IPsec Technology Details

SANOG 6: ISP/NSP Security

16-23 July, 2006

Merike Kaeo

merike@doubleshotsecurity.com

Agenda

- Cryptography 101
  - Fundamental concepts
  - Algorithms and their applicability
- IPsec Technology
  - Standards (how does it work)
  - Practical Concerns
- IPsec LAB

Is The Internet Insecure?

The Internet isn't insecure. It may be *unsecure*. Insecurity is a mental state. The users of the Internet may be insecure, and perhaps rightfully so……

- Simson Garfinkel

Crypto 101

- Cryptography Is Used For
  - Authentication Protocols
  - Data Origin Authentication
  - Data Integrity
  - Data Confidentiality
- Cryptographic Algorithms
  - Asymmetric (Public Key) Encryption
  - Symmetric (Secret Key) Encryption
  - Diffie-Hellman
  - Hash Functions

Crypto Notation

- P = Plaintext
- C = Ciphertext
- K = Key

Map plaintext to ciphertext: $C = K[P]$

Map ciphertext to plaintext: $P = K^{-1}[C]$

Building Blocks

- Crypto algorithm: specifies the mathematical transformation that is performed on data to encrypt/decrypt
- Stream cipher: encrypts a digital stream one bit at a time (RC4)
- Block cipher: transforms data in fixed-size blocks, one block at a time (DES, IDEA)

Properties of Good Crypto Algorithms

- Crypto algorithm is NOT proprietary
- Analyzed by public community to show that there are no serious weaknesses
- Explicitly designed for encryption

Exclusive–OR Function (X-OR)

1 xor 1 = 0

1 xor 0 = 1

0 xor 0 = 0

0 xor 1 = 1

Example 1: **0 1 1 0 0 1 0 1 xor'ed with 1 1 0 1 0 0 1 1**

RESULT: 1 0 1 1 0 1 1 0

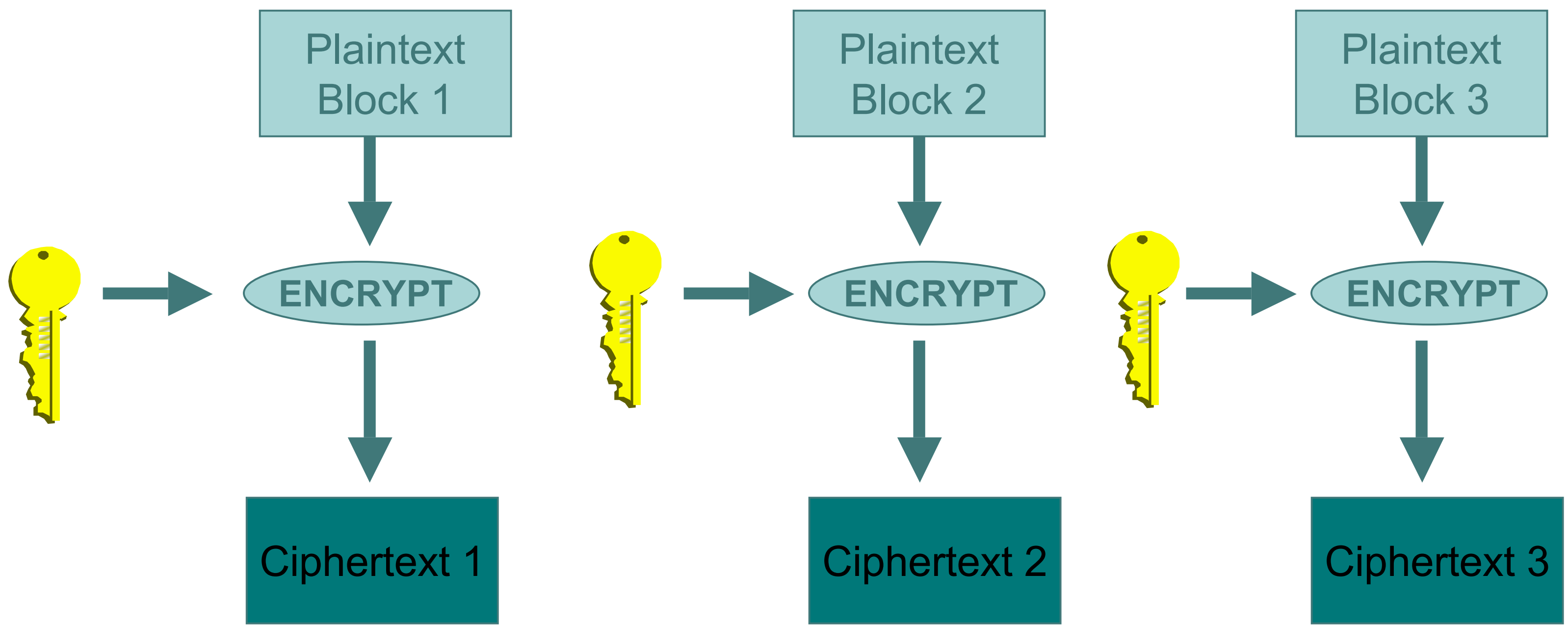
Example 2: **1 0 1 1 0 1 1 0 xor'ed with 1 1 0 1 0 0 1 1**

RESULT: 0 1 1 0 0 1 0 1

Block Cipher Modes

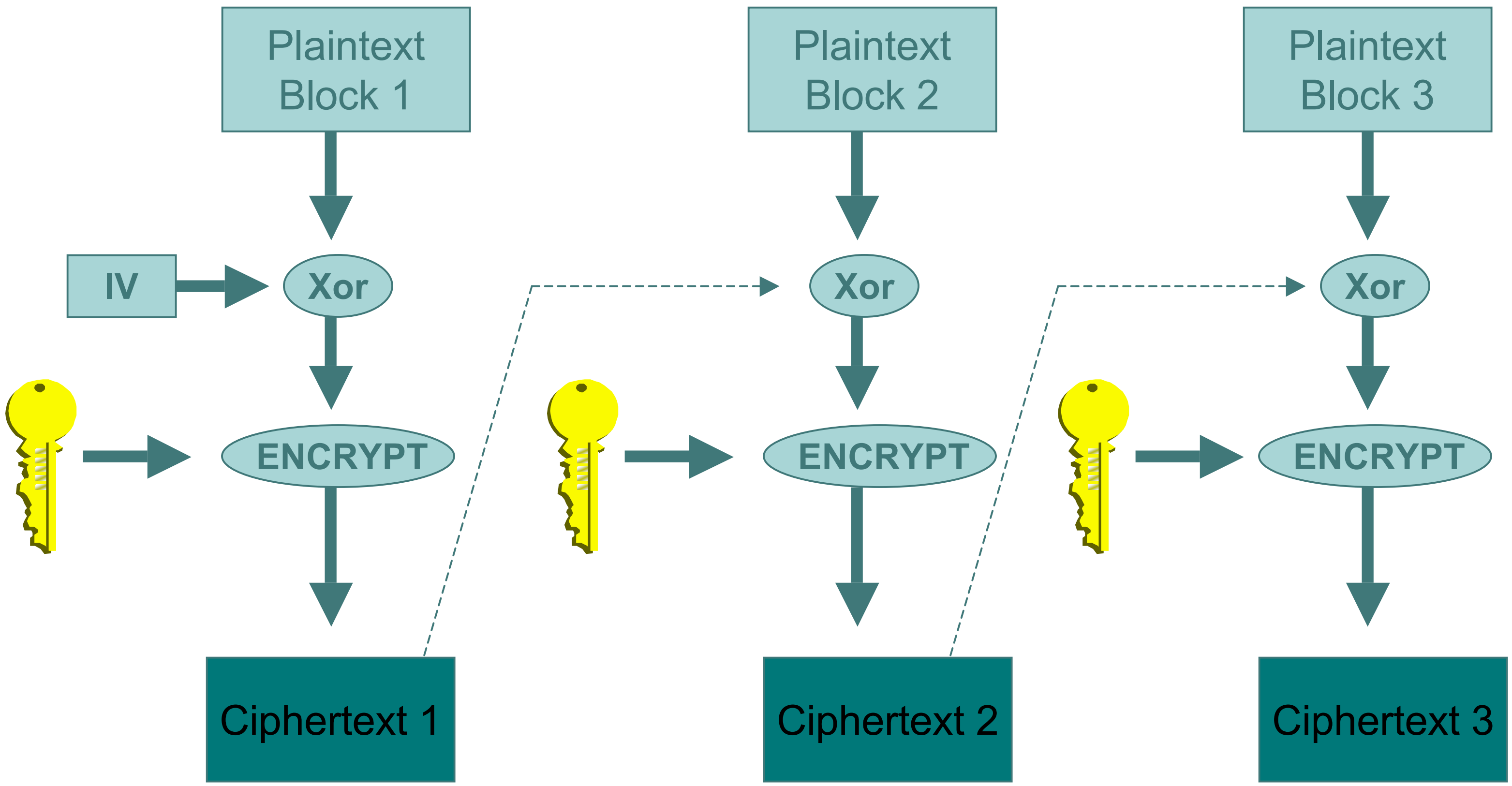
- Defines how the block cipher algorithm is applied to the data stream
- Four Basic Modes
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC)
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)

Electronic Code Book (ECB)

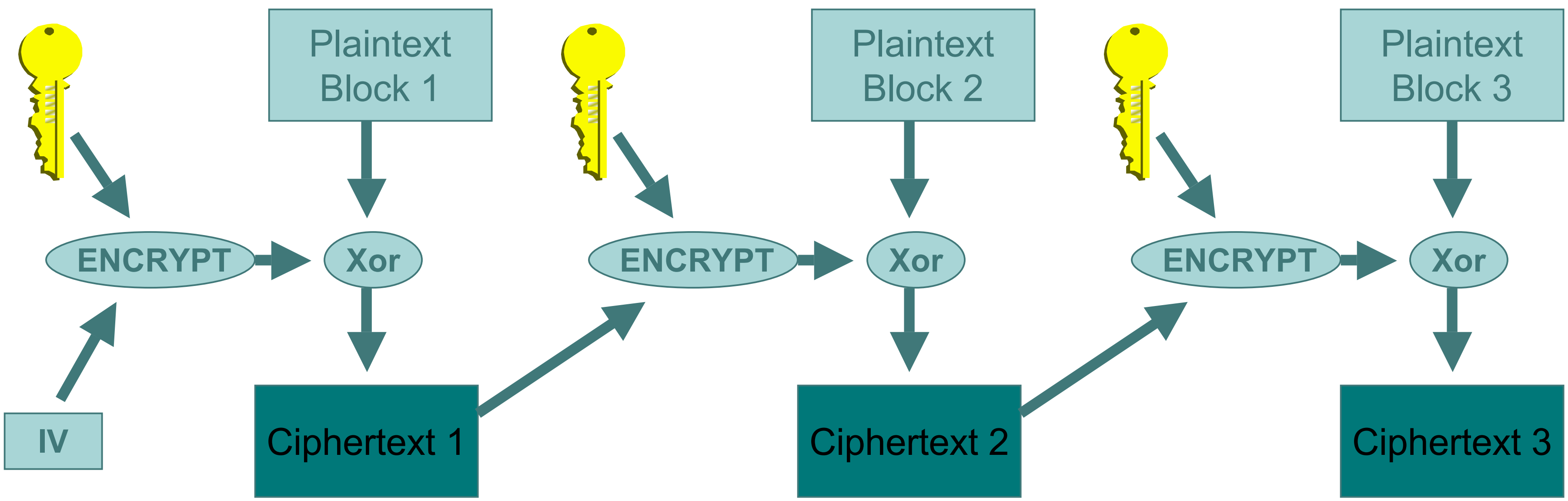


Problem: Identical plaintext blocks encrypted into identical ciphertext blocks when the same key is used; produces visible patterns

Cipher Block Chaining (CBC)

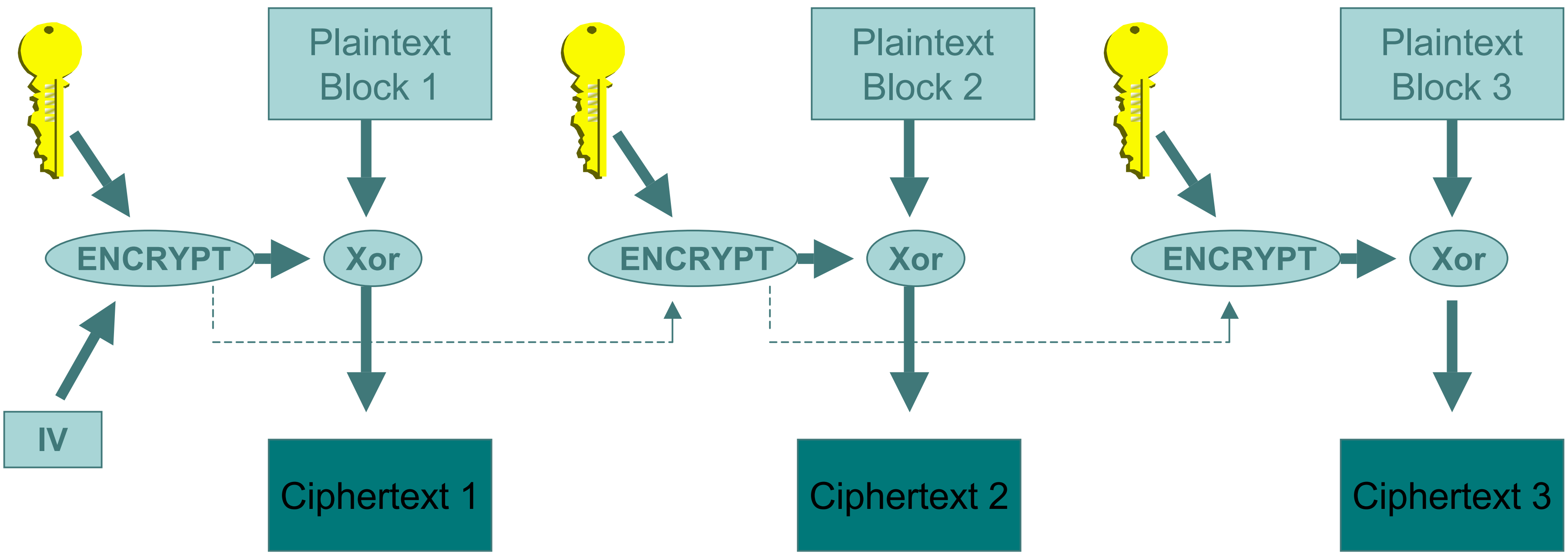


Cipher Feedback



- Uses the block cipher algorithm to generate a temporary key
- Can be adapted to work with smaller blocks to eliminate padding

