqdq - carry-less multiplication

AES-NI & CLMUL: FP in the kernel

- normally not used
- requires caller to save and restore fpu context
- requires caller to setup a clean fpu context
- cannot be safely used in the interrupt context
- involuntary context switches should be avoided
- fpu_kernel_enter()/fpu_kernel_exit()

AES-NI: The driver

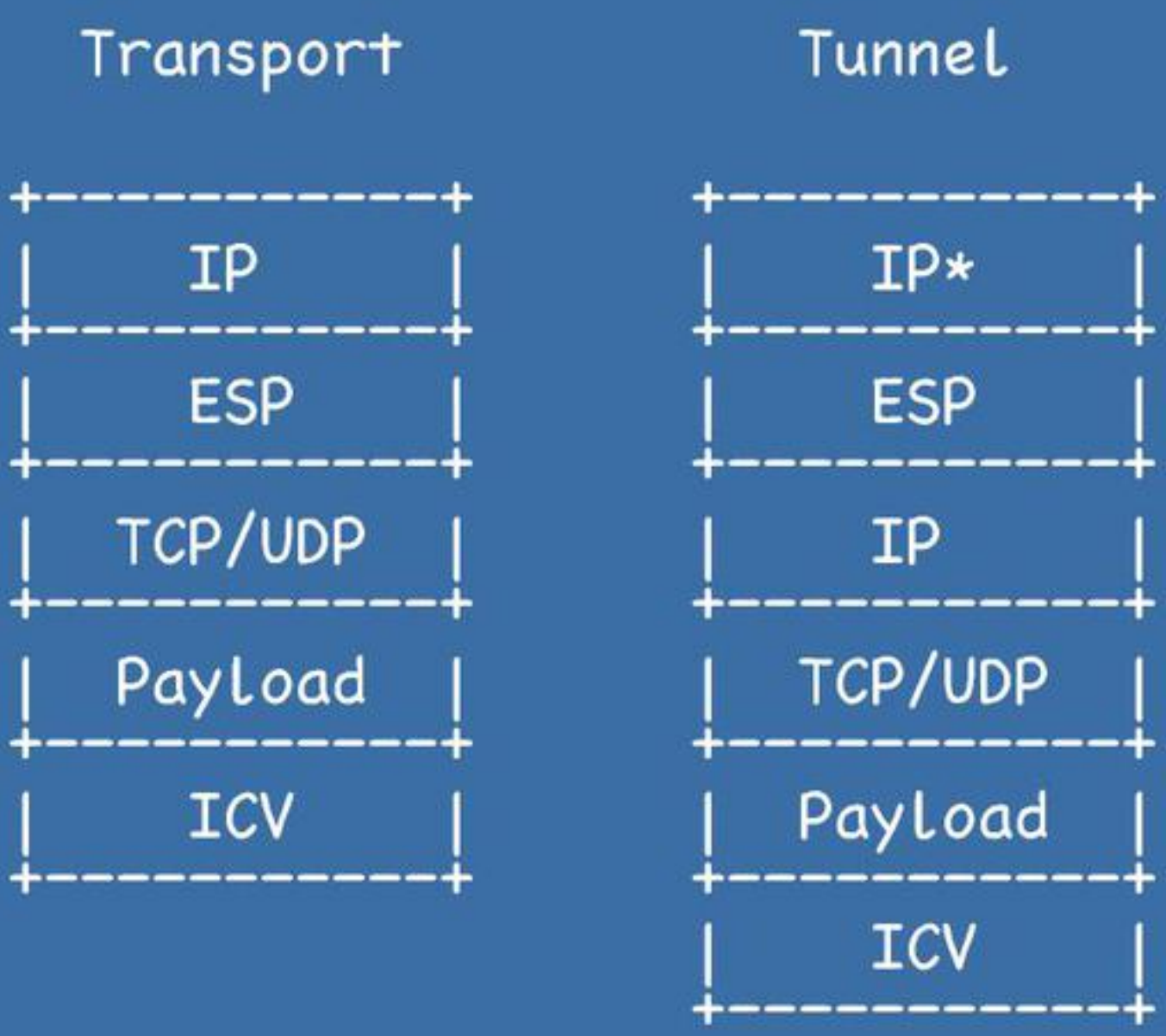
/sys/arch/amd64/amd64/aesni.c

- wrapper around assembly
- supports AES-CBC, AES-CTR, AES-GCM-16
- accelerated CBC and CTR modes since OpenBSD 4.9
- accelerated GCM mode since OpenBSD 5.1
- support for ESN since OpenBSD 5.3
- calls swcr for HMAC

