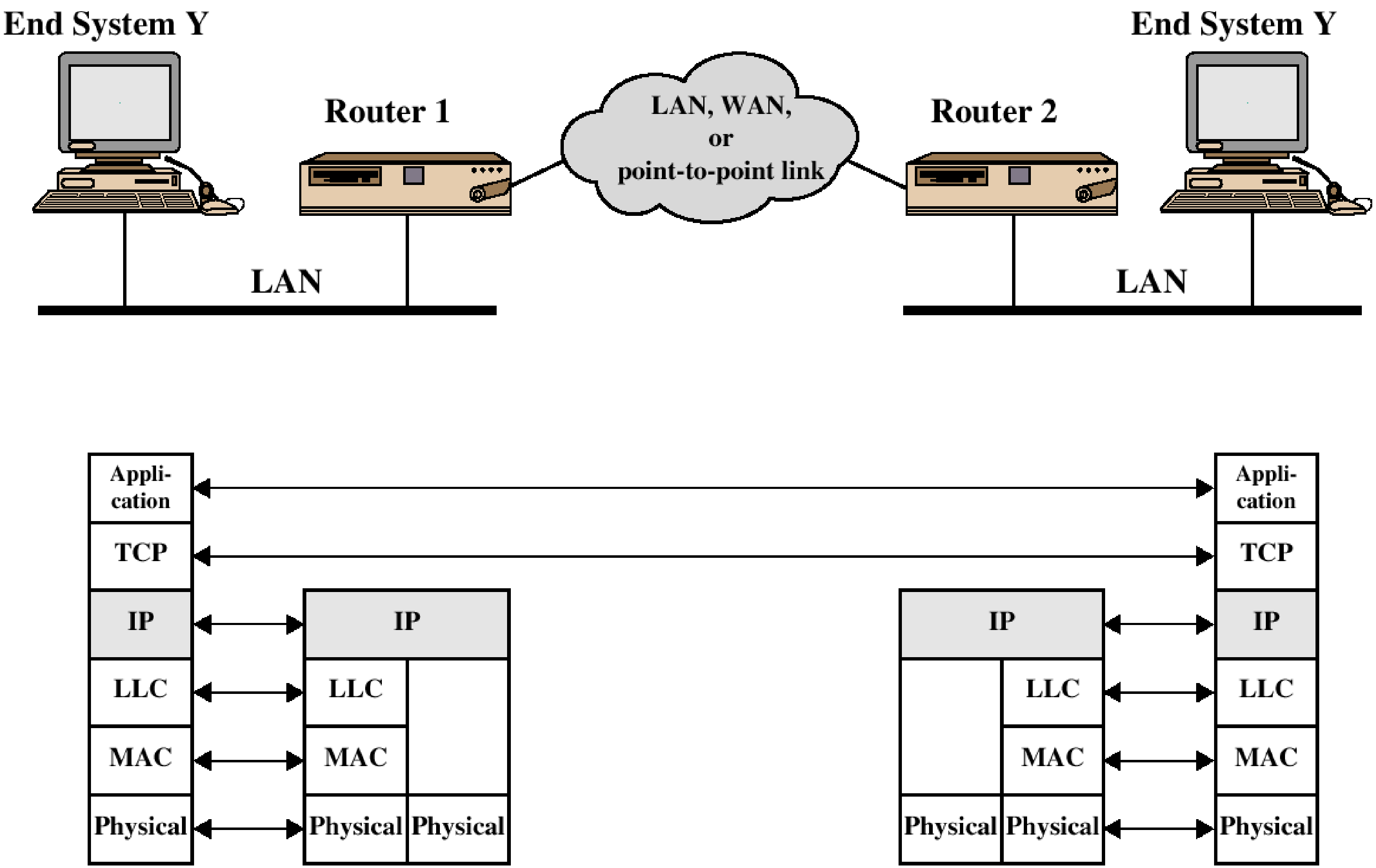
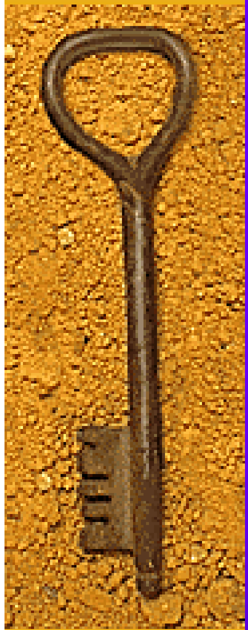
IPSec

TCP/IP Example

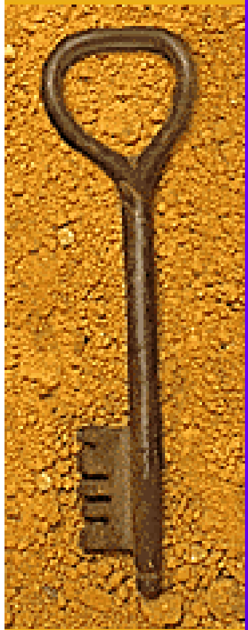




IP Security Issues

- Eavesdropping
- Modification of packets in transit
- Identity spoofing (forged source IP addresses)
- Denial of service

- Many solutions are application-specific
  - TLS for Web, S/MIME for email, SSH for remote login
- IPsec aims to provide a framework of open standards for secure communications over IP
  - Protect <u>every</u> protocol running on top of IPv4 and IPv6



IPsec: Network Layer Security

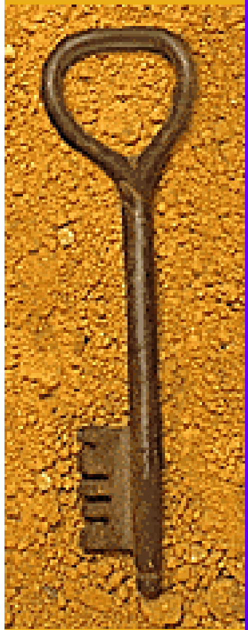
IPsec = AH + ESP + IPcomp + IKE

Protection for IP traffic
AH provides integrity and origin authentication
ESP also confidentiality

Compression

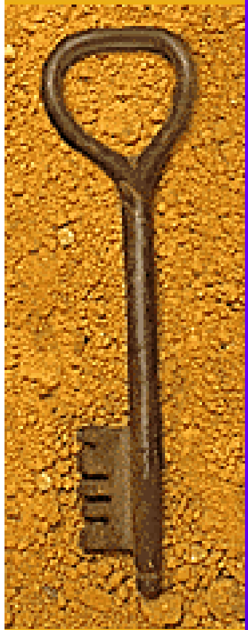
Sets up keys and algorithms for AH and ESP

- AH and ESP rely on an existing security association
  - Idea: parties must share a set of secret keys and agree on each other's IP addresses and crypto algorithms
- Internet Key Exchange (IKE)
  - Goal: establish security association for AH and ESP
  - If IKE is broken, AH and ESP provide no protection!



IPsec Security Services

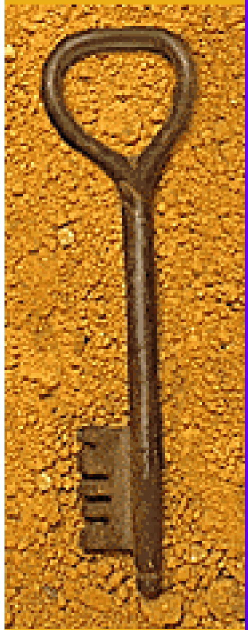
- Authentication and integrity for packet sources
  - Ensures connectionless integrity (for a single packet) and partial sequence integrity (prevent packet replay)
- Confidentiality (encapsulation) for packet contents
  - Also partial protection against traffic analysis
- Authentication and encapsulation can be used separately or together
- Either provided in one of two <u>modes</u>
- These services are transparent to applications above transport (TCP/UDP) layer

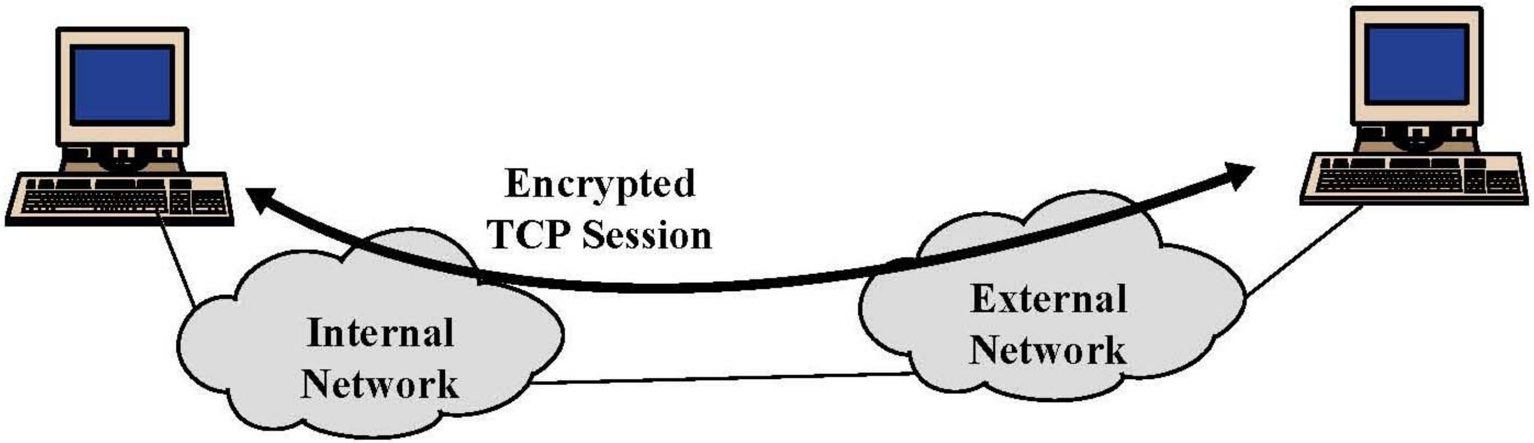


IPsec Modes

- Transport mode
  - Used to deliver services from host to host or from host to gateway
  - Usually within the same network, but can also be end-to-end across networks
- Tunnel mode
  - Used to deliver services from gateway to gateway or from host to gateway
  - Usually gateways owned by the same organization
    - With an insecure network in the middle

