# Network Security - ISA 656
# IPsec
# IPsec Key Management (IKE)

Angelos Stavrou

October 1, 2007

# What is IPsec, and Why?



- Network-layer security protocol for the Internet.
- Completely transparent to applications.
- TCP- or application-level retransmissions handle deleted or damaged packets.
- Generally must modify protocol stack or kernel; out of reach of application writers or users.

# History

**SP3** Layer 3 security protocol for SDNS.

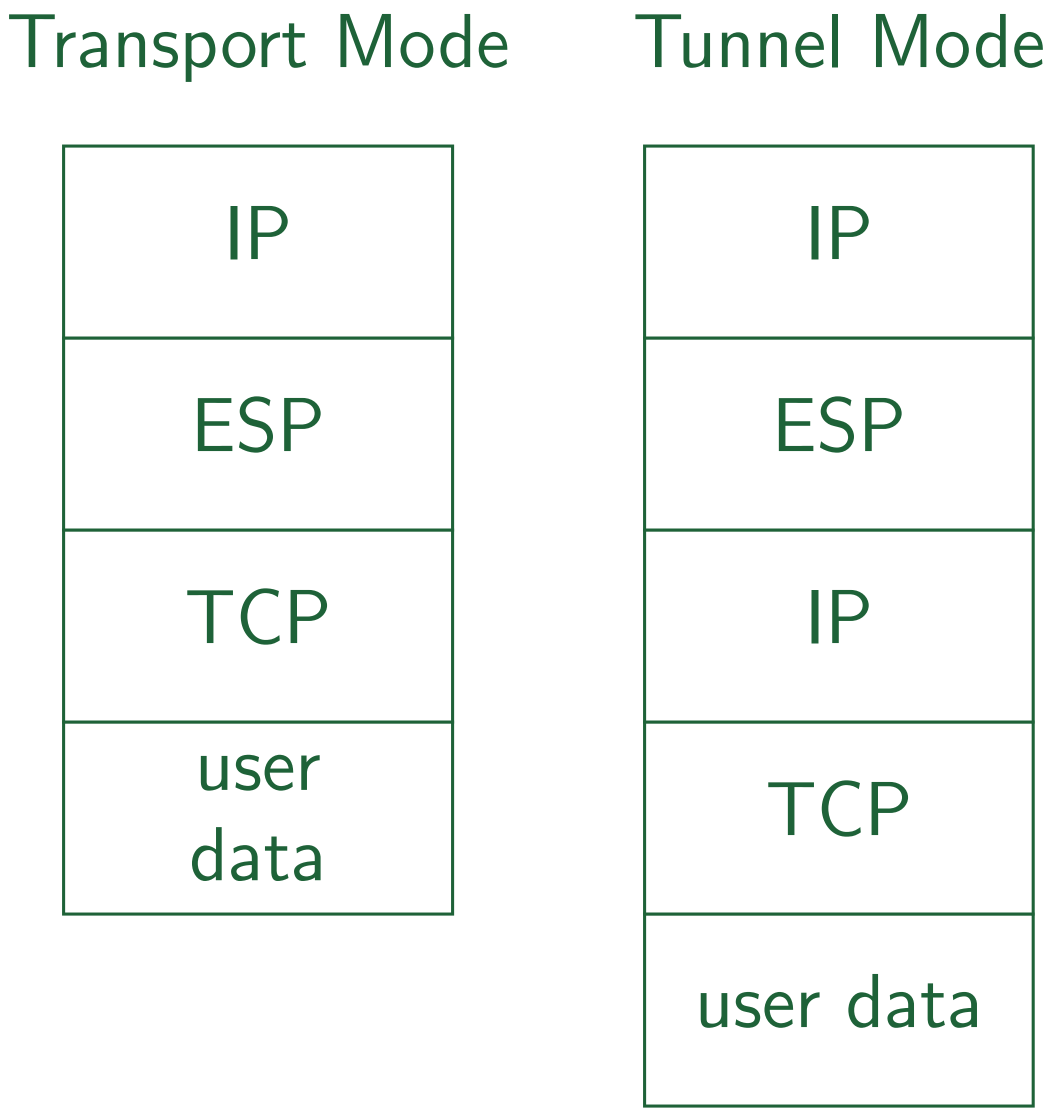
**NLSP** OSIfied version of SP3, with an incomprehensible spec.

