Introduction to IPsec

Charlie Kaufman
charliek@microsoft.com

IP Security (IPsec)

- IETF standard for Network Layer security
- Popular for creating trusted link (VPN), either firewall-firewall, or machine to firewall
- Done “at layer 3” (we’ll explain that later)
- Pieces include data packets (AH, ESP), authentication handshake (ISAKMP/IKE), and endless other documents

Terminology Nit…

- Cryptographic protection of data usually has two pieces:
  - Encryption, for confidentiality
  - Integrity protection, for authentication
- In this talk, I'll just say encryption and mean both!

Terminology Nit…

- Cryptographic protection of data usually has two pieces:
  - Encryption, for confidentiality
  - Integrity protection, for authentication
- In this talk, I'll just say encryption and mean both!
- "*We could do encryption without integrity protection, but it would be wrong, that's for sure*"….apologies to Richard Nixon

Distinction between IPsec and SSL/TLS Interesting

- Both “real time” security
  - Mutual authentication
  - SA (security association) establishment
  - encryption/integrity protection of conversation
- But important and subtle differences

IPsec vs. SSL/TLS

- IPsec philosophy: only change OS, don't change applications or API
- SSL/TLS philosophy: don't change OS, deployable as user process. TCP and below in OS, so works on top of TCP

SSL vs IPsec

- Layer 3 (IPsec) theoretically better
  - SSL: Rogue packet problem
    - TCP by definition, not involved in crypto
    - So attacker can generate TCP with (noncrypto) good checksum
      - TCP will accept it
      - Real data will be discarded as duplicate
    - Only recourse: break the connection
  - In contrast, each IPsec pkt ind. protected
  - Also, easier to build outboard crypto assist

However...

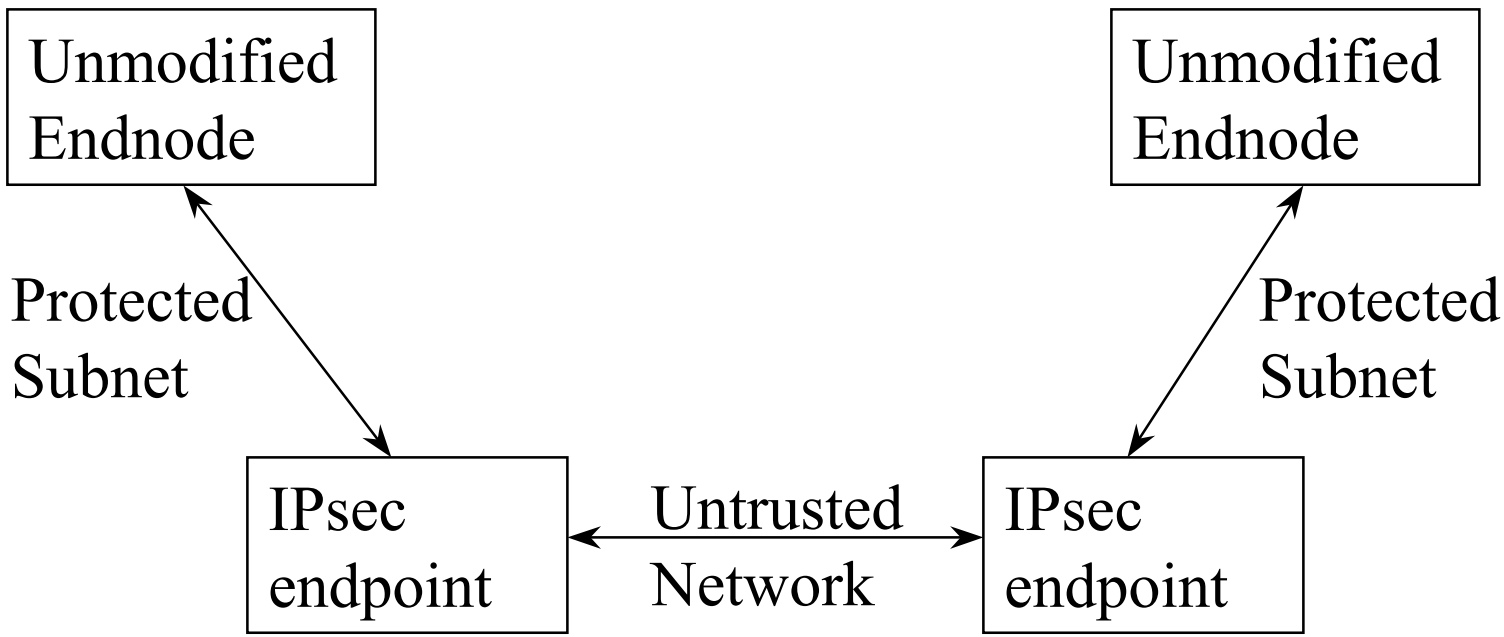
- If you don't change the API or the application:
  - the only thing IPsec can pass up is the IP address you're talking to
  - so IKE does all this PKI stuff to find out this is mary.smith.examplecompany.com, but can't tell app

What you do get

- Encryption of the traffic
- Ability to do filtering, based on a policy database
- Just as if there were a firewall between the two ends