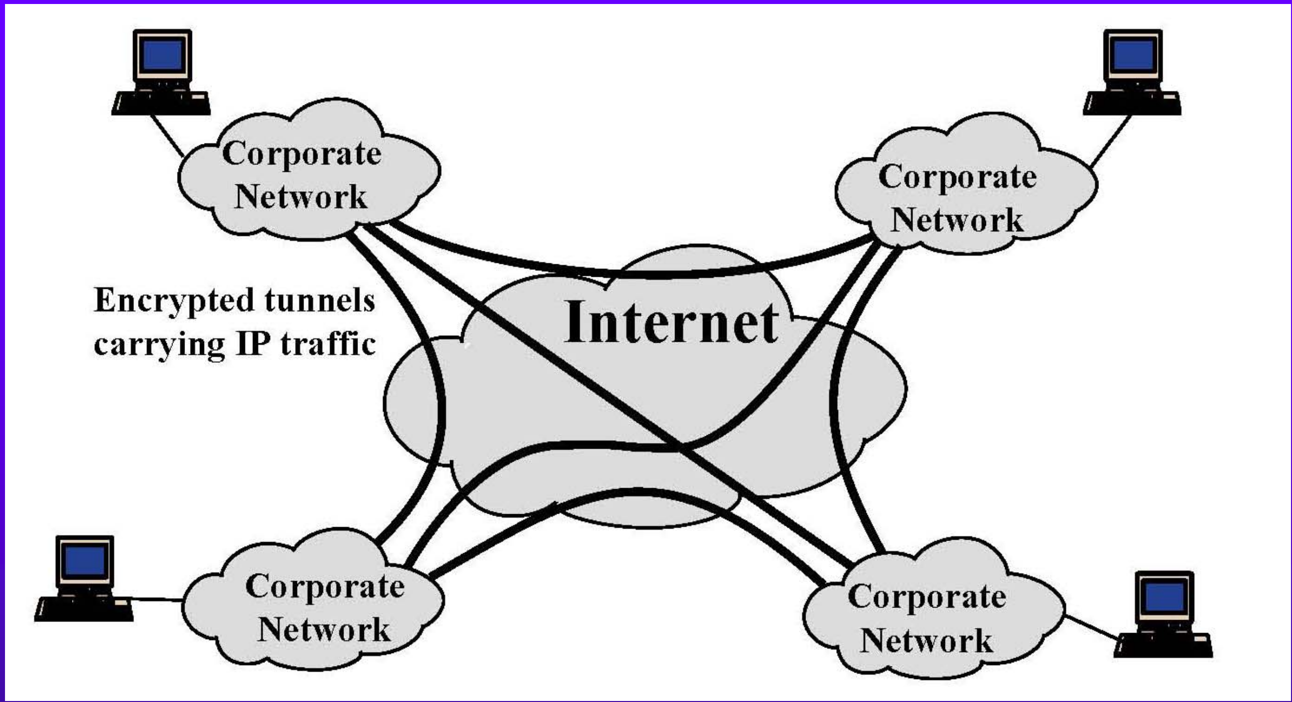
IPsec in Transport Mode





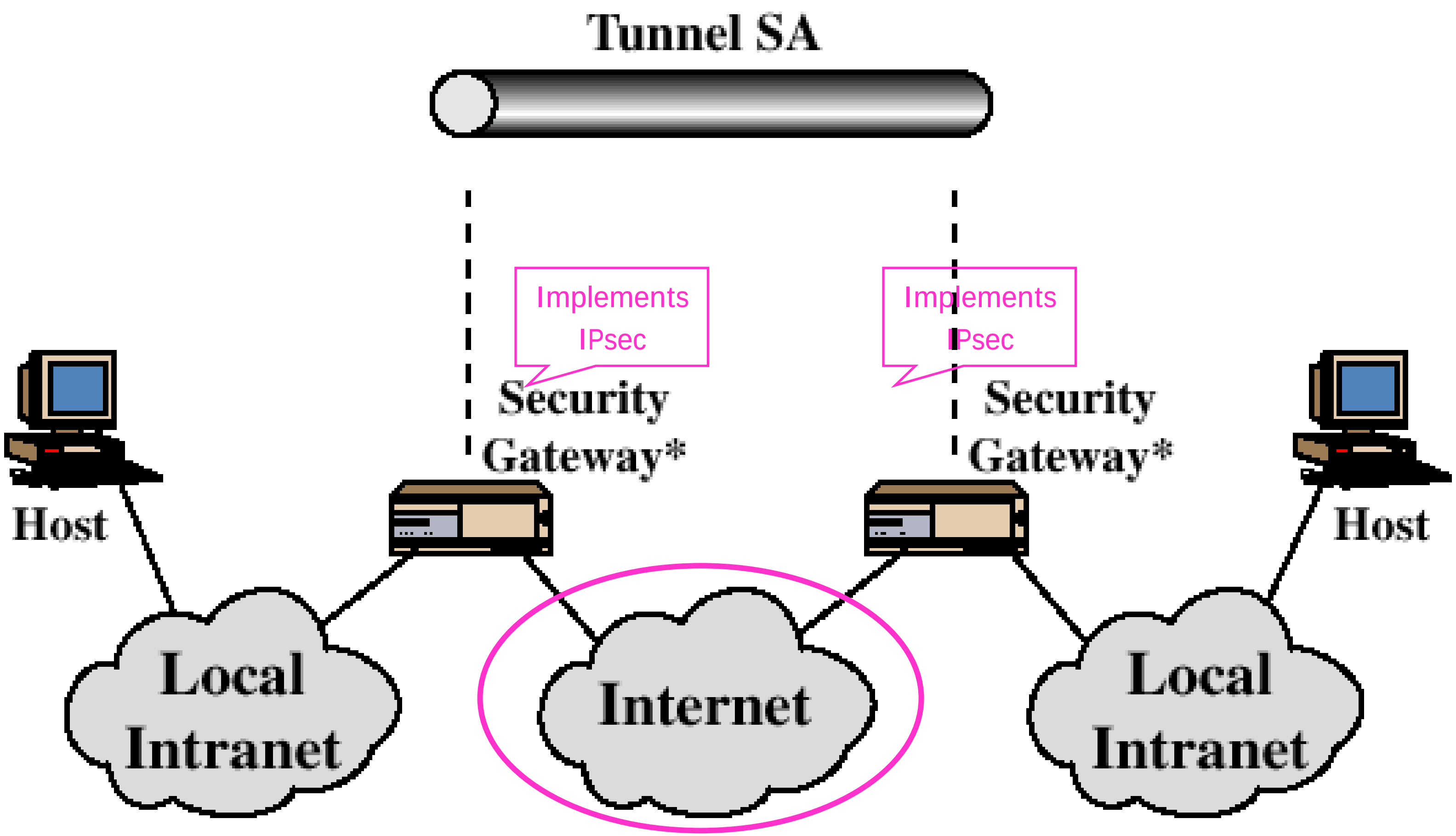
- End-to-end security between two hosts
  - Typically, client to gateway (e.g., PC to remote host)
- Requires IPsec support at each host

IPsec in Tunnel Mode

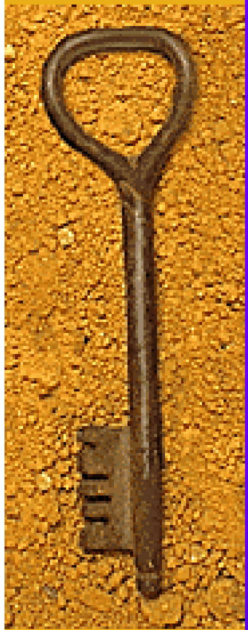


- Gateway-to-gateway security
  - Internal traffic behind gateways not protected
  - Typical application: virtual private network (VPN)
- Only requires IPsec support at gateways

Tunnel Mode Illustration



IPsec protects communication on the insecure part of the network



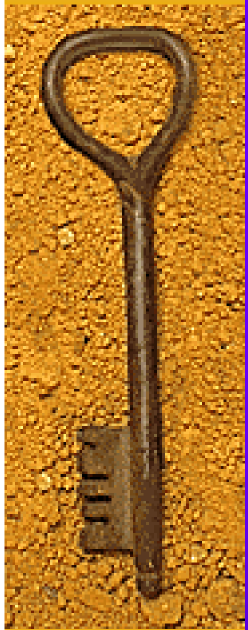
Transport Mode vs. Tunnel Mode

- Transport mode secures packet payload and leaves IP header unchanged

IP header (real dest)	IPSec header	TCP/UDP header + data

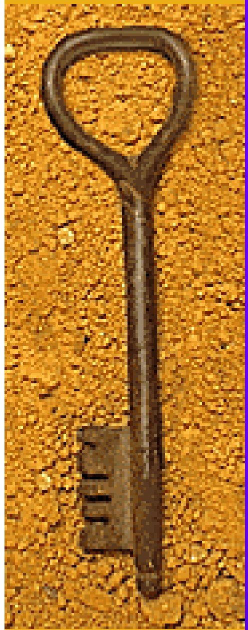
- Tunnel mode encapsulates both IP header and payload into IPsec packets

IP header (gateway)	IPSec header	IP header (real dest)	TCP/UDP header + data



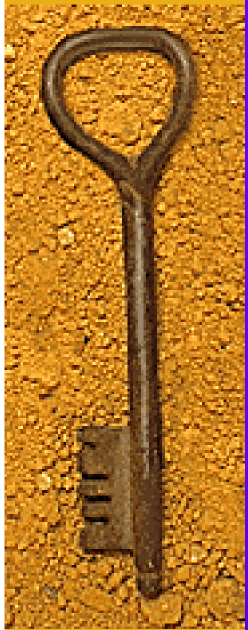
Security Association (SA)