Output Feedback



- Uses the block cipher algorithm to generate a key stream independent of the data being encrypted
- Can be adapted to work with smaller blocks to eliminate padding

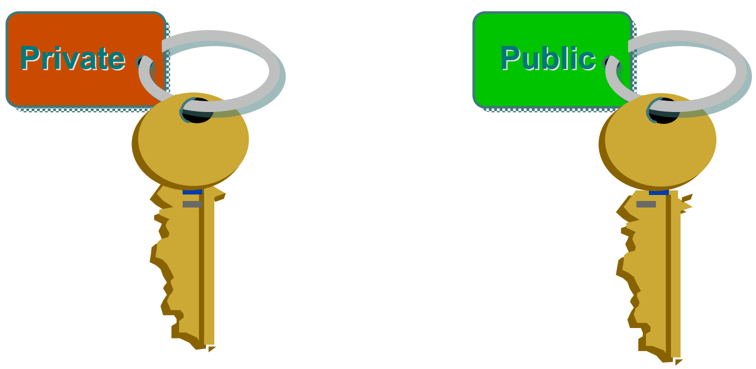
Selecting A Block Cipher Mode

- Small amounts of truly random data: ECB
  - Example: randomly generated keying material
  - Other modes can be used but ECB is most efficient
- Protocols with crypto integrity protection: CBC, CFB, OFB
- Arbitrary communications with arbitrary data: CBC, CFB
  - Repeated plaintext data is obscured
  - Constantly changing encryption keys defeat differential cryptanalysis attacks

Public Key Cryptography

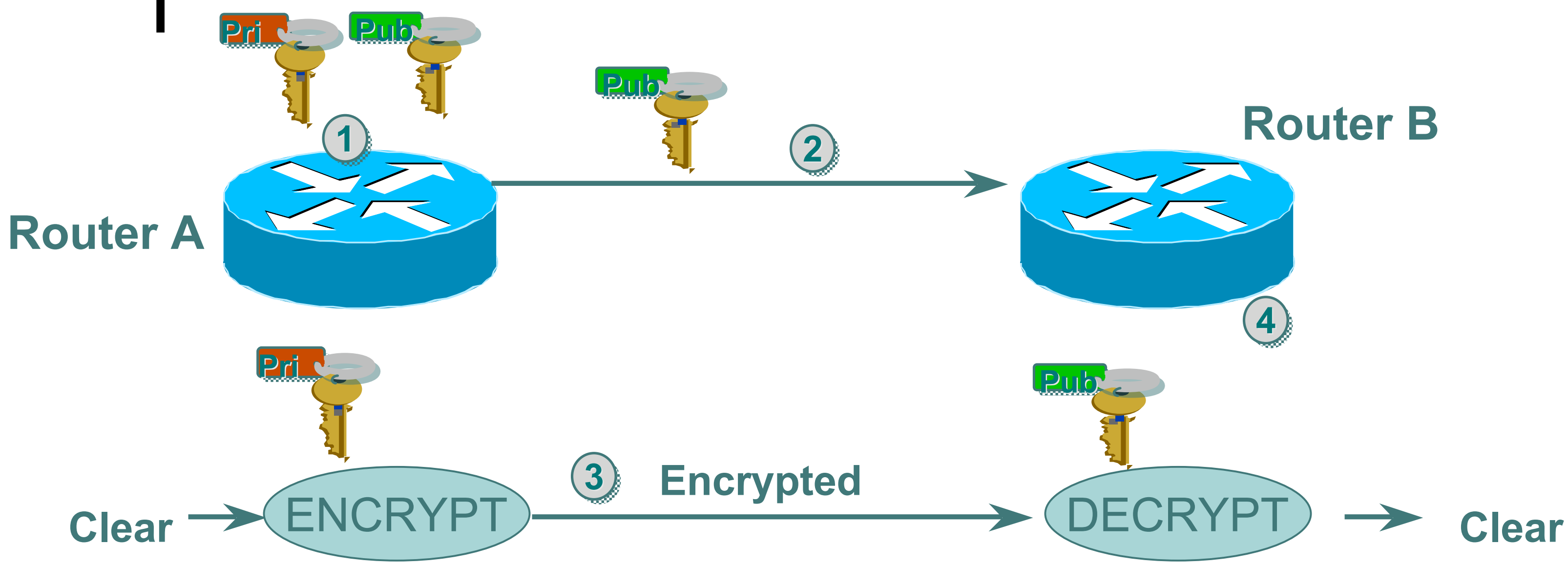
Uses a key pair (i.e. public/private keys)

- Keep private key private
- Anyone can see public key



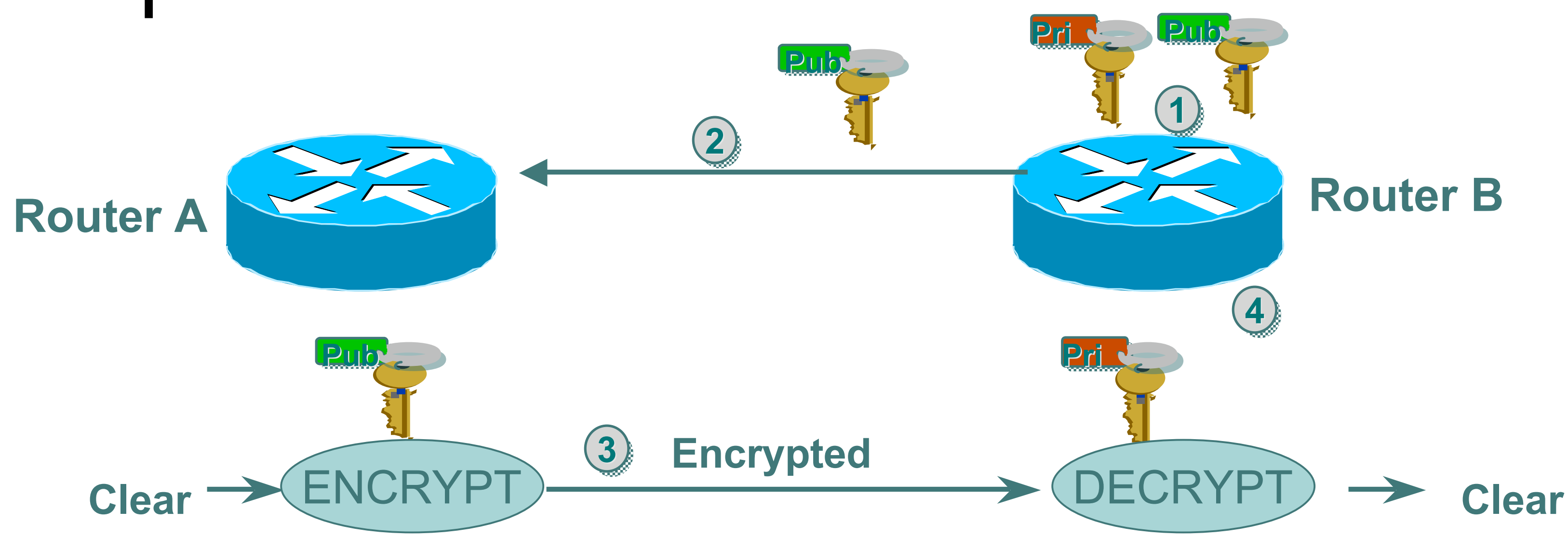
Computing Key pair is computationally expensive!!
Common Algorithms: RSA, El Gamal, Elliptic Curve

Data Origin Authentication



1. Router A generates public/private key pair
2. Router A sends its public key to Router B
3. Router A encrypts packet with its private key and sends encrypted packet to Router B
4. Router B receives encrypted packet and decrypts with Router A's public key

Data Integrity and Confidentiality

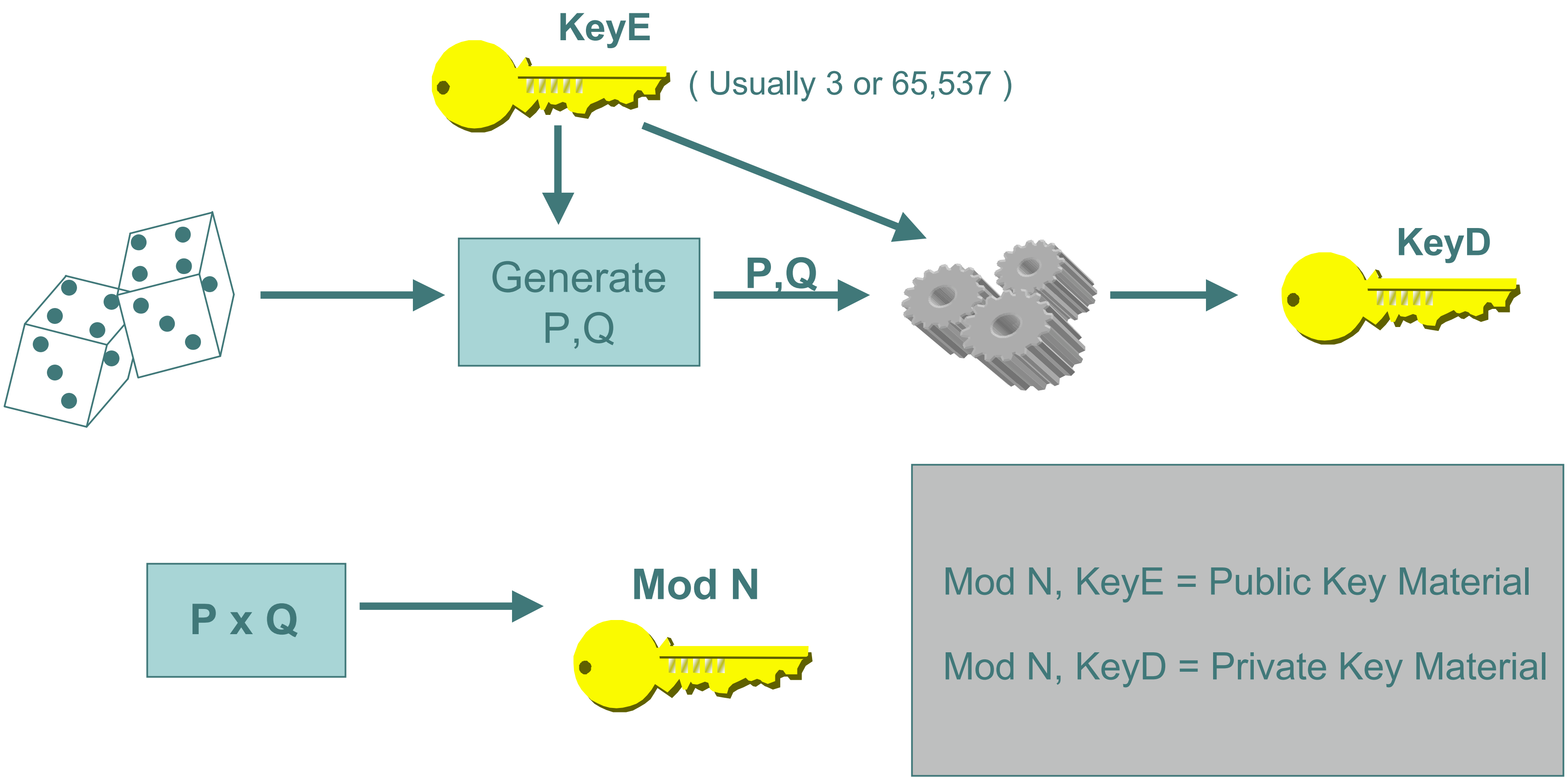


1. Router B generates public/private key pair
2. Router B sends its public key to Router A
3. Router A encrypts packet with router B's public key and sends encrypted packet to Router B
4. Router B receives encrypted packet and decrypts with its' private key

RSA Public Key Cryptography

- Based on relative ease of multiplying large primes together but almost impossible to factor the resulting product
- RSA keys: 3 special numeric values
- Algorithm produces public keys that are tied to specific private keys
- Public key operations can be made very fast but private key operations will be slow
- Provides both digital signatures and public-key encryption