**swIPe** UNIX implementation by Ioannidis and Blaze.

# IPsec Structure

- Nested headers: IP, ESP, AH, maybe another IP, TCP or UDP, then data.
- Cryptographic protection can be host to host, host to firewall, or firewall to firewall.
- Option for user-granularity keying.
- Works with IPv4 and IPv6.

# Packet Layout



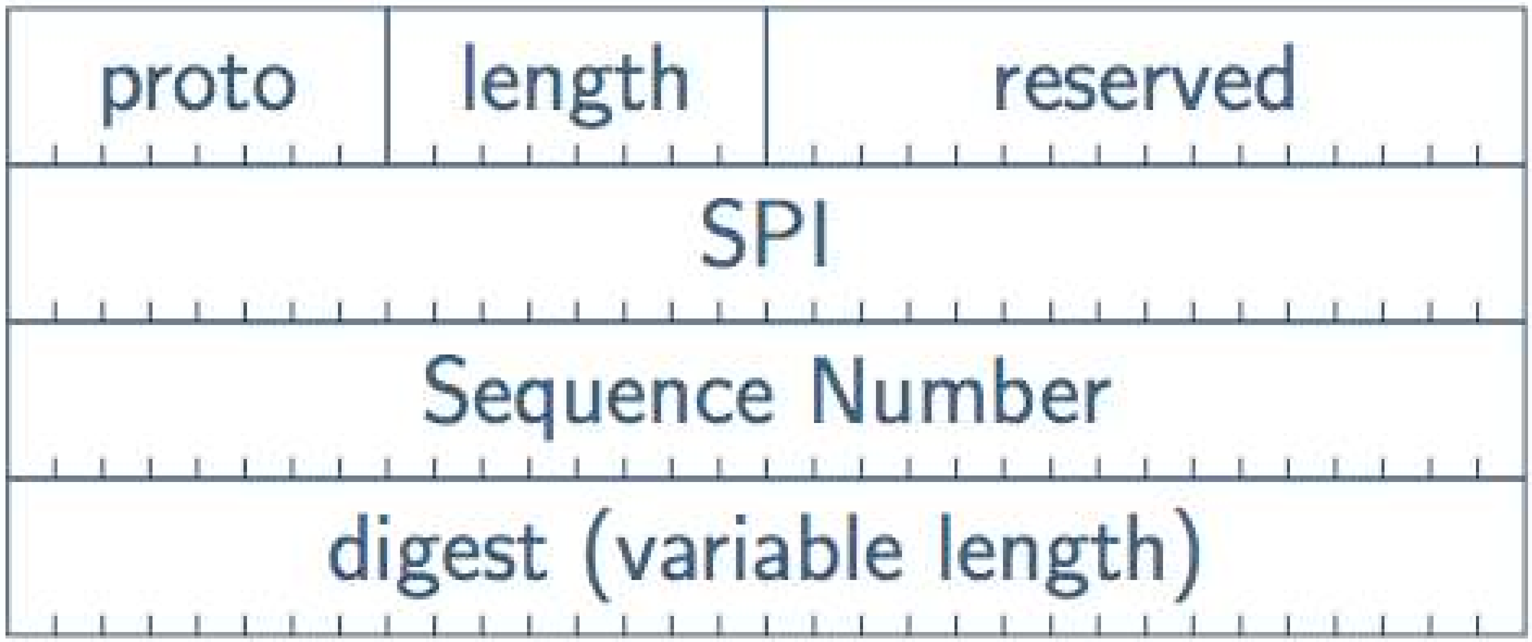
# Authentication Header (AH)



- Based on keyed cryptographic hash function.
- Covers payload and portion of preceeding IP header.
- Uses *Security Parameter Index* (SPI) to identify security association, and hence key, algorithm, etc.