- One-way sender-recipient relationship
- SA determines how packets are processed
  - Cryptographic algorithms, keys, IVs, lifetimes, sequence numbers, mode (transport or tunnel)
- SA is uniquely identified by SPI (Security Parameters Index)...
  - Each IPsec implementation keeps a database of SAs
  - SPI is sent with packet, tells recipient which SA to use
- ...destination IP address, and
- ...protocol identifier (AH or ESP)



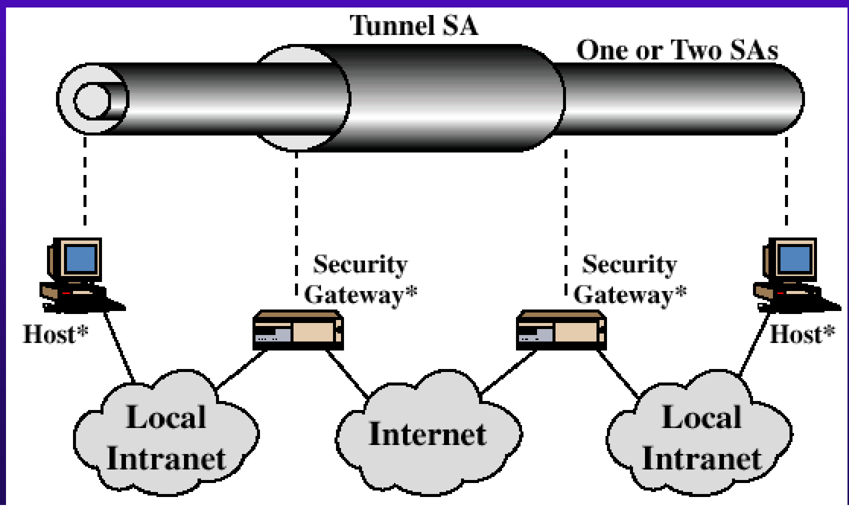
SA Components

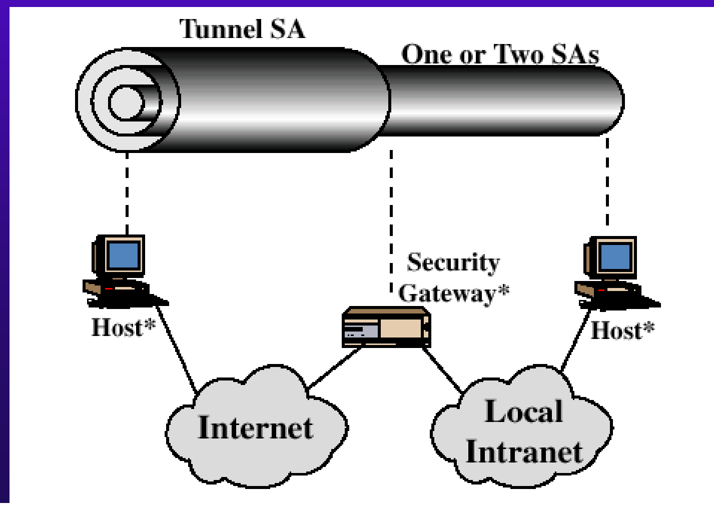
- Each IPsec connection is viewed as one-way so two SAs required for a two-way conversation
  - Hence need for Security Parameter Index
- Security association (SA) defines
  - Protocol used (AH, ESP)
  - Mode (transport, tunnel)
  - Encryption or hashing algorithm to be used
  - Negotiated keys and key lifetimes
  - Lifetime of this SA
  - … plus other info

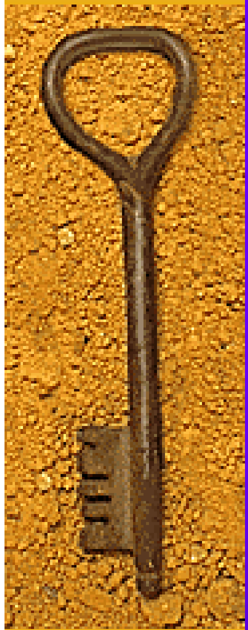


Security Association Issues

- How is SA established?
  - How do parties negotiate a common set of cryptographic algorithms and keys to use?
- More than one SA can apply to a packet!
  - E.g., end-to-end authentication (AH) and additional encryption (ESP) on the public part of the network







AH: Authentication Header

- Sender authentication
- Integrity for packet contents and IP header
- Sender and receiver must share a <u>secret key</u>
  - This key is used in HMAC computation
  - The key is set up by IKE key establishment protocol and recorded in the Security Association (SA)
    - SA also records protocol being used (AH) and mode (transport or tunnel) plus hashing algorithm used
    - MD5 or SHA-1 supported as hashing algorithms

IP Headers

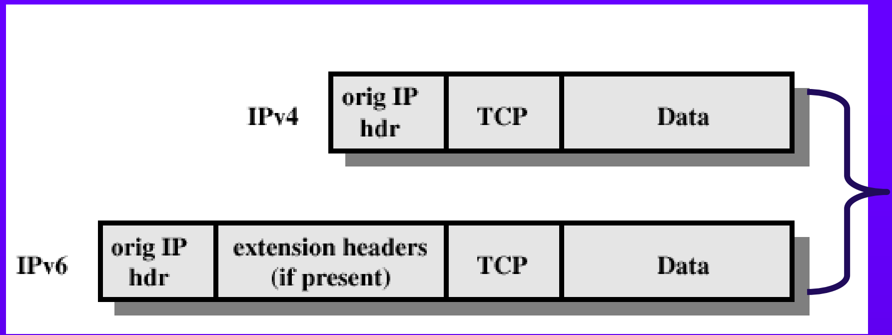
Version	Header Length	TOS	Packet length	Packet Id	Flags

Mutable Immutable Predictable

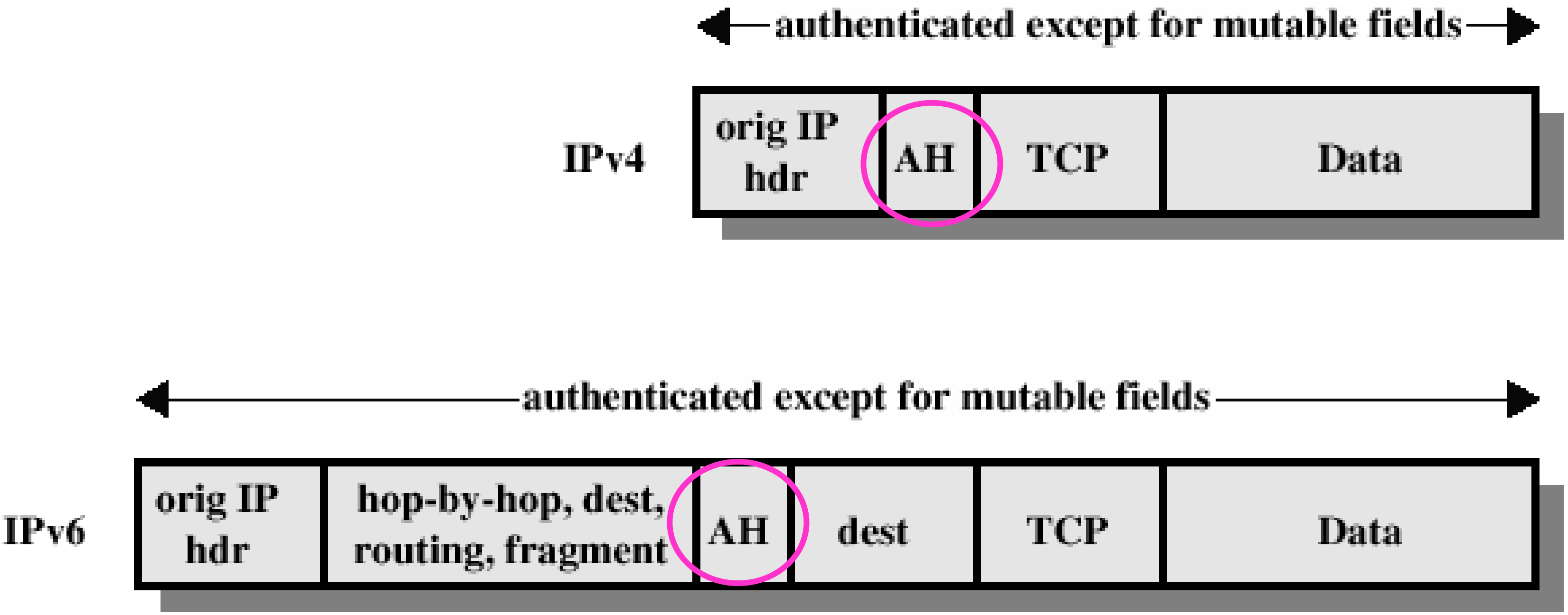
Fragment offset	TTL	Protocol number	Checksum	Source IP address	Destination IP address	Options

AH sets mutable fields to zero and predictable fields to final value and then uses this header plus packet contents as input to HMAC