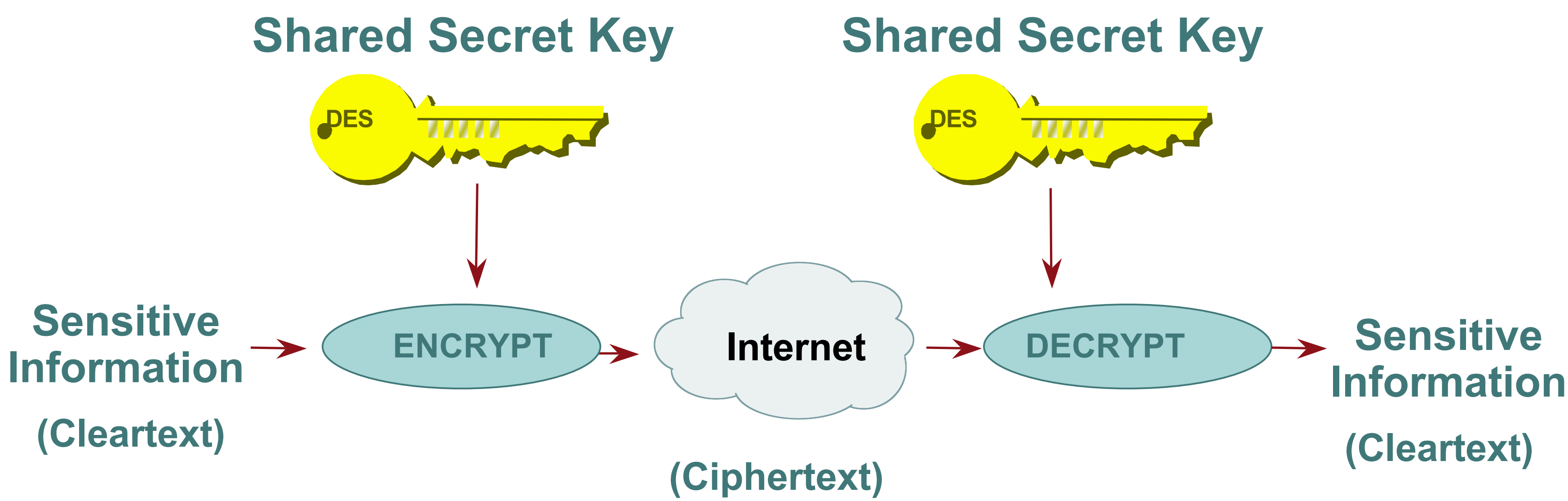
Generating RSA Keys



Secret Key Cryptography

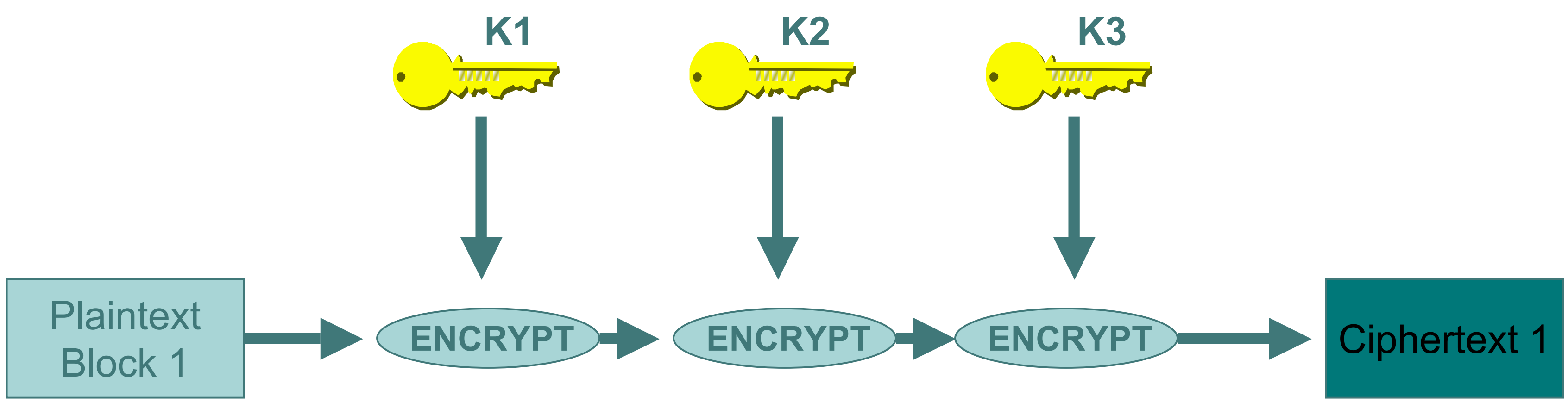
- Two operations which are inverses of each other
- Shared secret key is required to encrypt and decrypt messages
- A good secret key algorithms cannot be broken without knowing the key
- NOT good practice to use group shared secret keys - instead, use different shared secret between each pair of users)

Secret Key Encryption



Common Algorithms: DES, 3DES, AES, IDEA

Triple DES (3DES)



- Many applications use K3=K1, yielding a key length of 112 bits
- Interoperable with conventional DES if K1=K2=K3

AES

- Published in November 2001
- Rijndael algorithm developed by Dr. Joan Daemen and Dr. Vincent Rijmen
- Symmetric Block Cipher
  - 128 bit blocks
  - 3 key lengths: 128, 192, and 256 bits
  - symmetric and parallel
  - low memory requirement

More Secret Key Cryptography Uses

- Authentication
  - Challenge/Response
    - Initiator sends encrypted challenge (X)
    - Responder sends decrypted challenge (X) along with its own challenge (Y)
    - Initiator replies w/ decrypted challenge(Y)
- Integrity
  - Integrity check generated and verified with same key (verifier can forge this)

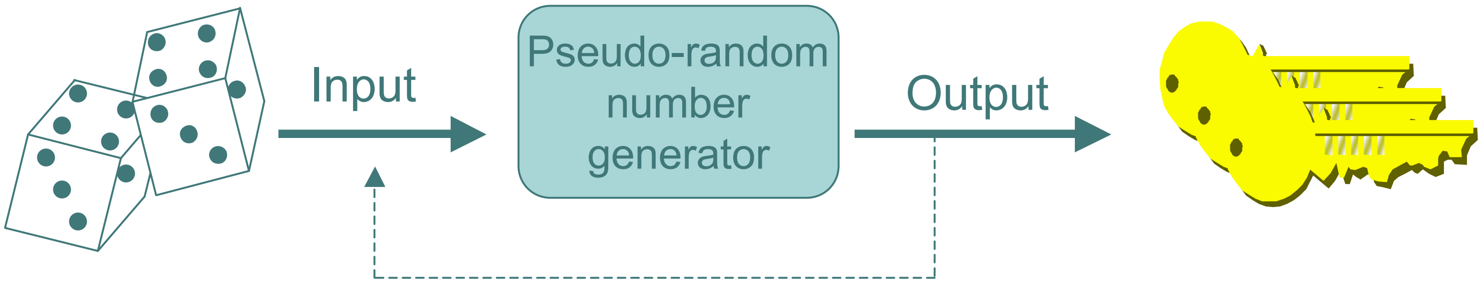
Key Length

Key Length (in bits)	Number of Combinations
40	$2^{40} = 1,099,511,627,776$
56	$2^{56} = 7.2 \times 10^{16}$
64	$2^{64} = 1.8 \times 10^{19}$
112	$2^{112} = 5.2 \times 10^{33}$
128	$2^{128} = 3.4 \times 10^{38}$
192	$2^{192} = 6.2 \times 10^{57}$
256	$2^{256} = 1.1 \times 10^{77}$

Longer Keys Are Better

- Brute Force attacks are ones where miscreants try all possible combination of keys to break algorithm
- Security depends on limited resources for the miscreants
- A good crypto algorithm is linear in computational resources for ‘good guys’ and exponential for ‘bad guys’
- Faster computers work for benefit of ‘good guys’ since can use longer keys more effectively

Producing Effective Keys



- Producing random seed value can be slow and inefficient
- PRNG used when generating many separate keys
- Properties of sequence #'s produced by a good PRNG
  - Equal chance that a given number falls anywhere within the range of numbers being generated
  - The sequence should not repeat itself

Scalability with Secret Key Cryptography

- Configuring shared secret keys easily becomes administrative nightmare
- Automated mechanism to securely derive secret keys => Diffie-Hellman

Diffie-Hellman Algorithm

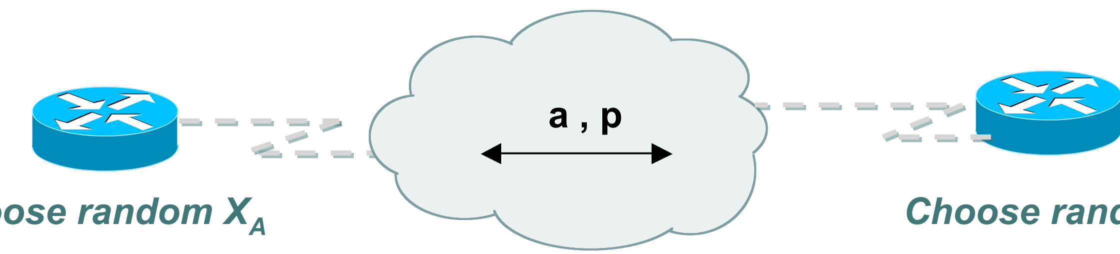
- Two entities can agree on a secret key while communicating over a public network
- Both peers choose a private number and from that compute a public number
- They use their own private number and the other's public number to derive the same shared secret
- Security based on principle that given a, p, (a^Xmod p) it is nearly impossible to derive X

Remember……

Nothing is impossible, only mathematically improbable

- The Avengers

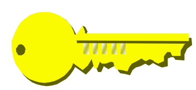
Deriving Secret Keys Using Public Key Technology (e.g., Diffie-Hellman)



Choose random X_A | *Choose random X_B*

$Y_A = (a^{X_A}) \bmod p \longrightarrow$ | $\longleftarrow Y_B = (a^{X_B}) \bmod p$

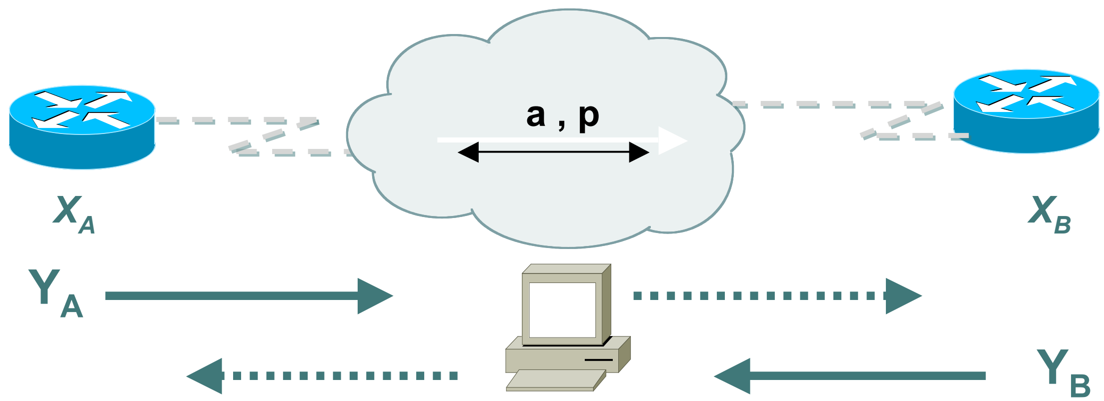
Compute: $(Y_B)^{X_A}$ | *Compute:* $(Y_A)^{X_B}$

$Z = a^{X_A X_B} \bmod p$  $Z = a^{X_A X_B} \bmod p$

By exchanging numbers in the clear, two entities can determine a new unique number (Z), known only to them

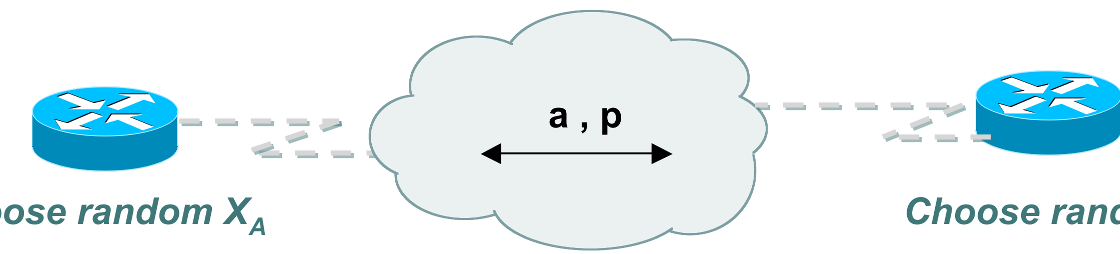
DH Man-in-the-Middle Attack

- Diffie-Hellman is subject to a man-in-the-middle attack
- Digital signatures of the ‘public values’ can enable each party to verify that the other party actually generated the value



=> DH exchanges need to be authenticated!!

Signed Diffie-Hellman



Choose random X_A

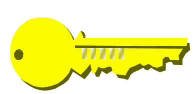
Choose random X_B

$Y_A = [(a^{X_A}) \bmod p]$ *signed with A's private key*

Verify A's signature

$Y_B = [(a^{X_B}) \bmod p]$ *signed with B's private key*

Verify B's signature



$Z = a^{X_A X_B} \bmod p$

Perfect Forward Secrecy

- Deriving new keying material without using previous parameters
- Limits decryption of a conversation if private key is escrowed or broken
- SSL does not use PFS
  - Sender uses secret key (K_{Secret}), encrypts it with Recipient's public key (B_{Pub}) and sends it to Recipient

Hash Functions

A *hash function* takes an input message of arbitrary length and outputs fixed-length code. The fixed-length output is called the *hash*, or the *message digest*, of the original input message.

