Network Security: Firewalls, IPSec

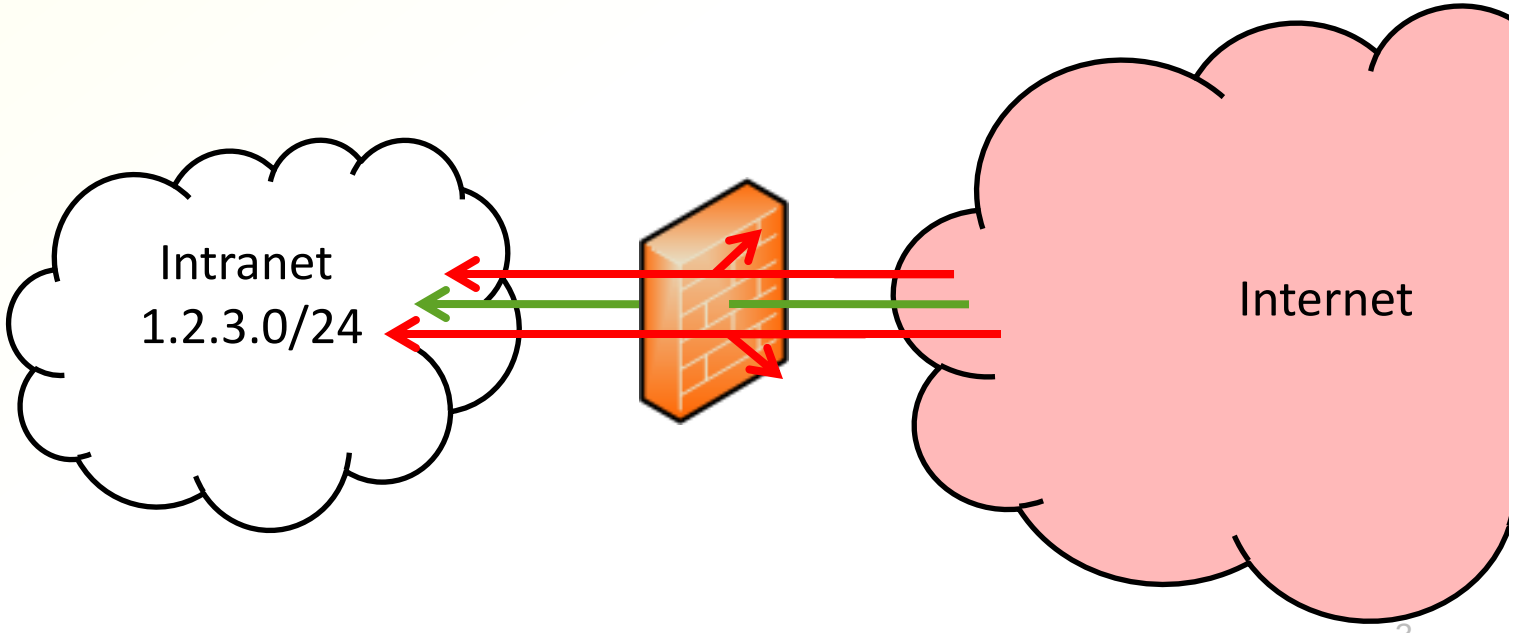
Tuomas Aura

T-110.5240 Network security
Aalto University, Nov-Dec 2010

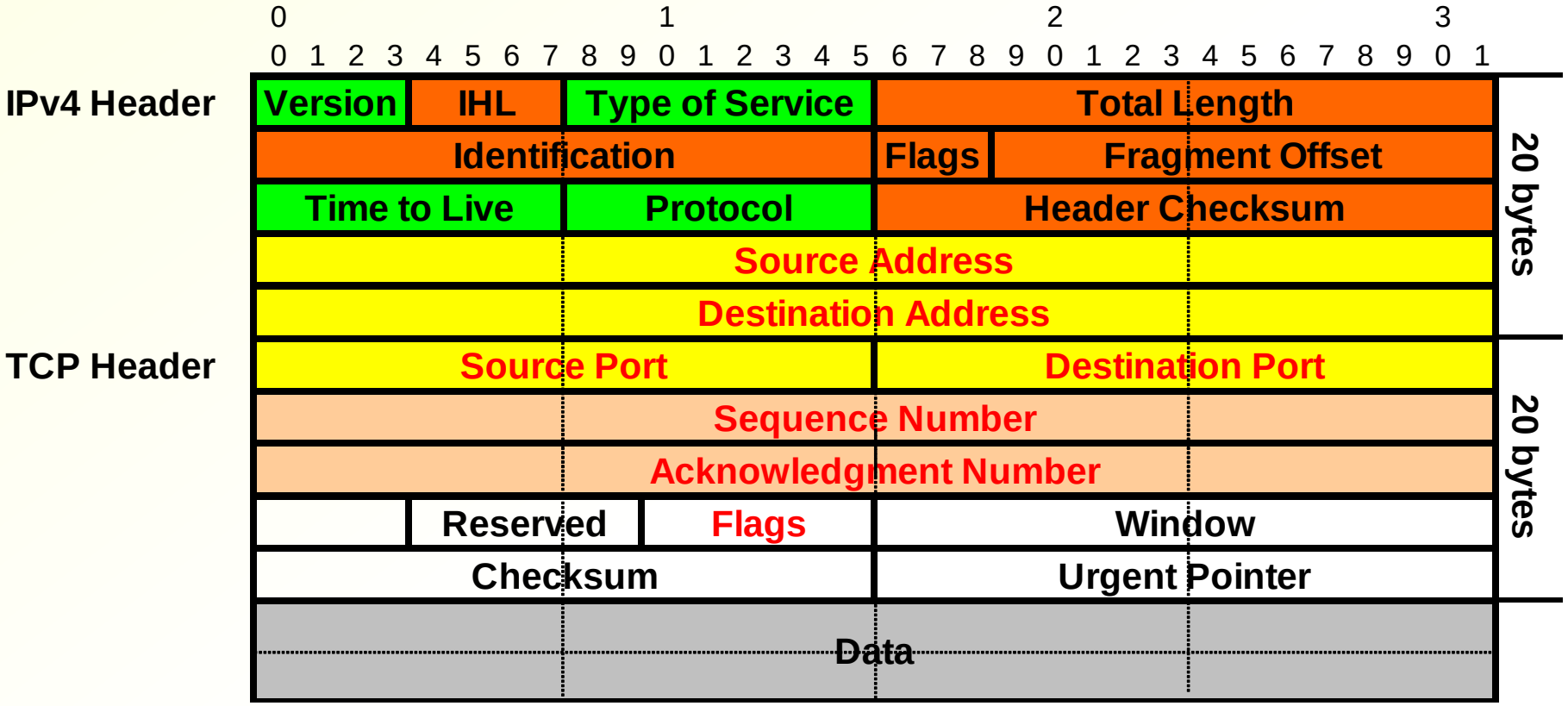
Firewalls:
Stateless packet filter

Firewall

- Perimeter defence:
  - Divide the world into the good/safe inside (intranet) and bad/dangerous outside (Internet)
  - Prevent anything bad from entering the inside
- Block communication that is evil, risky or just unnecessary



IPv4 and TCP headers



(TCP flags: CWR ECE URG ACK PSH RST SYN)

- Which field should a firewall use for filtering?

Stateless packet filter

- Allow or block IP packets based on their IP header fields and TCP/UDP port numbers
  - Fields with static locations in most IP packets: protocol (TCP/UDP/ICMP), source and destination IP address, source and destination port, TCP flags, ICMP type and code
- Packet filter is defined as a rule table
  - Linear list of rules
  - Each rule consist of conditions and an action
  - For each packet, the first matching rule is found
  - Two possible actions:
    allow (=accept, permit, bypass) or block (=drop, deny, discard), maybe also allow and log or block and log

Packet filter example (1)

- Example rule table: inbound email to our SMTP server 1.2.3.10

Protocol	Src IP	Src port	Dst IP	Dst port	Action	Comment
TCP	4.5.6.7	*	1.2.3.10	25	Block	Stop this spammer
TCP	*	*	1.2.3.10	25	Allow	Inbound SMTP
TCP	1.2.3.10	25	*	*	Allow	SMTP responses
*	*	*	*	*	Block	Default rule

- Note: The examples in this lecture are an abstraction and don't directly correspond to the way real firewalls are configured

Packet filter example (2)

- Allow web access from our subnet… not quite right!

Protocol	Src IP	Src port	Dst IP	Dst port	Action	Comment
TCP	1.2.3.0/24	*	*	80	Allow	Outbound HTTP requests
TCP	*	80	1.2.3.0/24	*	Allow	HTTP responses
*	*	*	*	*	Block	Default rule

- Allow only outbound connections:

Protocol	Src IP	Src port	Dst IP	Dst port	Flags	Action	Comment
TCP	1.2.3.0/24	*	*	80		Allow	Outbound HTTP requests
TCP	*	80	1.2.3.0/24	*	ACK	Allow	HTTP responses
*	*	*	*	*		Block	Default rule

(TCP packets, except the first SYN, have ACK flag set)

Packet filter example (3)