# AH Layout





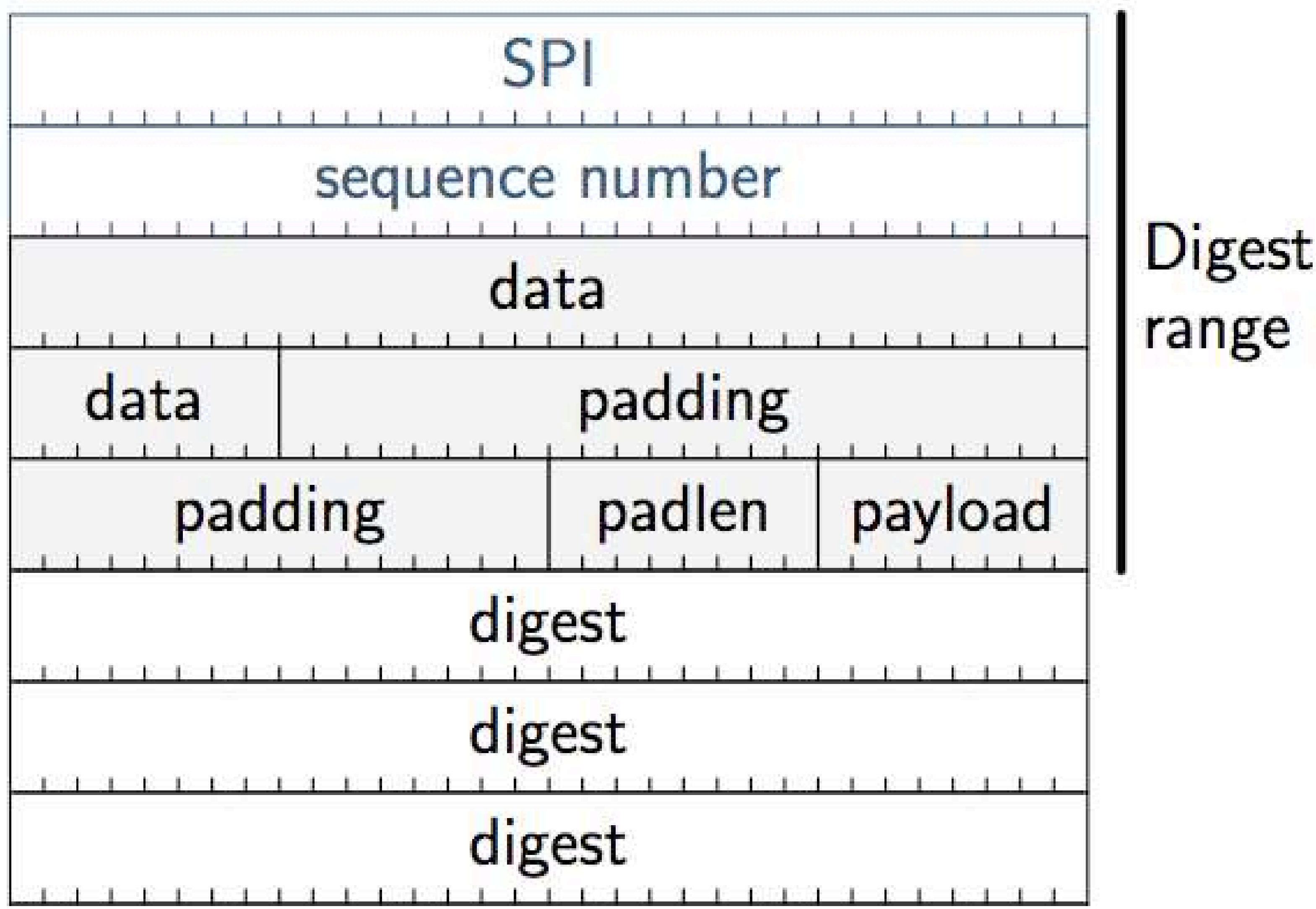
proto
length
reserved
SPI
Sequence Number
digest (variable length)

# Encapsulating Security Payload (ESP)

- Carries encrypted packet.
- An SPI is used, as with AH.
- Standard use of ESP is for DES in CBC mode.