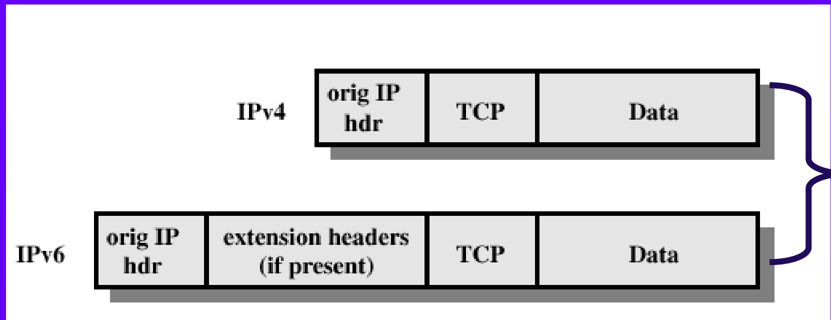
AH in Transport Mode



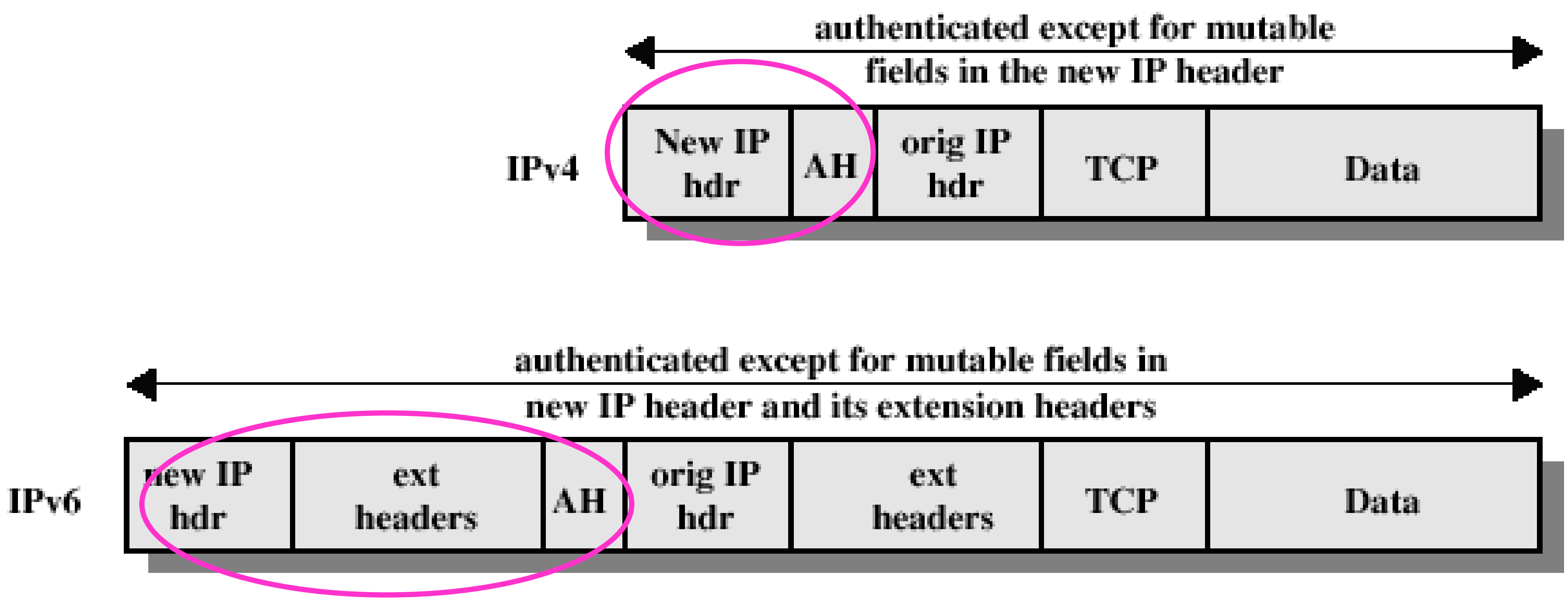
Before AH is applied

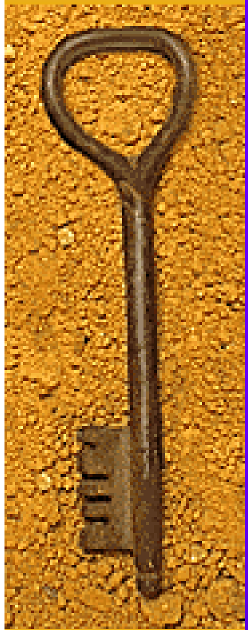


AH in Tunnel Mode



Before AH is applied





Authentication Header Format

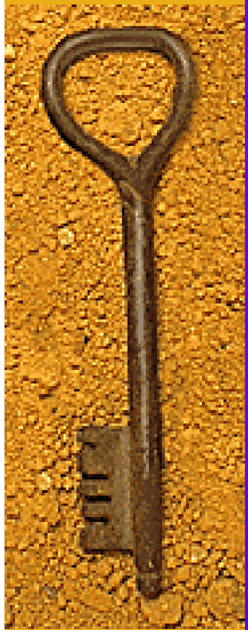
- Provides integrity and origin authentication
- Authenticates portions of the IP header
- Anti-replay service (to counter denial of service)
- No confidentiality

Next header (TCP)	Payload length	*Reserved*
Security parameters index (SPI)		
Sequence number		
ICV: Integrity Check Value (HMAC of IP header, AH, TCP payload)		

Identifies security association (shared keys and algorithms)

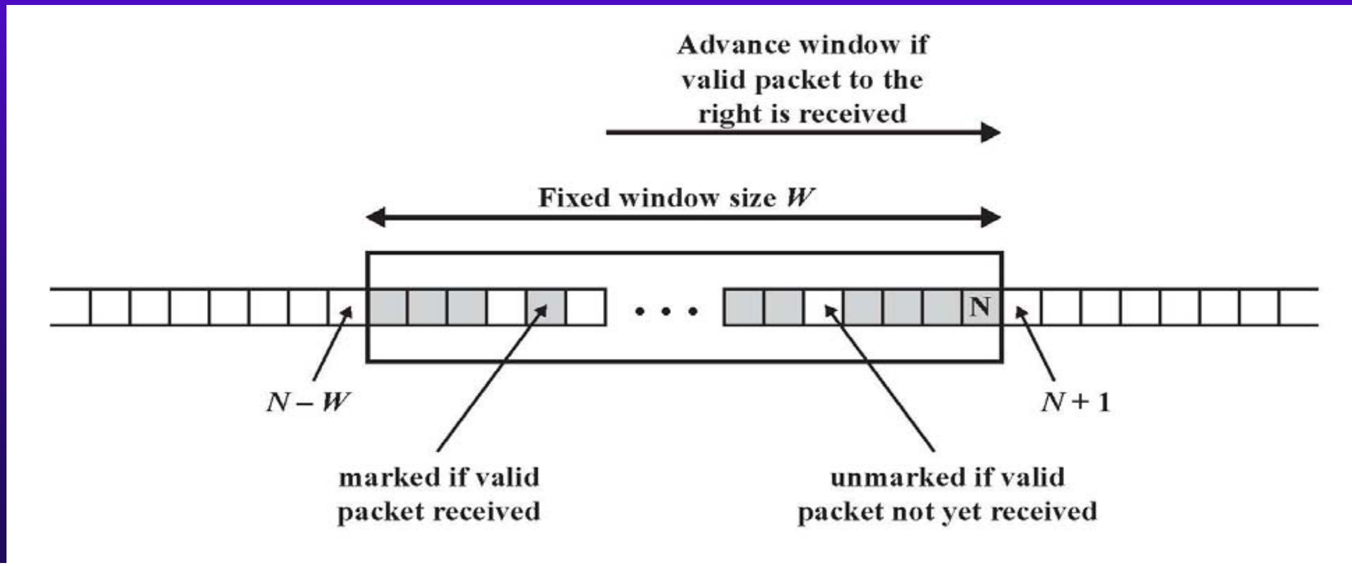
Anti-replay

Authenticates source, verifies integrity of payload

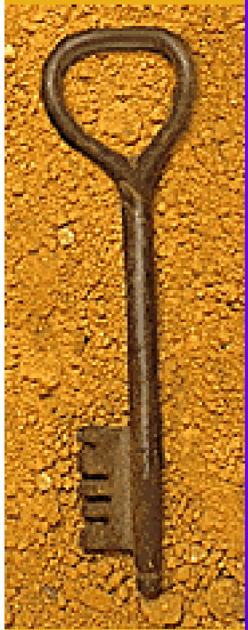


Prevention of Replay Attacks

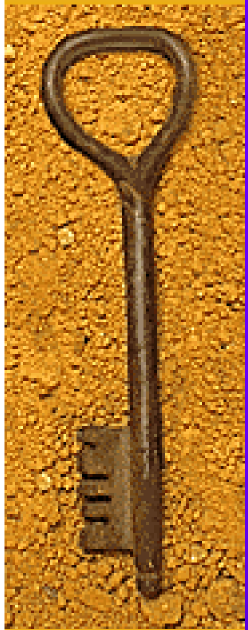
- When SA is established, sender initializes 32-bit counter to 0, increments by 1 for each packet
  - If wraps around 2^{32}-1, new SA must be established
- Recipient maintains a sliding 64-bit window
  - If a packet with high sequence number is received, do not advance window until packet is authenticated



ESP: Encapsulating Security Payload

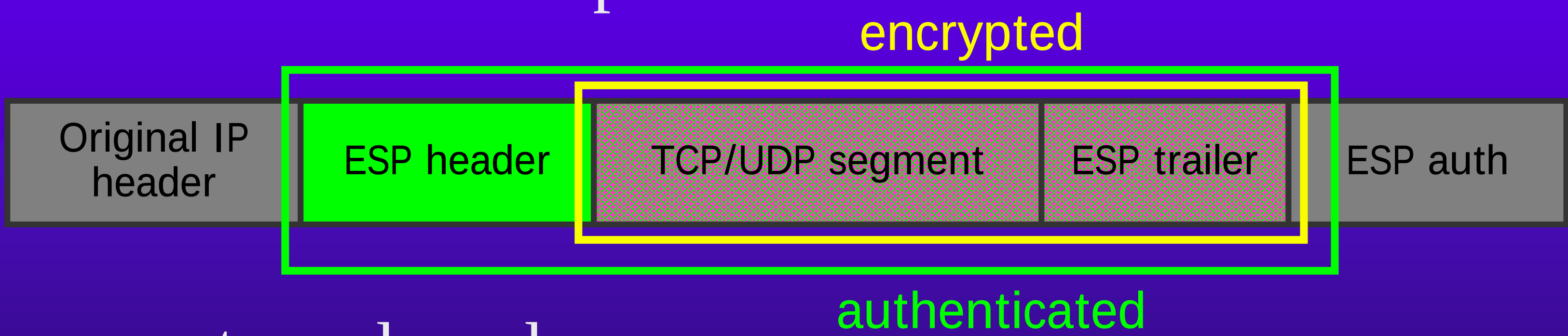


- Adds new header and trailer fields to packet
- Transport mode
  - Confidentiality of packet between two hosts
  - Complete hole through firewalls
  - Used sparingly
- Tunnel mode
  - Confidentiality of packet between two gateways or a host and a gateway
  - Implements VPN tunnels