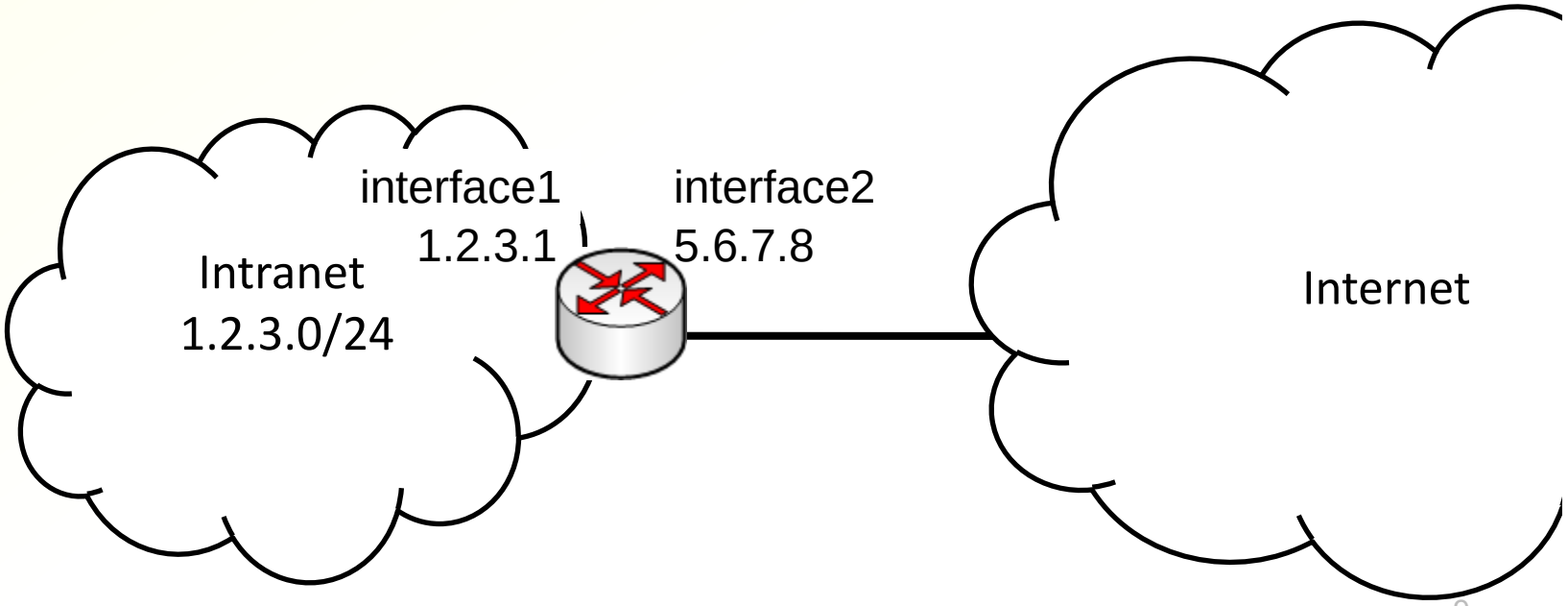
- University lab network 1.2.3.0/24 (address 1.2.3.0, netmask 255.255.255.0)
- HTTP/Mail/DNS server 1.2.3.10

Protocol	Src IP	Src port	Dst IP	Dst port	Flags	Action	Comment
UDP	*	*	*	53		Allow	DNS queries in and out
UDP	*	53	*	*		Allow	DNS responses
TCP	5.4.3.2	*	1.2.3.10	53		Allow	DNS zone transfer
TCP	*	*	1.2.3.10	25		Allow	Inbound SMTP
TCP	*	*	1.2.3.10	80		Allow	Inbound HTTP
TCP	1.2.3.121	*	*	*		Block	Bob's test machine
TCP	*	*	1.2.3.121	*		Block	Bob's test machine
TCP	*	*	1.2.3.0/24	22		Allow	Inbound SSH
TCP	1.2.3.0/24	*	*	*		Allow	All outbound TCP
TCP	*	*	1.2.3.4/24	*	ACK	Allow	All TCP responses
*	*	*	*	*		Block	Default rule

- Is this correct? Could we limit inbound DNS queries to the server?

Router as packet filter

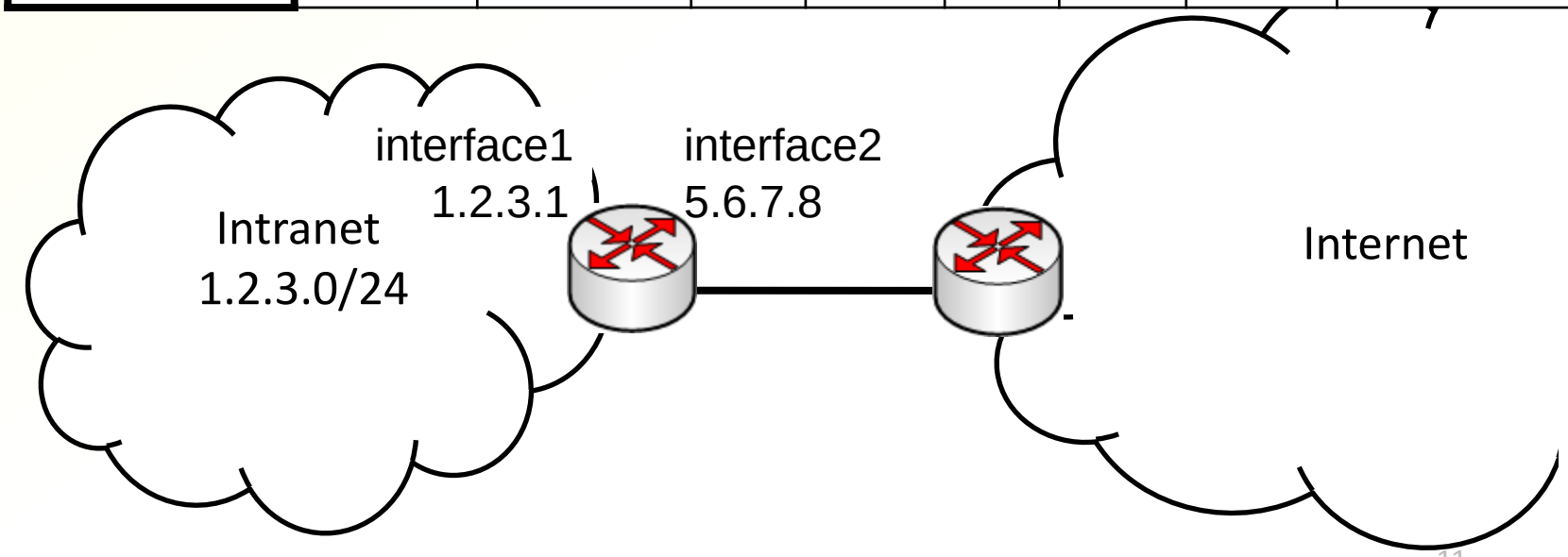
- Firewall rule table is similar to a routing table, with the option of dropping some packets
- Most routers can be used as a packet filter
  - Choice of filters may affect router throughput



Anti-spoofing filter example

- Filter based on input interface (part of the policy shown only):

Input interface	Protocol	Src IP	Port	Dst IP	Port	Flags	Action	Comment
2	*	1.2.3.0/24	*	*	*		Block	Ingress filter
2	*	5.6.7.8	*	*	*		Block	Router address
1	*	1.2.3.1	*	*	*		Block	Router address
1	*	1.2.3.0/24	*	*	*		Allow	Egress filter
1	*	*	*	*	*		Block	Default rule (If1)
…							…	



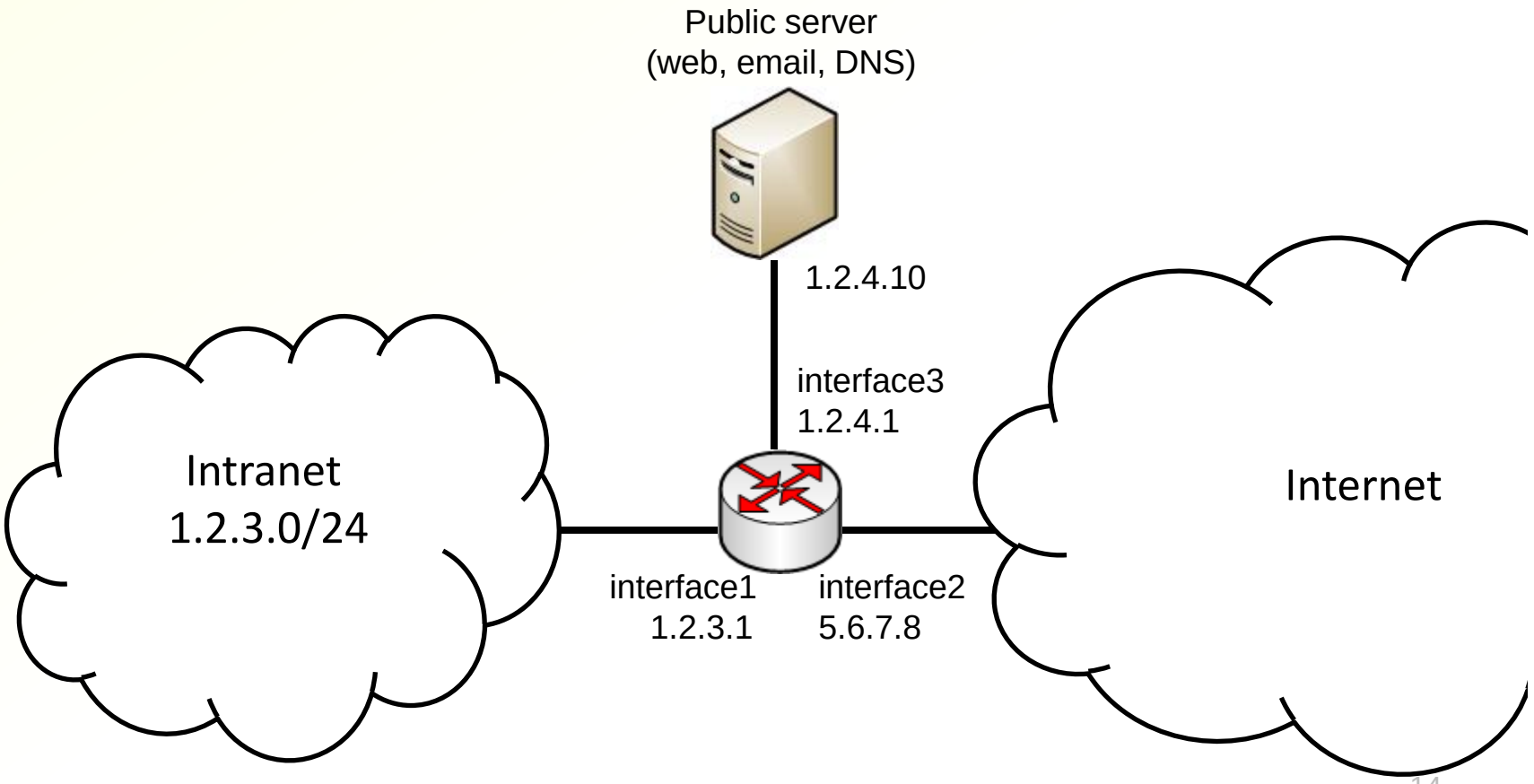
Dynamic packet filter

Dynamic firewall

- Stateful filter: change filtering rules based on previously seen packets
- Outbound TCP or UDP packet creates a pinhole for inbound packets of the same connection
  - Unlike stateless packet filter, can support UDP connections
  - TCP pinhole closed with connection, UDP after eg. 30 min
- May also allow inbound ICMP messages that match outbound traffic
- Support for special protocols:
  - FTP: firewall may sniff PORT command in FTP to open port for the inbound connections
  - X Windows