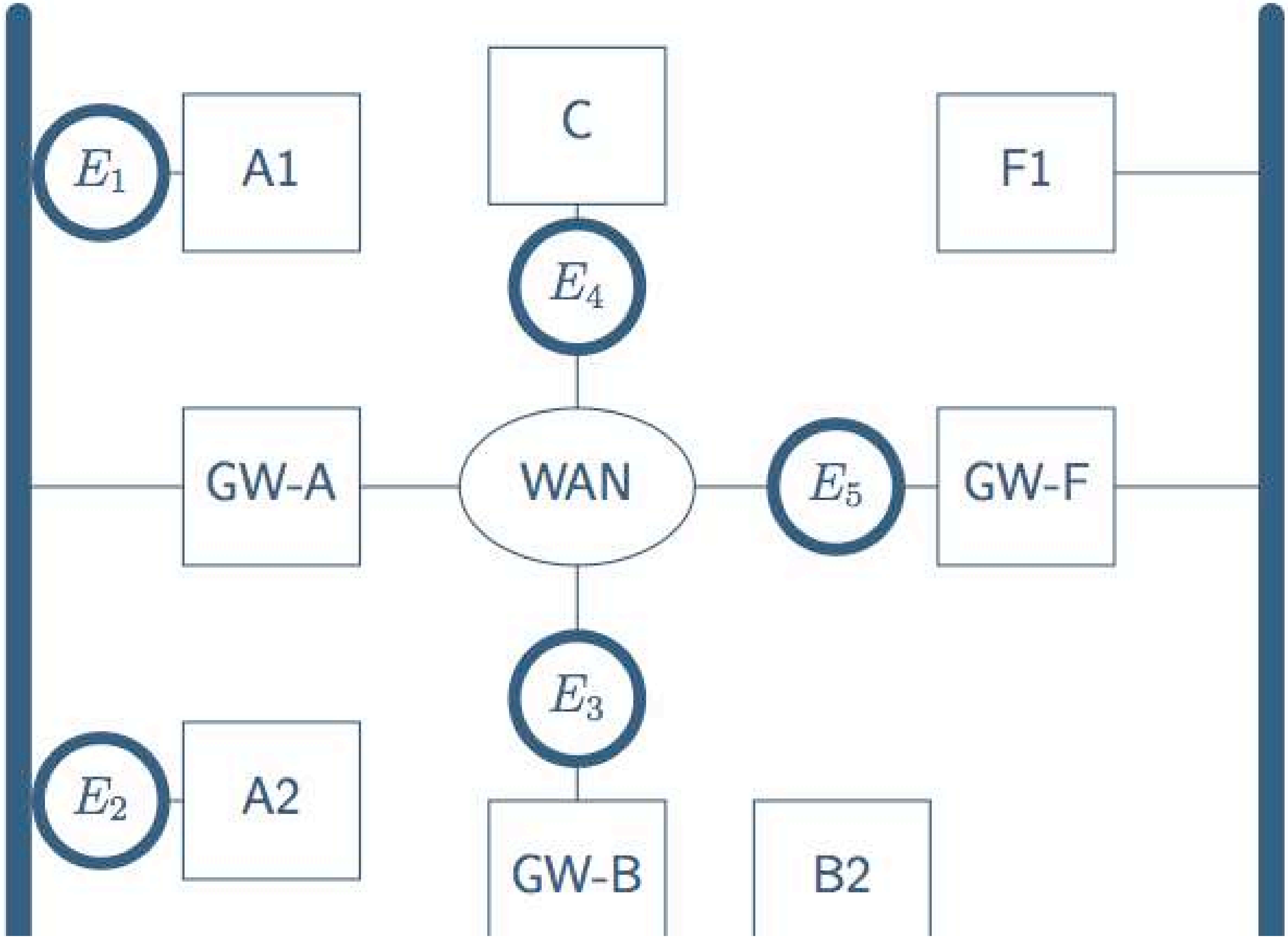
# ESP Layout




SPI
sequence number
data
data
padding
padding
padlen
payload
digest
digest
digest
Digest range

# Topologies





# Paths



- A1 to F1, F2:
  Encryptors $E_1$, $E_5$
- B1, B2, D1, D2 to F1, F2:
  Encryptors $E_3$, $E_5$
- A2 to C:
  Encryptors $E_2$, $E_4$

GEORGE MASON UNIVERSITY



# Uses for IPsec

- Virtual Private Networks.
- "Phone home" for laptops, telecommuters.
- General Internet security.