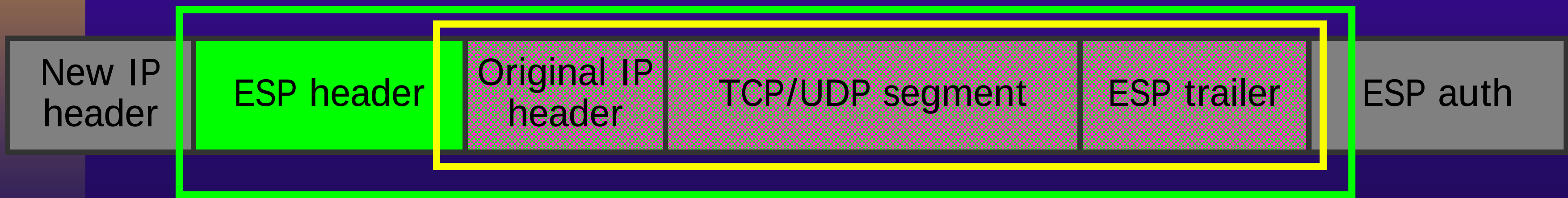


ESP Security Guarantees

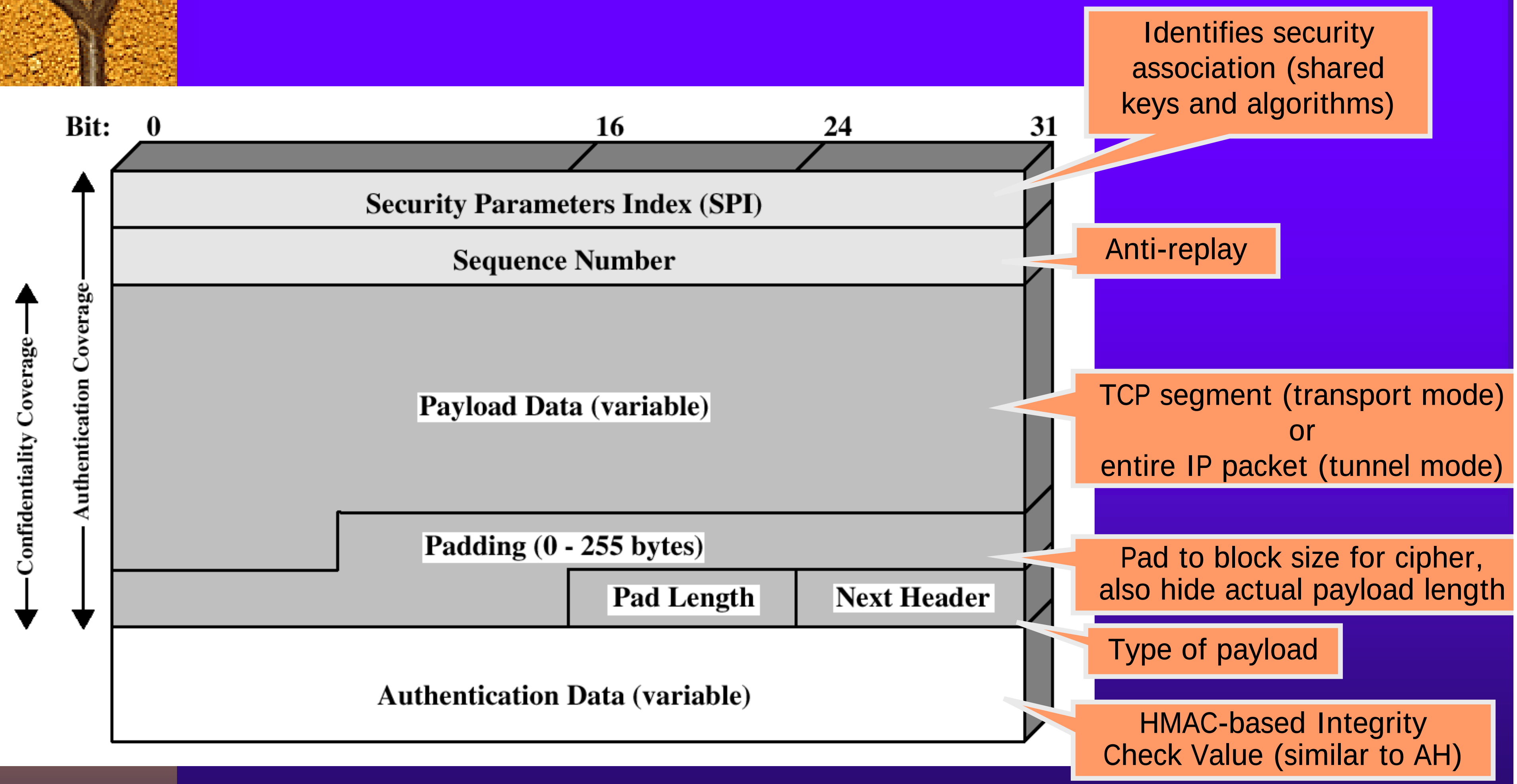
- Confidentiality and integrity for packet payload
  - Symmetric cipher negotiated as part of security assoc
- <u>Optionally</u> provides authentication (similar to AH)
- Can work in transport mode



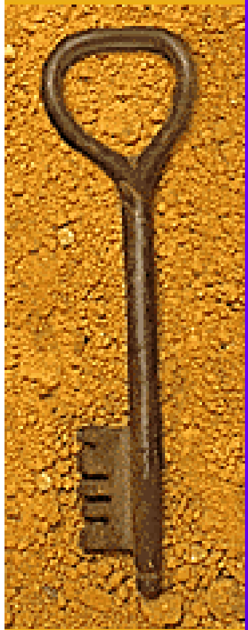
- ...or tunnel mode



ESP Packet



Bit: 0 16 24 31
Security Parameters Index (SPI)
Sequence Number
Payload Data (variable)
Padding (0 - 255 bytes)
Pad Length
Next Header
Authentication Data (variable)
Confidentiality Coverage
Authentication Coverage
Identifies security association (shared keys and algorithms)
Anti-replay
TCP segment (transport mode) or entire IP packet (tunnel mode)
Pad to block size for cipher, also hide actual payload length
Type of payload
HMAC-based Integrity Check Value (similar to AH)

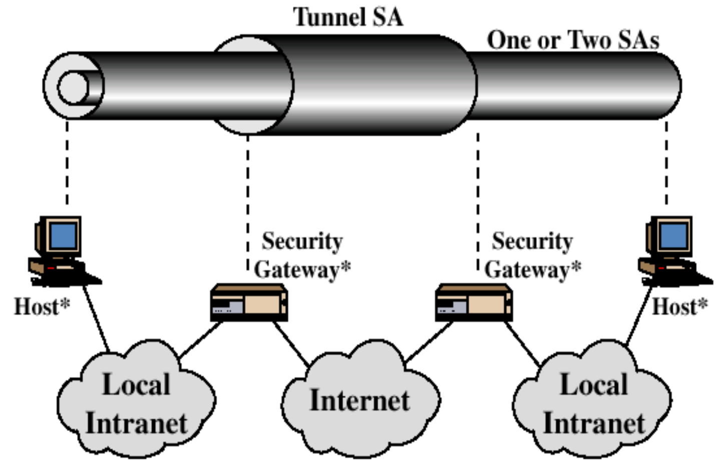


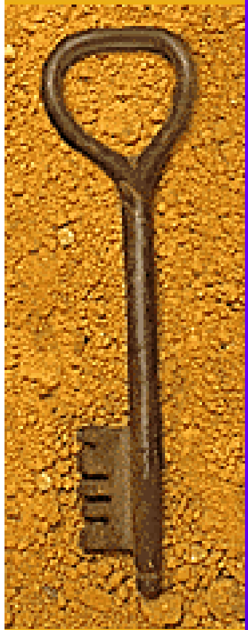
Virtual Private Networks (VPN)

- ESP is often used to implement a VPN
  - Packets go from internal network to a gateway with TCP / IP headers for address in another network
  - Entire packet hidden by encryption
    - Including original headers so destination addresses are hidden
  - Receiving gateway decrypts packet and forwards original IP packet to receiving address in the network that it protects
- This is known as a VPN tunnel
  - Secure communication between parts of the same organization over public untrusted Internet

ESP Together With AH

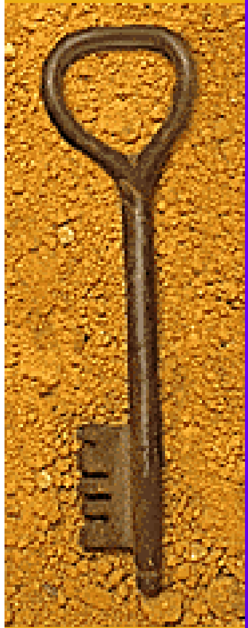
- AH and ESP are often combined
- End-to-end AH in transport mode
  - Authenticate packet sources
- Gateway-to-gateway ESP in tunnel mode
  - Hide packet contents and addresses on the insecure part of the network
- Significant cryptographic overhead
  - Even with AH





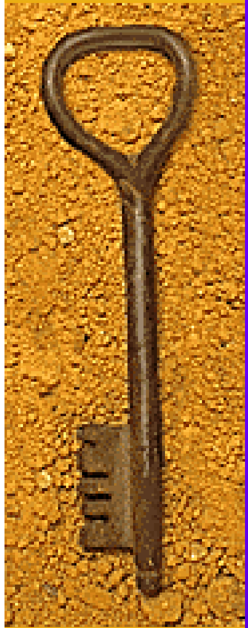
Secure Key Establishment

- Goal: generate and agree on a session key using some public initial information
- What properties are needed?
  - Authentication (know identity of other party)
  - Secrecy (generated key not known to any others)
  - Forward secrecy (compromise of one session key does not compromise keys in other sessions)
  - Prevent replay of old key material
  - Prevent denial of service
  - Protect identities from eavesdroppers
  - Other properties can you think of???