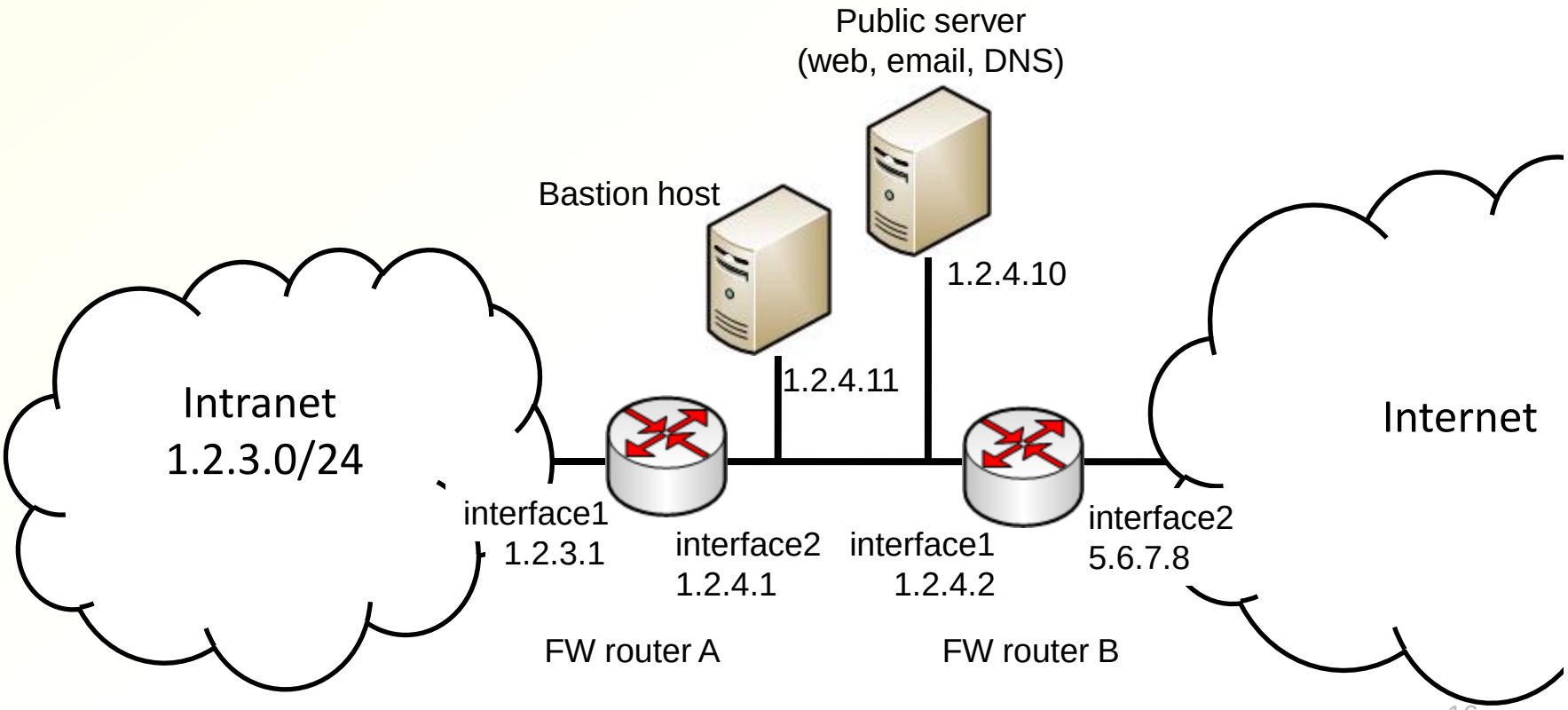
Typical network topology (1)

- Services accessible from the Internet are isolated to a demilitarized zone (DMZ), i.e. somewhere between the intranet and Internet



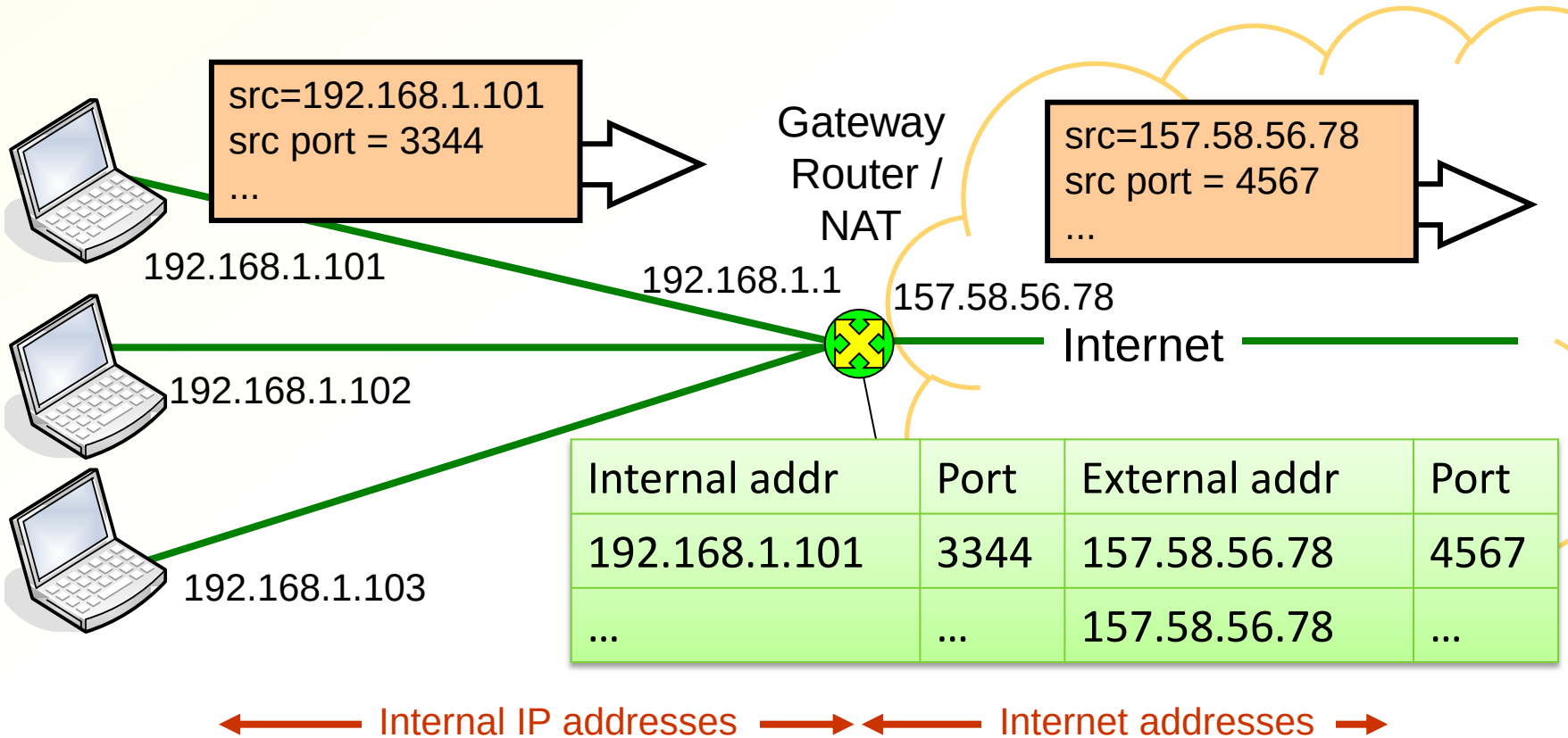
Typical network topology (2)

- Two-firewall configuration for isolating publicly-accessible services from the Internet
- All inbound connections use ssh and go through a hardened bastion host in the DMZ