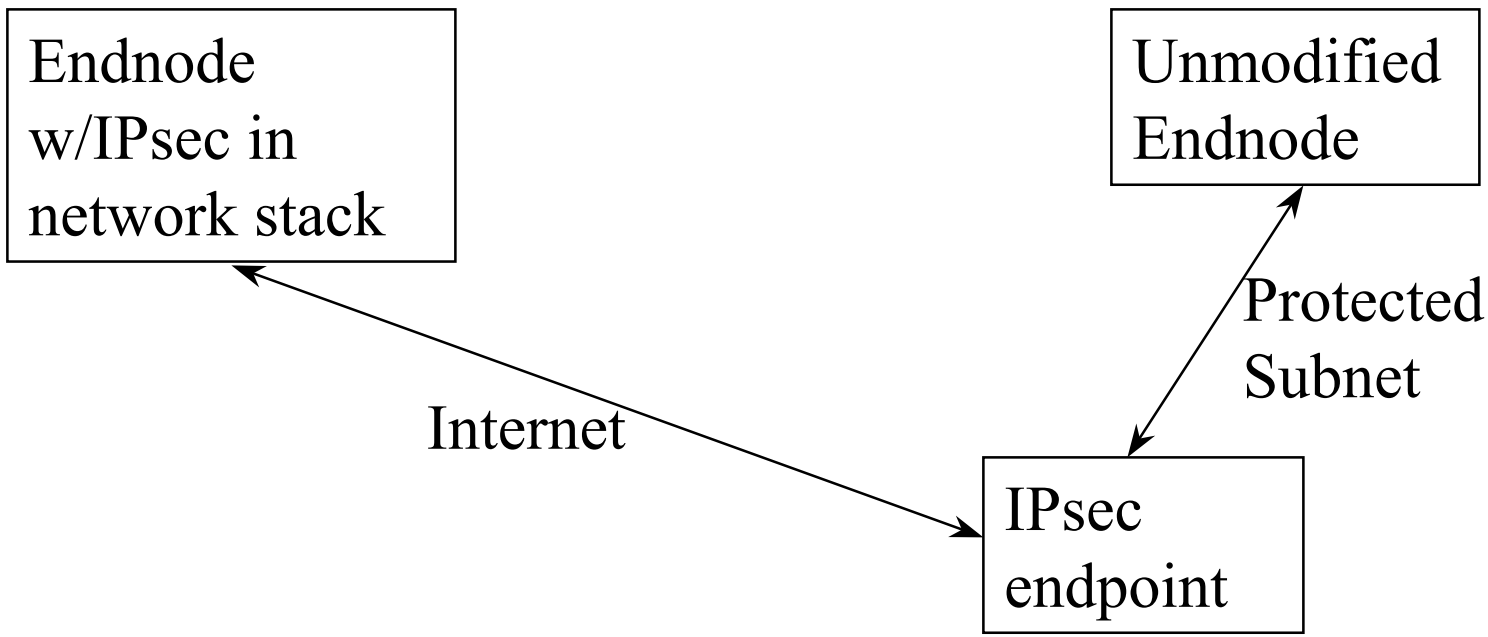
IPsec Scenario 1
Firewall to Firewall

- Corporate network connected through Internet



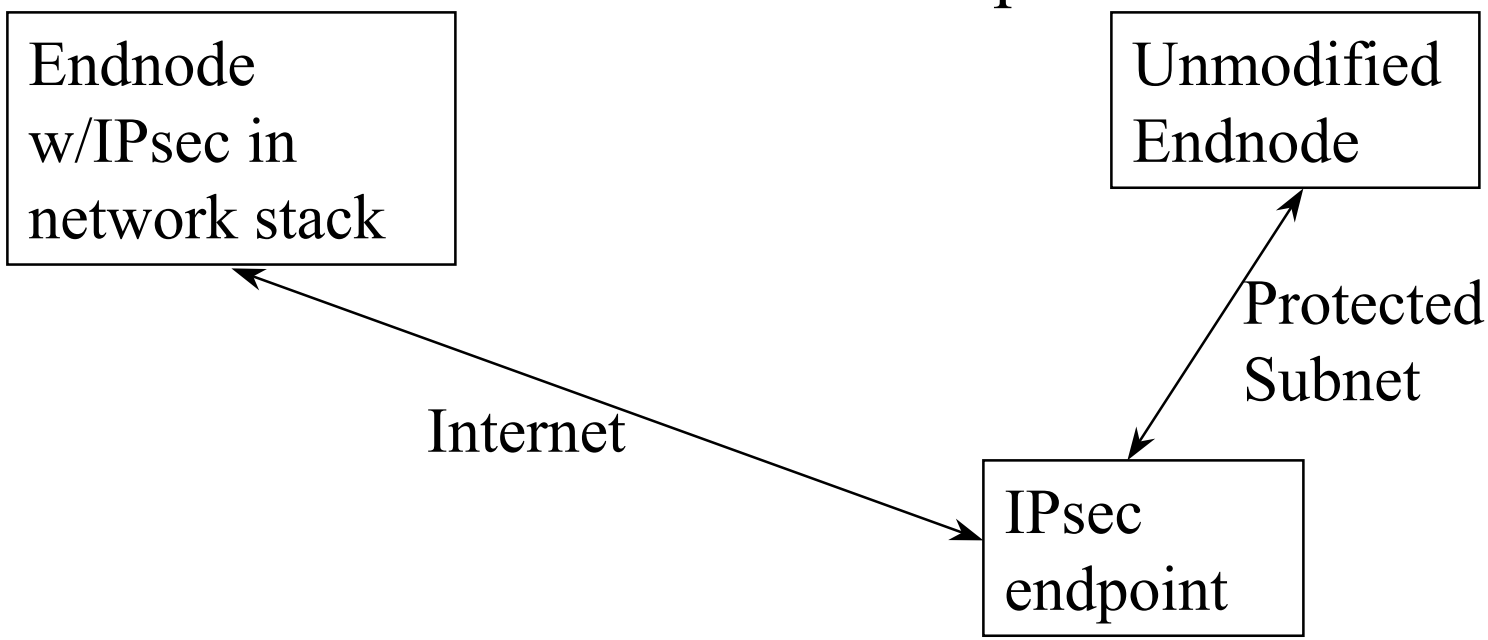
IPsec Scenario 2
Endnode to Firewall

- Mobile node connects home through Internet



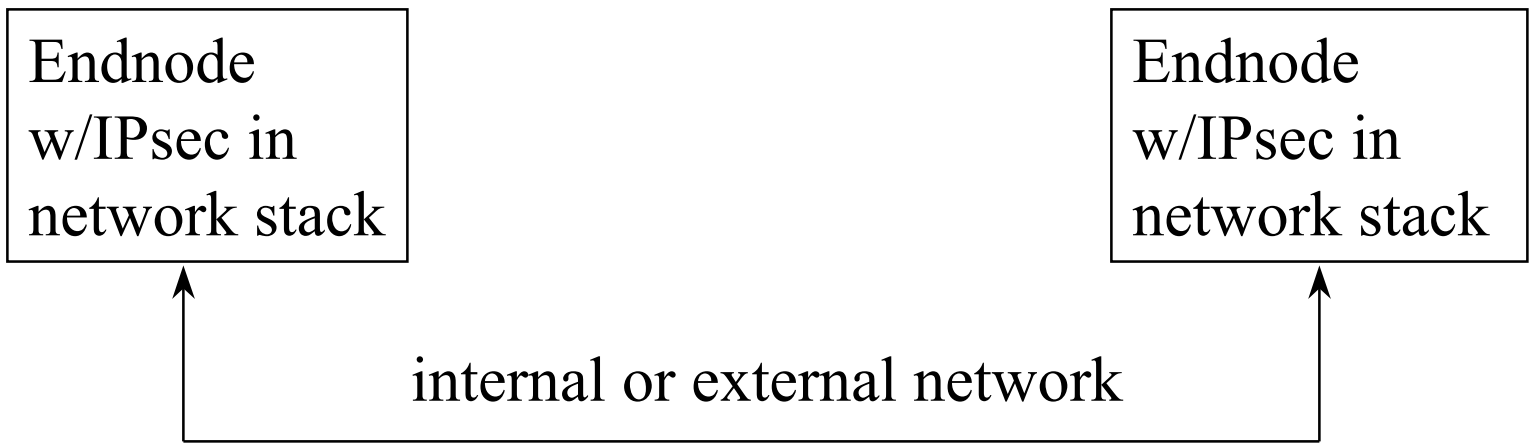
In Scenario 2, allocating an "internal" IP address

- Mobile node needs address in Protected Subnet that will be routed to IPsec endpoint



IPsec Scenario 3
End to End

- Two nodes don't need to trust the network

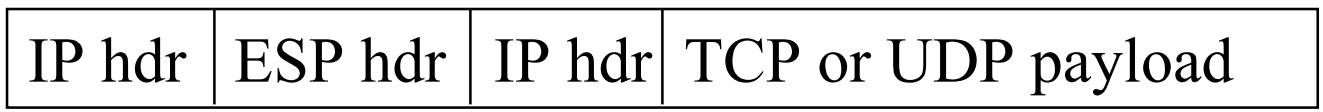


What does IPsec Protect?

- Protection from eavesdropping on the untrusted network