Common Algorithms: MD-5 (128), SHA-1 (160)

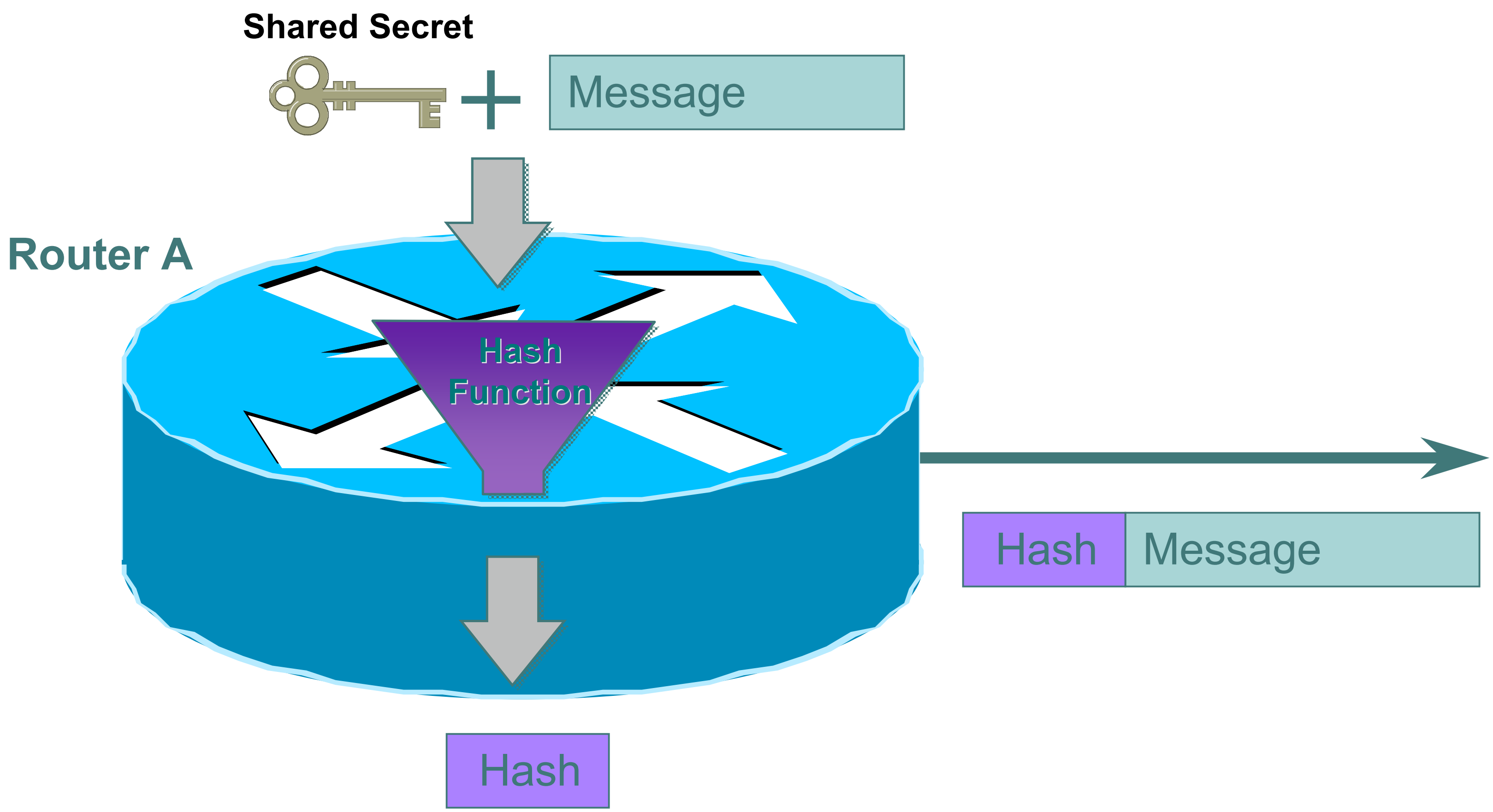
Digital Signatures

- A digital signature is a cryptographic hash appended to a packet
- Used to prove the identity of the sender and the integrity of the packet

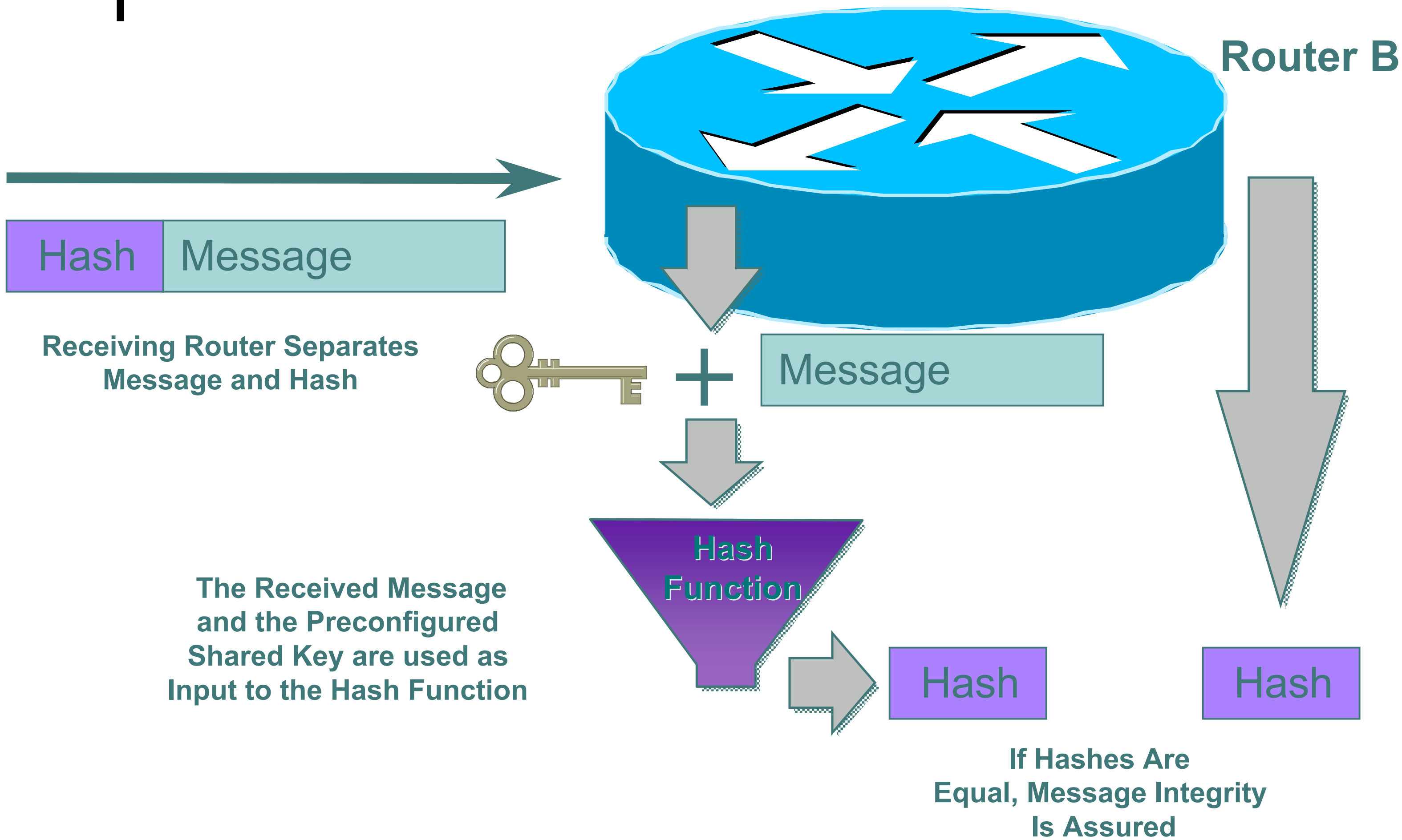
Digital Signatures

- Two common public-key digital signature techniques:
  - RSA (Rivest, Shamir, Adelman)
  - DSS (Digital Signature Standard)
- A sender uses its private key to sign a packet.
- The receiver of the packet uses the sender's public key to verify the signature.
- Successful verification assures:
  - The packet has not been altered
  - The identity of the sender

Integrity Check with Hash



Integrity Check With Hash



Computing a Keyed-MAC

- Message broken down into n blocks of 512-bits
- Shared secret key is xor'ed with specified array to produce K1
- Shared secret key is xor'ed a 2nd time with another specified array to produce K2

Hash1 = $(1^{st}\text{ block of message} + K1)_{MD5}$

Hash2 = $(\text{hash1} + K2)_{MD5}$

Hash3 = $(2^{nd}\text{ block of message} + \text{hash2})_{MD5}$

Hash(n+1) = $(n^{th}\text{ block of message} + \text{hash}[n])_{MD5}$

HMAC-MD5-96 / HMAC-SHA-96 -> last hash truncated to 96 bits!!

Crypto 101 Summary

- Public Key Encryption
  - Typically used for data origin authentication
  - Often combined with hash function
- Secret Key Encryption
  - Typically used for data confidentiality
- Diffie-Hellman Algorithm
  - Uses public-key cryptography to derive secret key
  - Exchanges need to be authenticated
- Hash Functions
  - Easy to compute
  - Typically used for data origin authentication and data integrity
- Digital Signatures
  - Combines hash functions with public key cryptography

IPsec

- Suite of protocols to secure IP traffic
  - Defined in RFC 2401-2409, RFC 2451
  - New updated standards soon
    - (architecture, AH, ESP)
- Components
  - AH (Authentication Header)
    - RFC requires HMAC-MD5-96 and HMAC-SHA1-96….older implementations also support keyed MD5
  - ESP (Encapsulating Security Payload)
    - RFC requires DES 56-bit CBC and Triple DES. Can also use RC5, IDEA, Blowfish, CAST, RC4, NULL
  - IKE (The Internet Key Exchange)

What Does IPsec Provide?

- Data integrity and data origin authentication
  - Data “signed” by sender and “signature” verified by the recipient
  - Modification of data can be detected by signature “verification”
  - Because “signature” based on a shared secret, it gives data origin authentication
- Confidentiality

What Does IPsec Provide?

- Anti-replay protection
  - Optional : the sender must provide it but the recipient may ignore
- Key Management
  - IKE – session negotiation and establishment
  - Sessions are rekeyed or deleted automatically
  - Secret keys are securely established and authenticated
  - Remote peer is authenticated through varying options

What is an SA?

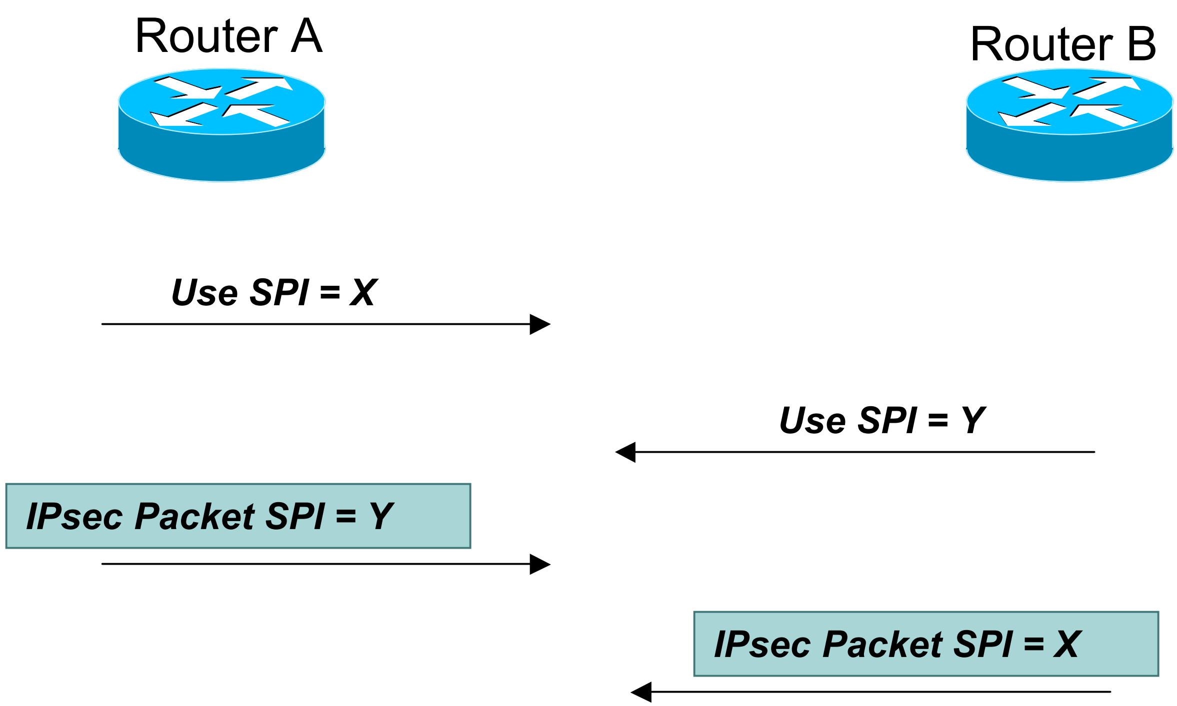
- Security Association groups elements of a conversation together
  - AH authentication algorithm and keys
  - ESP encryption algorithm and key(s)
  - Cryptographic synchronization
  - SA lifetime
  - SA source address
  - Mode (transport or tunnel)

A Security Association Maps:

- From a host or gateway
  - To a particular IP destination address
  - With a particular security protocol (AH/ESP)
  - Using SPI selected by remote host or gateway
- To a host or gateway
  - To (one of) our IP address(es)
  - With a particular security protocol (ESP/AH)
  - Using SPI selected by us

SPI (Security Parameter Index)

The SPI is selected by the remote peer



A SPI Represents an SA

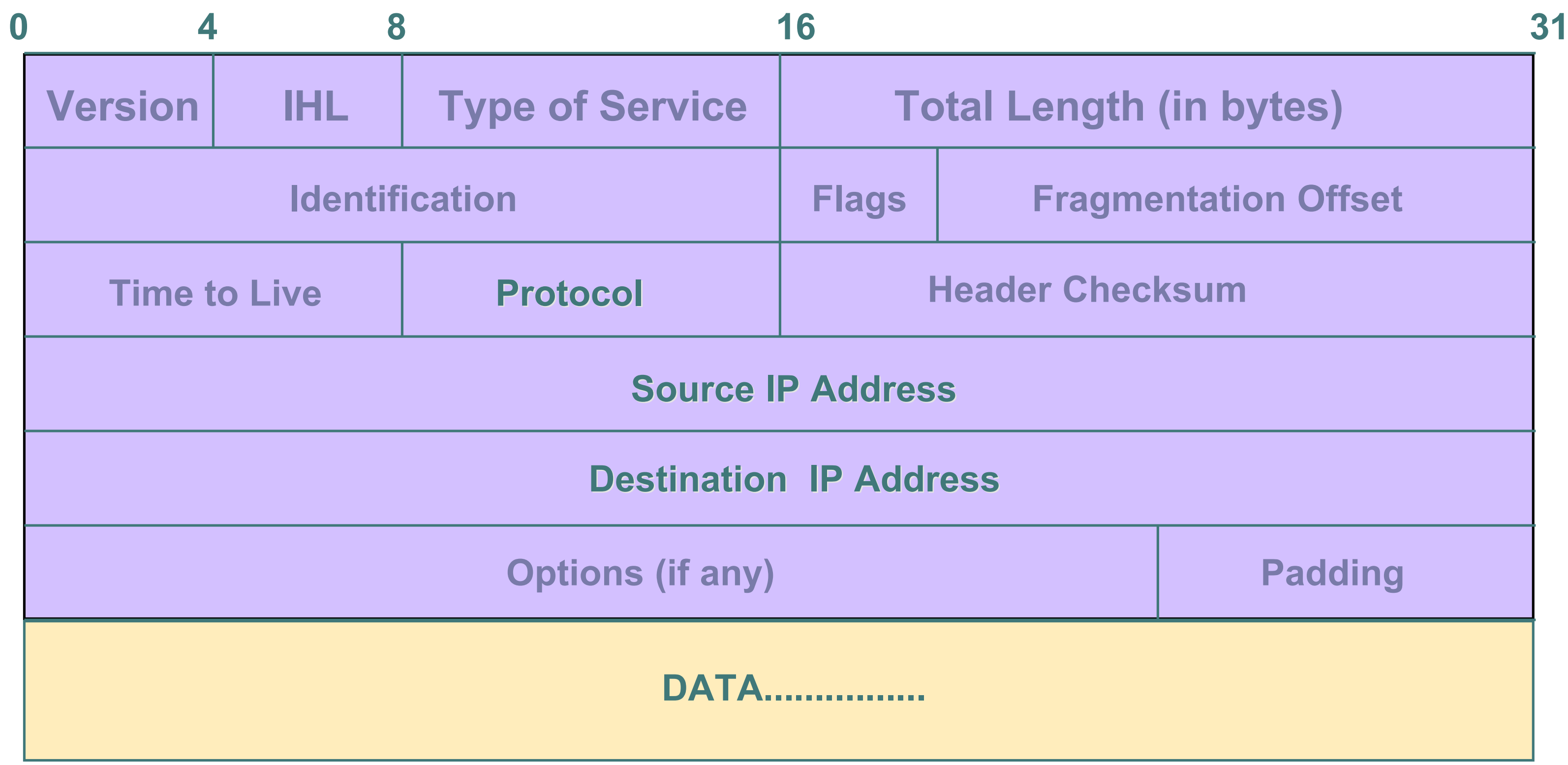
- The SPI is a 32-bit number
- The SPI is combined with the protocol (AH/ESP) and destination IP address to uniquely identify an SA
- An SA is unidirectional

When an ESP/AH packet is received, the SPI is used to look up all of the crypto parameters

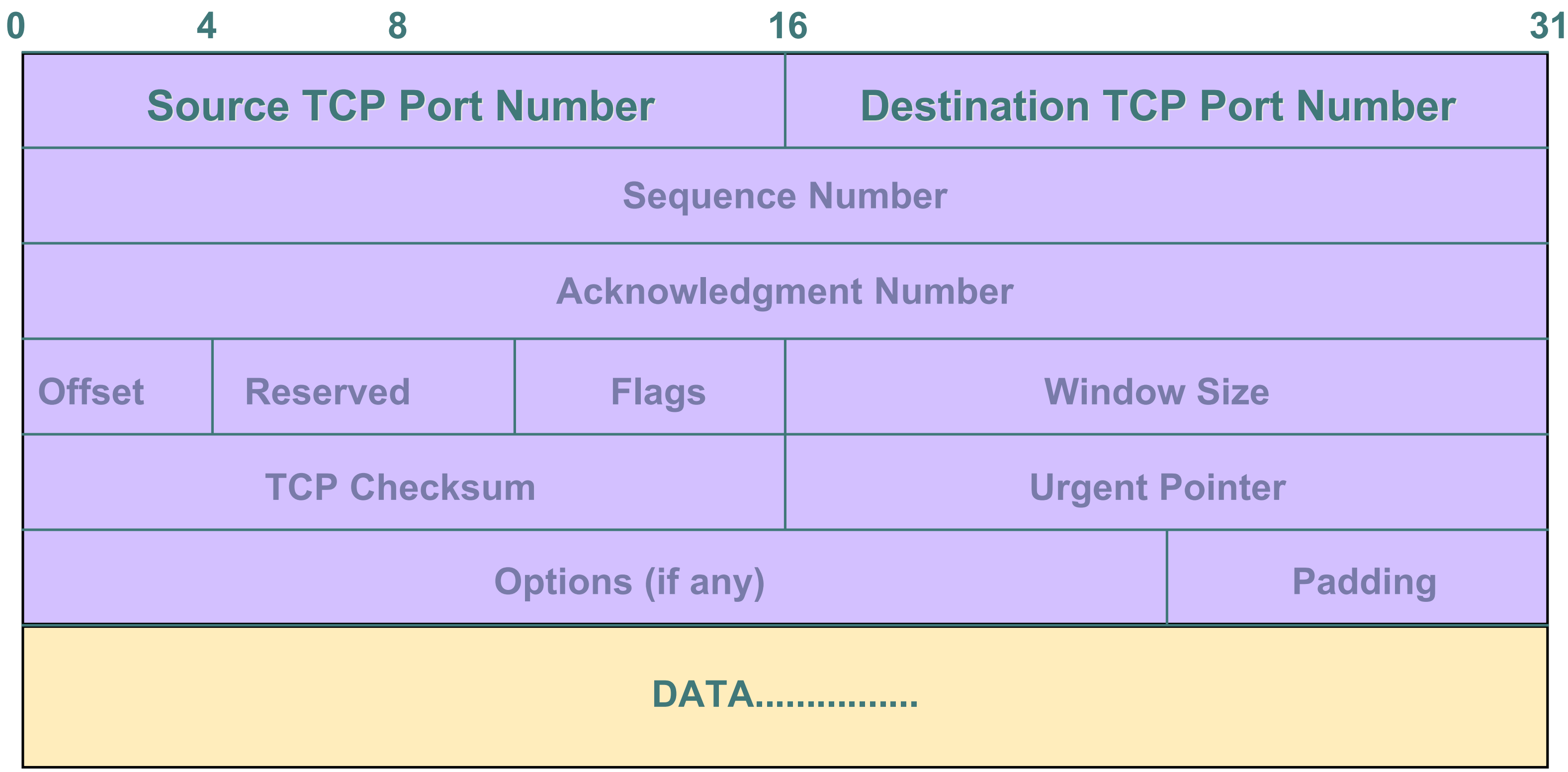
IPsec Traffic Selectors

- Selectors for traffic matches….what kind of traffic will be acted on how
- Selectors include:
  - IP address or range
  - Optional IP protocol (UDP, TCP, etc)
  - Optional layer 4 (UDP, TCP) port
- Selected traffic is either protected with IPsec or dropped

IP Header Format



TCP Header Format



IPsec Components

- AH
  - RFC requires HMAC-MD5-96 and HMAC-SHA1-96….older implementations also support keyed MD5
- ESP
  - RFC requires DES 56-bit CBC and Triple DES. Can also use RC5, IDEA, Blowfish, CAST, RC4, NULL
- IKE

Authentication Header (AH)

- Authentication is applied to the entire packet, with the mutable fields in the IP header zeroed out
- If both ESP and AH are applied to a packet, AH follows ESP

AH Header Format

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Next Header	Payload Length	Reserved
Security Parameter Index (SPI)		
Sequence Number		
Authentication Data [Integrity Value Check (ICV)]		

Next Header: which higher level protocol is (UDP,TCP,ESP) next

Payload Length: size of AH in 32-bit longwords, minus 2

Reserved: must be zero

SPI: arbitrary 32-bit number that specifies to the receiving device which security association is being used (security protocols, algorithms, keys, times, addresses, etc)

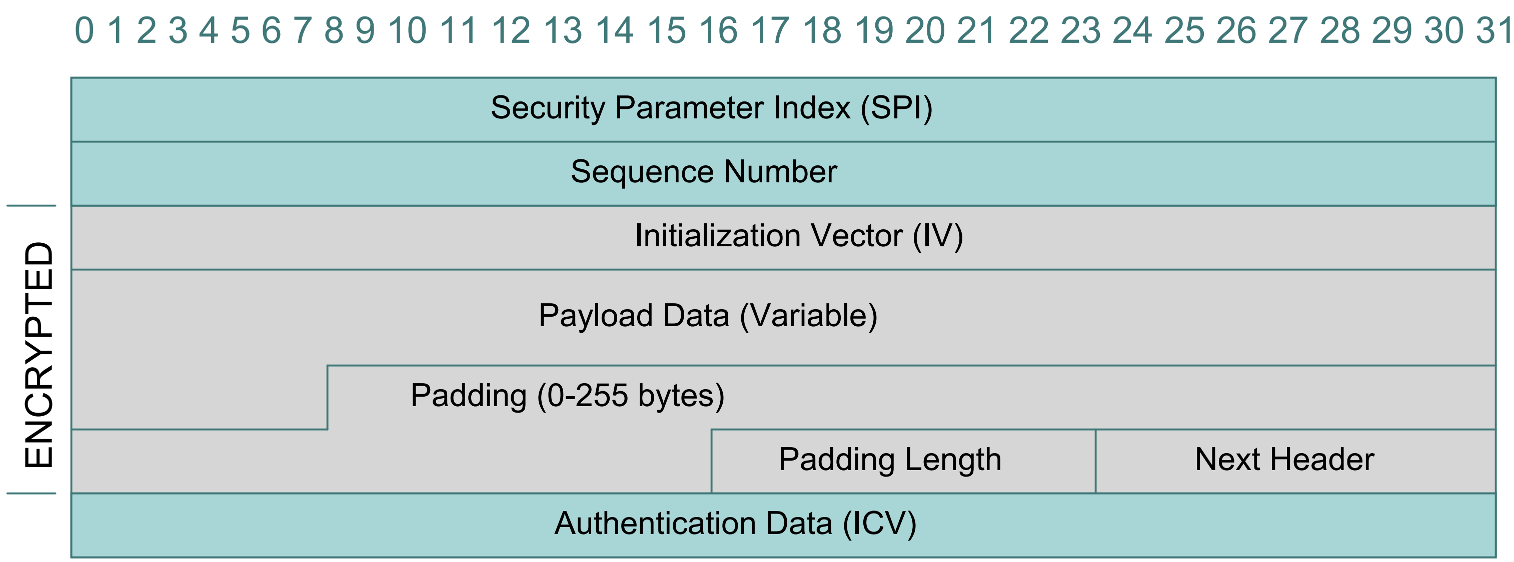
Sequence Number: start at 1 and must never repeat. It is always set but receiver may choose to ignore this field

Authentication Data: ICV is a digital signature over the packet and it varies in length depending on the algorithm used (SHA-1, MD5)

Encapsulating Security Payload (ESP)