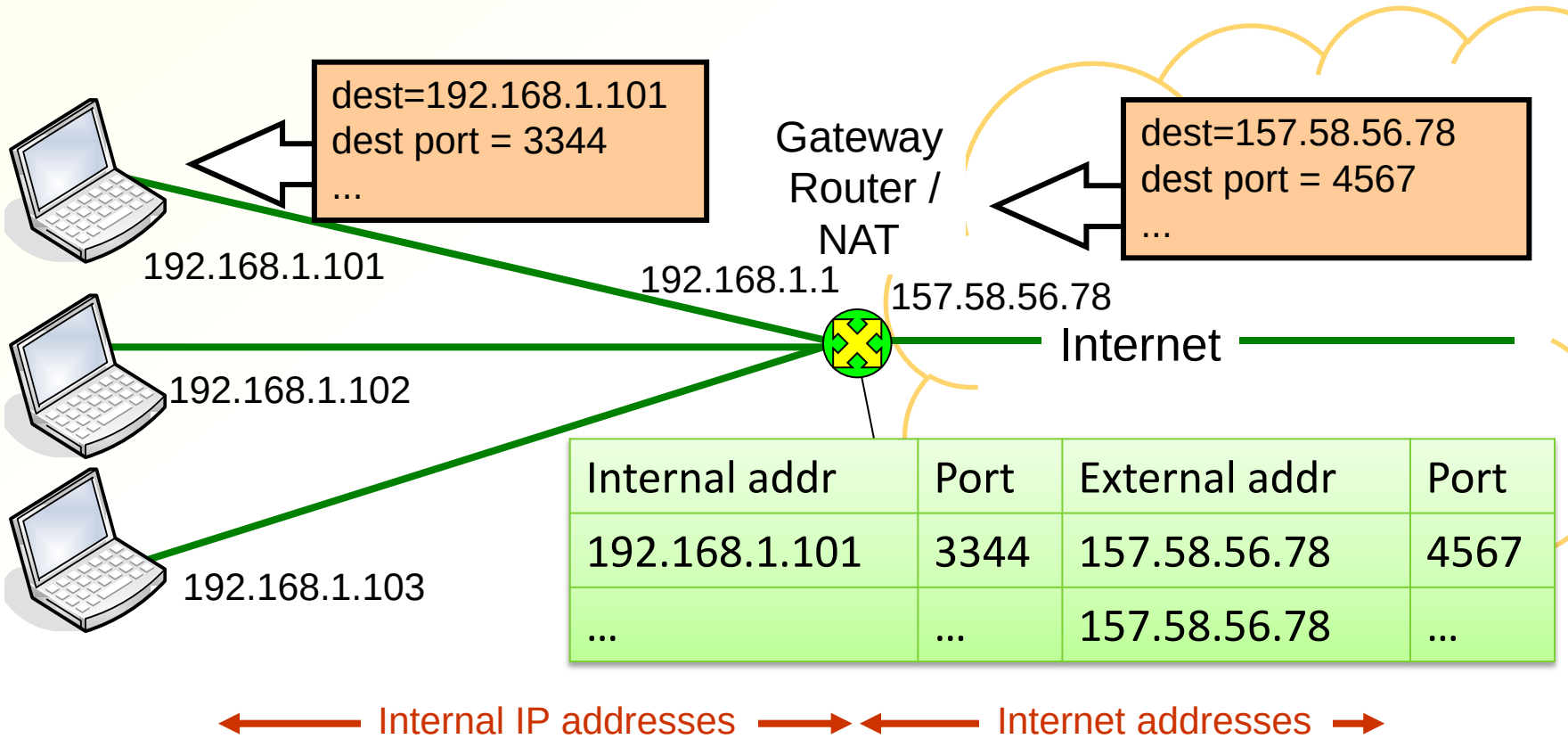
NAT

- IPv4 addresses are in short supply
- Native address translator (NAT) is a mechanisms for sharing one IPv4 address between multiple hosts
- Hosts behind NAT can only act as TCP or UDP clients



NAT

- IPv4 addresses are in short supply
- Native address translator (NAT) is a mechanisms for sharing one IPv4 address between multiple hosts
- Hosts behind NAT can only act as TCP or UDP clients



NAT as a firewall

- NAT maps internal <private IP addr, port> pairs to external <public IP addr, port> pairs and back
- NAT creates the mapping after seeing an outbound packet → a node on the intranet must initiate the connection→ NAT acts as a dynamic firewall
- NAT reference types:
  - Full cone NAT: NAT doesn't remember peer addresses
  - Port-restricted cone NAT: NAT remembers peer IP address and port and filters inbound packets
  - Symmetric NAT: different external port (and even address) depending the peer address and port
- Port-restricted and symmetric NATs function as firewalls
- Warning: real NATs rarely implement exactly one of the one of the reference types, and often have additional firewall functionality

Transport and application-layer firewalls

Circuit-level proxy

- Transport-layer proxy as a firewall
  - When an intranet client needs to connect to a server outside, it connects to the proxy instead
  - Proxy terminates TCP and UDP connections. Creates a second connection to the server on the Internet
  - Proxy is simpler than a host, easier to harden against attacks
  - Proxy can filter and normalizes connections
- SOCKS management protocol between client and firewall
  - Client requests new connections from firewall
  - Authentication and authorization of client requests, e.g. GSSAPI
  - Error messages to client
  - Supported by most web browsers
- Firewall router can be set up to forward only some connections to the proxy for closer inspection

Application-level firewall

- Application-level firewall filters application data
  - E.g. email gateway, intercepting web proxy
  - Need to implement the entire application protocol
- Telephone call blocking and barring vs. wiretapping
- Encrypted data cannot be filtered → what to do?
- Are latest applications and features supported?

Firewall issues

Firewall traversal

- Network admins prefer to block traffic by default → New applications and protocols will not work
- New applications will not gain popularity if an administrative decision is needed at each site → application developers (and users) do their best to circumvent firewalls
  - Web services over port 80, everything over port 443
  - Skype, Bittorrent etc.
- Question: Should all new network applications be standardized and get a port number from IANA, so that they can be filtered by the firewall?
  - Big debate in the 90s, now everything uses port 80

Firewall limitations

- May prevent people from doing their work
  - Try to convince a network admin to open a pinhole for your server
- Network admins are often reluctant to change firewall policies in case something breaks
- Makes network diagnostics harder
- Firewall configuration errors are common
- Coarse-grained filtering for efficient routing and administration
- Perimeter defence is ineffective in large networks
  - There are always some compromised nodes inside
- Potential unfiltered ingress routes that circumvent firewalls:
  - Dial-up modem connections in and out
  - Unauthorized access points
- Laptops move in and out of the intranet
- Security of home gateways and other network devices is questionable
- Most applications now use TCP port 80 or 443, or use other clever tricks to traverse firewalls

Exercises

- Why cannot ingress filtering ever stop all IP spoofing attacks?
- Do you find any mistakes or shortcomings in the firewall policy examples of this lecture? Can they be improved?
- Find out what kind of firewall capabilities your home gateway router/NAT has.
- Find the firewall configuration of a small network. Try to understand each line of the policy. Have compromises on security been made to achieve better performance, to make management easier, or because of limitations in the firewall platform?
- Write firewall policies for the Network topology example (2) in an earlier slide. What compromises will you have to make if the firewalls are stateless packet filters and do not support filtering based on the input interface.
- Stateless firewall typically allows all inbound TCP packets with the ACK flag set. On a 1 GB/s network, how difficult is it for external attackers to spoof some TCP packets that match the sequence numbers of an intranet TCP connection?

Related reading

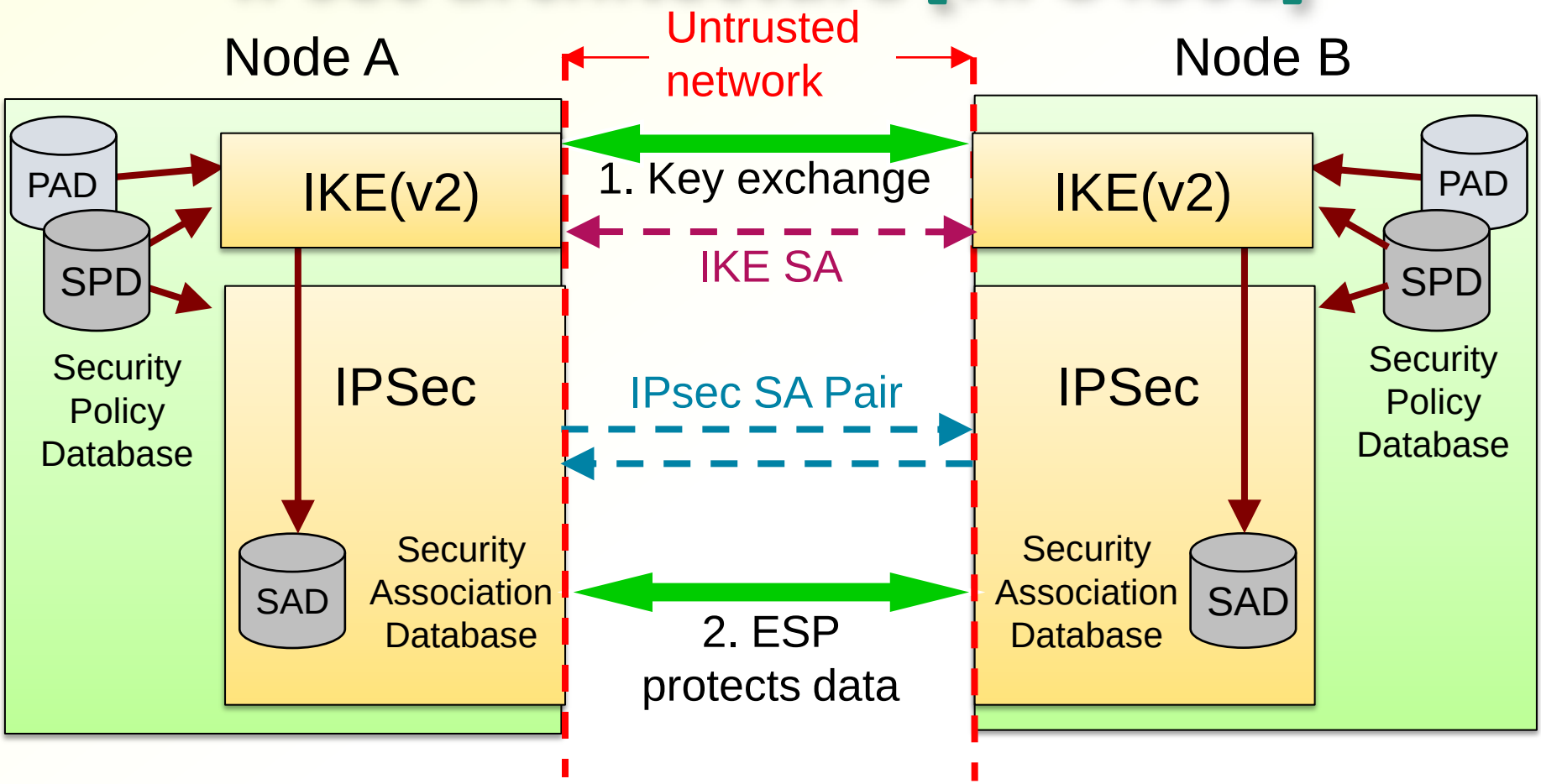
- William Stallings. Network security essentials: applications and standards, 3rd ed.: chapter 11
- William Stallings. Cryptography and Network Security, 4th ed.: chapter 20
- Kaufmann, Perlman, Speciner. Network security, 2nd ed.: chapter 23
- Ross Anderson. Security Engineering, 2nd ed.: chapter 21.4.2
- Dieter Gollmann. Computer Security, 2nd ed.: chapter 13.6