- In scenarios 1 & 2, connectivity only
  - control ‘admission’ to a protected network
- In scenario 3, potential for user and server authentication – mostly unrealized

Tunnel vs. Transport Mode

- In scenarios 1 & 2, IPsec payload is an IP packet complete with different addresses



- In scenario 3, IP endpoints have same addresses as IPsec endpoints, so second header not needed.

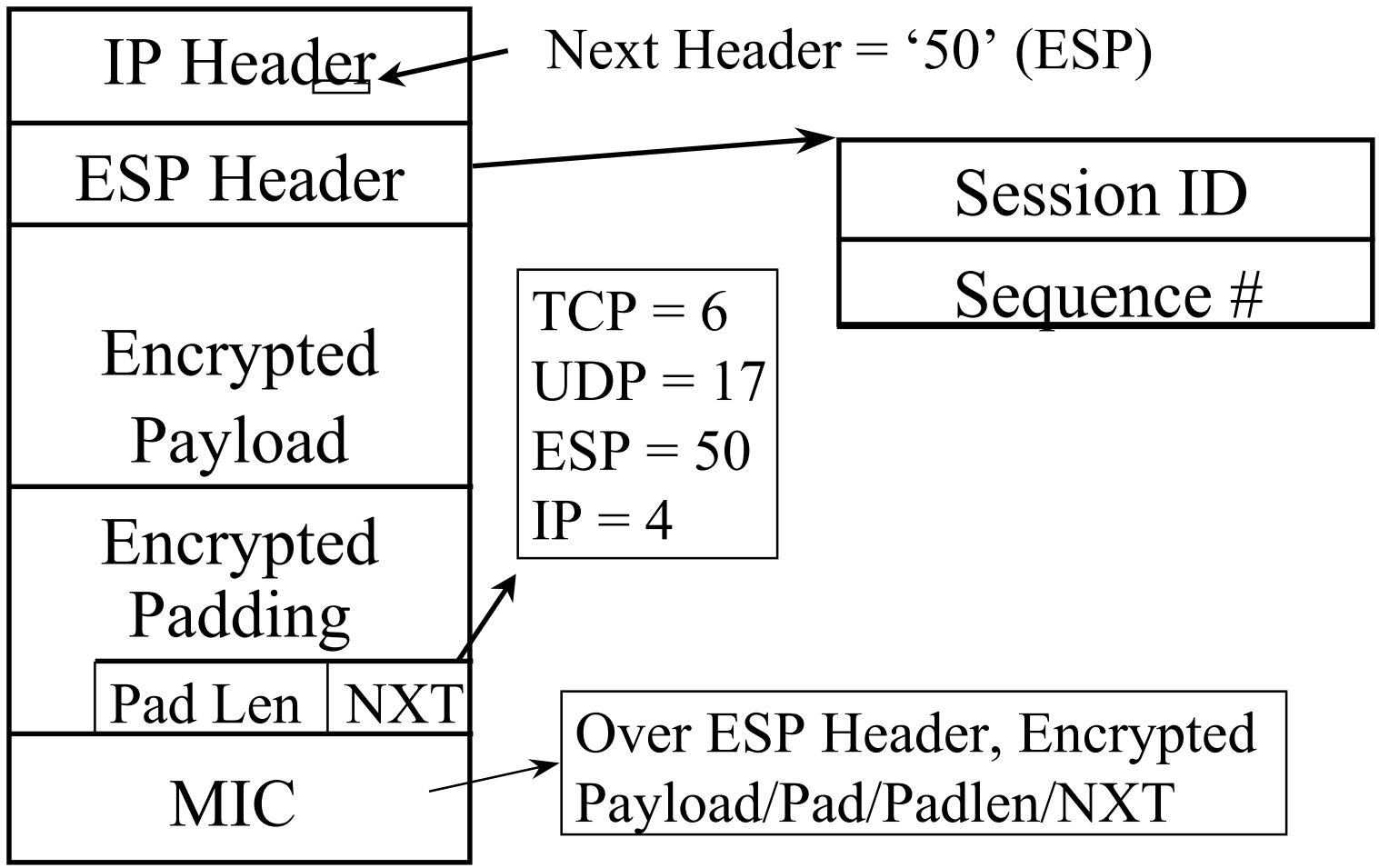
IKE vs. ESP vs. AH

- IPsec Security Association (SA) established using IKE
- Payload packets are encapsulated with ESP and/or AH
- IPsec Security Association could be configured manually (at least in theory) or using some other protocol