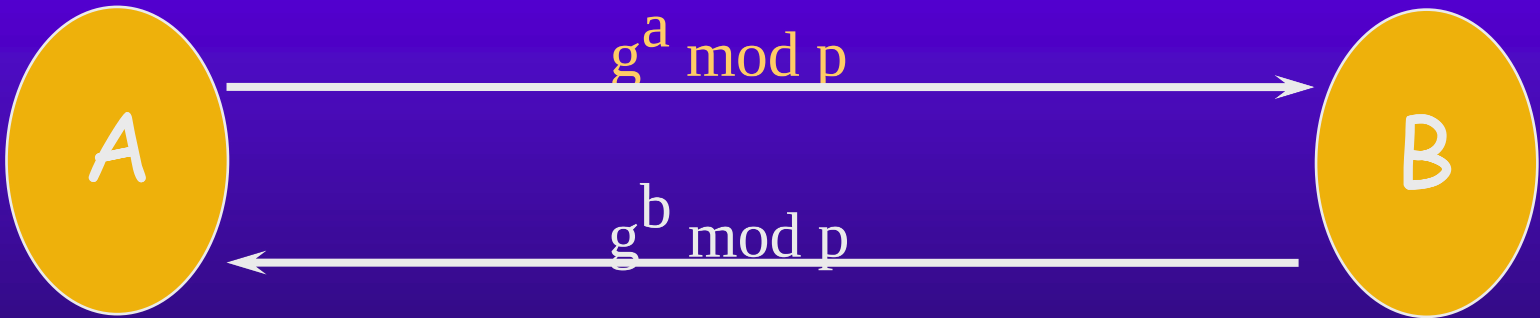
Key Management in IPsec

- Manual key management
  - Keys and parameters of crypto algorithms exchanged offline (e.g., by phone), security associations established by hand
- Pre-shared symmetric keys
  - New session key derived for each session by hashing pre-shared key with session-specific nonces
  - Standard symmetric-key authentication and encryption
- Online key establishment
  - Internet Key Exchange (IKE) protocol
  - Use Diffie-Hellman to derive shared symmetric key



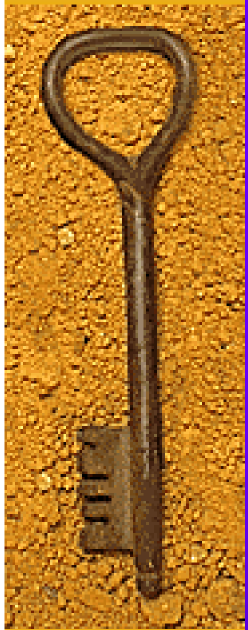
Diffie-Hellman Key Exchange

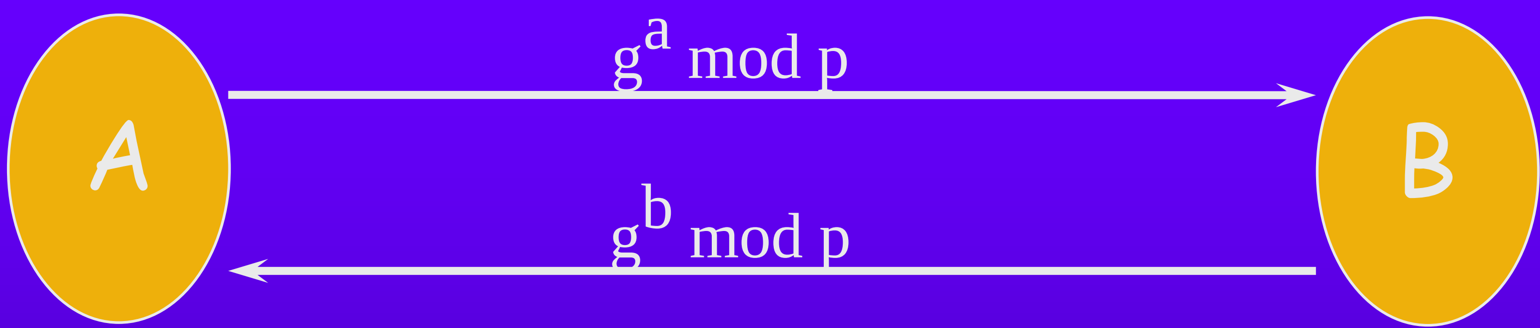
- Assume finite group $G = \langle S, \bullet \rangle$
  - Choose generator g so every $x \in S$ is $x = g^i$ for some i
  - Example: squares modulo prime p (even i)
- Protocol



Alice, Bob share g^{ab} mod p not known to anyone else

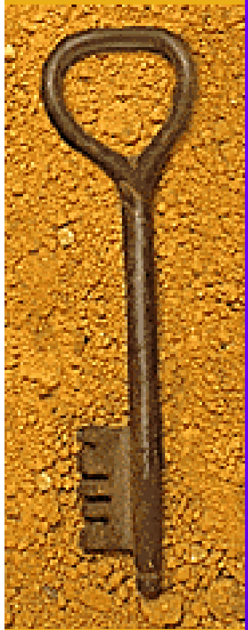
Diffie-Hellman Key Exchange



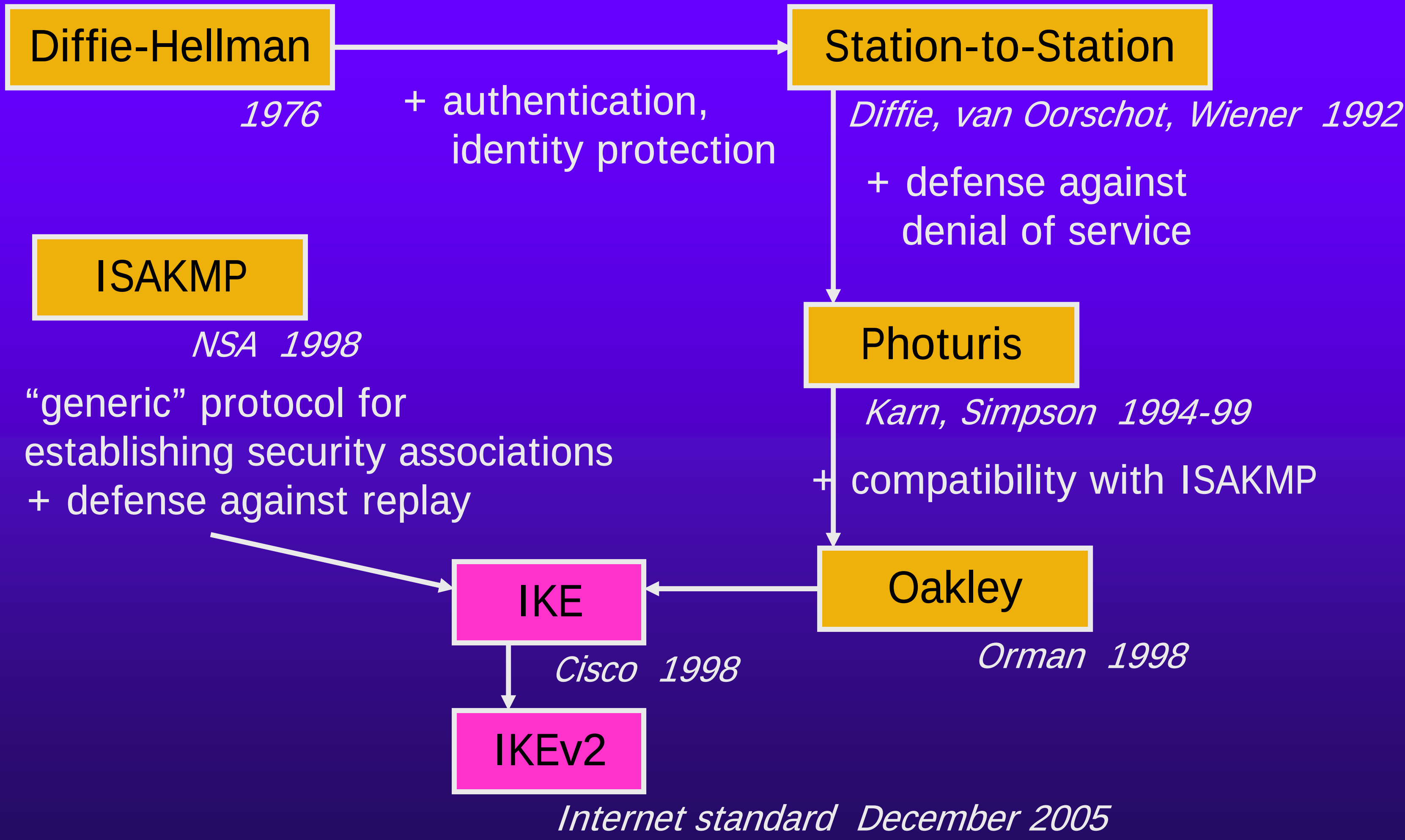


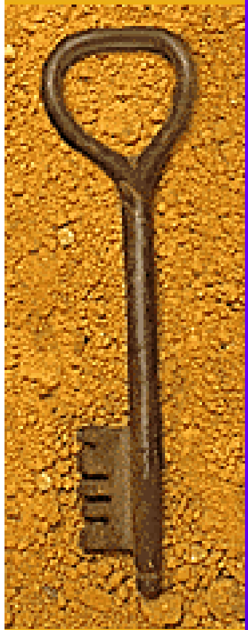
Authentication?	No
Secrecy?	Only against passive attacker
Replay attack?	Vulnerable
Forward secrecy?	Yes
Denial of service?	Vulnerable
Identity protection?	Yes