IPsec:
Architecture and protocols

Internet protocol security (IPsec)

- Network-layer security protocol
  - Protects IP packets between two hosts or gateways
  - Transparent to transport layer and applications
  - IP addresses used to as host identifiers
- Two steps:
  1. IKE creates security associations
  2. ESP session protocol protects data
- Specified by Internet Engineering Task Force (IETF)
  - Original goal: encryption and authentication layer that will replace all others
  - Sales point for IPv6; now also in IPv4

IPsec architecture [RFC4301]



- Security associations (SA) created by IKE, used by IPsec ESP
- Security policy guides SA creation and selection for use
- IPsec is part of the IP layer in the OS kernel; IKE is a user-space service (daemon)

Internet Key Exchange (IKE)

- IKE(v1) [RFC 2407, 2408, 2409]
  - Framework for authenticated key-exchange protocols, typically with Diffie-Hellman
  - Multiple authentication methods:
    certificates, pre-shared key, Kerberos
  - Two phases: Main Mode (MM) creates an ISAKMP SA (i.e. IKE SA) and Quick Mode (QM) creates IPsec SAs
  - Main mode (identity-protection mode) and optimized aggressive mode
  - Interoperability problems: too complex to implement and test all modes; specification incomplete
- IKEv2 [RFC 4306]
  - Redesign of IKE: less modes and messages, simpler to implement
  - Initial exchanges create the IKE SA and the first IPsec SA
  - CREATE_CHILD_SA exchange can create further IPsec SAs
  - EAP authentication for extensions
- Works over UDP port 500

Internet Key Exchange (IKEv2)

Initial exchanges:

1. I → R: HDR(A,0), SAi1, KEi, Ni
2. R → I: HDR(A,B), SAr1, KEr, Nr, [CERTREQ]
3. I → R: HDR(A,B), SK { IDi, [CERT,] [CERTREQ,] [IDr,] AUTHi, SAi2, TSi, TSr }
4. R → I: HDR(A,B), SK { IDr, [CERT,] AUTHr, SAr2, TSi, TSr }

A, B = SPI values that identity the protocol run and the created IKE SA

Nx = nonces

SAx1 = offered and chosen algorithms, DH group

KEx = Diffie-Hellman public key

IDx, CERT = identity, certificate

AUTHi = Sign_I (Message 1, Nr, h(SK, IDi))

AUTHr = Sign_R (Message 2, Ni, h(SK, IDr))

SK = $h(Ni, Nr, g^{xy})$ — a bit simplified, 6 keys are derived from this

SK { … } = E_{SK}(…, MAC_{SK}(…)) — MAC and encrypt

SAx2, TSx = parameters for the first IPsec SA (algorithms, SPIs, traffic selectors)

CERTREQ = recognized root CAs (or other trust roots)

Internet Key Exchange (IKEv2)

Initial exchanges:

1. I → R: HDR(A,0), SAi1, KEi, Ni
2. R → I: HDR(A,B), SAr1, KEr, Nr, [CERTREQ]
3. I → R: HDR(A,B), SK { IDi, [CERT,] [CERTREQ,] [IDr,] AUTHi, SAi2, TSi, TSr }
4. R → I: HDR(A,B), SK { IDr, [CERT,] AUTHr, SAr2, TSi, TSr }

A, B = SPI values that identity the protocol run and the created IKE SA

Nx = nonces

SAx1 = offered and chosen algorithms, DH

KEx = Diffie-Hellman public key

IDx, CERT = identity, certificate

AUTHi = $Sign_I$ (Message 1, Nr, h(SK, IDi))

AUTHr = $Sign_R$ (Message 2, Ni, h(SK, IDr))

SK = $h(Ni, Nr, g^{xy})$ — a bit simplified, 6 key

SK { … } = E_{SK}(…, MAC_{SK}(…)) — MAC and e

SAx2, TSx = parameters for the first IPsec

CERTREQ = recognized root CAs (or other

Secret, fresh session key?
Mutual authentication?
Entity authentication and
key confirmation?
Separation of long and short-term secrets?
Contributory?
Perfect forward secrecy?
Integrity check for initial negotiation?
Non-repudiation or plausible deniability?
Identity protection?
DoS protection?

IKEv2 with a cookie exchange

- Responder may respond to the initial message by sending a cookie
- Goal: prevent DOS attacks from a spoofed IP address

1. I → R: HDR(A,0), SAi1, KEi, Ni
2. R → I: HDR(A,0), N(COOKIE) // R stores no state
3. I → R: HDR(A,0), N(COOKIE), SAi1, KEi, Ni
4. R → I: HDR(A,B), SAr1, KEr, Nr, [CERTREQ] // R creates a state
5. I → R: HDR(A,B), SK{ IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr }
6. R → I: HDR(A,B), E_{SK} (IDr, [CERT,] AUTH, SAr2, TSi, TSr)

How to bake a good cookie? For example:
COOKIE = h($N_{R\text{-periodic}}$, IP addr of I, IP addr of R) where $N_{R\text{-periodic}}$ is a periodically changing secret random value know only by the responder R

Security Associations (SA)

- One IKE SA for each pair of nodes
  - Stores the master key SK = h(Ni, Nr, g^{xy}) for creating IPsec SAs
- At least one IPsec SA pair for each pair of nodes
  - Stores the negotiated session protocol, encryption and authentication algorithms, keys and other session parameters
  - Stores the algorithm state
  - IPsec SAs always come in pairs, one in each direction
- SAs are identified by a 32-bit security parameter index (SPI) [RFC4301]
  - For unicast traffic, the destination node selects an SPI value that is unique to that destination
- Node stores SAs in a security association database (SAD)

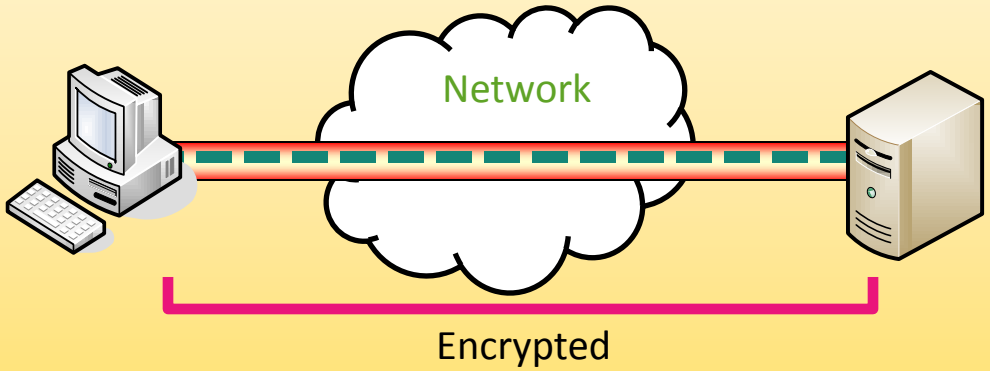
Session protocol

- Encapsulated Security Payload (ESP) [RFC 4303]
  - Encryption and/or MAC for each packet
  - Optional replay prevention with sequence numbers
  - Protects the IP payload (= transport-layer PDU) only
- ESP with encryption only is insecure
- Deprecated: Authentication Header (AH)
  - Do not use for new applications
  - Authentication only
  - Protects payload and some IP header fields