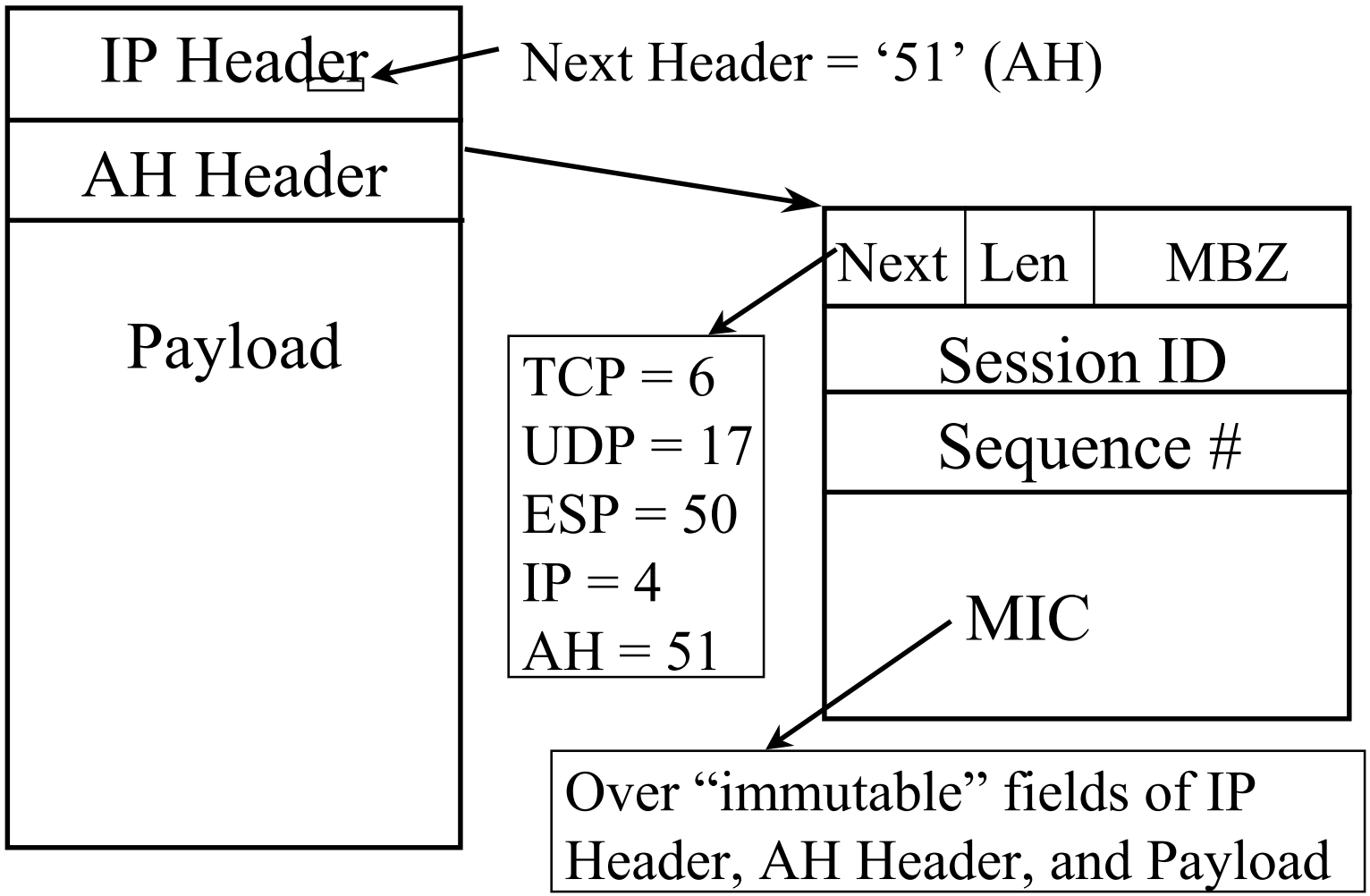
AH / ESP

- Extra header between layers 3 and 4 (IP and TCP) to give dest enough info to identify "security association"
- AH does integrity only - but also protects parts of IP header
- ESP does encryption and (optional) integrity protection (but only starting after IP header) … encryption "optional" too now

ESP
Encapsulating Security Payload



AH (Authentication Header)



ESP / AH

- Payload may be TCP, UDP, or some other ‘higher layer’ protocol (transport mode)
- Payload may be IP datagram (tunnel mode)
- Payload may be ESP/AH again (recursive encapsulation)
- If it’s important to protect IP header, ESP with tunnel mode will do that

Why AH?

- AH and ESP designed by different groups. AH designers were IPv6 supporters
- AH looks more like IPv6
- AH also protects “immutable” fields in IP header.
- Originally, ESP just encryption
- Encryption without integrity has flaws

Why AH, con't

- Then integrity protection added to ESP.
- Excuses for keeping AH
  - protects IP header (nobody has a credible security reason why, and ESP-tunnel can too.
  - Makes NAT harder, which pleases IPv6 fans)
  - with AH, firewalls and routers that want to look at layer 4 info (like ports) know it's not encrypted. With ESP, can't tell from packet