Participants can't tell $g^x \bmod p$ from a random element of G: send them garbage and they'll do expensive exponentiations



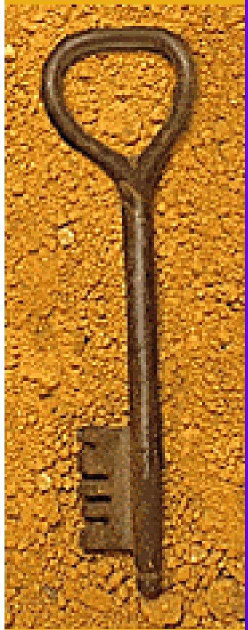
IKE Genealogy





Design Objectives for Key Exchange

- Shared secret
  - Create and agree on a secret which is known only to protocol participants
- Authentication
  - Participants need to verify each other's identity
- Identity protection
  - Eavesdropper should not be able to infer participants' identities by observing protocol execution
- Protection against denial of service
  - Malicious participant should not be able to exploit the protocol to cause the other party to waste resources

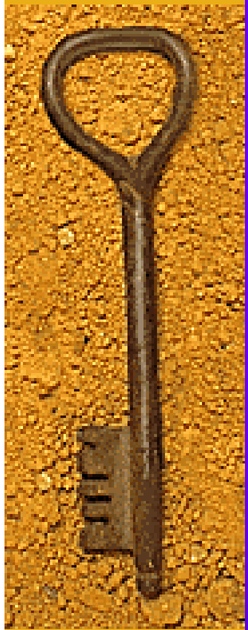


Ingredient 1: Diffie-Hellman

$A \rightarrow B: \quad g^a$

$B \rightarrow A: \quad g^b$

- Shared secret is g^{ab}, compute key as $k=hash(rand,g^{ab})$
  - Diffie-Hellman guarantees perfect forward secrecy
- Authentication
- Identity protection
- DoS protection



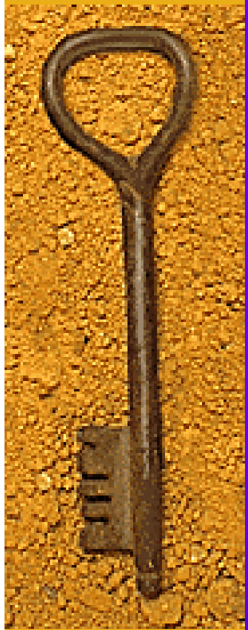
Ingredient 2: Challenge-Response

$$A \rightarrow B: \; m, A$$

$$B \rightarrow A: \; n, sig_B(m, n, A)$$

$$A \rightarrow B: \; sig_A(m, n, B)$$

- Shared secret
- Authentication
  - A receives his own number m signed by B's private key and deduces that B is on the other end; similar for B
- Identity protection
- DoS protection



DH + Challenge-Response

ISO 9798-3 protocol:

$A \rightarrow B$: g^a, A

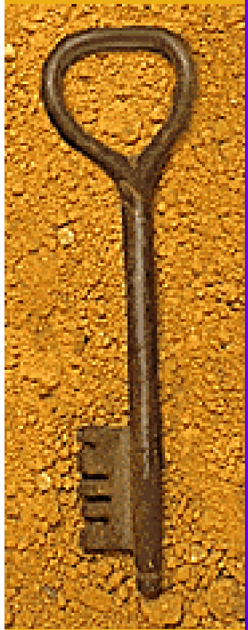
$B \rightarrow A$: g^b, $sig_B(g^a, g^b, A)$

$A \rightarrow B$: $sig_A(g^a, g^b, B)$

$m := g^a$

$n := g^b$

- Shared secret: g^{ab}
- Authentication
- Identity protection
- DoS protection



Ingredient 3: Encryption

Encrypt signatures to protect identities:

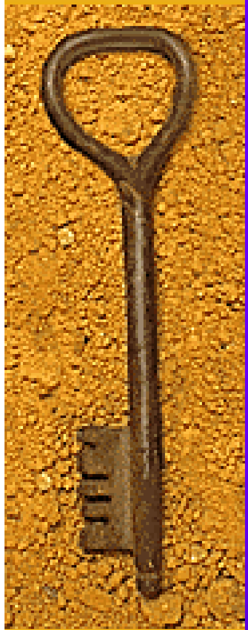
$$A \rightarrow B:\ \ g^a, A$$

$$B \rightarrow A:\ \ g^b, Enc_K(sig_B(g^a, g^b, A))$$

$$A \rightarrow B:\ \ Enc_K(sig_A(g^a, g^b, B))$$

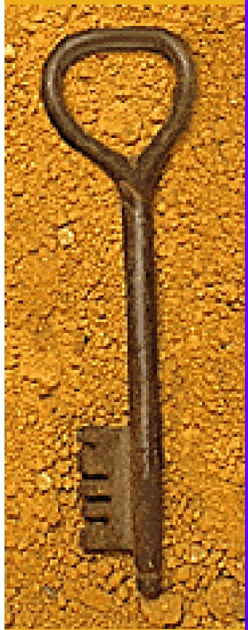
k=hash(g^{ab})

- Shared secret: g^{ab}
- Authentication
- Identity protection (for responder only!)
- DoS protection



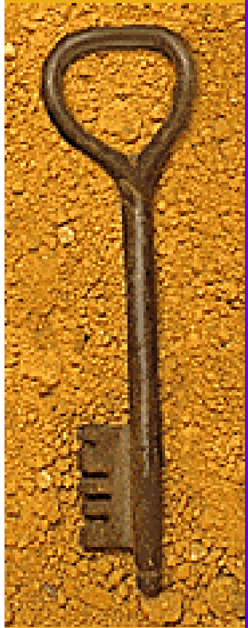
Refresher: DoS Prevention

- Denial of service due to resource clogging
  - If responder opens a state for each connection attempt, attacker can initiate thousands of connections from bogus or forged IP addresses
- Cookies ensure that the responder is stateless until initiator produced at least 2 messages
  - Responder's state (IP addresses and ports) is stored in an unforgeable cookie and sent to initiator
  - After initiator responds, cookie is regenerated and compared with the cookie returned by the initiator
  - The cost is 2 extra messages in each execution



Refresher: Anti-DoS Cookie

- Typical protocol:
  - Client sends request (message #1) to server
  - Server sets up connection, responds with message #2
  - Client may complete session or not (potential DoS)
- Cookie version:
  - Client sends request to server
  - Server sends hashed connection data back
    - Send message #2 later, after client confirms his address
  - Client confirms by returning hashed data
  - Need an extra step to send postponed message #2



Ingredient 4: Anti-DoS Cookie

"Almost-IKE" protocol:

Doesn't quite work: B must remember his DH exponent b for every connection

$A \rightarrow B$: g^a, A

$B \rightarrow A$: g^b, $hash_{Kb}(g^b, g^a)$

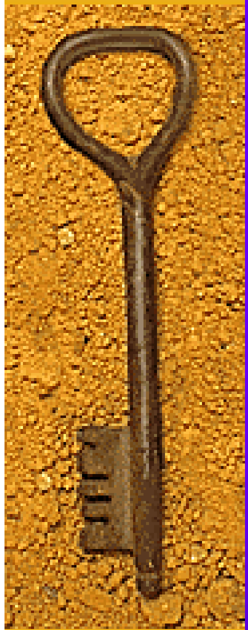
$A \rightarrow B$: g^a, g^b, $hash_{Kb}(g^b, g^a)$

$Enc_K(sig_A(g^a, g^b, B))$

$B \rightarrow A$: g^b, $Enc_K(sig_B(g^a, g^b, A))$

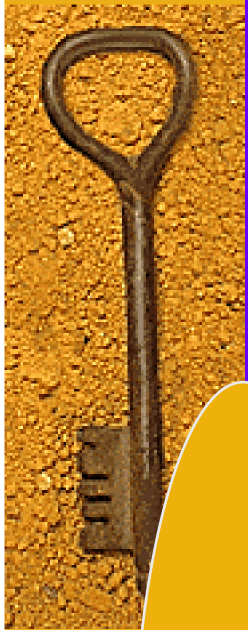
k=hash(g^{ab})

- Shared secret: g^{ab}
- Authentication
- Identity protection
- DoS protection?



Medium-Term Secrets and Nonces

- Idea: use the same Diffie-Hellman value g^{ab} for every session, update every 10 minutes or so
  - Helps against denial of service
- To make sure keys are different for each session, derive them from g^{ab} and session-specific nonces
  - Nonces guarantee freshness of keys for each session
  - Re-computing g^a, g^b, g^{ab} is costly, generating nonces (fresh random numbers) is cheap
- This is more efficient and helps with DoS, but no longer guarantees forward secrecy (why?)