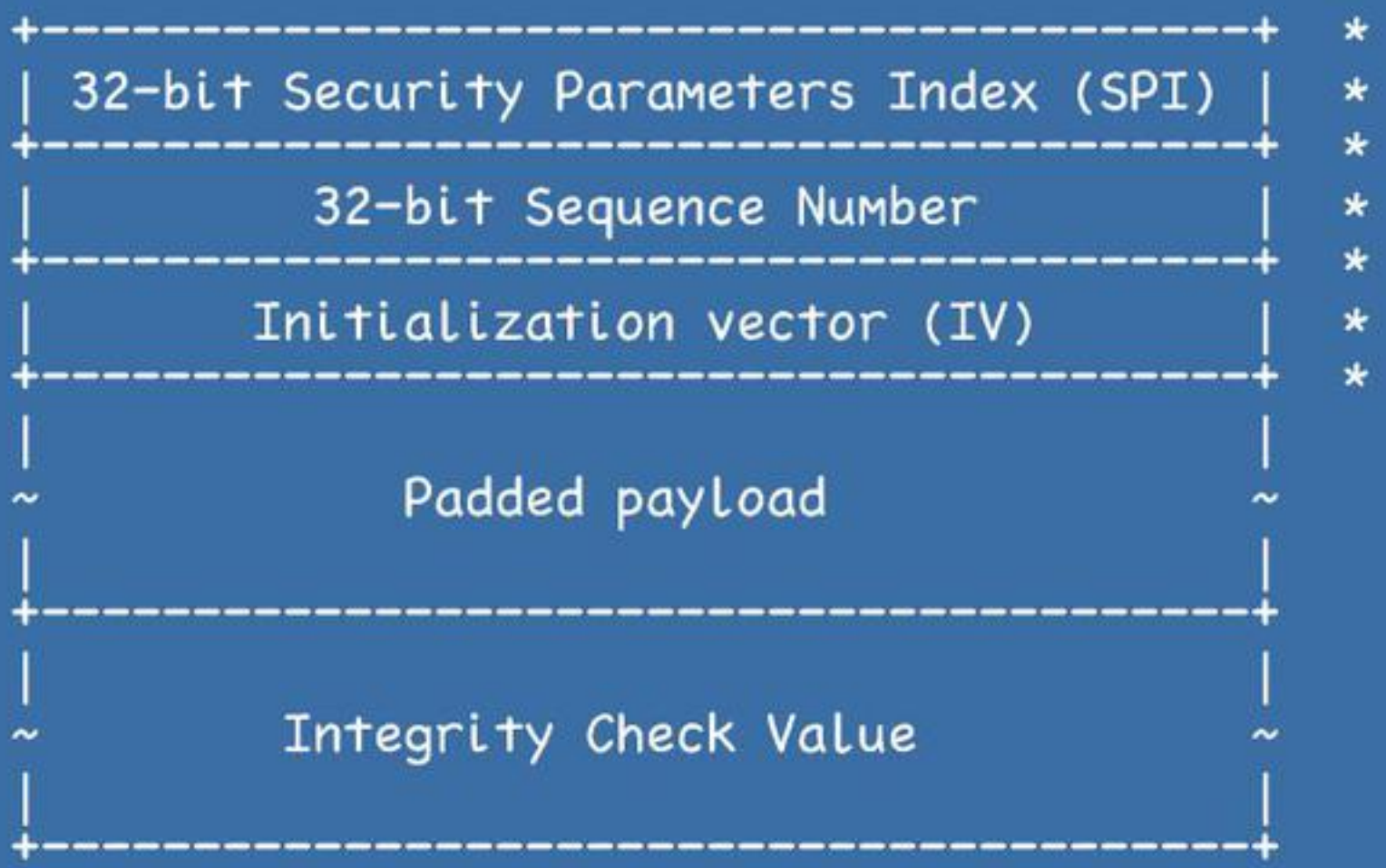
AES-NI: Future projects

- implement AES-XTS (done by jsing@)
- improve assembly
- port to i386
- evaluate AVX