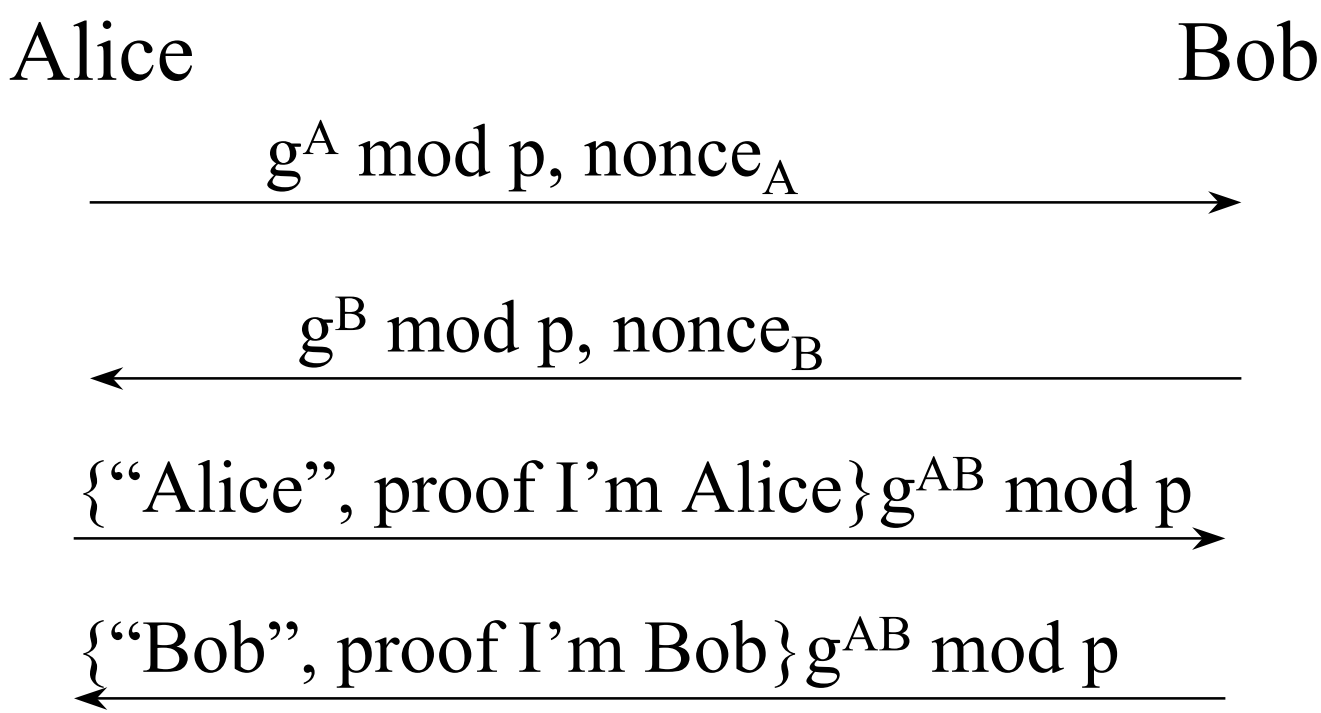
Why Not AH?

- IPsec already way too complex.
- AH implementation headache, makes IP complex (marking everything “mutable” or not)
- IP header can’t be integrity protected en route anyway (routers don’t know the key)
- You could peek inside ESP and almost always tell if it’s encrypted or not. A flag might be nice (reserved SPIs would work)

Internet Key Exchange (IKE)

- Resynchronize two ends of an IPsec SA
  - Choose cryptographic keys
  - Reset sequence numbers to zero
  - Authenticate endpoints
- Design evolved into something very complex

General idea of IKEv2



Functionality WG wanted

- Perfect Forward Secrecy
- Identity hiding
- Lots of authentication styles
- Work with NATs
- DHCP-like address allocation
- crypto negotiation
- filtering rules (“selectors”) negotiation (“Traffic over this SA must be between this set of IP addresses and layer 4 ports …)
- Two “phases” (next slide)

Phases

- Phase 1: expensive (when based on public keys) mutual authentication, establish SA between two machines
- Phase 2: leverage the phase 1 SA to create lots of “child-SAs”

Why Two Phases

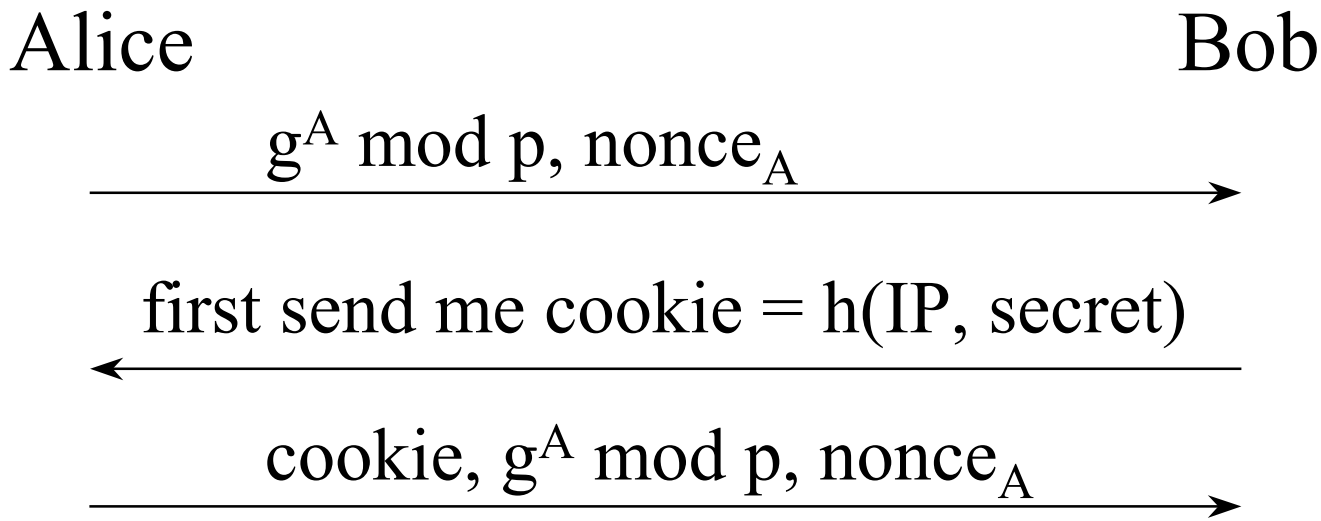
- We argued for removing this, but people wanted it for:
  - firewalls creating lots of VPNs for lots of customers…they feel safer if different SAs
  - different QOS, since might travel at different speeds, sequence numbers get far apart
  - makes rekeying faster
  - different SAs with different security properties

Conceptual IKE

- Diffie-Hellman for PFS
- Signed D-H to avoid man-in-middle attack
- Cookies for DoS protection

DoS Protection Using Cookies

- Avoid using memory or computation resources when pkts from forged IP addr's



An Intuition for Diffie-Hellman

- Allows two individuals to agree on a secret key, even though they can only communicate in public
- Alice chooses a private number and from that calculates a public number
- Bob does the same
- Each can use the other's public number and their own private number to compute the same secret
- An eavesdropper can't reproduce it

Why is D-H Secure?