(Simplified) Photuris

[Karn and Simpson]

Random number (identifies connection)

Hash(source & dest IP addrs, Cookie_I, ports, **R's local secret**)

I → R: Cookie_I

R → I: Cookie_I, Cookie_R, offer crypto

I → R: Cookie_I, Cookie_R, $g^a \bmod p$, select crypto

R → I: Cookie_I, Cookie_R, $g^b \bmod p$

I: *switch to $K=h(g^{ab} \bmod p)$*

R: *switch to $K=h(g^{ab} \bmod p)$*

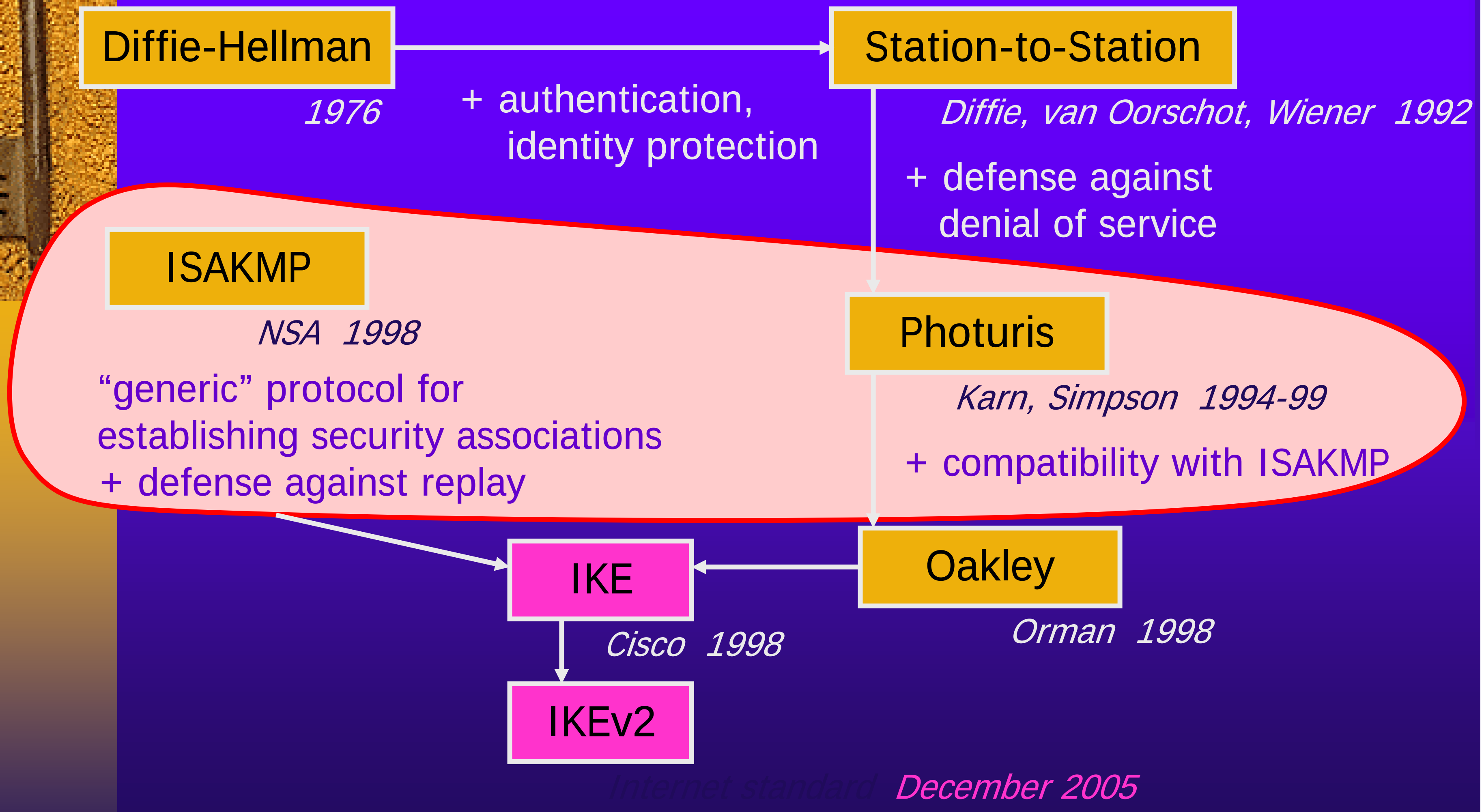
I → R: Cookie_I, Cookie_R, $\text{Enc}_K(\text{"I"}, \text{sig}_I(\textit{previous msgs}))$

R → I: Cookie_I, Cookie_R, $\text{Enc}_K(\text{"R"}, \text{sig}_R(\textit{previous msgs}))$

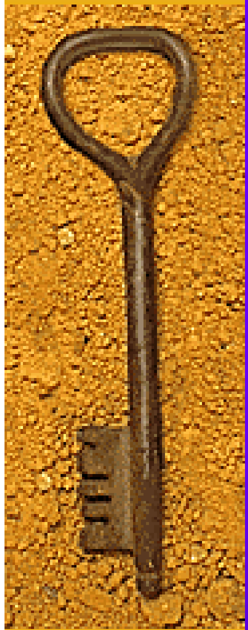
Alice is called "Initiator" for consistency with IKE terminology

Bob is called "Responder"

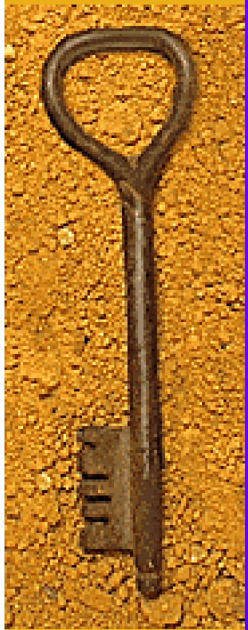
IKE Genealogy Redux



Cookies in Photuris and ISAKMP

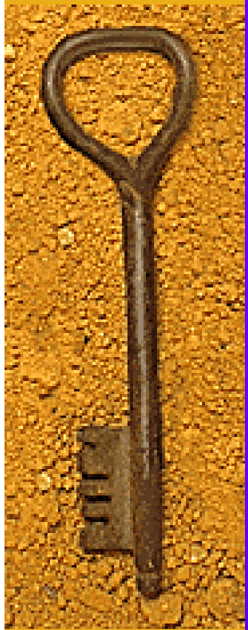


- Photuris cookies are derived from local secret, IP addresses and ports, counter, crypto schemes
  - Same (frequently updated) secret for all connections
- ISAKMP requires unique cookie for each connect
  - Add timestamp to each cookie to prevent replay attacks
  - Now responder needs to keep state (“cookie crumb”)
    - Vulnerable to denial of service (why?)
- Inherent conflict: to prevent replay, need to remember values that you’ve generated or seen before, but keeping state allows denial of service



IKE Overview

- Goal: create security association between 2 hosts
  - Shared encryption and authentication keys, agreement on crypto algorithms
- Two phases: 1st phase establishes security association (IKE-SA) for the 2nd phase
  - Always by authenticated Diffie-Hellman (expensive)
- 2nd phase uses IKE-SA to create actual security association (child-SA) to be used by AH and ESP
  - Use keys derived in the 1st phase to avoid DH exchange
  - Can be executed cheaply in "quick" mode
    - To create a fresh key, hash old DH value and new nonces



Why Two-Phase Design?

- Expensive 1st phase creates “main” SA
- Cheap 2nd phase allows to create multiple child SAs (based on “main” SA) between same 2 hosts
  - Example: one SA for AH, another SA for ESP
  - Different conversations may need different protection
    - Some traffic only needs integrity protection or short-key crypto
    - Too expensive to always use strongest available protection
  - Avoid multiplexing several conversations over same SA
    - For example, if encryption is used without integrity protection (bad idea!), it may be possible to splice the conversations
  - Different SAs for different classes of service

IKE: Phase One

Optional: refuse 1st message and demand return of stateless cookie

g^a mod p, crypto proposal, N_i

$Cookie_R$

$Cookie_R$, g^a mod p, crypto proposal, N_i

g^b mod p, crypto accepted, N_r

I

R

switch to $K=f(N_i,N_r,crypto,g^{ab} \text{ mod } p)$

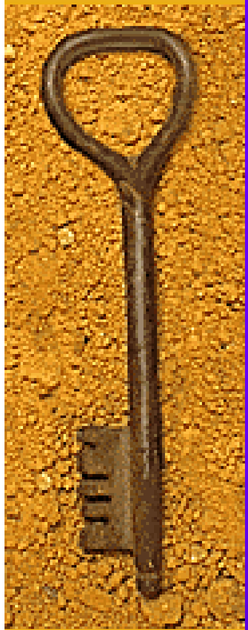
Enc_K(“I”, $sig_I(m_{1-4})$, [cert], child-SA)

Enc_K(“R”, $sig_R(m_{1-4})$, [cert], child-SA)

Initiator reveals identity first

Prevents “polling” attacks where attacker initiates IKE connections to find out who lives at an IP addr

Instead of running 2nd phase, “piggyback” establishment of child-SA on initial exchange



IKE: Phase Two (Create Child-SA)

After Phase One, I and R share key K

I

Enc_K(proposal, N_i, [g^a mod p], traffic) →

Crypto suites, protocol (AH, ESP or IPcomp)

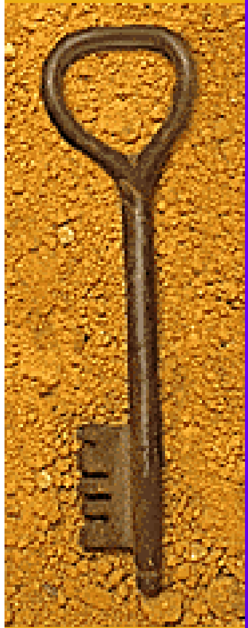
Optional re-key using old DH value and fresh nonces

IP address range, ports, protocol id

← Enc_K(proposal, N_r, [g^b mod p], traffic)

R

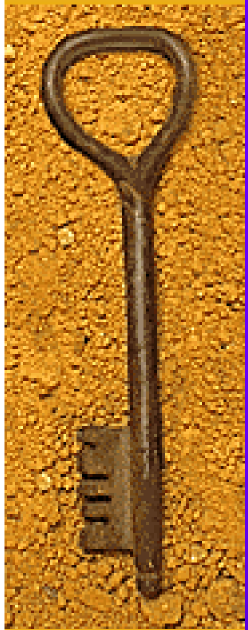
Can run this several times to create multiple SAs



Other Aspects of IKE

We did not talk about…

- Interaction with other network protocols
  - How to run IPsec through NAT (Network Address Translation) gateways?
- Error handling
  - Very important! Bleichenbacher attacked SSL by cryptanalyzing error messages from an SSL server
- Protocol management
  - Dead peer detection, rekeying, etc.
- Legacy authentication
  - What if one of the parties doesn't have a public key?



Current State of IPsec

- Best currently existing VPN standard
  - For example, used in Cisco PIX firewall, many remote access gateways
- IPsec has been out for a few years, but wide deployment has been hindered by complexity
  - ANX (Automotive Networking eXchange) uses IPsec to implement a private network for the Big 3 auto manufacturers and their suppliers