Session protocol modes

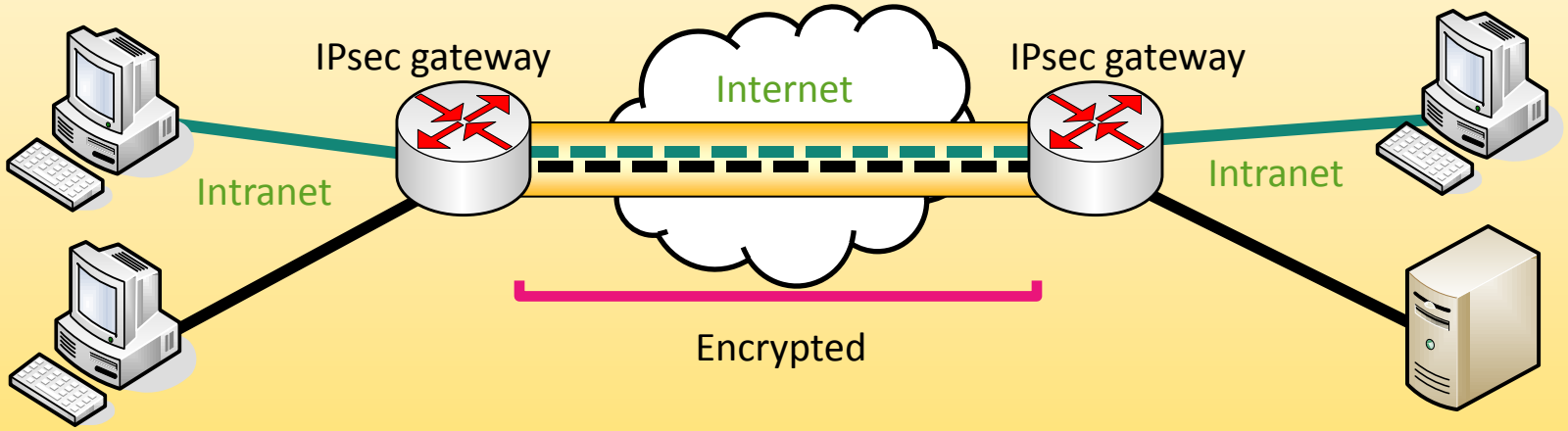
Transport mode

Encryption and/or authentication from end host to end host



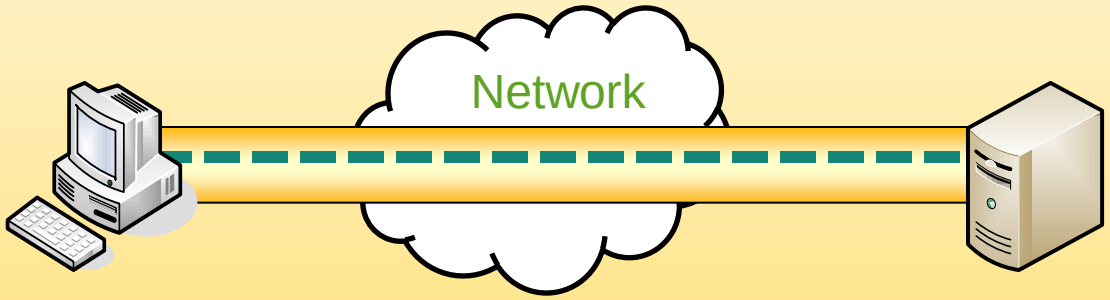
Tunnel mode

Encryption and/or authentication between two gateways

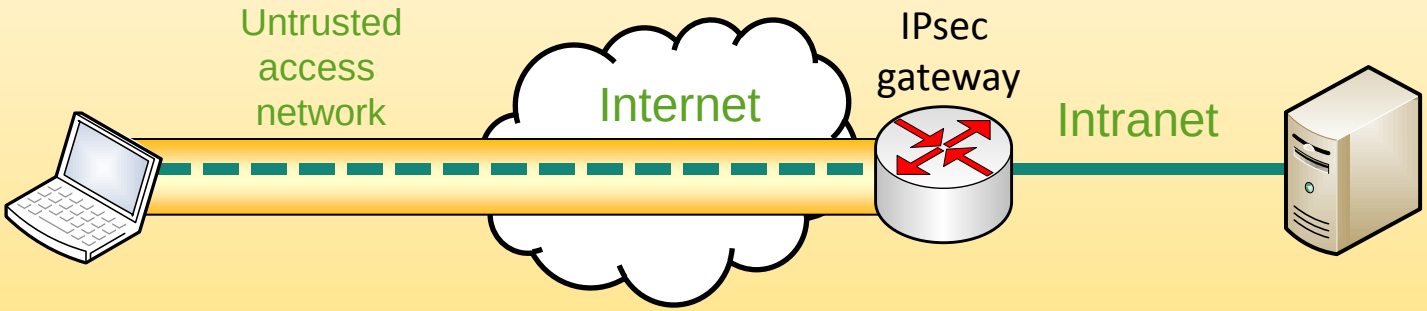


Using tunnel mode with hosts

Tunnel mode - between end hosts (equivalent to transport mode)

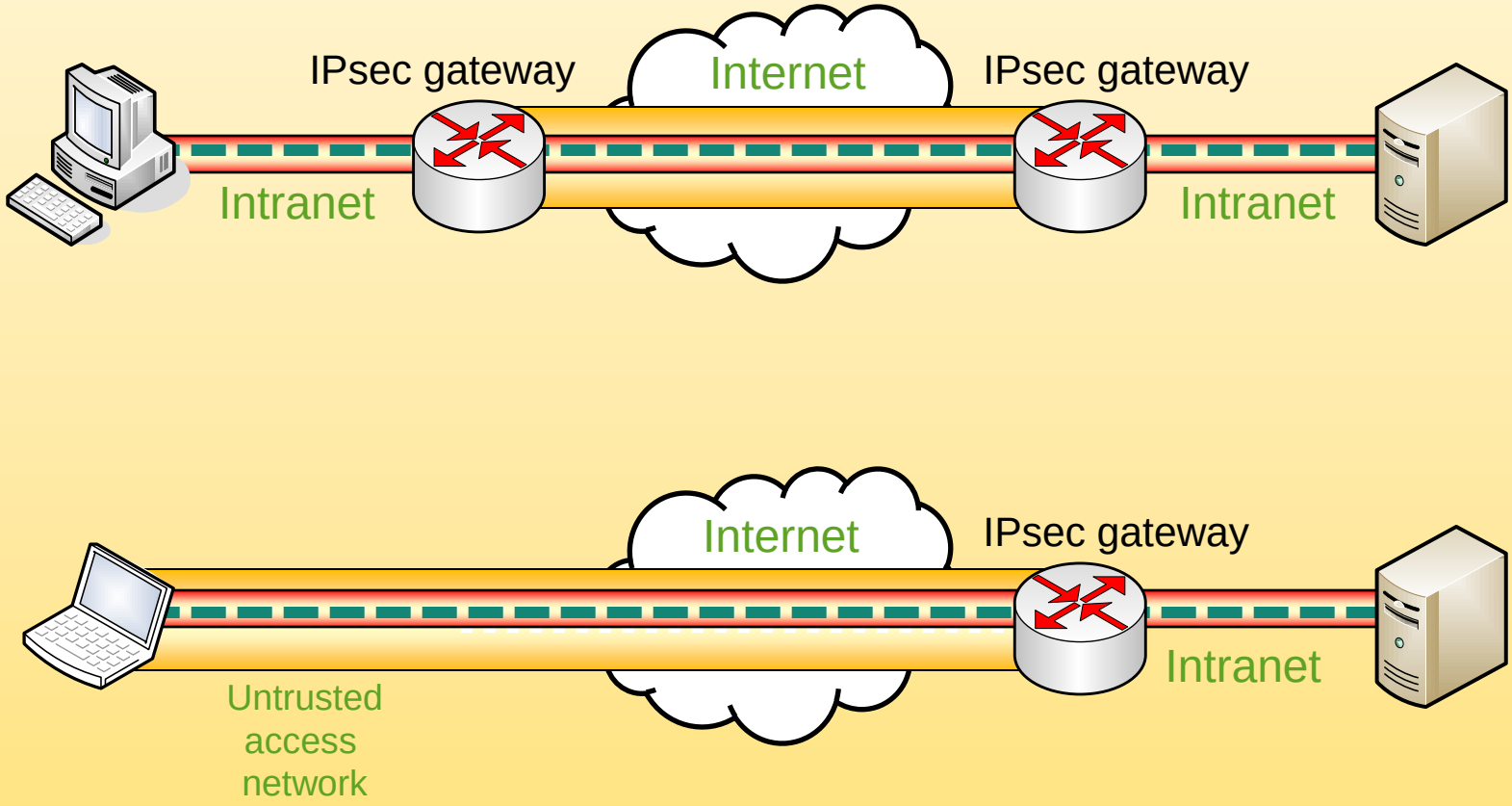


Tunnel mode - between a host and a gateway



Nested protection

Nested tunnel and transport mode

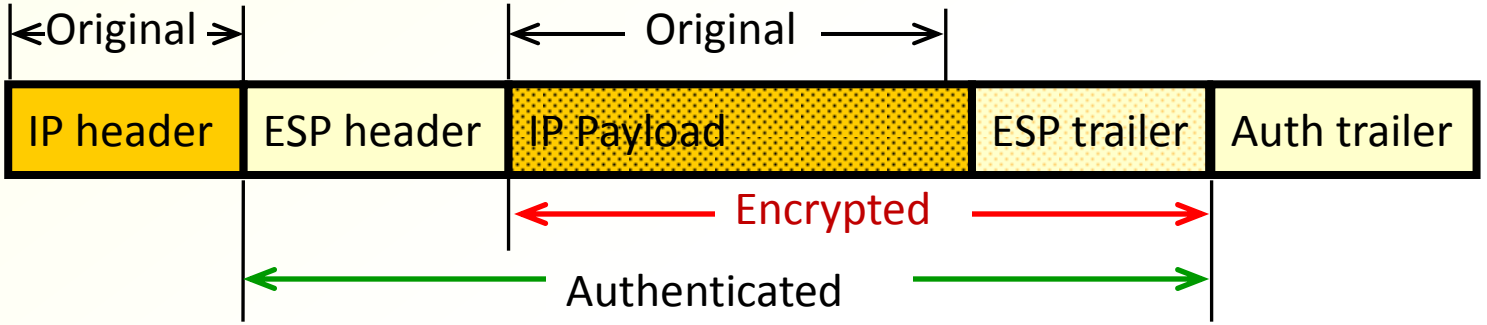


ESP packet format (1)

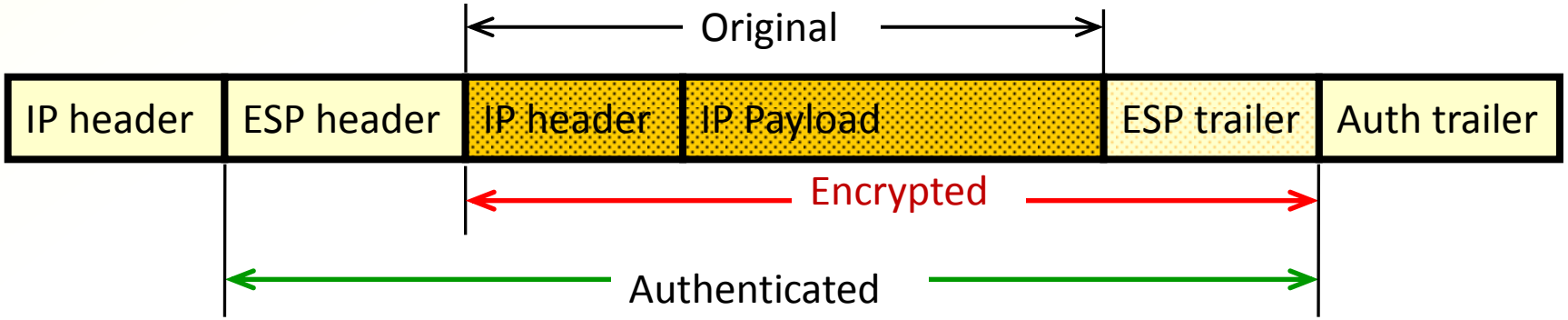
Original packet:

ESP header and trailer =
SPI + Sequence number + Padding
ESP authentication trailer =
message authentication code (MAC)

ESP in transport mode:



ESP in tunnel mode:



IPsec databases

- Security association database (SAD)
  - Contains the dynamic protection state
- Security policy database (SPD)
  - Contains the static security policy
  - Usually set by system administrators (e.g. Windows group policy), although some protocols and applications make dynamic changes
- Peer authorization database (PAD)
  - Needed in IKE for mapping between authenticated names and IP addresses
  - Conceptual; not implemented as an actual database
- Additionally, the IKE service stores IKE SAs:
  - Master secret created with Diffie-Hellman
  - Used for instantiating IPsec SAs

(Note: our description of SDP differs somewhat from RFC4301 and is probably closer to most implementations)

Security policy database (SPD)

- Specifies the static security policy
- Multi-homed nodes have a separate SPD for each network interface
- Policy maps inbound and outbound packets to actions
  - SPD = linearly ordered list of policies
  - Policy = selectors + action
  - The first policy with matching selectors applies to each packet
- Policy selectors:
  - Local and remote IP address
  - Transport protocol (TCP, UDP, ICMP)
  - Source and destination ports
- Actions: BYPASS (allow), DISCARD (block), or PROTECT
  - PROTECT specifies also the session protocol and algorithms
  - Packet is mapped to a suitable SA
  - If the SA does not exist, IKE is triggered to create one
  - SPD stores pointers to previously created SA
- Policies at peer nodes must match if they are to communicate

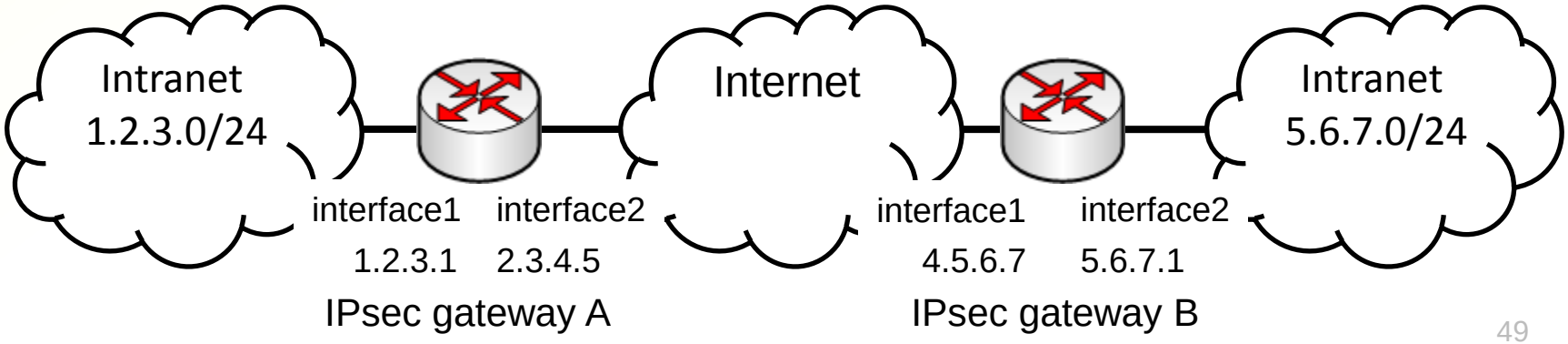
Gateway SPD/SAD example

SPD of gateway A, interface 2

Protocol	Local IP	Port	Remote IP	Port	Action	Comment
UDP	2.3.4.5	500	4.5.6.7	500	BYPASS	IKE
*	1.2.3.0/24	*	5.6.7.0/24	*	ESP tunnel to 4.5.6.7	Protect VPN traffic
*	*	*	*	*	BYPASS	All other peers

SAD of gateway 1

SPI	SPD selector values	Protocol	Algorithms, keys and algorithm state
spi1	UDP,1.2.3.0/24,5.6.7.0/24	ESP tunnel from 4.5.6.7	…
spi2	—	ESP tunnel to 4.5.6.7	…



IPsec policy implementation differences

- Historically, IPsec and firewalls have different models of the network:
  - Firewall is a packet filter: which packets to drop?
  - IPsec sits between the secure and insecure areas (host and network at IPsec hosts, intranet and Internet at IPsec gateways) and encrypts packets that leave the secure side

  Firewall and IPsec policies can, however, be unified
- In some IPsec implementations, the policy is specified in terms of source and destination addresses (like a typical firewall policy), instead of local and remote addresses

  → mirror flag is shorthand notation to indicates that the policy applies also with the source and destination reversed

Mirror	Protocol	Source IP	Port	Destination IP	Port	Action	Comment
yes	UDP	2.3.4.5	500	4.5.6.7	500	BYPASS	IKE
yes	*	1.2.3.0/24	*	5.6.7.0/24	*	ESP tunnel to 4.5.6.7	Protect VPN traffic
yes	*	*	*	*	*	BYPASS	All other peers

Some problems with IPsec

IPsec and NAT

- Problems:
  - NAT cannot multiplex IPsec: impossible to modify SPI or port number because they are authenticated

  → Host behind a NAT could not use IPsec
- NAT traversal (NAT-T):
  - UDP-encapsulated ESP (port 4500)
  - NAT detection: extension of IKEv1 and IKEv2 for sending the original source address in initial packets

  → Host behind a NAT can use IPsec

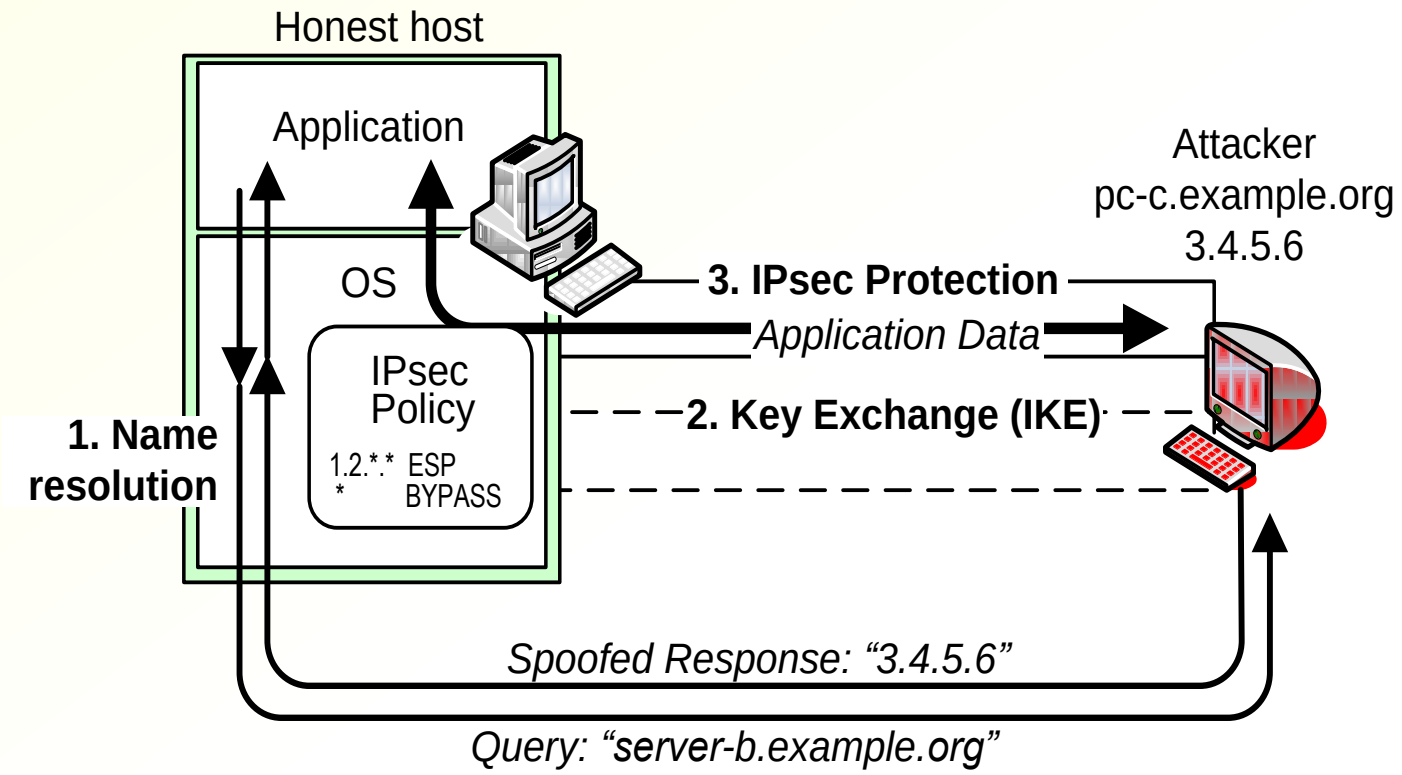
IPsec and mobility

- Problem:
  IPsec policies and SAs are bound to IP addresses.
  Mobile node's address changes
- Mobile IPv6 helps: home address (HoA) is stable.
  But mobile IPv6 depends on IPsec for the tunnel between HA and MN. → Chicken-and-egg problem
- Solutions:
  - IPsec changed to look up inbound SAs by SPI only
  - IPsec-based VPNs from mobile hosts do not use the IP address as selector. Instead, proprietary solutions
  - MOBIKE mobility protocol