- We assume the following is hard:
- Given g, p, and g^X mod p, what is X?
- With the best known mathematical techniques, this is somewhat harder than factoring a composite of the same magnitude as p
- Subtlety: they haven't proven that the algorithms are as hard to break as the underlying problem

Diffie-Hellman

Alice agree on g,p Bob

choose random A choose random B

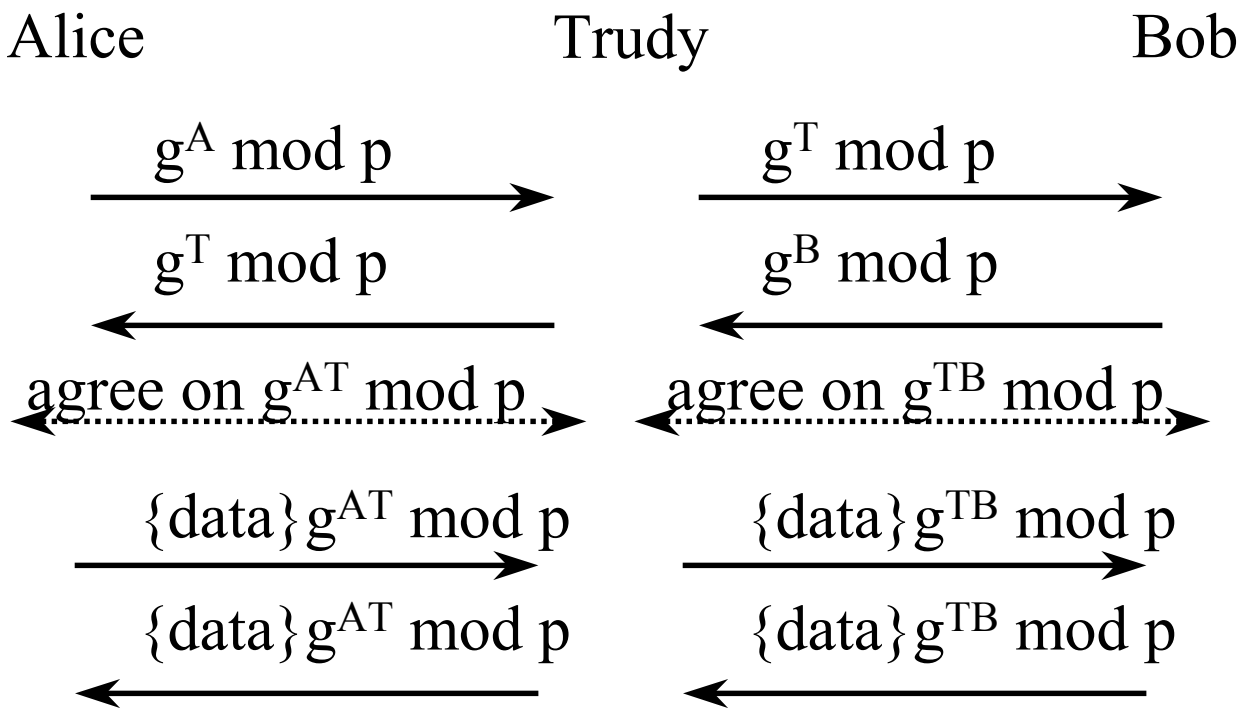
$g^A \bmod p$

$g^B \bmod p$

compute $(g^B \bmod p)^A$ compute $(g^A \bmod p)^B$

agree on $g^{AB} \bmod p$

Man in the Middle



Signed Diffie-Hellman (Avoiding Man in the Middle)

Alice

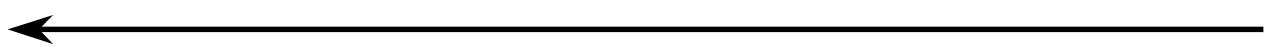
Bob

choose random A

choose random B

$[g^A \bmod p]$ signed with Alice's Private Key

$[g^B \bmod p]$ signed with Bob's Private Key



verify Bob's signature

verify Alice's signature

agree on $g^{AB} \bmod p$

But…if you have RSA keys...

- Why bother with Diffie-Hellman?
- Answer: PFS
  - If someone records the entire conversation, and later discovers Alice's and Bob's private keys, you don't want them to be able to decrypt
  - example without PFS (SSL): Alice chooses secret, encrypts it with Bob's PK, rest of session protected based on that secret