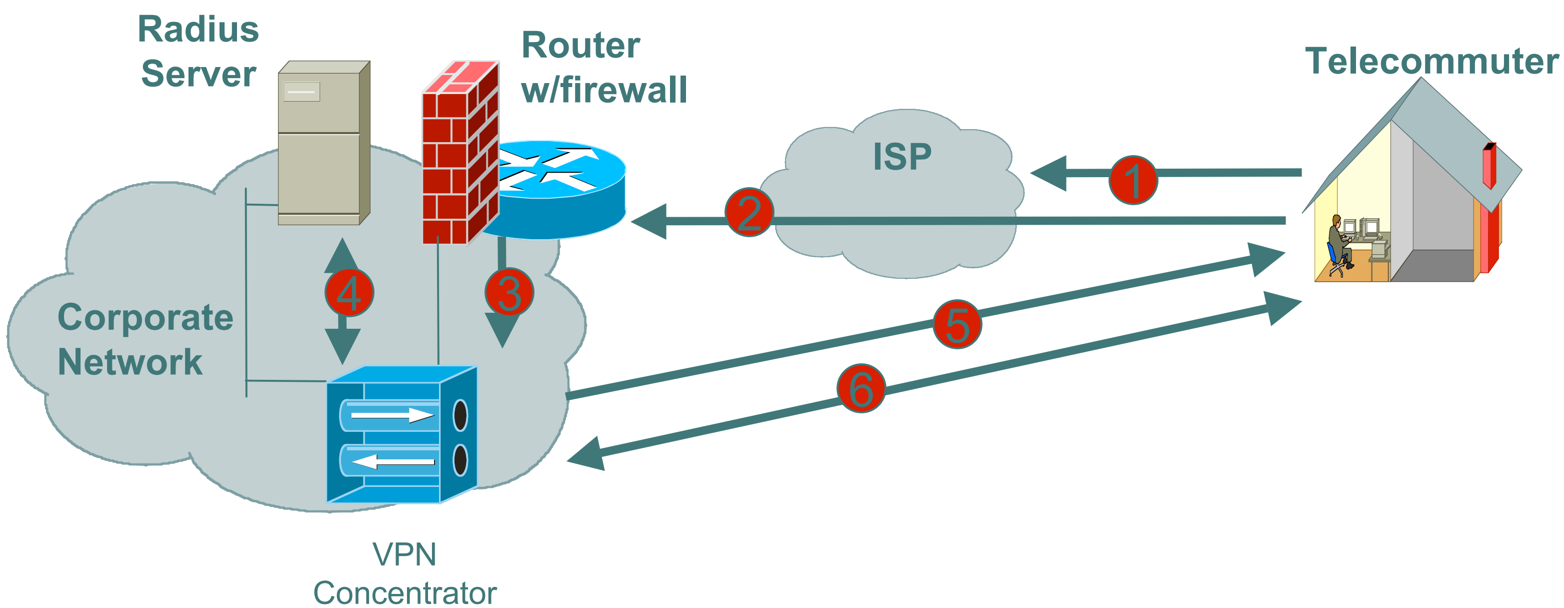
- Must encrypt and/or authenticate in each packet (null encryption)
- Encryption occurs before authentication
- Authentication is applied to data in the IPsec header as well as the data contained as payload

ESP Header Format



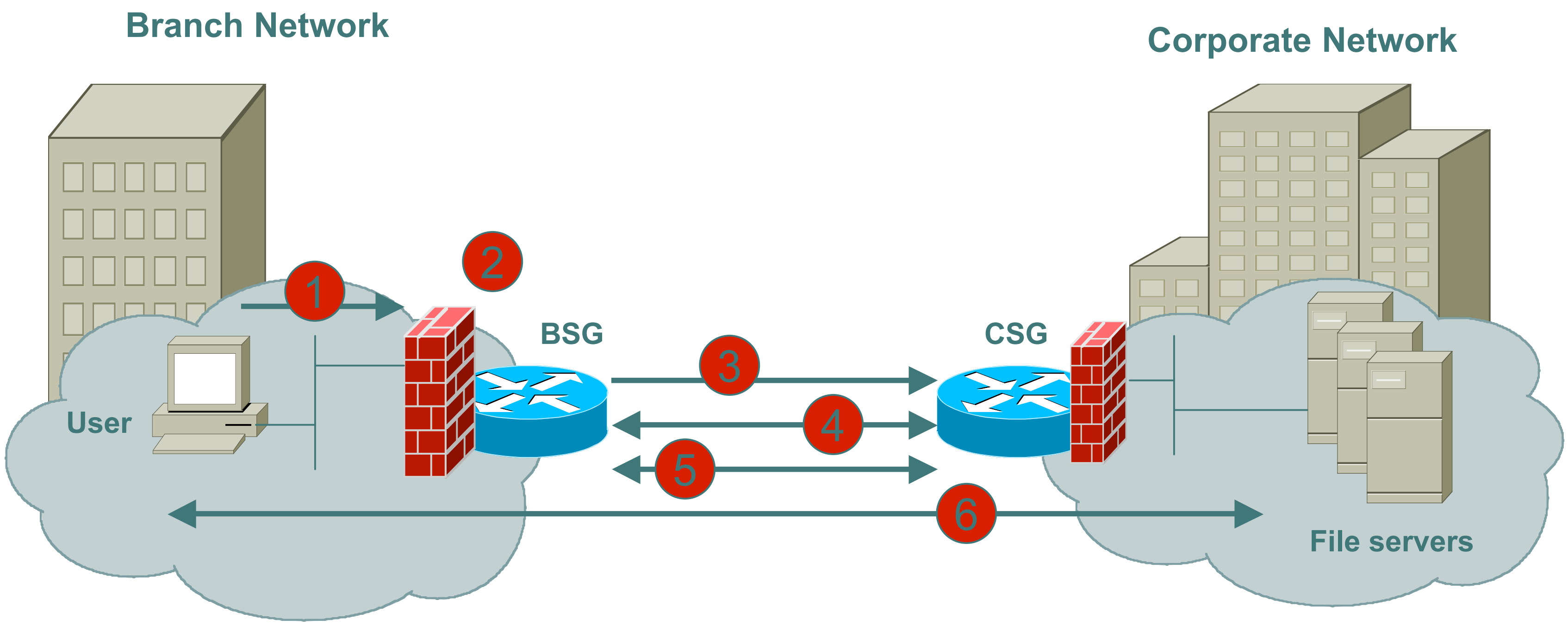
SPI: arbitrary 32-bit number that specifies SA to the receiving device
Seq #: start at 1 and must never repeat; receiver may choose to ignore
IV: used to initialize CBC mode of an encryption algorithm
Payload Data: encrypted IP header, TCP or UDP header and data
Padding: used for encryption algorithms which operate in CBC mode
Padding Length: number of bytes added to the data stream (may be 0)
Next Header: the type of protocol from the original header which appears in the encrypted part of the packet
Authentication Header: ICV is a digital signature over the packet and it varies in length depending on the algorithm used (SHA-1, MD5)

AH/ESP Transport Mode



IPsec AH/ESP protection for hosts end-to-end

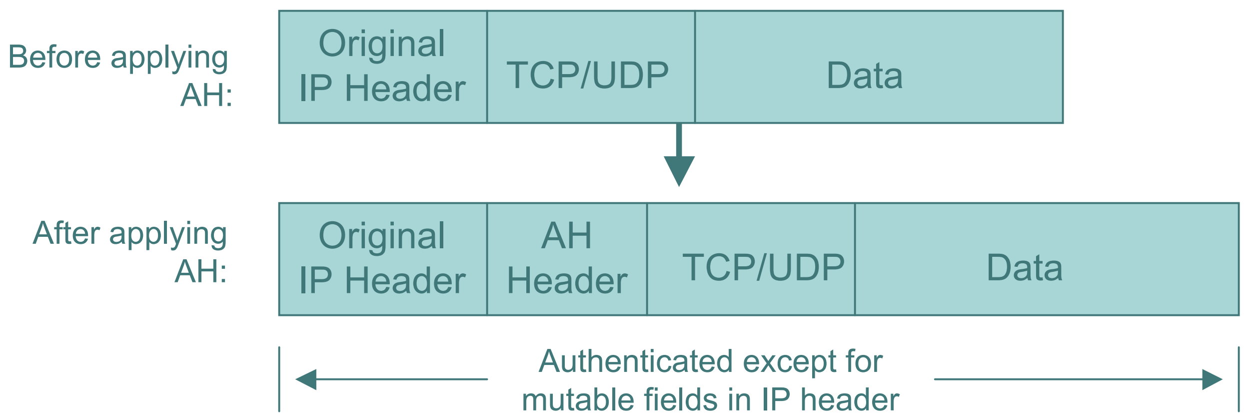
AH/ESP Tunnel Mode



IPsec AH/ESP protection for hosts or subnets behind security gateways

Packet Format Alteration for AH Transport Mode

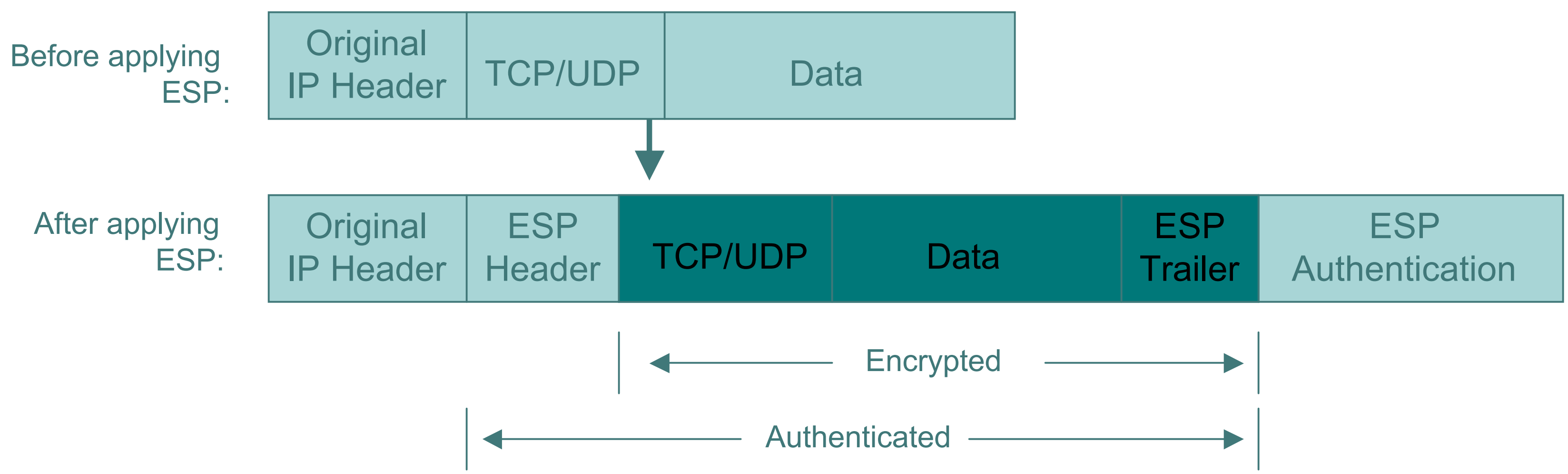
Authentication Header



- **ToS**
- **TTL**
- **Header Checksum**
- **Offset**
- **Flags**

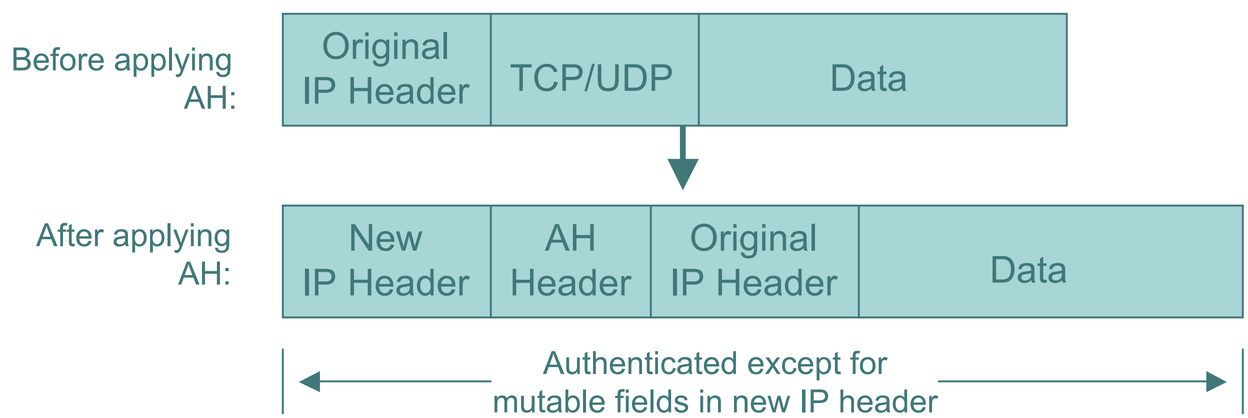
Packet Format Alteration for ESP Transport Mode