# IPsec and Firewalls



- Encryption is not authentication.
- Access controls may need to be applied to encrypted traffic, depending on the source.
- The source IP address is only authenticated if it is somehow bound to the certificate.
- Encrypted traffic can use a different firewall; however, co-ordination of policies may be needed.

# IPsec and the DNS



- IPsec often relies on the DNS.
  - Users specify hostnames.
  - IPsec operates at the IP layer, where IP addresses are used.
  - An attacker could try to subvert the mapping.
- DNSSEC may not meet some organizational security standards.
- DNSSEC — which isn't deployed yet, either — uses its own certificates, not X.509.

# Implementation Issues



- How do applications request cryptographic protection? How do they verify its existence?
- How do adminstrators mandate cryptography between host or network pairs?
- We need to resolve authorization issues.

# Key Management Requirements

# Why Key Management?

- Where do IPsec keys come from?
- Could we use static keys?
- What are the other requirements for key management?

# Static Keys



- In theory, static keys can be used; in practice, they have several disadvantages
- Primary disadvantage: they almost certainly will not be random enough
- (If they're passwords, attackers can launch a password guessing attack)
- History (and theory) suggest that it's a bad idea to encrypt too much plaintext with a single key
- You can't use replay protection with static keys

GEORGE MASON UNIVERSITY

# Replay Protection



- The first packet transmitted on an SA *must* be numbered 1
- Any time a machine reboots and loses knowledge of its sequence number status, it will restart from 1
- Besides, $2^{32}$ packets isn't that many; it *will* wrap around at some point
- Replays can be used to attack confidentiality

# SA Management

- We spoke of the SADB
- How does it get populated?
- We must negotiate it!

# Other Issues

- SA lifetime
- Dead peer detection
- SA tear-down
- Algorithm negotiation
- Other negotiations

# Internet Key Exchange (IKE)

# IKE

- *Very* complex protocol
- Does a lot, probably too much
- We'll just skim the surface, and we'll discuss IKEv2, which is simpler
- I'll be simplifying it, too. . .

# Basic Philosophy



- Two parties, *Initiator* and *Responder*
- First set up a *control SA* (known in IKEv1 as a *Phase 1* SA)
- Use the control SA to create *child SAs* (known as *Phase 2* SAs)
- Actual IPsec data is protected via child SAs
- Other control traffic can use the control SA

# Initial Exchange

- (Each message includes a random SPI, to distinguish between different IKE sessions.)
- Negotiate cryptographic algorithms
- Do a Diffie-Hellman exchange

$$I \rightarrow R: \quad \text{SA}_i 1, \text{KE}_i, N_i$$
$$R \rightarrow I: \quad \text{SA}_r 1, \text{KE}_r, N_r, [\text{Certreq}]$$

| | |
|---|---|
| SA | Crypto algorithm proposals and answer |
| KE | Diffie-Hellman exponential |
| $N$ | Nonce (random number) |
| Certreq | List of trust anchors (CAs) |

# What Do We Have?

- I has proposed several algorithms; R has accepted one of each category
- The two sides have a Diffie-Hellman shared secret. The Diffie-Hellman shared secret is combined with the two nonces to produce *seed keying material*. Any message $M$ protected by keying material derived from this will be written $\boxed{M}$
- Different keys are used in each direction
- I knows what CAs R trusts
- Neither side knows the other's identity yet

# Authentication



$$I \longrightarrow R: \quad \boxed{\mathsf{ID}_i, \mathsf{SA}_i 2, \mathsf{TS}_i, \mathsf{TS}_r, [\mathsf{Cert}]}, \mathsf{Auth}$$

$$R \longrightarrow I: \quad \boxed{\mathsf{ID}_r, \mathsf{SA}_r 2, \mathsf{TS}_i, \mathsf{TS}_r}, \mathsf{Auth}$$

Both sides send their own identities, the SA data for subsequent exchanges, *traffic selectors*, and an *authenticator*.

The authenticator is either an HMAC or a digital signature of the message (including the SPI) concatenated with the current sender's identity and the other party's nonce.

There are various other optional payloads for certificates, CAs, etc.