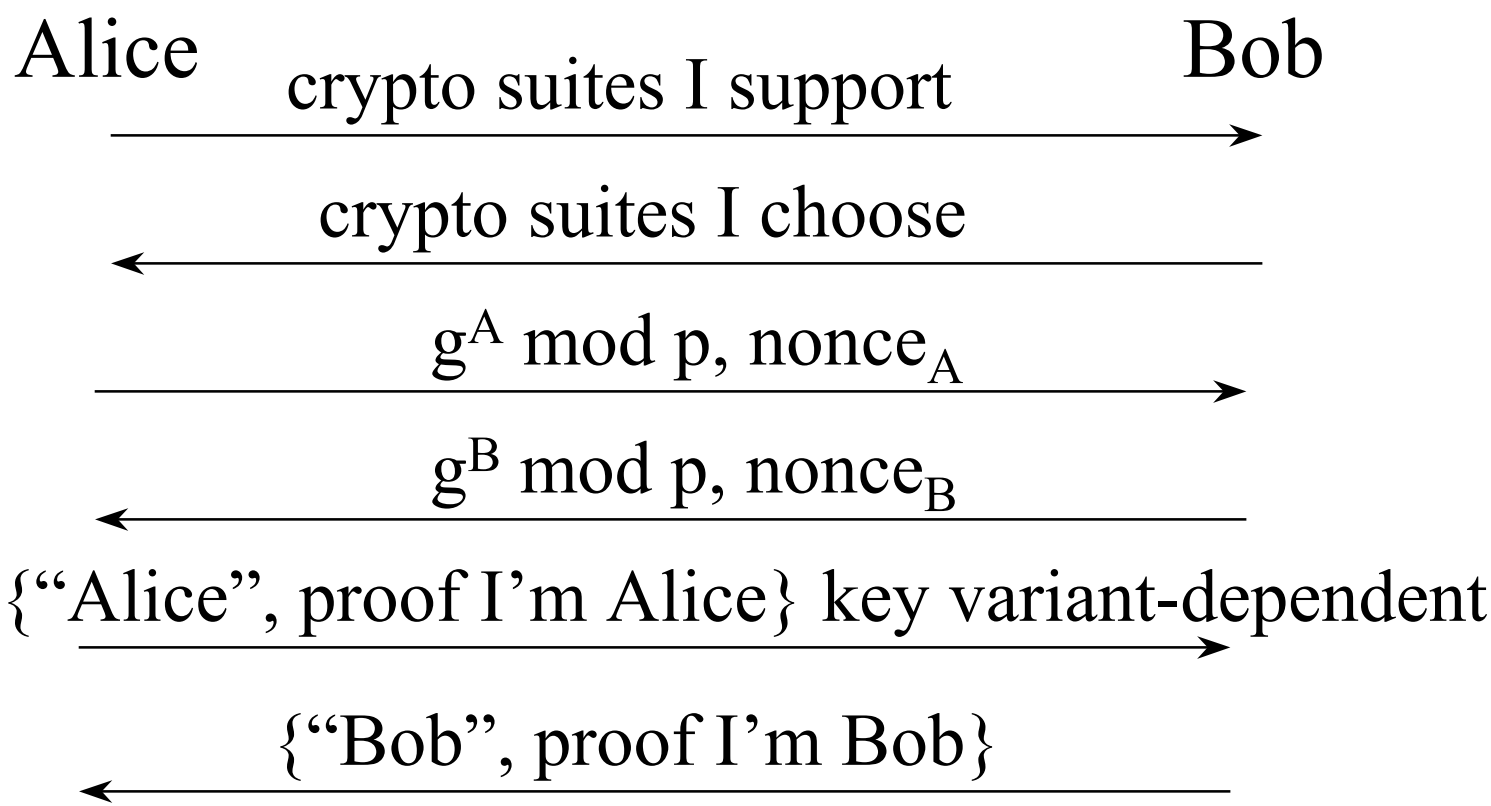
What are the nonces for?

- It's expensive to compute g^A mod p
- So, nice to reuse A (or for Bob to reuse B)
- Using nonces allows you to do that, and still get a new session key for each session
- Though if you remember A beyond a session, you lose PFS

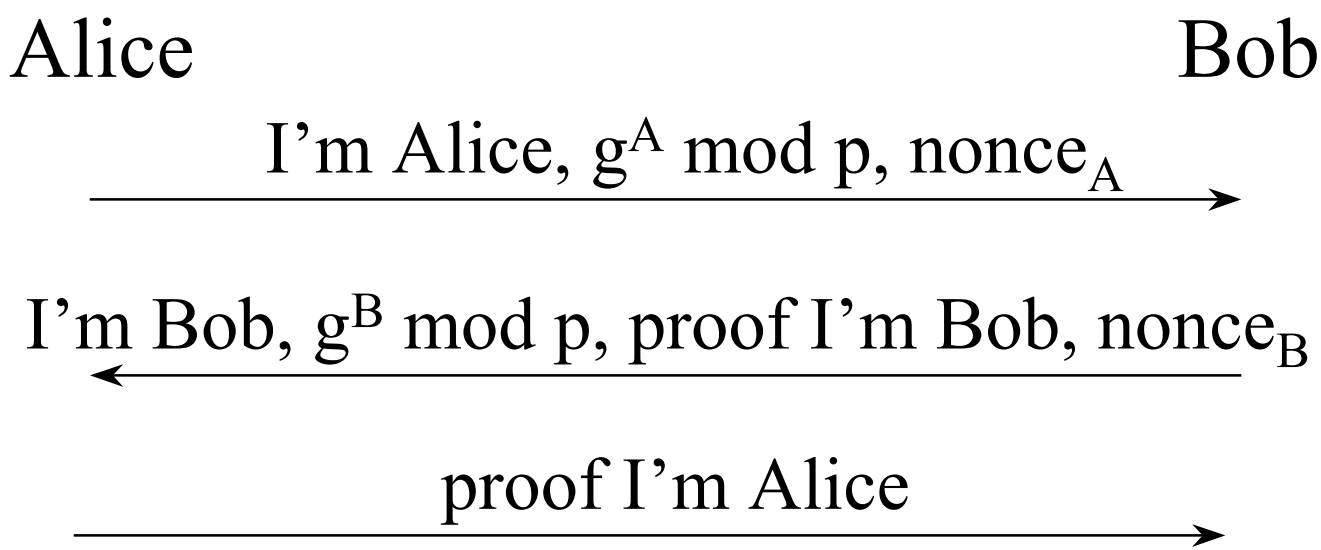
Now details. First IKEv1

- Two phases
- Phase 1 has 8 protocols!
  - two “modes”
    - aggressive: 3 msgs. mutual auth and get session key
    - main: 6 msgs. that, plus ID hiding
  - For each mode, a protocol for each key type
    - preshared secret key, signature PK, encryption PK (old crufty way), encryption PK (improved way)
- So, 9 protocols (4*2 phase 1, plus phase 2)

General Idea of IKEv1 Main-Mode



General Idea of IKEv1 Aggressive-Mode

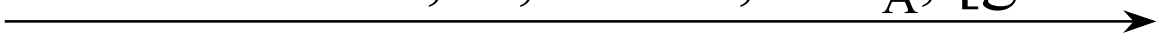


Which of 8 is “MUST”?