IPsec and Identifiers

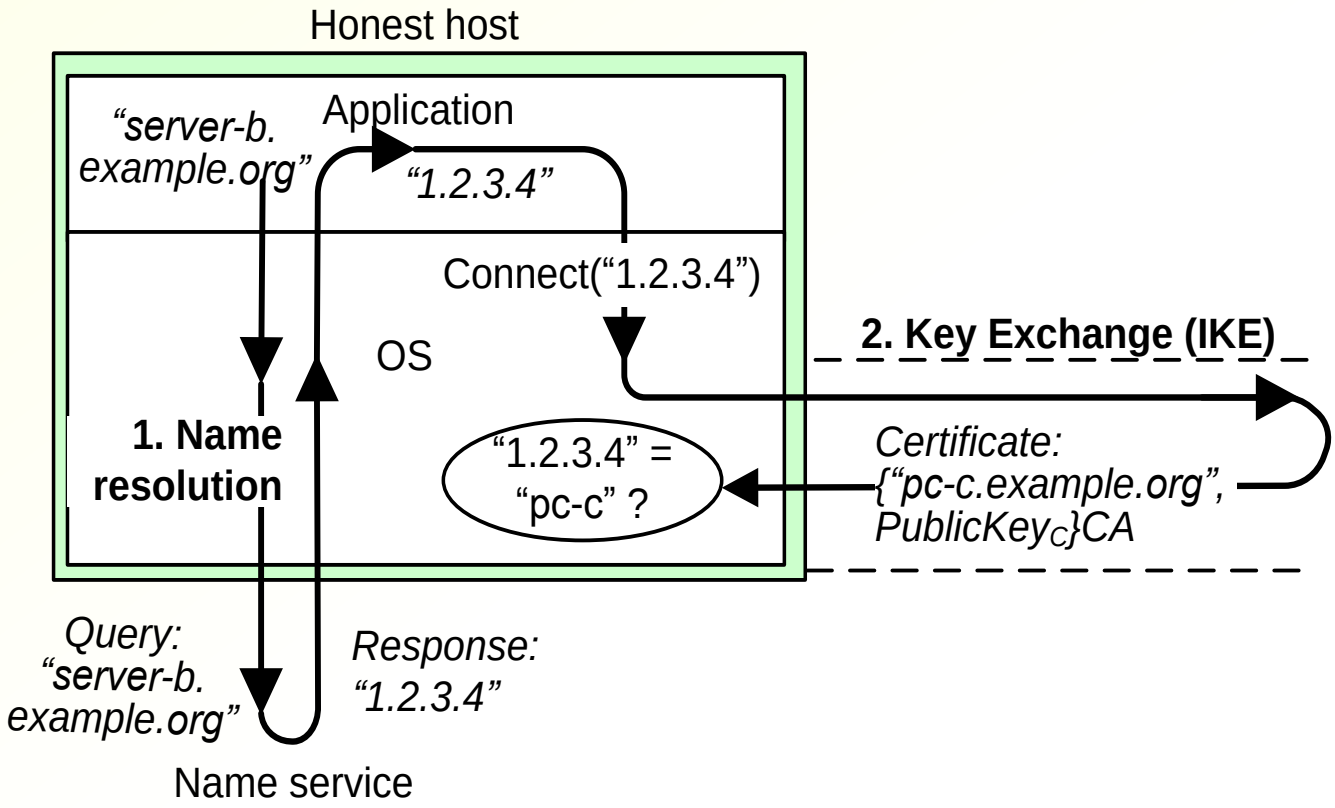
- Application opens a connection to an IP address. IPsec uses the IP addresses as policy selector
- IKE usually authenticates the remote node by its DNS name
- Problem: No secure mapping between the two identifier spaces: DNS names and IP addresses

Classic IPsec/DNS Vulnerability



- IPsec policy selection depends on secure DNS

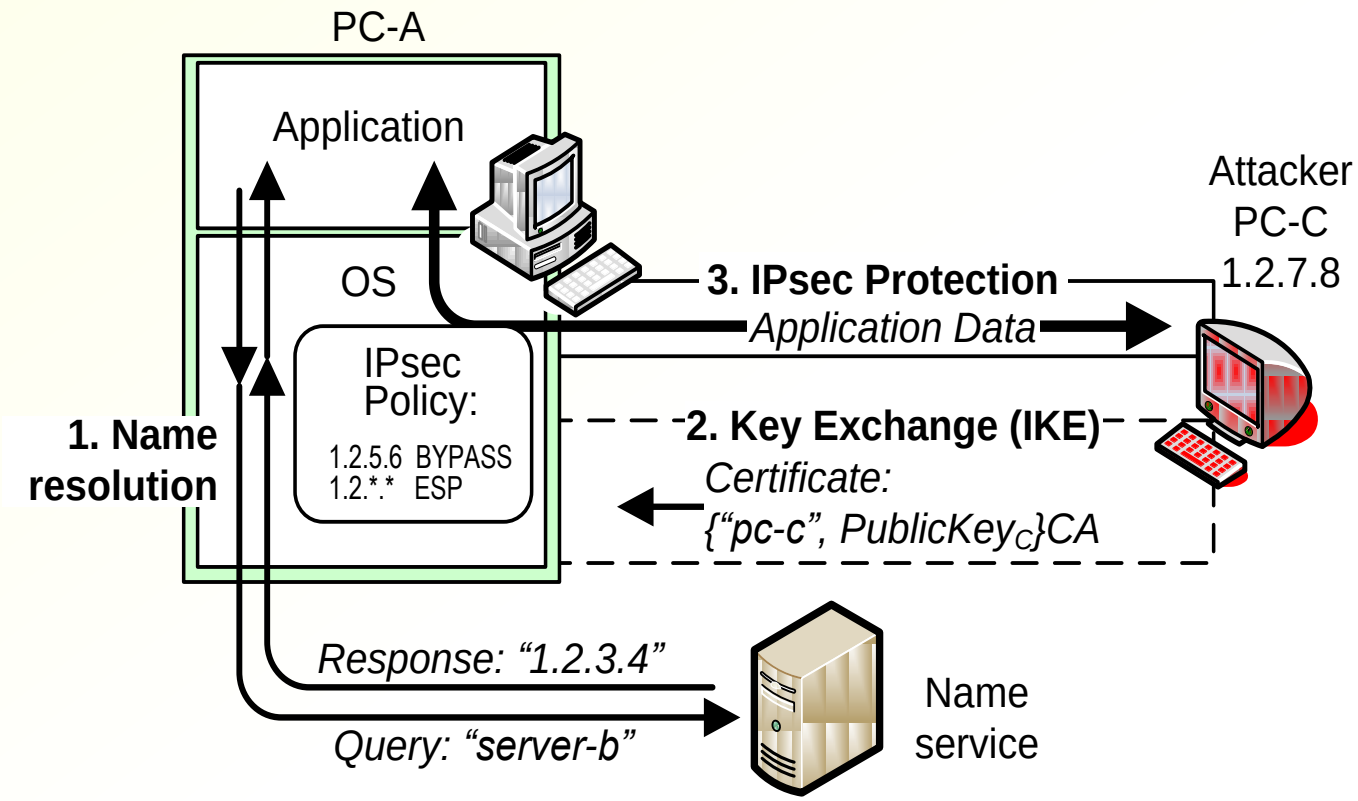
IPsec and Certificates



- Let's assume DNS is secure
- Another problem: IKE knows the peer IP address, not the peer name, but the certificate only contains the name

  → How does IPsec decide whether the certificate is ok?

IPsec and Certificates - Attack



- IPsec cannot detect the attack
- Secure NDS (forward lookup) does not help — why?
- Result: group authentication of those certified by the same CA
  → maybe ok for protecting an intranet where the goal is to keep outsiders out

Peer authorization database (PAD)

- IPsec spec [RFC4301] defines a database that maps authenticated names to the IP addresses which they are allowed to represent
  - How implemented? Secure reverse DNS would be the best solution — but it does not exist.
- Other solutions:
  - Accept that group authentication is ok — short-term solution
  - Secure DNS — both secure forward and reverse lookup needed, which is unrealistic
  - Give up transparency — extend the socket API so that applications can query for the authenticated name and other security state
  - Connect by name — change the socket API so that the OK knows the name to which the application wants to connect

Exercises

- For the IPsec policy examples of this lecture, define the IPsec policy for the peer nodes i.e. the other ends of the connections
- Try to configure the IPsec policy between two computers. What difficulties did you meet? Use ping to test connectivity. Use a network sniffer to observe the key exchange and to check that packets on the wire are encrypted
- Each SAD entry stores (caches) policy selector values from the policy that was used when creating it. Inbound packets are compared against these selectors to check that the packet arrives on the correct SA.
  - What security problem would arise without this check?
  - What security weakness does the caching have (compared to a fresh lookup through the SPD)?
  - Some IPsec implementations stored a pointer to the policy entry, instead of caching the selector. What weakness did this have?
  - RFC 4301 solves these problems by requiring that the SPD is **decorrelated**, i.e. that the selectors of policy entries not to overlap, i.e. that any IP packet will match at most one rule (excluding the default rule which matches all packet). Yet, the policies created by system administrators almost always have overlapping entries. Device an algorithm for transforming any IPsec policy to an equivalent decorrelated policy.

Related reading

- William Stallings. Network security essentials: applications and standards, 3rd ed.: chapter 6
- William Stallings. Cryptography and Network Security, 4th ed.: chapter 16
- Kaufmann, Perlman, Speciner. Network security, 2nd ed.: chapter 17 (not AH)
  - Note: chapter 18 on the older IKEv1 is historical
- Dieter Gollmann. Computer Security, 2nd ed.: chapter 13.3