Encapsulating Security Payload



Packet Format Alteration for AH Tunnel Mode

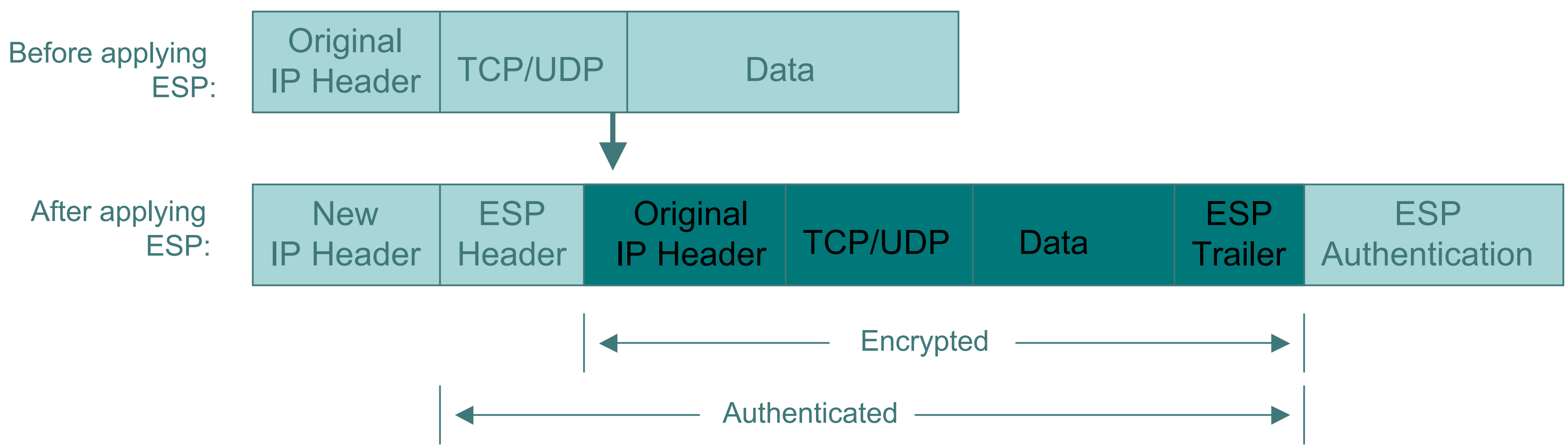
Authentication Header



- **ToS**
- **TTL**
- **Header Checksum**
- **Offset**
- **Flags**

Packet Format Alteration for ESP Tunnel Mode

Encapsulating Security Payload



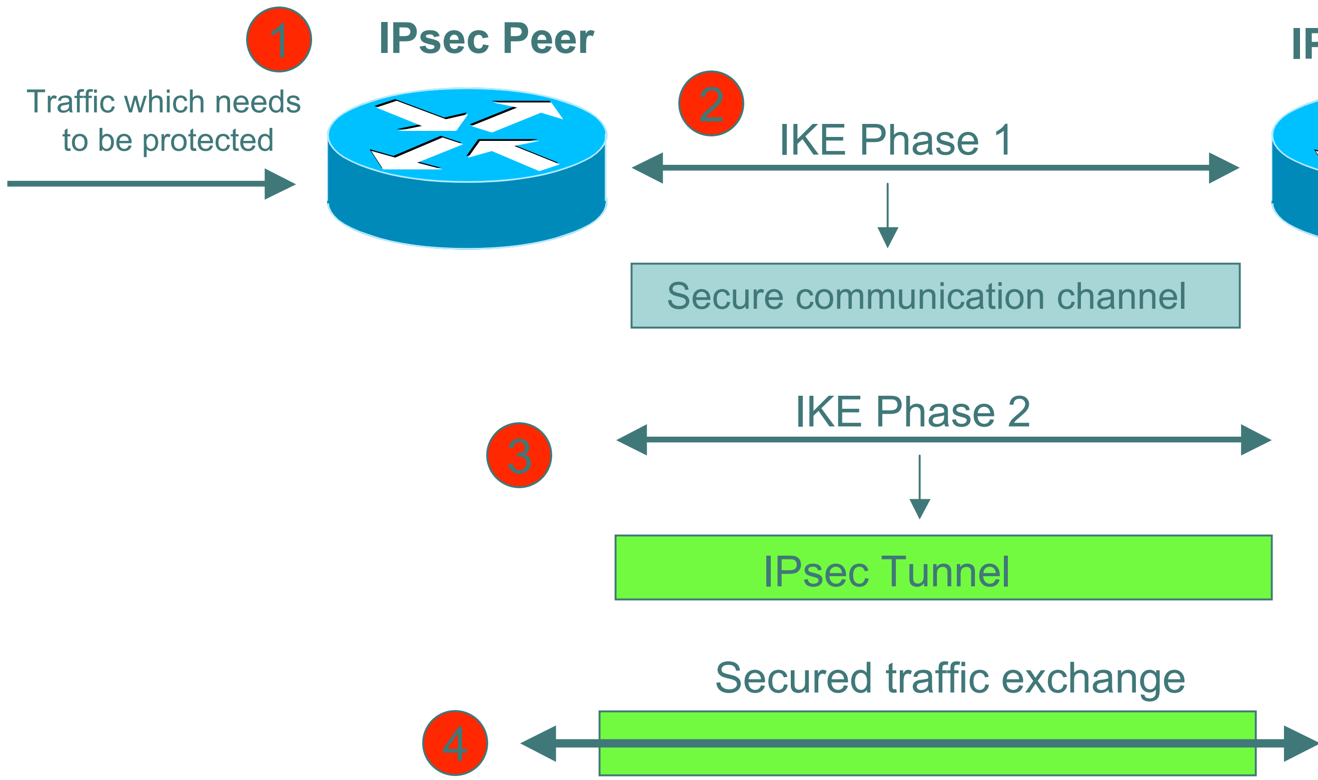
How Do You Get Your Crypto Keys?

- Manual keying
  - Easy way to get started
  - Difficult to administer
- IKE
  - Authenticates IPsec peers
  - Negotiates IPsec SAs
  - Establishes IPsec

Internet Key Exchange (IKE)

- Phase I
  - Establish a secure channel (ISAKMP/IKE SA)
  - Using either main mode or aggressive mode
- Phase II
  - Establishes a secure channel between computers intended for the transmission of data (IPsec SA)
  - Using quick mode

Overview of IKE



ISAKMP Header Format

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Initiator Cookie				
Responder Cookie				
Next Payload	Major Version	Minor Version	Exchange Type	Flags
Message ID				
Total Length of Message				

ISAKMP Message Format

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

ISAKMP HEADER		
Next Payload	Reserved	Payload Length
Payload		
Next Payload	Reserved	Payload Length
Payload		

Next Payload: 1byte; identifier for next payload in message. If it is the last payload It will be set to 0

Reserved: 1byte; set to 0

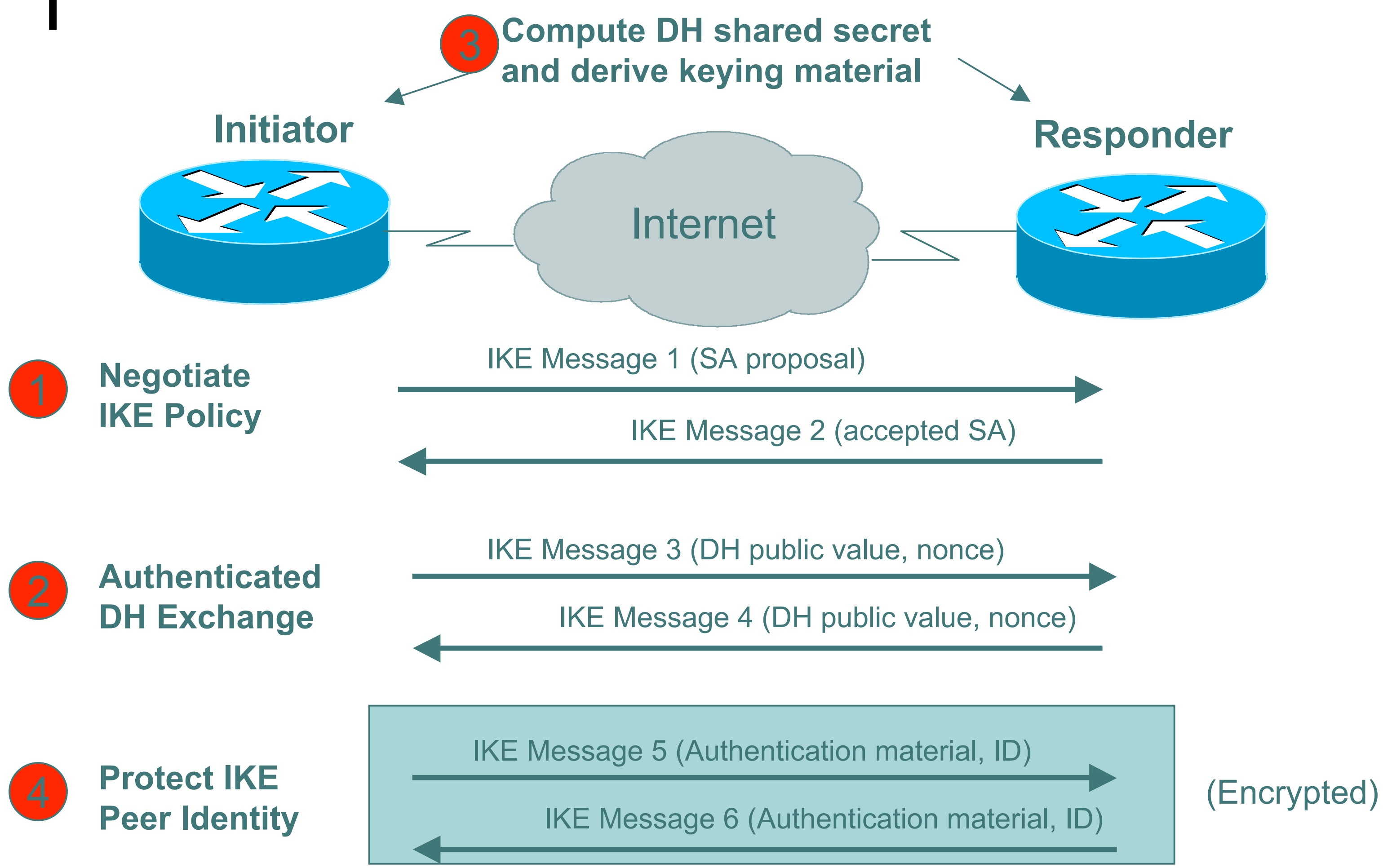
Payload Length: 2 bytes; length of payload (in bytes) including the header

Payload: The actual payload data

IKE Phase 1 Main Mode

- Main mode negotiates an ISAKMP SA which will be used to create IPsec Sas
- Three steps
  - SA negotiation (encryption algorithm, hash algorithm, authentication method, which DF group to use)
  - Do a Diffie-Hellman exchange
  - Provide authentication information
  - Authenticate the peer

IKE Phase 1 Main Mode



What Is Diffie-Hellman?

- First public key algorithm (1976)
- Diffie Hellman is a key establishment algorithm
  - Two parties in a DF exchange can generate a shared secret
  - There can even be N-party DF changes where N peers can all establish the same secret key
- Diffie Hellman can be done over an insecure channel
- IKE authenticates a Diffie-Hellman exchange 3 different ways
  - Pre-shared secret
  - Nonce (RSA signature)
  - Digital signature

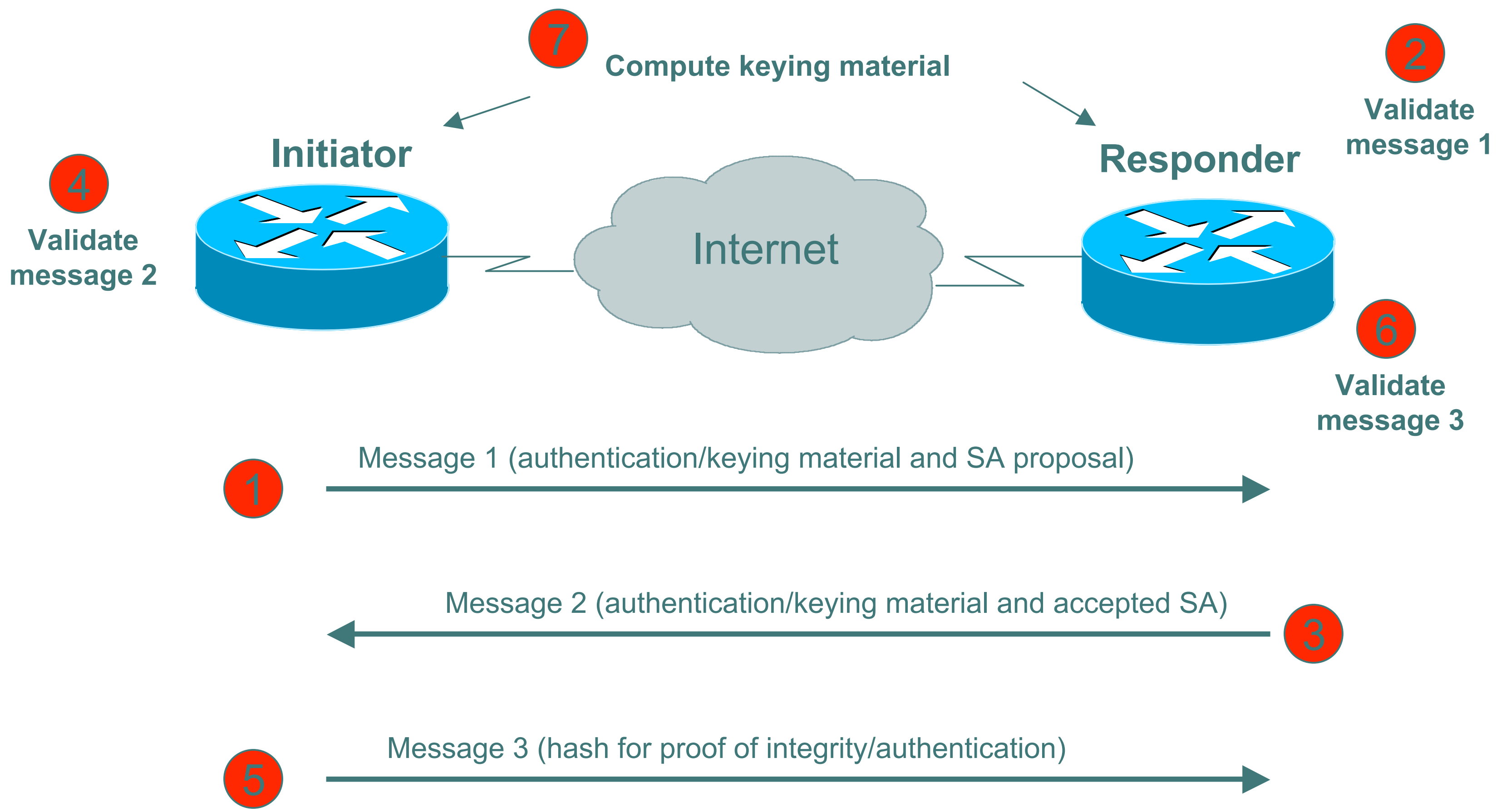
IKE Phase 1 Aggressive Mode

- Uses 3 (vs 6) messages to establish IKE SA
- No denial of service protection
- Does not have identity protection
- Optional exchange and not widely implemented