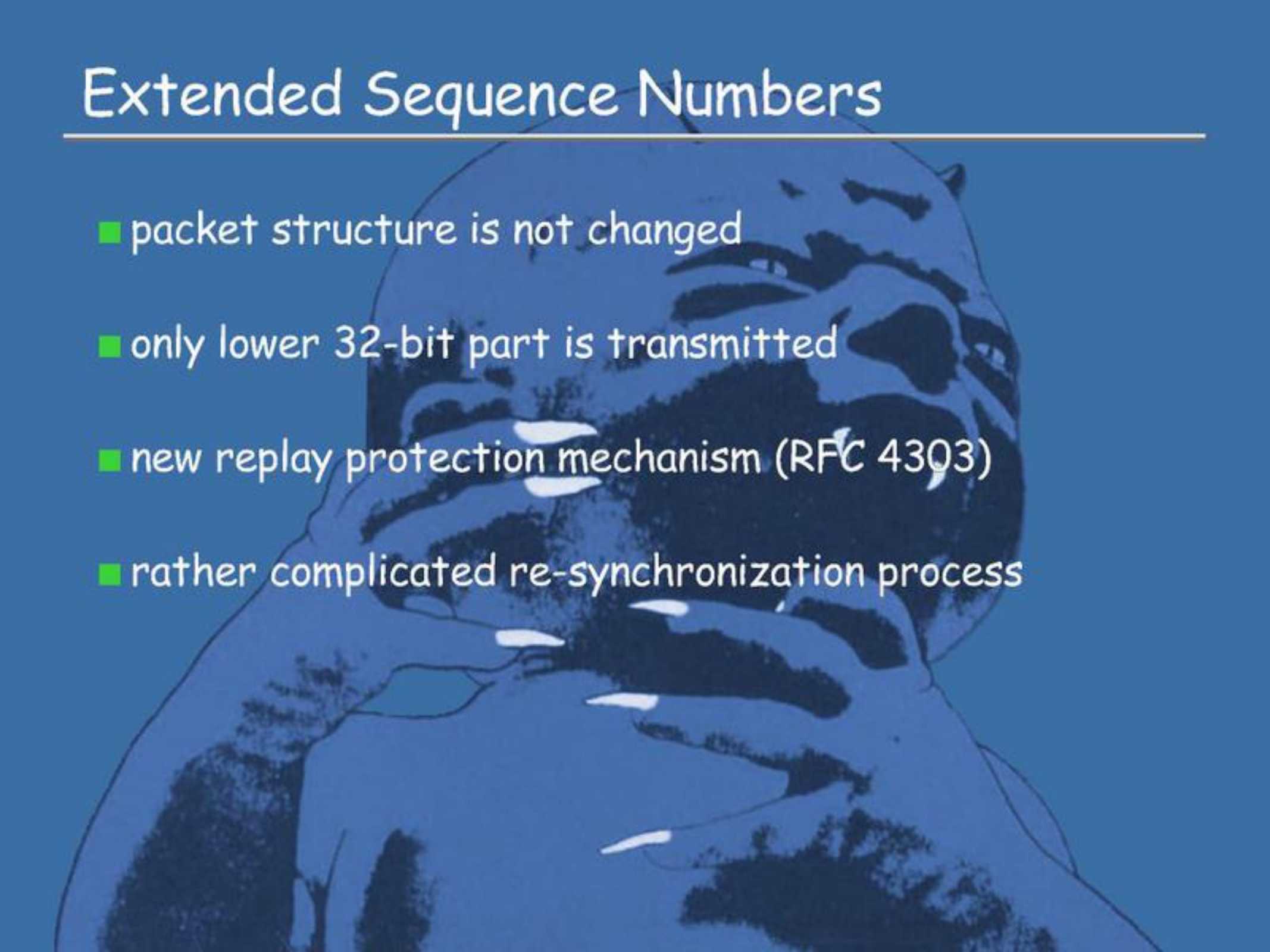
ESP


Transport
IP
ESP
TCP/UDP
Payload
ICV
Tunnel
IP*
ESP
IP
TCP/UDP
Payload
ICV

ESP


32-bit Security Parameters Index (SPI)
32-bit Sequence Number
Initialization vector (IV)
Padded payload
Integrity Check Value

Extended Sequence Numbers



- packet structure is not changed
- only lower 32-bit part is transmitted
- new replay protection mechanism (RFC 4303)
- rather complicated re-synchronization process

First attempt at ESN in OpenBSD 5.2

- only supported by the IKEv2 daemon iked(8)
- works fine with AES-CBC against Strongswan
- replay window size increased to 64 packets
- artificial replay distance of 1000 packets
- bug in the GCM support

AES-GCM and ESN in OpenBSD 5.3

RFC 4106 authors didn't pay attention to how ESN handling was described in RFC 4303 for non-combined modes:

hash higher 32-bit part *after* the rest of the packet

and specified ESN handling like this:

AES-GCM and ESN in OpenBSD 5.3

```
/*
 * Section 5 of RFC 4106 specifies that AAD construction consists of
 * {SPI, ESN, SN} whereas the real packet contains only {SPI, SN}.
 * Unfortunately it doesn't follow a good example set in the Section
 * 3.3.2.1 of RFC 4303 where upper part of the ESN, located in the
 * external (to the packet) memory buffer, is processed by the hash
 * function in the end thus allowing to retain simple programming
 * interfaces and avoid kludges like the one below.
 */
if (crda->crd_flags & CRD_F_ESN) {
    aadlen += 4;
    /* SPI */
    COPYDATA(outtype, buf, crda->crd_skip, 4, blk);
    /* loop below will start with an offset of 4 */
    iskip = 4;
    /* ESN */
    bcopy(crda->crd_esn, blk + 4, 4);
    /* offset output buffer blk by 8 */
    oskip = iskip + 4;
}
```

Replay protection

- everything larger than "x" is a new packet
- everything inside the window requires checking
- everything outside of the window is "too old"

Replay protection in the ESN world

- record number of packets that fail authentication
- retry using larger value for the upper half of ESN
- in the background or using a separate processor
- a "SHOULD" ...
- huge "thanks" to BBN

Plan for OpenBSD 5.3

- merge in markus@' diff to support large anti-replay windows
- verify that we're not doing anything stupid

Questions?