- Main mode, preshared key = $S_{Alice\text{-}Bob}$
- Alice sends: {“Alice”, proof} $f(S_{Alice\text{-}Bob})$
- Bob can’t decrypt that unless he knows who he’s talking to!
- So the WG said “your ID has to be your IP addr”
- But then why do 6-msg main mode to hide it?
- And it doesn’t work for “road warrior”
- So most IKEv1s are aggressive mode, preshared

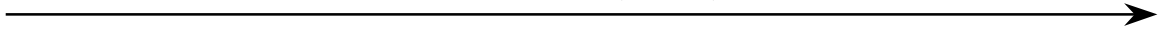
General idea of IKEv1 “quick mode” (phase 2)

IKE-SA, Y, traffic, SPI_A, $[g^A \bmod p]$



IKE-SA, Y, traffic, SPI_B, $[g^B \bmod p]$

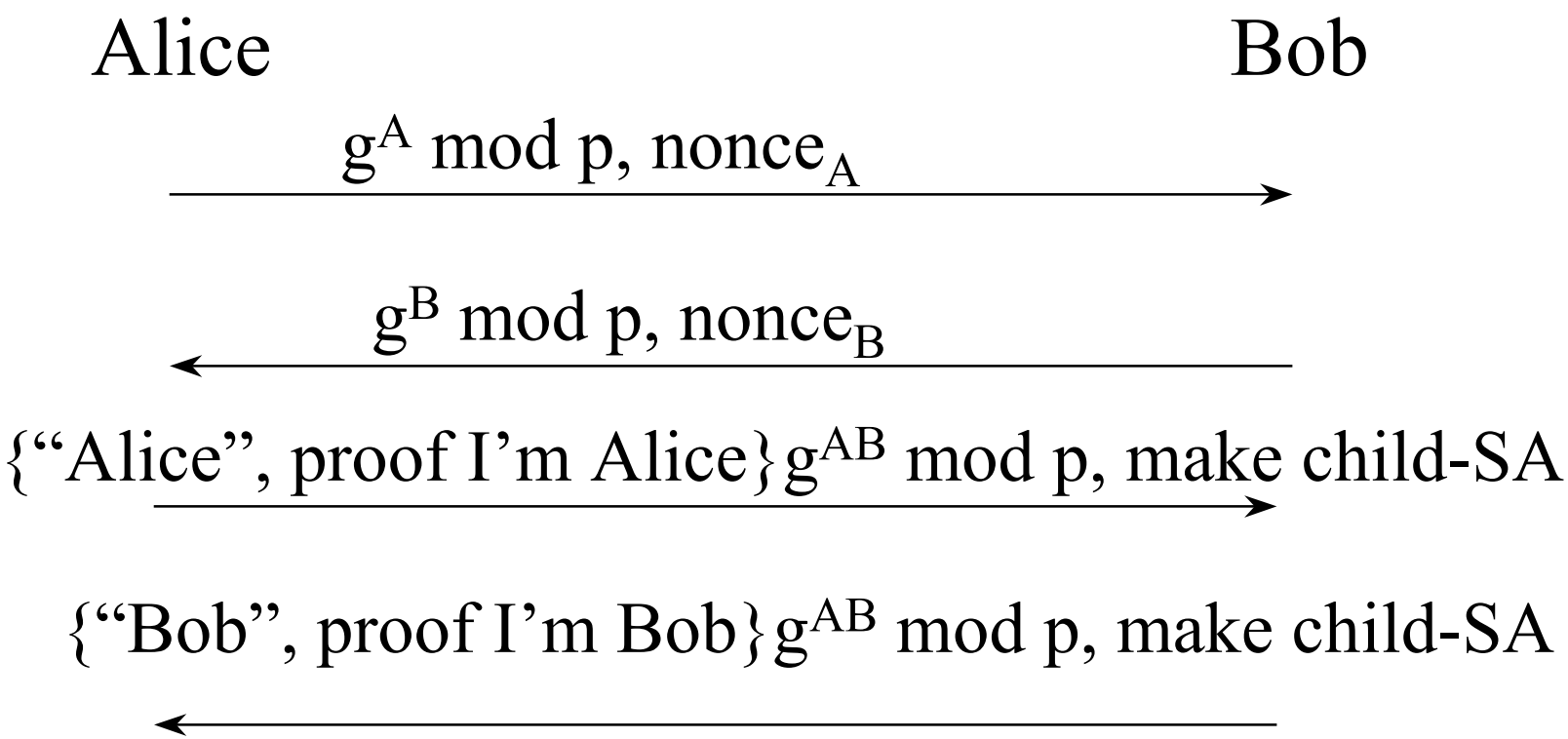
IKE-SA, Y, ack



IKEv2

- Greatly cleaned up and simplified
- Tried not to make gratuitous changes, so code reuse when possible
- Initial version much simpler, then added
  - NAT traversal, legacy authentication, internal address assignment
  - Copied those from how it was done in IKEv1

General idea of IKEv2



Traffic Restrictions

- IPsec policy: Traffic between these sets of IP adds, and protocol types, and ports, must have this sort of cryptographic protection
- Creating SA, specify “traffic selectors”
- IKEv1: Initiator proposes. Responder (if has more restrictive policy) can just say “no”
- IKEv2: allowed responder to narrow or say “single pair”

Working Through Firewalls and NATs

- Firewalls might not pass ESP packets
- Endnodes may share IP addresses (distinguishing them using ports)
- UDP encapsulation used to get through (make ports visible to NAPT)

Varying Authentication Methods

- X.509 certificates
  - Naming trusted certifiers
- User name and password
- SecurID or challenge/response cards
- Smart cards
- Kerberos

The Dream of IPsec End to End

- IPsec envisioned to replace SSL and be standard way to cryptographically protect all communications
- Protocol itself supports this
- Deployments don’t

What would it take?

- Policy encoding: how does a node know whether to require (or attempt) IPsec
- Certificate policies: what should be the requirements for certificates authenticating service X (or might keys be in DNS)?
- APIs – How does an application ask the OS the authenticated name of the other end of a connection?

What are the prospects?

- Not good… SSL and SSH “good enough”
- Hard policy and naming issues without an organizing force

Conclusions

- *Until a few years ago, you could connect to the Internet and be in contact with hundreds of millions of other nodes, without giving even a thought to security. The Internet in the '90's was like sex in the '60's. It was great while it lasted, but it was inherently unhealthy and was destined to end badly. I'm just really glad I didn't miss out again this time.*

  — from "Network Security: Private Communication in a Public World"