IKE Phase 2 Quick Mode

- All traffic is encrypted using the ISAKMP/IKE Security Association
- Each quick mode negotiation results in two IPsec Security Associations (one inbound, one outbound)
- Creates/refreshes keys

IKE Phase 2 Quick Mode



IKE Summary

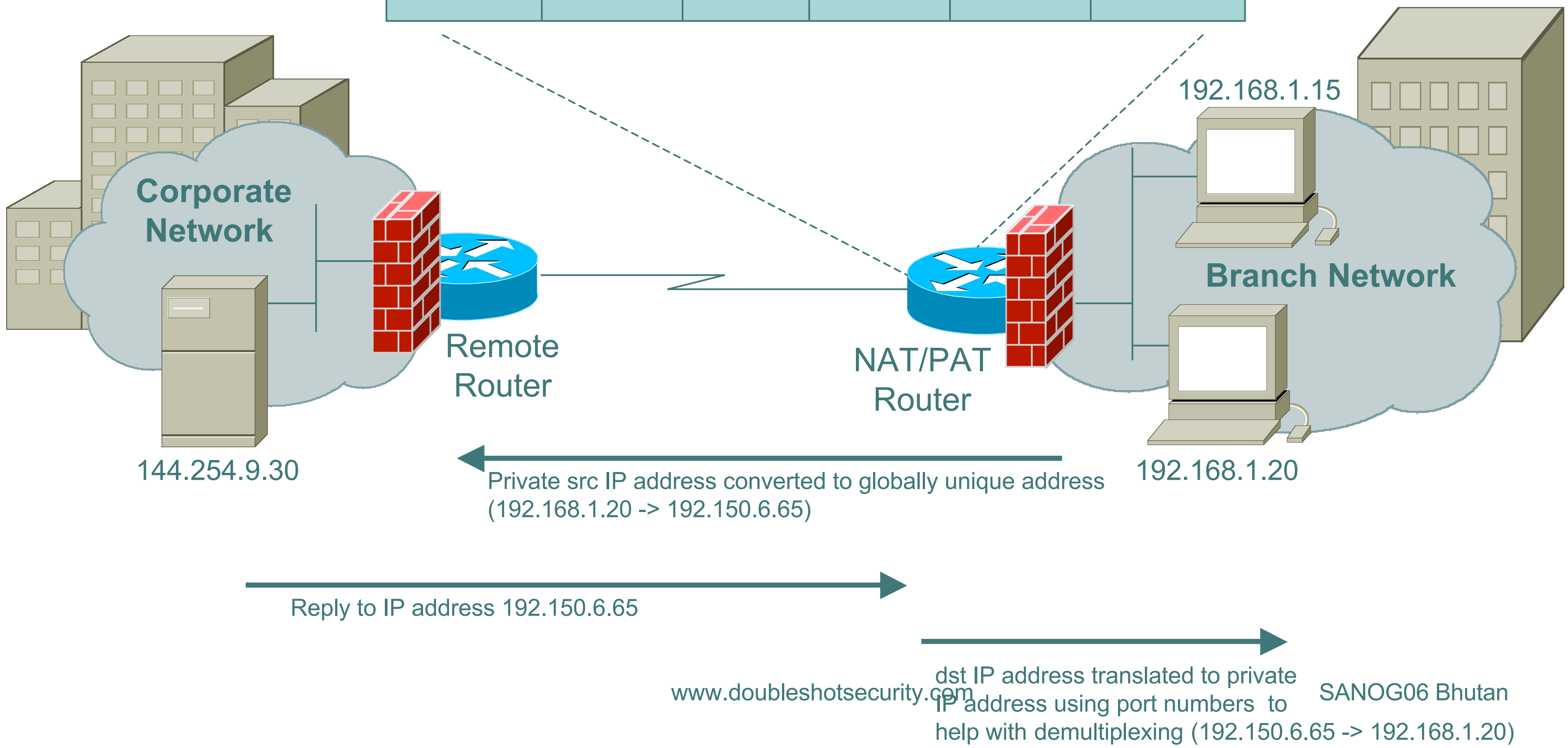
- Negotiates parameters to establish and secure a channel between two peers
- Provides mutual authentication
- Establishes authenticated keys between peers
- Manages IPsec SAs
- Provides options for negotiation and SA establishment
- IKEv2
  - User authentication
  - Dynamic addressing
  - NAT traversal

IPsec Issues

- Dynamic Addressing
- NAT/PAT
- Device vs User Authentication

NAT/PAT Problems

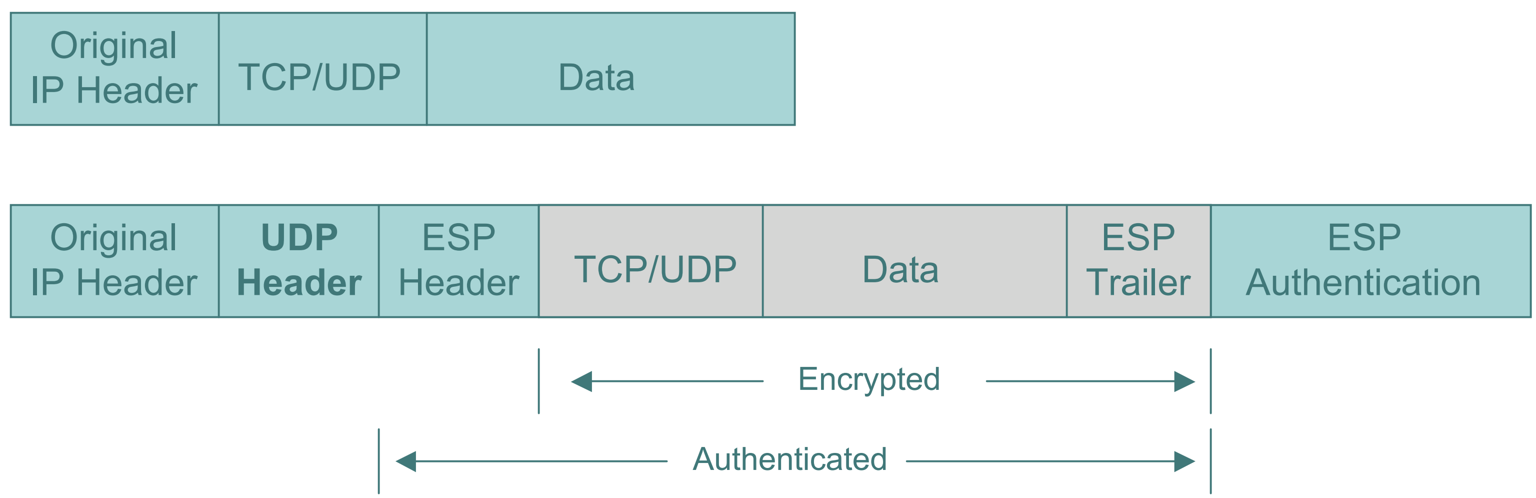
Original SRC IP	Translated SRC IP	Original SRC Port	Translated SRC Port	Original DST IP	Original DST Port
192.168.1.20	192.150.6.65	2654	6789	144.254.9.30	80
192.168.1.15	192.150.6.65	5876	6788	144.254.9.30	80



UDP Encapsulation of Transport Mode ESP Packets

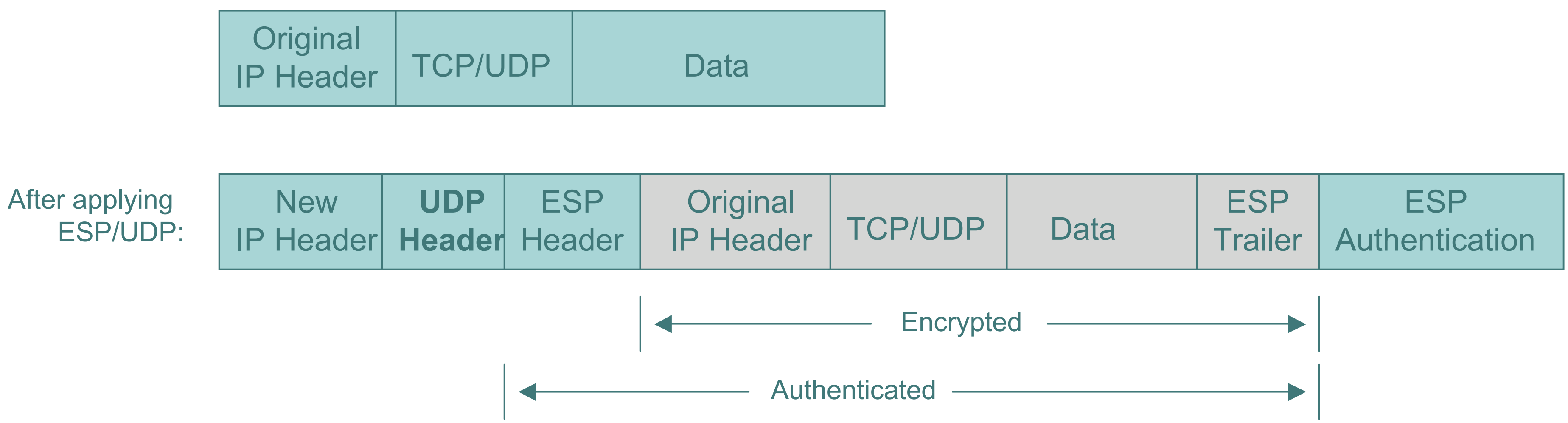
Transport Mode

After applying ESP/UDP:



UDP Encapsulation of Tunnel Mode ESP Packets

Tunnel Mode



Pretty Good IPsec Policy

- IKE Phase 1 (aka ISAKMP)
  - Main Mode
  - 3DES
  - SHA-1
  - DH Group 2 (MODP)
  - SA Lifetime (28880 seconds = 8 hours)
  - Pre-shared secret
- IKE Phase 2 (aka IPsec)
  - ESP Transport/Tunnel Mode
  - 3DES
  - SHA-1
  - PFS
  - DH Group 2 (MODP)
  - SA Lifetime (3600 seconds = 1 hour)

PFS- what is it?

- Perfect Forward Secrecy
- Doing new DH exchange to derive keying material

(DH used to derive shared secret which is used to derive keying material for IPsec security services)

Configuring IPsec

STEP 1 *Configure the IKE Phase 1 Policy (ISAKMP Policy)*

Cisco literature refers to IKE Phase 1 as the ISAKMP policy.
It is configured using the command:

crypto isakmp policy *priority*

Multiple policies can be configured and the priority number, which ranges from 1 to 10,000, denotes the order of preference that a given policy will be negotiated with an ISAKMP peer. The lower value has the higher priority. Once in the ISAKMP configuration mode, the following parameters can be specified are:

Encryption Algorithm
Hash Algorithm
Authentication Method
Group Lifetime

Configuring IPsec

STEP 2 *Set the ISAKMP Identity*

The ISAKMP identity specifies how the IKE Phase 1 peer is identified, which can be either by IP address or host name.
The command to use is:

crypto isakmp *identity* {*IP address* | *hostname*}

By default, a peer's ISAKMP identity is the peer's IP address.
If you decide to change the default just keep in mind that it is best to always be consistent across your entire IPsec-protected network in the way you choose to define a peer's identity.

Configuring IPsec

STEP 3 *Configure the IPsec AH and ESP Parameters*

The AH and ESP parameters are configured with the following commands:

crypto ipsec transform-set *transform-set-name* <transform 1> <transform 2>
mode [tunnel | transport]
crypto ipsec security-association lifetime seconds *seconds*

STEP 4 *Configure the IPsec Traffic Selectors*

The traffic selectors are configured by defining extended access-lists. The *permit* keyword causes all IP traffic that matches the specified conditions to be protected by IPsec

Configuring IPsec

STEP 5 *Configure the IKE Phase 2 (IPsec SA) Policy*

This step sets up a crypto map which specifies all the necessary parameters to negotiate the IPsec SA policy. The following commands are required:

crypto map *crypto-map-name seq-num* ipsec-isakmp
match address *access-list-id*
set peer [*IP address* | *hostname*]
set transform-set *transform-set-name*
set security-association lifetime seconds *seconds*
set pfs [group1 | group 2]

Configuring IPsec

STEP 6 *Apply the IPsec Policy to an Interface*

The configured crypto map is then applied to the appropriate interface using the crypto map *crypto-map-name* command.
It is possible to apply the same crypto map to multiple interfaces.
This case would require the use of the command:

crypto map *crypto-map-name* local-address *interface-id*

Using this command, the identifying interface will be used as the local address for IPsec traffic originating from or destined to those interfaces sharing the same crypto map. A loopback interface should be used as the identifying interface.

IPsec LAB

- IPsec configuration on Cisco Routers
- IPsec configuration on UNIX
- Secure Telnet between Cisco and UNIX host using IPsec