# What Do We Have?

- Both sides know the other's identity
- Both sides have authenticated the other
- Both sides have shared seed key material
- I has proposed a traffic selector; R has accepted a possibly-narrower one

# Traffic Selectors

- A *traffic selector* is a list of IP addresses and port numbers that are to be protected by the SA
- $TS_i$ specifies source addresses and ports; $TS_r$ specifies destination addresses and ports
- I proposes a certain range of traffic it wishes to protect
- R may agree to a narrower range
- This lets I — possibly a laptop — have a simple, "protect everything" configuration; the central gateway can narrow the scope of protection if desired

# Child SAs



- The control SA can now be used to create child SAs for actual user traffic

$$I \rightarrow R: \boxed{\text{SA}, N_i, [\text{KE}_i], [\text{TS}_i, \text{TS}_r]}$$

$$R \rightarrow I: \boxed{\text{SA}, N_r, [\text{KE}_r], [\text{TS}_i, \text{TS}_r]}$$

- Send new nonces for use in calculating keying material. For greater forward secrecy, send an optional new Diffie-Hellman exponential.
- Optionally negotiate new traffic selectors

# Rekeying

- Any SA can be rekeyed
- To rekey an SA, send a Rekey message with an SA identifier, new nonces, and perhaps new Diffie-Hellman exponentials
- Omit traffic selectors

# SA Lifetime

- SAs do not have negotiated lifetimes
- When either side thinks an SA has been around for long enough, it negotiates a new SA
- Net effect: SA lifetime is the shorter of the two sides' preferences
- *After* the new one is set up, delete the old SA

# Other Control Messages



- IKE "ping" — see if the other side is still alive
- Delete SA
- Obtain a remote IP address
- Check version information
- Error messages

# Timeouts

- IKE runs over UDP
- Each side must therefore implement its own timers and retranmissions
- It's reasonable to keep a cache of recently-received and -transmitted messages — when a duplicate request arrives, retransmit the cached copy

# Denial of Service



- What if an attacker attempts to exhaust R's CPU time or memory?
- CPU time: force it to calculate many D-H exponentials
- Memory: create initial SAs; don't authenticate them

# Defenses

- To prevent CPU time attacks, it's permissible to reuse D-H exponentials for a short while (though it hurts perfect forward secrecy)
- To prevent memory attacks, watch for too many incomplete SAs
- When these start to occur, reject new requests and send a *cookie* instead
- These are stateless, cryptographically sealed messages bound to the sender's IP address
- Require that such a cookie be returned with the actual first message
- Guards against spoofed IP address attacks

# Using IKE

- A host is configured with an initial protection SPD
- When a packet is to be sent that matches the SPD, IPsec searches for an existing SA
- If there is none, a request is sent to the local IKE daemon
- The IKE daemon attempts to create an SA, and updates the SAD
- (On some systems, this may result in updating the SPD)
- The packet is then transmitted

# Some Attacks

# Attacks!

- I keep talking about subtle attacks
- Let's look at some old ones. . .

# Splicing Attack

- Suppose that (a) ESP is being used with no authentication, (b) no sequence numbers, and (c) the good guy and the bad guy can send traffic on the same SA
- The bad guy intercepts a good guy's packet, sends a UDP packet with checksums turned off, and intercepts it, too
- The attacker then uses CBC splicing to replace the end of the UDP packet with the good guy's packet, and reinjects it
- The receiving IPsec sees this packet, decrypts it, and passes it to the bad guy's UDP listener

# Defenses

- Use ESP authentication
- Use ESP sequence numbers, to prevent reinjection of the UDP packet (though there are other variants that make that less useful)
- Use a separate SA for each connection

# Using a Separate SA?

- If you use separate SAs for each connection, it makes life easier for traffic analysts
- It can also aid cryptanalysts

# Probable Plaintext Attacks



- How does a cryptanalyst know if a guess at the key was correct?
- What should the packet look like?
- Compare certain fields from two packets for the same connection — they should match
- Source and destination IP address must match exactly
- Probabilistically, most bits of counters (such as TCP sequence numbers) will match: if you add 512 to a 32-bit number, probability is .97 that the high-order 18 bits remain unchanged, and the low-order 9 bits are always unchanged
- Other fields can be matched as well

# Defenses

- Not easy!
- Try avoiding per-connection SAs
- Don't use ciphers that are weak enough that this is a useful attack. . .